Norsk Informatikkonferanse

Artikkelsamling

NIK & UDIT 2019

UiT, Narvik 25 – 27 Nov 2019

NIKT er en viktig nasjonal møteplass for forskere, undervisere, master- og PhD-studenter innen informasjons- og kommunikasjonsteknologi, informatikk, informasjonsvitenskap og informasjonssystemer.

About the conference series

This conference proceedings is part of the NIKT-conference series, that normally take place in November each year, where the venue changes each year. The conference calls for contributions from researchers and practitioners to make presentations and publish papers ongoing projects and new results within the broad field of information security. PhD-students are encouraged to present their research activities.

This paper collection was recovered, assembled and republished in a new issue due to OJS migration errors. Editor: S. F. Mjølsnes, .2.March, 2025.

NIKT Journal series e-ISSN: 1892-0721 https://www.ntnu.no/ojs/index.php/nikt

NIKT Journal is accredited to level 1 in the Norwegian register of authorized academic publishing channels.

Contents

Forord til NIK 2019 og UDIT 2019

Arne Lakså[1], Birgit R. Krogstie[2] og Knut Collin[1]
[1] Institutt for datateknologi og beregningsorienterte ingeniørfag, UIT Norges arktiske universitet
[2] Institutt for informatikk og e-læring, Norges teknisk-naturvitenskapelige universitet

Norsk Informatikkonferanse (NIK) har vært et årlig arrangement siden 1988. Den 32. utgaven av NIK ble arrangert av UIT Norges arktiske universitet, Institutt for datateknologi og beregningsorienterte ingeniørfag. Konferansen fant sted på Quality Hotel Grand Royal Narvik**,** som del av Norsk IKT konferanse for forsking og utdanning.

NIK er en bred nasjonal konferanse for formidling av forskning og avansert utviklingsarbeid innen informatikk. Konferansen inviterer til innsending av artikler innen ulike sjangre og tema relatert til teoretisk og anvendt informatikk. NIK åpner for tverrfaglighet og mangfold i forhold til metodiske tilnærminger og anvendelsesområder.

Denne utgaven av tidsskriftserien Norsk Informatikkonferanse inneholder to seksjoner. Den første samler alle bidrag som ble presentert på NIK 2019. Den andre er dedikert til et eget spor for Utdanning og Didaktikk i IT-fag (UDIT), som ble arrangert for femte gang i år, i samarbeid med NOKOBIT. UDIT har hatt sin egen programkomité, bidragene er fordelt på NIK og NOKOBIT sine tidsskrifter.

Norsk Informatikkonferanse er godkjent som publiseringskanal på nivå 1 i det norske systemet for dokumentasjon av vitenskapelig publisering og blir for tiden offentliggjort i gjennom Open Journal Systems hos Bibsys. Samlet fikk NIK 29 komplette innsendte bidrag og aksepterte 16. Samlet mottok UDIT 18 bidrag og aksepterte 12.

NIK programkomité 2019

- Arne Lakså, UIT, Narvik (leder)
- Knut Collin, UIT, Narvik (nestleder)
- Susanne Koch Stigberg, Høyskolen i Østfold
- Eric B. Jul, Universitetet i Oslo
- Lars Michael Kristensen, Høgskulen på Vestlandet
- Tor-Morten Grønli, Høyskolen Kristiania
- Vladimir Oleshchuk, Universitetet i Agder
- Anh Nguyen Duc, Høgskolen i Sørøst-Norge
- Anders Andersen, Universitetet i Tromsø
- Reggie Davidrajuh, Universitetet i Stavanger
- Rune Hjelsvold, NTNU, Gjøvik
- Anne C. Elster, NTNU, Trondheim

Følgende personer har hjulpet til med fagfelle vurdering av innsendte bidrag til NIK 2019: Asbjørn Danielsen, Arild Steen, Helge Fredriksen, Øyvin Halfdan Thuv, Børre Bang, Marius Geitle, Marius Akerbæk, Vigdis Holen, Sukalpa Chanda, Michael Riegler, Andreas Biørn-Hansen, Maben Rabi og Siri Fagernes.

UDIT Programkomité 2019

- Birgit R. Krogstie, NTNU (leder)
- Hugo Nordseth, Nord universitet (co-chair)
- Erlend Tøssebro, Universitetet i Stavanger (co-chair)
- Ingrid Chieh Yu, Universitetet i Oslo (co-chair)
- Per Andersen, Universitetet i Agder
- Kirsti Berntsen, NTNU
- Morten Goodwin, Universitetet i Agder
- Tor-Morten Grønli, Høgskolen Kristiania
- Ibrahim A. Hameed, NTNU
- Erik Hjelmås, NTNU
- Rune Hjelsvold, NTNU
- Basel Katt, NTNU
- Per Lauvås, Høgskolen Kristiania
- Yngve Lindsjørn, Universitetet i Oslo
- Omid Mirmotahari, Universitetet i Oslo
- Robin Munkvold, Nord universitet
- Ottar Osen, NTNU
- Ragnhild Runde, Universitetet i Oslo
- Helga Dis Sigurdardottir, Nord universitet
- Guttorm Sindre, NTNU
- Trond O. Skevik, Nord universitet
- Girts Strazdins, NTNU

Narvik/Trondheim,
november 2019

Arne Lakså
Birgit Rognebakke Krogstie
Knut Collin

Pass/Fail Grading and Educational Practices in Computer Science

Torstein J. F. Strømme
Department of Informatics, University of Bergen, Norway
torstein.stromme@uib.no

Abstract

Binary (pass/fail) grading have been shown to have benefits with respect to mental health and collaboration, and is argued to promote a deep approach to learning. However, diverging results with respect to academic achievement suggests that the full benefits of binary grading are contingent on underlying factors, such as how the teaching and learning activities in the course are designed.

We here present experiences and student feedback for an intermediate level course in computer science that is graded using pass/fail, and which is highly successful both in terms of of student enjoyment and academic achievement. Survey results also indicate that students apply a deeper learning approach towards the course than average.

Drawing on examples and findings from this course, we argue that the following three practices makes a binary graded course in computer science successful: a) a sufficiently high bar for passing, b) clear course requirements, and c) the use of formative assessment.

1 Introduction

The assessment is arguably the most important landmark that students use when deciding how to work with a class, and what to focus on in their studies [1]. How assessment is done has therefore attracted great attention in education research. For instance, highly regarded principles such as constructive alignment [2] and integrated course design [3] both place heavy emphasis on making assessments align well with the intended learning outcomes, in order that the perceived authenticity will motivate the students to go deeper with the material. It is also well established that formative assessment, the practice of letting assessment be an integral part in the process of learning and working with the material, carries significant benefits [4].

In addition to the assessment format and its frequency, there is also a question of grading, and in particular the grading scale. Across the world there are multiple traditions in higher education when it comes to grading scales, for instance the six-tier A-F scale used in the ECTS framework in Europe, or the five-tier scale (A/B/C/D/F) commonly used in the United States. However, in many of these systems there is also an option of using a binary (pass/fail) grading scale for individual courses. In Norway, Jørgensen and

This paper was presented at the NIK-2019 conference; see http://www.nik.no/.

Bråten [5] recently argued that a ternary (pass/fail/honors) grading scheme should also be made available as an option.

Proponents of binary (and sometimes ternary) grading scales argue that they give less stress and anxiety among students, increase collaboration and sparks a more intrinsic motivation than finer-grained scales do, all without a decline in academic achievement [5] (see also [6, 7, 8, 9, 10]). However, some studies also find that academic achievements actually do decline under binary grading schemes [11, 12], and that students with pass/fail grades suffer in the job market [13]. It is also argued that the stress caused by competition in the job market will cause students with only pass/fail grades to pick up more extracurricular activities in order to distinguish themselves in job applications, reducing the time available for studies [14, 15].

Most of the positive effects of binary grading described in recent literature have occurred within the context of medicine in North America, and it is possible that the particularly rigid traditions in medical studies versus other subjects explains why the results appears to be diverging. In any case, it is clear that simply changing the grading scale is not a magic wand one can use without regard to how the course is otherwise designed. The current paper is a first look into what can make a pass/fail -course successful in the context of computer science.

Aim: To describe educational practices which are favorable when designing a successful pass/fail course in computer science.

2 The Course

We begin by describing the course, called *Algorithms Engineering* (course code INF237). It yields 10 ECTS credits upon completion, and is taught every spring semester. It is the only regular course with a pass/fail grading scheme in the Department of Informatics at University of Bergen (UiB), Norway. The course is practically oriented and involves a lot of programming, however a good understanding of the theoretical algorithms is important. There are no formal requirements for the course, but it is highly recommended to have completed one or two courses in algorithms and data structures prior to enrolling. It is typically taken the last semester of bachelor studies, but almost 40% of students are on the master level. The course is not mandatory for any degree, however it is recommended for students pursuing a masters degree in algorithms.

There is no final exam in the course, but the final grade is decided by work submitted throughout the semester. All submissions are entirely auto-graded[1], so a student will at every point throughout the semester be aware of exactly how (s)he is doing within a few seconds after submitting a task on an assignment. Even though the feedback from the course judge is limited in its level of detail, the learning process is fully integrated with the assessment – in this sense the assessment is formative. There is no penalty for attempting the same task many times before getting it accepted.

In order to pass the course, a student is required to successfully pass 13 assignments, one for each week of lecture. Typically, each assignment consists of four tasks, and the requirement for passing the assignment is to successfully solve at least two of them. The assignment for a particular topic is released the same day as the lecture on that topic, and is accepting submissions in a three to four week period.

In addition, there are two oral presentations in which the students will be asked to explain a chosen part of their submitted work. Some students perceive this oral

[1] Complete source code for each task is submitted to the online judge Kattis at https://uib.kattis.com.

presentation as an exam, however the primary purpose of the presentations is to detect cheating, and most students find the presentations quite easy in retrospect. During the presentation, students will be asked to present some of their submitted tasks, for instance solutions which the instructors find either suspicious or particularly interesting.

With respect to cheating, plagiarism detection is automatically conducted by the course judge. All submissions for a problem are checked against each other, including (if the task is publicly available) thousands of submissions from around the world[2]. Instructors are made aware of possible plagiarism cases in the graphical user interface of the course judge, and a few cheating cases have been detected each year.

Students in the course are also required to participate in at least two programming contests, and have the option to participate in four or five. If a student participates in more than two contests, (s)he will receive wild cards which can be used to skip one task in an assignment. Two wild cards may not be used towards the same assignment.

Over the years we cover in this paper (2017, 2018 and 2019), the course was taught by three different instructors and had different sets of exercises, but maintained essentially the same structure as described above. In 2018 the course had two different instructors due to paternity leave, where the author taught the first half of the course, and the same instructor as in 2018 taught the second half of the course. A third instructor taught the course in 2019.

3 Methodology and background data

A digital survey was distributed to all students who had successfully solved at least half the assignments in either 2017, 2018 or 2019 (respectively 16, 39 and 29 students, a total of 84). This includes students who ultimately failed or dropped the course due to either cheating or simply not finishing all the assignments (5 students). The survey was sent to all respondents during summer 2019, so those who took the course in 2017 and 2018 needed to recall experiences much further back than those who took the course in 2019.

The questionnaire consisted of three main sections, and there was an opportunity to make separate comments on each section of the submission. There was also a field where the respondents could provide general constructive feedback about the course if they wished.

In the first section we ask the respondents to compare the course to other courses they have undertaken in computer science. Claims (e.g. "The course is easy") are presented to the respondents, and answers are reported on an modified Likert scale where the alternatives are:

(1) This statement applies much more to INF237 than to my other courses.
(2) This statement applies slightly more to INF237 than to my other courses.
(3) This statement applies roughly the same to INF237 as to my other courses.
(4) This statement applies slightly less to INF237 than to my other courses.
(5) This statement applies much less to INF237 than to my other courses.

In the second section, we ask a few items specifically on experiences with the pass/fail grading scale. Answers here are recorded on a standard Likert scale.

[2]While the course judge Kattis allow us to create tasks which are private to the UiB domain, most tasks are picked from a large database of problems open for everyone at https://open.kattis.com

The third and final section of the survey is an edited version of Biggs' revised two-factor study process questionnaire (R-SPQ-2F) [16], in which the students are asked to compare their study habits in Algorithms Engineering to their computer science studies in general. The intention is to get an indication for how the course itself affects deep and surface learning, as opposed to measuring the learning approach of the student mass in general.

In our edition of the R-SPQ-2F questionnaire, the statements respondents face are the same as in the original (except the question 18, which is not applicable for Algorithms Engineering and was removed), however the 5-point scale ranging from "always or almost always true of me" to "never or only rarely true of me" was replaced by a 5-point scale where the options were instead:

(1) This statement applies much more to INF237 than to my studies in other courses.
(2) This statement applies slightly more to INF237 than to my studies in other courses.
(3) This statement applies roughly the same to INF237 as to my studies in other courses.
(4) This statement applies slightly less to INF237 than to my studies in other courses.
(5) This statement applies much less to INF237 than to my studies in other courses.

For each of deep approach and surface approach, half the statements apply to them as described in [16]. In order not to confuse our results with those of a proper R-SPQ-2F questionnaire, we will simply report the average value answered to the statements in each of the two categories.

4 Results

Response rate for the survey was 52% (for years 2017, 2018 and 2019 respectively 56%, 54% and 48%). One response was removed from the data set, since the answers appeared inversely correlated to the written comments of the same submission. Hence, final sample size used in the analysis is $n = 44$, out of which 42 answered every multiple choice item.

Demography

Gender	86% male, 9% female, 5% prefer not to say
Age*	Average 23.3, SD 2.57, min 19, max 31
Course year	20% 2017, 48% 2018, 32% 2019
Towards degree*	39% master, 59% bachelor, 2% other

Table 1: Background of respondents. *At the time when course was taken.

There was no item on the survey where respondents across different years answer significantly different, except one: Respondents who took the course in 2017 recommend it slightly more (2017 average: 1.0, 2018 average: 1.57, 2019 average: 1.54).

Of students responding, 85% belong to the Department of Informatics at University of Bergen (UiB), whereas the others were equally distributed between the Department of Information Science and Media Studies at UiB and the Department of Computing, Mathematics and Physics at collaborating institution Western Norway University of Applied Sciences (HVL).

Claims about the course	Ave	1	2	3	4	5
The objectives of the course are clear.	1.61	50	41	7	2	0
The course requires a high cognitive level of reasoning.	1.70	43	43	14	0	0
The course requires memorizing many details.	3.52	5	16	25	32	23
The course is easy.	4.20	0	5	11	43	41
The course is fun.	1.34	73	20	7	0	0
The course has a high workload.	1.48	59	34	7	0	0
The course is relevant to my future career.	2.16	27	41	23	7	2
I learned a lot in this course.	1.52	52	43	5	0	0
I recommend this course.	1.44	67	26	2	5	0

Table 2: About the Course. The scale ranges from 1 ("this statement applies much more to INF237 than to my other courses") to 5 ("this statement applies much less to INF237 than to my other courses"). Values on the right are percentages.

About the course

Respondents generally report favorable views of the course, saying that the course has clearer objectives, is more fun, and more relevant to their future careers than a typical computer science course (see Table 2). 93% of respondents report that they recommend it more than other courses. It is also a pretty strong consensus that the course is hard, has a high workload, and requires a high cognitive level of reasoning, but also that one learns more in the course than what is normal. Some typical comments include:

> Toughest course I took in UiB but also the one I probably learned the most from and it was highly beneficial for my future career. (2018 master student)

> I would have recommended the course based on what we learn, but it requires a lot of work, so it can't be recommended for anybody. (2019 bachelor student)

> I liked the course, and i really liked the course style, it forces you to work with the course evenly over the whole semester, and that was good for me. And the topic of the course is really fun also! (2018 bachelor student)

> I felt the course was very good. Best class the entire semester. It was interesting and challenging, but not overly challenging either. (2019 bachelor student)

> This is one of top three courses I have taken at university (...). It was a very heavy subject, but it was luckily engaging enough that I never considered giving up. I believe, though, that I wouldn't have made it through without a group of people to work and discuss with. (2017 bachelor student)

> Love that you use everything you learn. It was my favourite course. Think I spent 90% of my time on this course, and winded up only getting 20 [ECTS credits] that semester. On the downside, I have nothing but a – pass – to show for all my efforts in it. (2018 master student)

Claim	Ave	1	2	3	4	5
I would have preferred an A-F grading scheme in INF237 over the current pass/fail scale.	4.07	2	9	11	34	43
I would have preferred a pass/fail grading scheme in the computer science core courses (for example INF100/101/102) over the current A-F scale.	3.80	5	7	34	14	41
I prioritized other courses than INF237 because they used an A-F grading scheme	4.30	0	9	7	30	55
If all my other courses also had a pass/fail grading schemes, I would have spent more time on INF237.	3.88	5	12	16	26	42
For a course with a pass/fail grading scheme to be successful, it is important that it is challenging to receive a passing grade.	1.88	37	42	16	5	0
For a course with a pass/fail grading scheme to be successful, it is important to receive continuous feedback on how you are performing throughout the semester.	1.60	53	33	14	0	0

Table 3: About the pass/fail grading scale. The scale ranges from 1 (“I strongly agree”) to 5 (“I strongly disagree”). Values on the right are percentages.

About the pass/fail grading scale

Respondents generally agree that pass/fail is a suitable scale for the course, even though 13% of respondents would have preferred an A-F scale. The main criticism of the binary grading scale is that a pass/fail does not reflect the work they put into the course when calculating grade averages (in Norway, pass is treated like a C when calculating grade point average). Consider these comments:

> I thought the pass/fail was unsuitable for inf237. The amount of work that I put in was probably enough to get an A or a B, but instead I got a pass... The amount of work put into inf237 also affected the grades in my other courses, so my average became lower than I wanted. (2018 master student)

> The only downside of pass/fail, is that it counts as a C when calculating average grades. It is therefore hard to justify using that much time on the class. The reason I still did was because I enjoyed it and found it challenging. (2017 master student)

On the other hand, a significant majority (76%) prefer binary grading in the course. Yet, when it comes to the introductory core courses in computer science, the picture is entirely different; only 11% of respondents report that they would like them be graded pass/fail, whereas 57% prefer the finer grained A-F scale. There is in other words a huge gap between the support for binary grading in Algorithms Engineering versus in the core courses in computer science.

As for traits that the respondents believe are important in order to make a course with binary grading successful, 78% agree that it should be challenging to receive a passing grade, and 87% agree that it is important to receive continuous feedback on performance throughout the semester. A recurring comment was also the importance of clear and objective criteria for passing:

> Very clear conditions for passing the course [makes a course suitable for pass/fail grading]. Students are pretty much hardwired to optimize for good grades, and in courses where what constitutes a pass is more nebulous, you spend less time learning and more time obsessing over what the instructor is looking for, and what is enough to pass. I've occasionally spent entire days on such discussions, and it is very counterproductive. It worked well in INF237, because the objectives were very clear (...). (2017 bachelor student)

> The course objectives should be unambiguously measurable, meaning the criteria must be clearly defined. INF237 does one thing correctly, the fact that with each task the student knows if he/she will pass or fail the course. This is good because the student knows what he/she did not understand early on the course. (2019 bachelor student)

Other comments mention that classes without final exams are a good fit for binary grading, but that good systems for detecting plagiarism are important, and that many small assignments is more suitable than a few large ones. We also note that students generally do not down-prioritize the course compared to courses with fine-grained grading scales. See Table 3.

Study habits

$n = 42$	Score	SD	Min	Max	α
Deep approach	2.42	0.51	2.26	2.57	0.77
Surface approach	3.57	0.47	3.43	3.72	0.74

Table 4: Study habits. The scale ranges from 1 to 5, where values below 3 indicate that this approach is relatively more applied in Algorithms Engineering, and values above 3 indicate that this approach is relatively more applied in other courses. Min and max values indicate the 95% confidence interval for the score, and α is the Cronbach alpha -value.

Our modification of Biggs' R-SPQ-2F questionnaire is not well suited to analyze the actual level of deep learning and surface learning of the respondents, nor is it the intended purpose. It rather attempts to measure how a particular course influence these learning strategies. We find that a deep approach is applied more in Algorithms Engineering than in other courses, whereas a surface approach is applied more in other courses, see Table 4. The Cronbach alpha for deep approach was 0.77, and for surface approach was 0.74 (slightly higher than those reported by Biggs for the R-SPQ-2F, respectively 0.73 and 0.64 [16]).

Even though the majority report having a deeper than average approach to Algorithms Engineering, it is not the case for everyone. A respondent writes:

> The main reason I answer in the direction of not spending time on learning topics in depth or spending time on studying topics on your own is that for me 237 was a course where my time was stretched to my very limit (...). If 237 was slightly easier, I would definitely spend more time learning more about everything from the course. (2018 master student)

There were also a few respondents that were confused about the questionnaire itself, since the modification of the answer alternatives did not allow the respondent to express their full view on the issue. For instance, we received this feedback:

> [This section was] a bit hard to answer, since the scale does not take into account whether I agree to the claim at all (that is, neither in INF237 nor other subjects). I therefore answered 3 both when I agreed and disagreed to a claim, if the claim applied equally much to INF237 as other courses. (2017 bachelor student)

Course statistics

There are a few statistics available from the online course judge (Kattis), indicating how students have worked with the course. We report here the extent to which students work more to improve on an assignment after it is already accepted, as this indicates that the student is using a deep approach to their studies. This is something every single student have actually done to some extent, though exactly how much varies. There are two basic ways to work on a submission after it is already accepted:

The first way is to submit improved versions of already accepted assignment tasks. On average, a student submits such improved versions at least 10.8 times during the semester. We remark that this is merely a lower bound, since students also have the option of submitting their code to a public judge for which we do not collect statistics. Both on the public and on the course judge there is a scoreboard ranking the best solutions by execution time, adding an element of gamification some students appreciate.

The second way to go beyond the requirements on an assignment, is to solve more than the required number of tasks. This requires significant extra effort, as the difficulty of the extra task is typically higher. A median student solves 3 tasks (average: 3.8) more than the minimum during the course, which corresponds to roughly 11% more work than necessary (though this is hard to quantify in terms of working hours). Only 5% of students passing the course have not done any excess tasks.

5 Discussion

Algorithms Engineering is a binary graded course where students have frequent and regular assessments with automatic grading, which we regard as a type of formative assessment. The bar for passing the course is high and rigid, and for many students this means that relatively many working hours are required to pass. However, the failure rate is low - among students who completed at least half the assignments (that is, who do not drop the course early), only 3 out of 83 students have been unable to successfully complete the course.

What separates strong from weak students is thus not primarily what is learned, but how much time is spent learning it. In this sense, the course achieves the elusive goal of outcome-based education: By the end of the educational experience, every student have achieved the intended goal [17]. A “pass” in the course is hence not merely a diploma for effort and participation, it is a true certificate that the candidate knows the topic.

The course appears to encourage a deep approach among students, and discourages a surface approach (see Table 4). This is also supported by the extra work students put into the course beyond what is required; every single student in the course have done some unnecessary improvement to their submission at some point throughout the semester, and only 5% of passing students solved only the minimum required number of tasks. Since this also requires solving conceptually harder tasks than necessary, it demonstrates a genuine interest and a deep approach to the course.

We believe the course offers some valuable ideas for designing courses that work well with a pass/fail grading:

A challenge. In order to facilitate a good learning experience, it is important that the learning outcomes of the course are (perceived to be) significant. This view is also supported by a large majority of the students who took the course (Table 3). While binary grading may under some circumstances make students sloppy, our experience is that it can also sharpen the students when the bar is high.

Clear requirements. While clear requirements is an advantage in all courses, it is of particular importance when the difference between two adjacent grades is so large. If a student receive "fail", the student him/herself should ideally know so before the instructor. Students repeatedly mention this as an important trait of a well-designed pass/fail -course. Clear requirements in combination with effective strategies to detect cheating also means that a course is perceived as fair.

Formative assessments. Students need to always be aware of how well they fare compared to the course requirements. This is important for continuous motivation throughout the semester, and for a good experience as a whole. An overwhelming majority of students agree (Table 3).

While it is not directly investigated in this paper, we also believe that the instructor and course staff taking a supportive role "helping the students fight the inflexible requirements" is a big positive. Due to the rigidity of the course requirements it has indeed been possible to assume this role in the vast majority of student-instructor interactions. We also see that many students appreciate the supportive atmosphere in the course, where discussing the exercises with one another is encouraged. A competitive gamification element in the course still exists, but it is detached from the final grade.

The above mentioned traits are not equally present everywhere in computer science education, and the wide gap between the support of binary grading for one class versus others (see Table 3) suggests that not all courses are created equal. If we still wish to pursue the benefits of binary grading (for instance with respect to mental health, deep approach learning and collaboration), then a complete redesign might be necessary in many courses. On the other hand, the traits that makes pass/fail work well for Algorithms Engineering can certainly have positive effects in the context of finer grading scales as well; *any* course can be challenging, have clear requirements and use formative assessment.

Binary grading does, however, make it easier and more natural to follow those principles. For instance, clear course requirements are much easier to formulate for a binary scale than for a finer scale (though not impossible, as seen for instance in Nilson's specifications grading [18]). By lightening the burden of marking assessments, pass/fail can also free more resources to provide formative assessment. Lastly, we believe a certain level of challenge arise naturally when combining frequent formative assessment with binary grading.

And while binary grading supports a high bar for passing, clear course requirements and formative assessment, the converse is doubly true: A binary graded course without a high bar can make students uninterested and prioritize other subjects perceived to have higher learning outcomes. A binary graded course without clear requirements and formative assessment can leave the student flustered, frustrated and unmotivated. We believe these principles and binary grading are mutually supporting each other.

Limitations

The sample size for this survey is only $n = 44$, though the trends seems strong enough that the reported results are still significant.

Our modification of the R-SPQ-2F is not a validated instrument, and we do not know how the scores will look if students in other courses were to answer a similar survey. The results are therefore merely an indication that deep approaches are employed more. Moreover, there was a slight confusion over how the questions should be answered.

Students electing to take the course are often aware beforehand what they are signing up for, including that the course has a high workload and very stringent requirements for passing. This self-selection process will bias any results we record, and it is hence not straightforward to determine which effect the course format itself is responsible for. It is, for instance, entirely plausible that the particular format of the course attracts specifically students who appreciate pass/fail as a grading scale. However, the wide gap in support for binary grading in Algorithms Engineering versus the introductory core courses indicate that it is not obviously so.

It is also plausible that the reputation of being a work-intensive course combined with offering small rewards (or even punishment) in terms of calculated grade point average attracts a type of student that is naturally predisposed to deep learning approaches. However, if the deep and surface approaches were merely traits of the individual student, one would expect them to employ deep learning approaches equally much in all courses.

Conclusion

Binary graded courses in computer science can be successful with respect to both student enjoyment and academic achievement, and also stimulate deep approaches to learning by employing the following educational practices:

- Set the bar for passing sufficiently high.
- Provide clear course requirements.
- Use formative assessments.

We argue that there is a synergy between these principles which work especially well with pass/fail grading. The combination of formative assessments and clear course requirements ensures that every student can closely monitor his/her own progress and allocate the required resources to work on the topic. This allows raising the bar for a pass without losing many students, ensuring high academic achievement. After all, students appreciate challenges as long as they overcome them.

Further work

The course discussed in this paper is a free elective with a certain reputation among students, giving rise to a bias in the student mass. It would be interesting to draw experiences from a mandatory course and investigate to which extent the current findings still hold. A next step might be a qualitative study which spans across multiple classes that has binary grading; this would provide a better background for further research in the area.

Another open question is the relationship of stress/anxiety and difficulty. While existing literature have established that binary grading reduce stress and anxiety [10], it is not entirely clear how this relates to the perceived difficulty of obtaining a pass. What happens when the challenge becomes too challenging?

Thanks

Many thanks to Daniel Lokshtanov and Jan Arne Telle who were the other instructors of the course during the period of this study. We would also like to thank the many instructors before them who helped develop Algorithms Engineering into what it is today.

References

[1] Arild Raaheim. Eksamensrevolusjonen. råd og tips om eksamen og alternative vurderingsformer. *Nye perspektiver på evalueringsformer i universitetspædagogik*, page 87, 2016.

[2] J. Biggs and C. Tang. *Teaching For Quality Learning At University*. SRHE and Open University Press Imprint. McGraw-Hill Education, 2011.

[3] L. Dee Fink. *Creating significant learning experiences: An integrated approach to designing college courses*. John Wiley & Sons, 2013.

[4] Neal Kingston and Brooke Nash. Formative assessment: A meta-analysis and a call for research. *Educational measurement: Issues and practice*, 30(4):28–37, 2011.

[5] C. Jørgensen and H. Bråten. Can 'passed with distinction' as a new grading scale favour the transition towards formative assessment? *Nordic Journal of STEM Education*, 3(1):7–11, 03 2019.

[6] Daniel E. Rohe, Patricia A. Barrier, Matthew M. Clark, David A. Cook, Kristin S. Vickers, and Paul A. Decker. The benefits of pass-fail grading on stress, mood, and group cohesion in medical students. *Mayo Clinic Proceedings*, 81(11):1443 – 1448, 2006.

[7] Robert A Bloodgood, Jerry G Short, John M Jackson, and James R Martindale. A change to pass/fail grading in the first two years at one medical school results in improved psychological well-being. *Academic Medicine*, 84(5):655–662, 2009.

[8] Darcy A Reed, Tait D Shanafelt, Daniel W Satele, David V Power, Anne Eacker, William Harper, Christine Moutier, Steven Durning, F Stanford Massie Jr, Matthew R Thomas, et al. Relationship of pass/fail grading and curriculum structure with well-being among preclinical medical students: a multi-institutional study. *Academic Medicine*, 86(11):1367–1373, 2011.

[9] Lynne S Robins, Joseph C Fantone, Mary S Oh, Gwen L Alexander, Marshal Shlafer, and Wayne K Davis. The effect of pass/fail grading and weekly quizzes on first-year students' performances and satisfaction. *Academic Medicine*, 1995.

[10] Laura Spring, Diana Robillard, Lorrie Gehlbach, and Tiffany A Moore Simas. Impact of pass/fail grading on medical students well-being and academic outcomes. *Medical education*, 45(9):867–877, 2011.

[11] Richard M. Gold, Anne Reilly, Robert Silberman, and Robert Lehr. Academic achievement declines under pass-fail grading. *The Journal of Experimental Education*, 39(3):17–21, 1971.

[12] David E Suddick and Robert E Kelly. Effects of transition from pass/no credit to traditional letter grade system. *The Journal of Experimental Education*, 50(2):88–90, 1981.

[13] John A. Dietrick, Michael T. Weaver, and Hollis W. Merrick. Pass/fail grading: A disadvantage for students applying for residency. *The American Journal of Surgery*, 162(1):63 – 66, 1991.

[14] Ray Guo and Jennifer Muir. Torontos honours/pass/fail debate: The minority position. *University of Toronto Medical Journal*, 85(2):95–98, 12 2008.

[15] Jan Frich, Knut E.A. Lundin, and Ingrid Os. Karaktersystemet avveining mellom ulike hensyn. *Tidsskrift for Den norske legeforening*, 134(1):14–15, 01 2014.

[16] John Biggs, David Kember, and Doris YP Leung. The revised two-factor study process questionnaire: R-spq-2f. *British journal of educational psychology*, 71(1):133–149, 2001.

[17] William G Spady. *Outcome-Based Education: Critical Issues and Answers.* ERIC, 1994.

[18] Linda Nilson. *Specifications grading: Restoring rigor, motivating students, and saving faculty time*. Stylus Publishing, LLC, 2015.

Den tause kunnskapen i IT-studia

Hans Georg Schaathun
⟨georg@schaathun.net⟩
NTNU — Noregs Teknisk-Naturvitskaplege Universitet

Samandrag

Michael Polanyi er kjend for omgrepet *taus kunnskap*. Han viste at me ofte veit meir enn me kan fortelja. Dette ser me tydleg i profesjonsstudia. Studentane treng ein praktisk kompetanse som førelesarane slit med å setja ord på. Denne artikkelen oppsummerer litt av det som har vore skrive om kva taus kunnskap er, kvifor han er taus og korleis han vert formidla. Me argumenterer for at det er den tause kunnskapen som skil menneske frå maskin og dermed er det som studentane våre treng i ein kvar jobb som ikkje snart vert robotisert.

1 Innleiing

Ei av dei største utfordringane for oss i utdanningssystemet er balansen mellom teori og praksis. Me ser det i konflikten mellom industrikompetanse og forskarkompetanse ved tilsetjingar. Me ser det ved prosjektbasert læring, der både studentar og sensorar kan vera usikre på kva vurderingskriterium som ligg til grunn når det innleverte arbeidet skal få karakter. Svært sjelden finn man tydleg formulerte vurderingskriterium som kunne sikra objektive og forutseielege karakterer.

Ei anna utfordring, som slektar på den fyrste, er manglande samanheng mellom teori og praksis. Den teoretiske kunnskapen er proposisjonell, dvs. me kan formulera han i ord, og ein stor del av forskarkompetansen handlar nettopp om å skriva (og lesa) for å omsetja kunnskap i ord. Den praktiske kunnskapen er gjerne taus (*tacit knowledge* som Michael Polanyi kalla det), og mange dyktige praktikarar maktar ikkje å forklara kva dei gjer eller kvifor dei gjer som dei gjer. Slik opererer praktikarane og teoretikarane på kvar sin kognitive arena og kan bidra me svært lite til studentane utover sin eigen arena.

Taus kunnskap er vanskeleg å handtera i pensum. Regelverket for høgare utdanning krev skriftlege læringsutbytebeskrivingar og skriftleg sensorvegleiing som skal sikra objektiv og etterprøvbar sensur. Taus kunnskap er nettopp taus fordi me ikkje maktar å forklara kva me veit. Korleis kan me då skriva sensorvegleiing og emneskildring?

I denne artikkelen tek me for oss konseptet taus kunnskap saman med det beslekta omgrepet *techne* frå Aristoteles, gjerne omsett som kunst, samt danningsomgrepet frå tysk og skandinavisk didaktikktradisjon. Me viser kvifor kunst og taus kunnskap er noko anna enn vitskapleg kunnskap, og kvifor det er minst like viktig i eit profesjonsstudium. Målet er å setja ord på ein del av utfordringane som me har i IT-utdanningane, og vonleg inspirera til vidare diskusjon.

Denne artikkelen vart presentert på konferansen NIK-2019; sjå http://www.nik.no/.

2 Tilbake til Aristoteles

Aristoteles (1999, Bok 6, Kap. 1) delte den rasjonelle sjela i to delar, den *vitande* delen (ἐπιστημονικὸν, epistêmonikon) og den *overvegande* delen (λογιστικόν, logistikon). Den vitande delen tek seg av kunnskap om ting som ikkje kunne ha vore annleis, medan den overvegande delen søkjer sanning om det som kan vera annleis. Dette skiljet er fundamentalt. Kunnskap eller *episteme* (ἐπιστήμη) handlar om evige sanningar. Overveging, på den andre sida, søkjer innsikt i einskildtinga og dei unike falla som avheng av tid og stad. I informatikkjargon kan me seia at kunnskap handlar om klassene, medan den overvegande delen skal forstå instansane.

Til den overvegande delen høyrer mellom anna *techne* (τέχνη), som gjerne vert omsett til kunst eller kunnen og er evna til å skapa gode og riktige ting. Dei praktiske IT-faga våre handlar nettopp om å skapa. Studentane våre skal skapa programsystem som skal bidra til «det gode livet». Dyden som studentante skal læra seg i slike fag er dermed *techne*. *Techne* dreier seg ikkje om å finna løysingar som alltid er sanne, men om å finna løysingar som er *gode* i den konkrete, unike situasjonen som ein har framfor seg.

I skiljet mellom *episteme* og *techne* kan me kanskje sjå det kjende skiljet mellom teori og praksis, men då skal me vera nøye med kva me meiner med praksis. Episteme er kunnskap om ting som ikkje kunne vera annleis. Me kan gjerne bruka praktiske øvingar og studentaktive læringsformar for å formidla *episteme*, t.d. i aritmetikken. Oppgåver som $812 + 373 =$ eller $1243 : 17 =$ har eitt svar som ikkje kunne vera annleis. Når studentane øver på å rekna for hand er det difor stadig *episteme* dei lærer, sjølv om oppgåva på sett og vis er praktisk og i alle fall studentaktiv.

Døme nærare vårt eige fag finn me i algoritmeteorien. Som matematikken, er algoritmeteorien formell. Både objekt og problem er veldefinerte, og algoritmane som me studerer, er universelt gyldige innanfor den formelle modellen. Dijkstras algoritme finn alltid den kortaste stigen i ein graf (på visse veldefinerte vilkår). Vidare kan me finna universelle sanningar om køyretid og kompleksitet. Alt dette er *episteme*, og når me bed studenten om å implementera Dijkstra i python eller i assembly, er *episteme* stadig det primære målet. Studenten implementerer algoritmar for å hugsa algoritmen, ikkje for å læra å programmera.

Techne kjem inn i biletet når me hentar problemet frå røynda, heller enn frå den abstrakte grafmodellen. Tak t.d. eit logistikkproblem. Me skal finna kortaste køyreveg herfrå til Gokk. Utfordringa er ikkje algoritmen; den er universell. Utfordringa er heller å modellera geografien som ein vekta graf slik at den universelle algoritmen kan brukast. Her nyttar det ikkje berre å vita, me er nøydde til å overvega. Kva meiner me med kortast? Tid eller distanse? Kor detaljert skal modellen vera? Presisjon må vegast opp mot køyretid. Kor lenge vil brukaren venta på utrekninga? Korleis presenterer me resultatet for brukaren? Alt dette krev merksemd om enkelttinga i det konkrete logistikkproblemet. Difor er *episteme* ikkje nok.

Aristoteles (1999, Bok VI, Kap. 8) var ikkje særleg optimistisk til utdanning for den overvegande sinnsdelen. Unge mennesker kan nok «bli geometere og matematikere og flinke i slikt,» meinte han,

> *men klokt kan ikke et ungt menneske være. Grunnen til dette er at klokskap også gjelder enkeltting, noe man får kjennskap til gjennom erfaring, mens den unge jo er uerfaren. For det er bare nok tid som gir erfaring.*

Me skal ikkje hevda at skiljet mellom *techne* og *episteme* er like svart/kvitt som

Aristoteles ser ut til å meina i *Etikken*. Det er det grunnleggjande skiljet mellom kunnskap om det universelle og forståing for enkelttinga som er interessant. Konsepta *techne* og *episteme* kan me då sjå på som ytterpunkt og som representantar for hhv. overvegande og vitande sinnsevner. Gråsonen kan ein evt. koma tilbake til seinare. Me skal utdjupa dette skiljet med hjelp av nokre meir moderne tenkjarar, og drøfta kva det kan fortelja om utdanninga i IT-fag i dag.

3 Den tekniske rasjonaliteten

Akademia har tradisjonelt hatt lite tid til *techne*. Idealet for framveksten av moderne vitskap har vore universelle sanningar og kontekstfrie resultat. Vitskapen har òg vorte idealet for utdanninga. Skular og universitet søkjer å formidla universell sanning. Som me har høyrd ein del sosialøkonomar formulera det, *røynda er berre eit uinteressant spesialtilfelle*. Schön (1983) er ein av dei som har studert problema denne tradisjonen har skapt for profesjonsutdanningane.

Profesjonsutdanningane

Profesjonane i Schöns vokabular omfattar so vidt forskjellige disiplinar som arkitektur, design, ingeniørfag, kliniske fag, osv. Desse profesjonane vert ofte omtalt som design, i tråd med ordbruken etter Simon (1996, p. 111):

> *Everyone designs who devises courses of action aimed at changing existing situations into preferred ones.*

Kjenneteiknet på design er altso at ein skaper nye og betre løysingar eller situasjonar. Dette står som motsats til vitskapen der ein søkjer å forstå eller forklara situasjonen slik han er. Her kjenner me att definisjonen av *techne* som skapande sinnsevner, til skilnad frå *episteme* eller vitskap. I denne ordbruken står då IT-faga fram som designfag.

Der har vore eit sterkt press gjennom meir enn 100 år for å gjera profesjonane mest mogleg vitskaplege. Yrkesgruppene har fått status gjennom den vitskaplege kunnskapen som dei legg til grunn. Moore (1970, p. 55f) skil mellom *avocations* og *professions*. Ein *avocation* er det motsette av ein profesjon, fordi han baserer seg på innarbeidd og tradisjonell praksis (customary practice) og vert utvikla gjennom individuell prøving og feiling.

> *The profession, on the other hand, involves the application of general principles to specific problems, and it is a feature of modern societies that such general principles are abundant and growing.*

Utdanninga i dei generelle prinsippa frå vitskapen fekk forrang, medan bruken på spesifikke problem gjerne vart sett på som triviell. Ingeniørskulane utvikla seg til ingeniørvitskaplege høgskular, der vitskapsfolk kunne nyta den høgaste statusen basert på heilt andre verdiar enn dei som ligg til grunn for profesjonen som studentane vart utdanna til (Schön, 1983, p. 27).

I 1983, då Schön gav ut den banebrytande boka si, kunne han observera ein aukande skepsis til og kritikk mot profesjonane gjennom to tiår. Trass i den suksessen som vitskapane hadde hatt i å frambringa ny kunnskap, var ikkje vitskaplege fundament nok til å sikra tillit til profesjonane. Både profesjonsmiljøa sjølve og folk flest hadde sidan 1963 vortne meir og meir kritiske til profesjonane (Schön, 1983, p. 39). Profesjonelle vurderingar er ikkje alltid nødvendige sanningar som ikkje kunne vera annleis, sjølv om dei byggjer på slike nødvendige sanningar frå vitskapen. I

det konkrete, individuelle tilfellet kan ulike ekspertar gjerne koma fram til ulike konklusjonar, sjølv om ingen tvil herskar rundt den underliggjande og generelle vitskapen. Kva tillit kan me då ha til profesjonskunnskapen?

Reflektert praksis

Ein sentral observasjon hjå Schön (1983) er at dyktige profesjonsutøvarar møter nye problem som unike fall og ikkje som instansar av generelle klasser som er dekt av eksisterande kunnskap. Unntak er regelen. Universelle sanningar strekk ikkje til.

Lækjaren kan kjenna ei lang rekkje sjukdomar og diagnosar som vitskapen har skildra i detalj, men pasientane er ofte ikkje læreboksdøme. Mange pasientar har fleire sjukdomar og atypiske symptom, som er vanskelege å skilja frå kvarandre, og ein kur som verkar positivt på ei av lidingane kan verka negative på den andre (Schön, 1983, s. 64f). Pasienten er ein «enkeltting», og lækjaren må sjå etter det særeigne i den individuelle pasienten. Fleire vitskaplege teoriar kjem saman i ein ny og unik kombinasjon, og lækjaren treng *techne* for å overvega slutningar frå ulike teoriar.

Den rådande epistemologien for profesjonane kalla Schön (1983) for den *tekniske rasjonaliteten*. Etter dette synet dreier profesjonspraksis seg berre om å løysa instrumentelle problem vha. vitskaplege teoriar. Det går greitt so lenge problema er velformulerte, men i røynda, skriv Schön, er dei fleste problem kaotiske og uoversiktlege. Veldefinerte problem er noko som profesjonsutøvaren konstruerer sjølv. Å finna riktig spørsmål er ofte både viktigare og vanskelegare enn å svara på det.

Behovet for å omforma problem er stort i faget vårt. Stadig ser me digitaliserte prosessar som er meir tungvinte og tidkrevjande enn gamle papirbaserte prosessar (Olsen, 2014). Grunnen er at dei digitale løysingane let dei gamle papirbaserte prosessane definera problemet, og gjerne kopierer prosessen steg for steg. Til dømes skriv Olsen (2014):

> *Da bilprodusenten Ford skulle effektivisere sine innkjøp skaffet de seg et avansert datasystem som sendte ordrer og andre dokumenter elektronisk. Toyota, derimot, fjernet behovet for disse dokumentene! Ved å gi leverandørene innsyn i Toyotas datasystem ble ordrer unødvendige.*

Ford har altso gått ut frå ei universell forståing av korleis ordreprosessen foregår. Dei har løyst problemet ut frå den same problemforståinga som låg til grunn då papirprosessen vart etablert. Toyota har på den andre sida vore merksame på enkelttinga, dvs. det som er særeige for den tida som kjem. Ein ordreprosess i det 21. århundredet er ikkje det same som ein ordreprosess i det 20., og dette tek Toyota omsyn til. Toyota har altso omforma problemet ved å sjå på enkelttinga, heller enn å føresetja det universelle. Dei stør seg på *techne* og ikkje berre på *episteme*.

Innanfor designtradisjonen er der kultur for å stilla spørsmål ved problemet (Norman, 2013, s. 220), og mange designarar vert upopulære fordi dei bruker altfor lang tid før dei tek til å løysa problemet. Grunnen er likevel openberr. Altfor ofte gjer me som Ford og løyser eit problem basert på føresetnader frå ei anna tid eller ein annan bransje.

Refleksjon-i-handling

Schön byggjer bøkene sine på ei lang rekkje observasjonar av praksis. Han observerer at utøvarane utviser stor kunnskap og forståing når dei stiller spørsmål ved

problemet, og gjerne forkastar det til fordel for eit anna problem. Denne kunnskapen er i stor grad taus. Dvs. utøvaren kan ofte ikkje setja ord på kva han gjer eller korleis han tenkjer, sjølv om det er tydleg at han *er* svært kyndig i det han gjer. Det er kunnskap *i handling*, eller *techne*, som det ikkje er trivielt å kopiera for dei ukyndige.

Eit av hovudmåla for Schön er å gjera ein del av den tause kunnskapen eksplisitt. Han føresleg ein epistemologi som han kallar *reflection-in-action*. Epistemologi viser etymologisk til *episteme*, men likevel er det ikkje her tale om å skaffa kunnskap om ting som ikkje kunne vore annleis. Schön gjev oss ein «epistemologi» for *techne*[1]. Målet hans er kunnskap som er objektiv ved at ein kan oppdaga feil (Schön, 1987, s. 79). Dette er eit objektivitetskriterium som høver for *techne*.

Epistemologien byggjer Schön opp rundt fellesdraga i vidt forskjellige profesjonar, med psykoterapi og arkitektur som dei mest detaljerte døma. Refleksjon-i-handling omfattar to kognitive element:

1. ein eksperimentell metodologi.

2. eit repertoar av døme samla gjennom eigne røynsler.

Eksperimenta i Schöns metodologi er ikkje lab- og felteksperiment som me er mest vane med. Dei har meir til felles med tankeeksperimenta som Einstein var kjend for, og som me finn i litteraturen tilbake til Ørsted i 1811 (Brendel, 2004).

Arkitektar og designarar presenterer gjerne tankeeksperimentet i perspektivskisser og planteikningar. Psykologen i Schöns døme eksperimenterer med hypotetiske forklaringsmodellar som er uttrykt som narrativar. Programutviklarar kunne bruka klassediagram og andre formar for diagram (med eller utan den strenge syntaksen frå t.d. UML). I tankeeksperimentet kan ein testa mange ulike variasjonar i eit konsept på kort tid.

Det som er vesentleg for objektiviteten i prosessen i fylgje Schön, er for det fyrste at ein utbroderer konsekvensane av kvar variasjon i tilstrekkeleg detalj til å kunna sjå inkonsistensar og andre negative og positive fylgjer. Når ein føresleg eit klassediagram i UML må ein t.d. vera nøye med å definera mål og meining ved kvar klasse og samanhengane mellom dei. I tankeeksperimentet kan ein forestilla seg kva som ville skje dersom ein sidan må endre den eine klassa og kva utviklingsbehov det kunne resultera i. Høg kopling og manglande koherens vert då noko røynleg og konkret, fordi ein kan leva seg inn i *konsekvensane* for framtidig arbeid, og ikkje berre prata om arkitekturteoretiske konsept.

Dette tyder ikkje at ein må modellera alt før ein programmerer. Somme tider vert tankeeksperimentet for krevjande slik at ein treng ein prototyp for å vurdera konsekvensane. Vidare er metodikken like relevant ved utviding og refaktorering som ved opprinneleg design.

Denne eksperimentelle metodologien kan lærast (Schön, 1983; Schön, 1987). Kunsten er å utdjupa den tenkte situasjonen i slik detalj at ein verkeleg kan leva seg inn i fylgjene, og leva seg fullt og heilt inn i forestillinga *som om* den tenkte situasjonen var verkeleg. Det er vanskelegare å sjå korleis eksperten vel kvart trekk, for her byggjer han på heile porteføljen av døme som er samla gjennom ein lang karriere.

[1]Ein epistemologi for techne høyrest ut som ei sjølvmotseiing i det greske vokabularet som Aristoteles brukte. Me kan verta freista til å kalla det ein *technologi*, men diverre, dét ordet er allereie teke til ei anna meining.

Taus kunnskap

Mykje av kompetansen som Schön observerer hos profesjonsutøvaren er taus. Omgrepet «taus kunnskap» vart introdusert av Polanyi (1983), som observerte at *me kan vita meir enn me kan fortelja*. Sjølv om me kan kjenna igjen eit andlet blant millionar, tyder ikkje det at me kan skildra andletet me kjenner att, eller forklara kva me ser etter. Det motsette av taus kunnskap vert gjerne kalla *proposisjonell kunnskap*, dvs. kunnskap som kan formulerast i ord, gjerne i logiske utsegner (proposisjonar).

Hendrix (1947) viste at taus kunnskap, som ho kalla *unverbalized awareness*, enklare let seg overføra til nye problem enn proposisjonell kunnskap. Ho studerte matematikkdidaktikk på barneskulenivå, og observerte at elevar som lærte å rekna gjennom oppgåver og døme, vart flinkare til å overføra kunnskapen enn dei som lærte generelle rekneregler fyrst. Ved å starta med oppgåver og døme studerer ein enkeltting og byggjer opp ein form for *techne*. Dei som lærer generelle reglar fyrst, får dei universelle sanningane (*episteme*) som proposisjonell kunnskap. Hendrix viste vidare at det skader forståinga om den tause kunnskapen for tidleg vert gjort proposisjonell.

Sjølv om den tause kunnskapen vert opparbeidd gjennom å studera enkeltting, so handlar han ikkje om detaljkunnskap om enkelttinga. Elevane i Hendrix sin studie etablerer ein generell rekneteknikk sjølv om dei ikkje kan setja ord på han, og der er ingen grunn til å tru at dei hugsar dei einskilde oppgåvene dei har vore gjennom. Polanyi (1983) viser òg at inngåande studie av detaljane kan øydeleggja heilskapsforståinga som ligg i den tause kunnskapen.

Den tause kunnskapen er ein svart boks, der me berre kan observera handlingar og skaparverk som kunnskapen leier til. Der kan godt vera generelle mekanismar inni boksen, men me manglar orda til å forklara mekanismane. Taus kunnskap kjem til kort når me ynskjer å forvissa oss om at universelle utsegner er sanne. Enkelttinga kan me observera og studera i verda rundt oss. Universalitet er derimot ein abstraksjon som me konstruerer og uttrykkjer gjennom ein eller annan form for språk. Vitskapleg kunnskap, eller *episteme*, er difor proposisjonell.

4 Danning og utdanning

Utdanningsfilosofien skil gjerne mellom *danning* og *utdanning*. Dette skiljet finst på tysk og alle dei skandinaviske språka, men ikkje på fransk og engelsk der *education* dekkjer båe tydingane (Kemp, 2013). Danningsaspektet har ein tendens til å verta fortrengt i utdanningssystemet. I fylgje Kemp (2013, s. 180f) er skiljet mellom danning og utdanning det same som mellom forståing og forklaring:

> *Forklaring skulle gælde det gentagelige og lovmæssige i erkendelsen, mens forståelse skulle gælde de individuelle éngangsfænomener, der ikke kunne begribes med nogen kausalforklaring.*

Her kjenner me att skiljet mellom den vitande og den overvegande sinnsdelen frå Aristoteles. Det går an å vita dei gjentakelege og lovmessige forklaringane. Den overvegande sinnsdelen kan derimot forstå enkeltting der forklaringane kjem til kort. Kemp (2013, s. 181) skriv vidare at

> *uddannelsens indhold er en viden, der kan oplagres og gemmes væk, hvis ingen har brug for den, og tages op og anvendes, når der bliver behov for det.*

Ordet «viden» er den alminnelege omsetjinga av *episteme*, og det er naturleg å lesa han slik at *techne* fell utanfor. Den tause kunnskapen fell tydleg utanfor, sidan han ikkje kan lagrast opp og gøymast som proposisjonell kunnskap.

Det er danninga som tek for seg den tause kunnskapen, forståing og *techne*. Kemp (2013, s. 181) skriv at danninga «må ses som en historisk proces med utallige momenter av éngangsfænomener» og gjev berre meining om «den udvikles, bæres og overtages af mennesker, der gøres og gør sig dannede». Kemp meiner altso at *techne* ikkje berre vert utvikla av kvar enkelt gjennom røynsler, slik som Aristoteles såg ut til å meina. Me utviklar *techne* (òg) gjennom ein historisk prosess, eller fagtradisjon, som spenner generasjonar. Ein generasjon kan overføra taus kunnskap til neste, men berre gjennom direkte kontakt mellom menneske.

Utdanning må gjerne inngå som ein del av danningsprosessen (Kemp, 2013, s. 181). Når sinnet skal overvega løysingar på eit konkret problem, kan vitskapleg kunnskap gjerne vera ein del av vurderingsgrunnlaget, ved sidan av taus kunnskap. Som Ricœur skriv må ein somme tider forklara meir for å forstå betre (Kemp, 2013, s. 182). Det er derimot ikkje gode forklaringar og *episteme* som kjenneteiknar dei dyktige studentane som er klare for kritiske oppgåver i arbeidslivet etter studiet.

Læring som etterlikning

Både Schön (1987) og Kemp (2013) dreg fram etterlikning som læringsmetode for å tilegna seg den tause kunnskapen. Studenten etterliknar meisteren. Dette er, seier dei begge, kontroversielt på eit vis. Ingen liker etterlikningar. Der er noko billig og mindreverdig ved ein kopi. Kva er då bra med etterlikning?

Etterlikning er ikkje det same som etteraping (Kemp, 2013, s. 205). Den etterlikninga som me søkjer er den skapande etterlikninga (Kemp, 2013, s. 200), som Aristoteles (1902) kalla *mimesis*, eit omgrep som er mykje brukt i filosofihistoria. Schön (1987, s. 108) omtaler etterlikninga (*imitation*) som ein selektiv konstruksjon. Når me etterliknar, vel me ut det som er vesentleg for oss og konstruerer det på nytt med vårt eige uttrykk.

Den som aper etter meisteren liknar likevel ikkje meisteren, for meisteren er original. Innan sveinen aper etter er meisteren gått vidare. Utfordringane som sveinen skal løysa er ikkje dei same som meisteren løyste. Som Schön fortel oss er kvar profesjonelle utfordring unik. Sveinen som blindt kopierer meisteren tek ikkje den unike situasjonen på alvor. For å gjera ein god jobb må sveinen *tenkja* som meisteren tenkte og og tolka førebiletet inn i si eiga konkrete og unike situasjon. Tankesettet kan vera ei etterlikning, men løysinga som han finn kan ikkje vera det. Løysinga skal vera særeigen for si eiga tid og stad.

Når me ser etterlikninga som som ei kognitiv etterlikning, ser me òg langt forbi mykje av kritikken mot mimesistanken, der ein har fokusert på mimesis som etterlikning av fysiske handlingar og andre ytre faktorar[2]. Etterlikninga er ikkje ei sovepute for å sleppa å tenkja sjølv, for det er nettopp i tanken at ein skal etterlikna.

Live-koding som undervisingsform

Innanfor programutvikling ser me fleire førelesarar som har innført det som dei kallar *live*-koding som førelesingsform. Førelesaren løyser då ei programmeringsoppgåve frå kateteret, med skjermvising på prosjektor. Han arbeider som regel utan manuskript

[2]For ein gjennomgang av kritikken mot mimesistanken, kan ein sjå Kemp (2013).

og det er ikkje på førehand opplagt kva løysing som vert vald ved kvar krossveg. Det er tvert imot viktig at studentane ser at tvil og overveging er ein naturleg og uunnverleg del av prosessen. Studentane vert gjerne inviterte med på råd.

Det interessante ved *live*-koding er at det viser fram kunnskapsformar som elles sjelden kjem til syne i undervisinga. Studentane vert vitne til meisteren sin kunnskap-i-handling, hans *techne*. Meisteren formidlar prosess heller enn produkt, og studentane vert inviterte til å etterlikna tankeprosessen meir enn sluttproduktet. Når meisteren må overvega ulike designval, kan han formidla at der ikkje finst nokon fasit. Svara han kjem fram til, kunne gjerne ha vore annleis. Ulikt meir teoretiske fag, må studentane søkja etter noko anna enn *episteme*.

Der er ein underliggjande fare i all meisterlæring, for at meisteren opptrer so meisterleg at prosessen vert skjult (Schön, 1983, s. 104), berre resultatet vert synleg. Me ser ofte studentar som trur at læraren forventar at det *veit* svaret. Når det ikkje er mogleg å vita, gissar dei heller enn å overvega. Schön (1987, s. 322) legg vekt på at tradisjonelle disiplinar må undervisast slik at utforskingsmetodikken vert synleg. Berre då kan studentane etterlikna prosessen heller enn produktet. Dette kan ein oppnå i *live*-koding med tilstrekkeleg tålmod og omtanke.

Praktisk kompetanse og aktiv læring

Sokalla moderne undervisingsformar legg stor vekt på aktiv læring og praktisk kompetanse. Stundom vert det framstilt som om aktiv læring automatisk gjev praktisk kompetanse. Den føregåande drøftinga gjev derimot grunnlag for å problematisera dette inntrykket.

Den praktiske kompetansen er overvegande og ikkje vitande. Studenten som vert sett til å rekna aritmetikk eller implementera kjende algoritmar, er absolutt aktiv, men han arbeider med problem der me *veit* løysinga. Der kan vera fleire alternative løysingar (t.d. ei rekursiv formulering og ein løkkeimplementasjon), men i dei fleste tilfelle går oppgåva ut på bruka éi løysing for å hugsa ho. Inga overveging trengst for å løysa oppgåva. Studenten endar opp med ny *episteme*, han hugsar ei løysing som (i essens) ikkje kunne vera annleis.

Motsett ser me at *live*-koding ikkje er meir student-aktivt enn (andre) tradisjonelle førelesingar. Likevel er det overvegande kompetanse, *techne*, som vert formidla, i langt større grad enn kunnskap. Studenten får sjå korleis førelesaren tenkjer og overveg i ein praktisk situasjon, og kan etterlikna tankesettet.

Dette skiljet mellom *techne* og *episteme* kan moglegvis kasta nytt lys på eksisterande kontroversar innanfor aktiv læring. Det er ei kjend sak at empirien ikkje gjev eintydig støtte til aktiv læring. Mayer (2004) er ein særleg fyrig kritikar, medan Prince (2004) peiker på blanda resultat når ein ser på fleire ulike læringsutbyte og viser til eit døme frå medisin. Studiar tyder på at prosjektbasert læring (PBL) gjev ei viss forbetring på klinisk kompetanse og ei viss forverring på standardiserte eksamenar. Prosjektbasert læring er ein læringsaktivitet som tek utgangspunkt i enkelttinga, og det er rimeleg å tru at ein då trenar dei overvegande evnene som ein treng i klinisk praksis. Dersom dei standardiserte eksamenane testar kunnskap om ting som ikkje kunne vera annleis, so er det kanskje ikkje overraskande om dei fordrar læringsaktivitetar som fokuserer på generell, proposisjonell kunnskap. Kan henda er styrken og ulempa ved PBL slett ikkje studentaktivitet men fokuset på *techne* over *episteme*.

Hendrix (1961) drøfta tre ulike læringsformar som alle kunne gå under nemninga

discovery learning. Den induktive metoden bruker oppgåver og døme, gjerne i ein nøye planlagd sekvens, med sikte på at studentane sjølv skal oppdaga og formulera dei generelle reglane. Undervisinga fokuserer dermed på enkeltting, medan meininga med læringa er generell og proposisjonell kunnskap. Samantreffsmetoden (*incidental method*) bruker praktiske prosjekt, der ein vonar at studentane skal støyta på og læra seg den generelle teorien ved behov. Igjen er målet generell og proposisjonell kunnskap. Den tredje metoden, som me nemnde tidlegare som *unverbalized awareness*, er den einaste der målet er taus kunnskap. Denne tause kunnskapen er meir effektiv enn proposisjonell kunnskap, sjølv om proposisjonell kunnskap er mogleg, seier Hendrix (1947).

Debatten handlar ofte om det er best å fortelja studenten det han treng vita, eller om det er best å lata han finna det ut sjølv (Lee and Anderson, 2013). Av Schön, Polanyi og Aristoteles har me derimot lært at der finst kunnskap som me kan fortelja til studenten, og annan kunnskap som det ikkje er råd å fortelja. Når me drøftar aktiv læring må me kunna skilja mellom desse to, og me skal for all del ikkje gløyma den kompetansen som ikkje kan forteljast. På ein eller annan måte må den òg formidlast.

5 Vegen mot IT-danning

Som nemnd inngår læring av *techne* i danninga, heller enn utdanninga. Me lærer frå både Kemp (2013) og Gadamer (1960) at danning er noko som vert utvikla gjennom ein kultur- eller fagtradisjon. Utan tradisjon finst der ingenting som studenten kan danna seg inn i. Kva kan me då seia om IT-tradisjonen?

Fagtradisjonen i IT-fag

IT-faga har eit særleg problem. Fagmiljøa våre har ikkje ein felles fagtradisjon som me byggjer på, og dei tradisjonane som evt. finst vert i liten grad formidla. I staden er der mange tradisjonar og ingen semje om kva rolle kvar tradisjon skal ha.

Den gongen datamaskina var ny for rundt 70 år sidan var det naturleg at utviklinga kom i matematikk- og elektronikkmiljø. Ein trengte elektronikkingeniørar for å byggja og vedlikehalda maskinvaren og matematikarar til å analysera og programmera samansette aritmetiske prosessar. Det er difor ikkje overraskande at me 30 år seinare fekk informatikkinstitutt skilt ut frå matematikkinstititutt og dataingeniørstudium skilt ut frå elektrostudium.

Allereie på 1960-talet stod datamaskina fram som eit universelt verkty som alle fag har bruk for. Ved Universitetet i Bergen var det so tydleg at ein gjekk inn for eit datastudium ved Historisk-Filosofisk Fakultet (Bagge, 1996). Svein Nordbotten vart utnemnd til professor i informasjonsvitskap frå 1971. Då hadde fakultetet nettopp vorte delt, og sjølv om informasjonsvitskapen var tenkt å tena både humaniora og samfunnsvitskap, var prof. Nordbotten sosiolog og faget vart fyrst forma i fagtradisjonen hans. Sidan kom der eit eige studium i humanistisk informatikk.

Informasjonsvitskap er rett nok særeige for Bergen, og mange tilsette som sjølve kjem frå informatikktradisjonen har vore med å (om)definera kva informasjonsvitskap skal vera i dag. Situasjonen illustrerer likevel at datafag kan byggja på meir enn éin tradisjon. Andre plassar finn ein datafag med røter i handelsfagstradisjonar, lingvistikk og andre fag.

Studieprogramma i datafag vert drege mellom alle desse ulike tradisjonane. Ikkje berre har me den same konflikten som Schön har peikt på, der studentane

søkjer danning innanfor profesjonstradisjonen medan undervisarane står i ein vitskapstradisjon. I tillegg er det uklart kva vitskapstradisjonar som kan vera relevante for profesjonen.

Svært mange av dei som underviser og datafag i dag har sine røtene sine i den matematiske informatikktradisjonen. Dei har lært den logisk-deduktive forskingsmetoden frå matematikken, og forstår faget ut frå det. Når ein utviklar komplekse informasjonssystem i industrien, er der mange andre relevante fag og metodeverk. Frå statsvitskap og handelsfag kan ein henta teoriar for organisasjonen og prosessen som informasjonssystemet skal støtta oppunder. I sosiologien finn ein metodikkar for å evaluera systemet i bruk.

Mange studieprogram søkjer å formidla kompetanse frå alle desse faga, men det er slett ikkje alltid dei har undervisarar som sjølv er danna innanfor den same fagtradisjonen. Dersom det berre hadde vore *episteme* dei skulle formidla, kunne dette kanskje ha gått, fordi episteme kan skrivast ned og hentast fram igjen i ein ny kontekst. Når me derimot skal formidla *techne* og taus kunnskap, må me vera bevisste på den rolla som kvar tradisjon skal ha. Det gjeld både dei ulike akademiske tradisjonane og profesjonstradisjonen.

Ein risiko som ein lauper når tradisjonen vert gløymt er at dagens sanningar, eller *best practice*, framstår som evige sanningar, og ikkje som det dei er, nemleg eit tilfeldig punkt på ei lang utviklingsline. Studenten kan lett misforstå eit resultat av *techne* som *episteme*. Me finn eit døme i smidige metodar. Hohl et al. (2018) peiker på skilnaden på «being agile» og «doing agile». Det er lett å læra seg ein av dei kjende smidige metodikkane, og gjera seg smidig etter boka. Når ein les historia bak smidige metodologiar ser me derimot ein reaksjon mot nettopp boka. Den som *er smidig* frigjer seg frå boka og ser på enkelttinga som det dei er. Difor har me òg fått mange smidige metodologiar som har oppstått i litt ulike historiske kontekstar. Kjennskap til variasjonane i tradisjon bidreg til repertoaret av døme som Schön viser til, og er difor nyttig for å utvikla og velja metodikk til konkrete prosjekt i framtida.

Eksamen som ein turingtest

Eit anna perspektiv på kunnskap kan me få gjennom forskinga på kunstig intelligens. Debatten om kva kunstig intelligens er, har pågått i fleire mannsaldrar. Kva krev me av ei tenkjande maskin?

Eit av dei mest kjende kriteria for ein tenkjande maskin er turingtesten, introdusert i 1950 av Turing (2009) som eit tankeeksperiment. Ein intervjuar stiller spørsmål til eit menneske og til ei maskin, utan på førehand å vita kven som er maskina. Maskina består testen om han kan imitera mennesket utan at intervjuaren kan sjå forskjell. Ei slik maskin må seiast å vera tenkjande.

Lat oss no snu dette på hovudet. Kva krev me av ein tenkjande student? I ei tid der mange yrke vert erstatta av robotar, er det klart at ein student som ikkje kan meir enn ei maskin på eksamen har dårlege sjansar i arbeidslivet. Dersom eksamen ikkje er egna til å skilja menneske frå maskin, vert ikkje studenten vurdert på kompetanse som han treng i yrkeslivet. Ein eksamen som ikkje fungerer som turingtest er dermed irrelevant for vurderinga av avgangsstudentar. Kva er det i utdanninga som me ynskjer å læra studentane og som me ikkje kunne ha lært til ei maskin?

Datamaskina utmerker seg på proposisjonell og generell kunnskap. Alt maskina bruker av program og modellar er i prinsippet proposisjonelle og kan skrivast

ut som logiske proposisjonar. Datamaskina får aldri del i den tause kunnskapen, fordi kunnskap som vert programmert per definisjon vert proposisjonell. Dersom studentane våre skal ha ein plass i arbeidslivet, må styrken deira liggja i den tause kunnskapen og i forståinga av enkelttinga, altso *techne*.

Me har tidlegare kritisert matematikkundervisinga for å utdanna *computers* (Schaathun and Moe, 2019). På den tida, rundt 1950, då Turing skreiv om turingtesten, var nettopp dét ein vanleg fyrste jobb for unge ingeniørar. Kunnskapen dei trong som *computer* var proposisjonell kunnskap, om samanhengar som ikkje kunne vera annleis (i alle fall i stor grad), men teknologien den gongen var ikkje avansert nok til å automatisera alt som var proposisjonelt. I dag kan maskinene meir, men matematikkpensum har til samanlikning endra seg lite i den same perioden. Når me ser på matematikkeksamenar på ingeniørstudia, so er der få oppgåver som testar evner som ikkje maskinene kan gjera betre enn studentane. Kvifor er det slik?

Turingtesten er stadig eit tankeeksperiment. Opprinneleg handla eksperimentet om kva maskina kan gjera i framtida. No handlar eksperimentet vel so mykje om kva plass mennesket har når maskinene vert stadig meir avanserte. Dette er ein viktig tanke i studie- og læringsdesign. Kva lærer studentane som gjer dei betre enn maskiner? Korleis vurderer me denne ikkje-maskinelle kompetansen på eksamen?

Sjølv om det varierer mykje mellom fag og emne, er der skremmande mange eksamenar som kan løysast maskinelt. Det er eit paradoks som krev ein kritisk diskusjon.

6 Konklusjon

Både Aristoteles og Schön fortel oss at den skapande kompetansen byggjer på forståing om enkeltting. Vitskapleg kunnskap fokuserer på universelle sanningar. Det er kunnskap om ting som ikkje kan vera annleis. Slik kunnskap kan vera naudsynt, men det er ikkje tilstrekkeleg for å løysa praktiske problem, og me kan trygt overlata det meste av vitande kompetanse til datamaskiner.

Studentane treng kunnskap og forståing for enkelttinga, som dei møter i praksis, og det er viktig å understreka at det er noko meir enn praktiske øvingar og aktiv læring. Gjennom praksis kan me òg trena *episteme*. Dersom me skal formidla taus kunnskap må me ha eit tydleg fokus på å forstå enkelttinga og løysa problem som ikkje er dekte av generelle teoriar.

Skiljet mellom vitande, generell og proposisjonell kunnskap på den eine sida, og taus kunnskap og forståing for enkelttinga på den andre, reiser fleire brennaktuelle spørsmål rundt utdanningane våre.

- Lærer me studentane våre noko meir enn datamaskinene kan?
- Kva for ein taus kunnskap treng me å formidla?
- Korleis legg me til rette for formidling av taus kunnskap?
- Korleis vurderer me den tause kunnskapen på eksamen?

Ingen problem er løyst i denne artikkelen, men me vonar at drøftinga og kartleggjinga av dei filosofiske konsepta, kan inspirera til ein ny diskusjon rundt (ut)danningsopplegget.

Me skal ikkje hevda at den tause kunnskapen manglar heilt i utdanningane, men det er sjelden tydleg kva taus kunnskap studentane skal ha, korleis han vert vurdert eller korleis han vert formidla. Særleg kan me uroa oss over dei mange studentane som freistar å lesa på eiga hand. Dei finn ikkje den tause kunnskapen i bøkene, men

likevel gjer mange av dei det godt på eksamen. Her er der mange opne spørsmål for framtidig gransking.

Referansar

Aristoteles. *Poetics*. London. MacMillan and Co., Ltd., 1902.

Aristoteles. *Etikk*. Gyldendal Akademisk, 1999. Omsett av Anfinn Stigen.

Sverre Bagge. Samfunnsvitenskapenes historie. In *Universitetet i Bergens historie*, volume 2. Universistetet i Bergen, 1996.

Elke Brendel. Intuition pumps and the proper use of thought experiments. *Dialectica*, 58(1): 89–108, 2004. doi: 10.1111/j.1746-8361.2004.tb00293.x. URL `https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1746-8361.2004.tb00293.x`.

Hans-Georg Gadamer. *Wahrheit und Methode*. Tübingen, 1960.

Gertrude Hendrix. A new clue to transfer of training. *The Elementary School Journal*, pages 197–208, 1947.

Gertrude Hendrix. Learning by discovery. *The Mathematics Teacher*, 54(5):290–299, May 1961.

Philipp Hohl, Jil Klünder, Arie van Bennekum, Ryan Lockard, James Gifford, Jürgen Münch, Michael Stupperich, and Kurt Schneider. Back to the future: origins and directions of the "agile manifesto" – views of the originators. *Journal of Software Engineering Research and Development*, 6(1):15, November 2018. ISSN 2195-1721. doi: 10.1186/s40411-018-0059-z. URL `https://doi.org/10.1186/s40411-018-0059-z`.

Peter Kemp. *Verdensborgeren: pædagogisk og politisk ideal for det 21. århundrede*. Hans Reitzels Forlag, 2. reviderede udgave edition, 2013.

Hee Seung Lee and John R Anderson. Student learning: What has instruction got to do with it? *Annual review of psychology*, 64:445–469, 2013.

Richard E Mayer. Should there be a three-strikes rule against pure discovery learning? *American psychologist*, 59(1):14, 2004.

Wilbert E. Moore. *The professions: roles and rules*. Russel Sage Foundation. New York, 1970.

Don Norman. *The design of everyday things: Revised and expanded edition*. Constellation, 2013.

Kai A. Olsen. Låst i papirets byråkrati. *Bergens Tidende*, September 2014. `https://www.bt.no/btmeninger/kronikk/i/m5k3L/laast-i-papirets-byraakrati`.

Michael Polanyi. *The Tacit Dimension*. Peter Smith, Gloucester, Massachussets, 1983. First published Doubleday & Co, 1966.

M. J. Prince. Does active learning work? a review of the research. *Journal of Engineering Education*, 93(3):223–231, 2004.

Hans Georg Schaathun and Jan Gunnar Moe. Kva er eigentleg målet i matematikken? In Reidar Lyng, editor, *MNT-Konferansen*, Nordic Journal of STEM Education. March 2019. `https://www.ntnu.no/ojs/index.php/njse/article/view/2992`.

Donald A Schön. *Educating the reflective practitioner*. Jossey-Bass San Francisco, 1987.

Donald A. Schön. *The Reflective Practioner*. Ashgate Arena, 1983.

Herbert A Simon. *The sciences of the artificial*. MIT press, 1996.

Alan M. Turing. *Computing Machinery and Intelligence*, pages 23–65. Springer Netherlands, Dordrecht, 2009. ISBN 978-1-4020-6710-5. doi: 10.1007/978-1-4020-6710-5_3. URL `https://doi.org/10.1007/978-1-4020-6710-5_3`. Annotated version of the paper from the British journal *Mind* 1950.

Investigating students' journey through a computer science program using exam data: three new approaches

Madeleine Lorås
NTNU
Hallvard Trætteberg
NTNU
Kshitij Sharma
NTNU

Abstract

A computing student will over the first three years of their studies complete approximately 20 exams and even more attempts due to failures and retakes. Details about all exam attempts are stored in a national database called Common Student System (Felles studentsystem, FS). Although access to FS, in general, is restricted, anonymized data about exam attempts can be provided for research and is a potential goldmine of data that, if used right might be a useful tool for educators and teachers. In this study, we explore this data in an attempt to conceptualize three new approaches to assessing student performance. Firstly, we relate students' final grade point average (GPA) to their performance in all courses in the first two years. Additionally, we propose a new indicator of student performance called "struggle factor," which is calculated using the number of exam attempts. Lastly, we investigate how students perform in different course subjects and types. Both the proposed use of FS data and the new approaches to performance indicators are relevant for educators wanting to understand the educational design of a study program and the students' journey.

Key words: Performance, Exam data, Educational design, Computing education

Introduction

In this study, we explore the FS data on our own students in an attempt to gain insight into student performance, both success, and failure. The overall research objective was to investigate what ways general institutional data can be used to assess student performance and aspects of the educational design. The research questions were as follows:

- How can FS data be extracted and managed ethically and practically?
- What insight into student performance can FS data give that could help in educational design?

In this context, student performance means both academic success, failure, and progression of the individual student, as well as the context of such behavior. For example, grades, attempts, and failed exams are indicators of performance, while the context relates to what kind of courses are challenging (e.g., computer science, mathematics or unrelated courses, project work or individual).

An essential underlying motivation for this work was the hunt for analytic tools on the study program level that can give educators insights into the educational design of study programs, both regarding practical implementation and quality assurance. Educators in this context means teachers and lecturers in various courses, as well as study program leaders, education managers and administrators working with teaching quality. In Norwegian higher education, as well as the rest of the world, universities and colleges are required to adhere to corporate governance, which entails finding good educational indicators to report, as well as useful planning and management tools. This contribution is an attempt to use already existing data in a new way, from educators who have experience "from the inside" of study program development and management.

Methodology

This research is designed as a retrospective quantitative study [3, 6]. The FS database allows us to go back over ten years, which makes it a useful dataset for a longitudinal study. Since the data is from the past, and any new data will be looking to the past as well, it must have a retrospective view. Lastly, the data is only quantitative as the exam submissions are not part of the database. Such studies are considered useful for establishing relationships and enables the dynamics of change and flow to be caught [3].

Population and program

The study investigates two undergraduate study programs; program A and B. Both these programs accept approximately 150 students each year and are considered relatively hard to get into, although the admission grade point average (GPA) for program B is significantly higher than A [8]. The gender balance for program A is consistent around 20%, while Program B has about 30% (+- 5% from year to year).

As you can see from Table 1, the educational design of the two programs are very similar. The main differences are related to the mathematics requirements and the number of optional courses. Program A has fewer mandatory mathematics courses than program B, which in turn opens up for more freedom to choose courses. For the first two years, however, the course plan is fixed.

Table 1: Educational design of Program A and B

Course	General description	Semester in program	
		Program A	Program B
Philosophy	Philosophy	1	1
CS1	Intro to programming, Python	1	1
Web 1	Introduction to web technology	1	
Math 1	Basic mathematics level 1	1	1
Discrete math	Discrete mathematics	2	1
CS2	Object oriented programming, Java	2	2
Arduino lab	Programming and technology, Arduino	2	2
Networks	Networks	2	4
Math 2	Basic mathematics level 2		2
Circuits	Electric circuits		2
Computers	Computers and digital design	3	3
Algorithms	Algorithms and data structures	3	3
SD1	Software development 1	3	
Security	Security in ICT systems	3	
Digital society	Digital society	3	
IoT lab	Programming and technology, IoT		3
SD2	Software development 2	4	4
HCI	Human computer interaction	4	4
DB	Databases	4	4
Statistics	Statistics		4
Web 2	Web development 2	5	5

Prog.lang	Programming languages	5	5
Low level	Low level programming	5	5
Cognitive arc	Cognitive architectures	5	5
Infosys	Information systems	5	5
AI1	Intro to AI	5	5
Info retrieval	Information retrieval	5	5
Software arc	Software architecture	6	6
OS	Operating systems	6	6
AI2	AI methods	6	6
Compilers	Compiler construction	6	6
Data mining	Data warehouses and data mining	6	6
Bachelors	Bachelors project	6	
Security 2	Software security	6	
Analytics	Intelligent text analytics	6	
Management	Technology management		6
Physics	Basic physics		6

Key:

Unique for one of the programs	The same for both programs	Same course, but different semesters

Data collection

In order to collect the necessary data in an ethical yet practical way, certain precautions had to be made. Although exam results are considered public information, the detailed data we were aiming for is not readily available. Therefore, we decided that there was a need to use the FS database; however, there were two main issues with that. Firstly, academic staff does not have direct access to FS for valid privacy reasons. Hence, a member of the administration who does have such access would need to extract the data on our behalf, which leads to a second issue: we would be collecting the data without informed consent from the students. With these issues in mind, we prepared a plan for anonymizing the data, which was approved by Norwegian Centre for Research Data (NSD).

The data set was extracted from FS by a member of the administration. The data set at this point uses the student's student number as an identifier, and in order to anonymize the data, this was replaced by a random identifier. This anonymized dataset did not contain any directly identifiable personal information and was therefore satisfactory to use in research. The random identifier does not relate to the original student number; hence, the researchers had no way of identifying a student.

Data set

The data set from FS contains exam results from students in two different study programs, classes starting in 2011-2015. That is five classes of students who have completed the first three years. In total, that is 1 809 students, who have completed 38 024 exams in 44 unique courses. For many study programs, students will have more and more freedom to choose courses as they progress. Therefore, this study limited the time span to the first three years and only looked at mandatory courses in those three years.

The data extracted from FS consisted of every exam attempt so that a student will appear several times in the data set. However, we are in this case, also interested in following a student as opposed to a course. Therefore, the data needed to be restructured so that every row in the dataset represented one student and their individual progression. For the purpose of analysis, it was useful to view both versions of the data, which is visualized in Figure 1:. Data set 1 shows the original data, exam-based, with each individual exam result per row. While data set 2 contains student-based data, with a student per row and corresponding courses. In addition, data set 2 includes individual variables such as GPAs, program, class, etc.

Figure 1: Visualization of the two different data sets

Data set 1		
Student 1	Course A	**Grade**
Student 1	Course B	…
Student 1	Course C	…
Student 2	Course B	…
Student 3	Course A	…
Student 3	Course C	…
…		
Student N	Course X	Grade

Data set 2					
Student 1	**Course A**	**Course B**	**Course C**	**…**	**Individual variables**
Student 2	…	…	…		…
Student 3	…	…	…		…
Student 4	…	…	…		…
Student 5	…	…	…		…
Student 6	…	…	…		…
…					
Student M	Course X	Course Y	Course Z	…	Individual variables

Variables

With the two datasets described above, there are many different variables that could be calculated in order to learn more about student performance and educational design. Performance variables can generally be viewed at four levels: institution, study program, course, and individual. Traditionally, institution and program-level variables are concerned with throughput and dropout rates. Courses are commonly assessed by pass/fail rates and grade distributions, while individuals are often measured by GPA and failure rates. In this study, we are only able to address program, course and individual levels and will focus on individual performance. Table 2 gives an overview of possible variables.

Table 2: Overview of possible performance variables

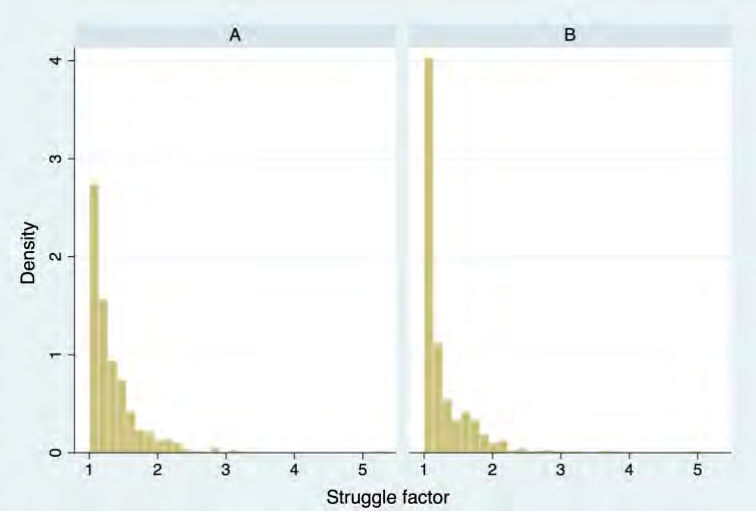
	Overall	**Year**	**Semester**	**Program**
Program	Within a program: the average grade overall	Within a program: the average grade for one year of exams	Within a program, the average grade of each semester	Drop-out rate
Course	For each course, the overall average grade for all years Pass/fail rates	For each course, the average grade for each year taught	For each course, the average grade for each year taught	For each course, the average grade for one year within a program
Individual	For each student, what is their overall average grade	For each student, what is their average grade per year	For each student, what is their average grade per semester	

GPA

For the analysis in this study, we have chosen to examine two GPA variants: the overall GPA for the student and the 6th semester GPA. The overall GPA was calculated by averaging the individual student's grades for all courses in the first three years. The 6th semester GPA, on the other hand, only consisted of grades in semester six. The reason for wanting to examine both these GPA variants was that the 6th-semester courses are on a higher level and therefore should

represent the final competence level. Whereas the overall GPA includes all courses, and in that way, might include low points periods in the students' academic progression. In other words, looking at only semester six GPA will exclude the possibility that students needed a few semesters to "get the hang of things."

Figure 2: Histograms of struggle factor for programs A and B.



The struggle factor

In addition to the GPA indicators, we have also explored a new approach. We have observed as educators that some students need several attempts to pass a course, or to get an acceptable grade. We were, therefore, curious if this could be a possible approach to performance that does not directly relate to GPA. To calculate this "struggle factor" we used the following formula:

$$Struggle\ factor = \frac{Total\ number\ of\ exam\ attempts}{Total\ number\ of\ courses\ completed}$$

Calculating the struggle factor like this for our data set resulted in a number between 1 and 5.2, for each student in the dataset. A struggle factor of 1 means that the student had the same number of exam attempts as courses completed, which indicates no struggle. A struggle factor of 5.2, on the other hand, indicates that the student has attempted 5.2 more exams than courses completed. In Figure 2 you can see that most students have a struggle factor of between 1 and 2. However, some students have a higher struggle factor, which is why the right tale of the histogram is so long.

It is important to note here that the assessment regime at NTNU allows students to retake exams for different reasons, which is why the number of attempts can be higher than the number of courses. Firstly, students who take an exam and fail will, of course, have the option to retake that exam. The same goes for students who are unable to take an exam because of medical reasons. A second possibility is retaking an exam to improve a grade, which is also possible. The last possibility is what is often referred to as a "tactical retake", where students can intentionally choose not to receive a grade in an exam in order to qualify for the retake. This is often something students can do when they believe they will not get the grade they aimed for, or for some other reason, wanted access to the retake exams. Retake exams are organized in the summer, and students only have access to these if they failed or did not receive a grade. If a

student wants to improve a grade, they would have to take the exam the next time the course is offered.

Course types and sets

A different perspective on performance is looking at how students perform in different course types. On the highest level, we can group the course into computer science (CS), mathematics and other courses (physics, philosophy, management, etc.). From Table 1, we can see that there are 30 CS courses, four mathematics courses and three other courses across both programs. Furthermore, some courses can be viewed as sets that build on each other. For example, CS1 and CS2 or the different mathematics courses.

Analysis and results

When it comes to exploring ways to evaluate student performance using FS data, we will in the following section present three approaches. Firstly, we visualize the first two years of courses with respect to the overall and 6th semester GPA. Secondly, we explore the struggle factor and what insights it can give into student performance. Lastly, the courses were grouped by type, and the differences evaluated. Both the method of analysis and the results will be presented for each approach.

Study program heat maps

The dependent variables for these approaches are the overall GPA (calculated using grades from all three years), the 6th semester GPA and the struggle factor. The independent variables in this analysis are the grades received in the mandatory courses of the first two years. Since the third year includes several optional courses, we were not able to analyze the data in a coherent way. Additionally, pass/fail courses were omitted since there was no variance in grade. For this analysis, we used data set 2.

A Pearson's correlation matrix was calculated to model the two study programs [2]. In order to visualize this, the correlations are summarized with a heat map as shown in Figure 3 and Figure 4, for programs A and B, respectively. In this heatmap, darker colors indicate a stronger positive correlation, while lighter colors are negative correlations. Every color shade relates to a specific Pearson correlation coefficient (r), as labeled on the right. Note that programs A and B have different coefficients. All variables are listed on both axes', hence creating the matrix; however, we have only included the upper half in order to make the map more readable. The first three variables from the top are the dependent variables; overall GPA, 6th semester GPA and struggle factor. Notice that since the struggle factor ranges from 1-5.2, meaning a high score equals a high level of struggle, the correlations are naturally negative. The remaining variables are the various courses in chronological order from first semester courses on the left/top to fourth-semester courses on the right/bottom (see Table 1 for details about the courses). In the following sections, we list the results for each dimension of the analysis; GPA, struggle factor and course types and sets.

GPA results

- The 6th semester GPA and overall GPA are highly correlated, as indicated by the dark color of the top left correlation box.
- Most courses seem to have a high correlation to overall GPA, as indicated by all the dark correlation boxes. Exceptions will be further discussed in the course type section.
- The correlations between courses and the 6th semester GPA are not as strong as the overall GPA, as indicated by the fact that the whole row for the 6th semester GPA is lighter than the overall GPA.
- For program A the relation between 6th semester GPA and the different courses is weaker than for Program B, as indicated by the lighter color of the 6th semester row.

Figure 3: Heat map plot of the first two years for Program A. N=265

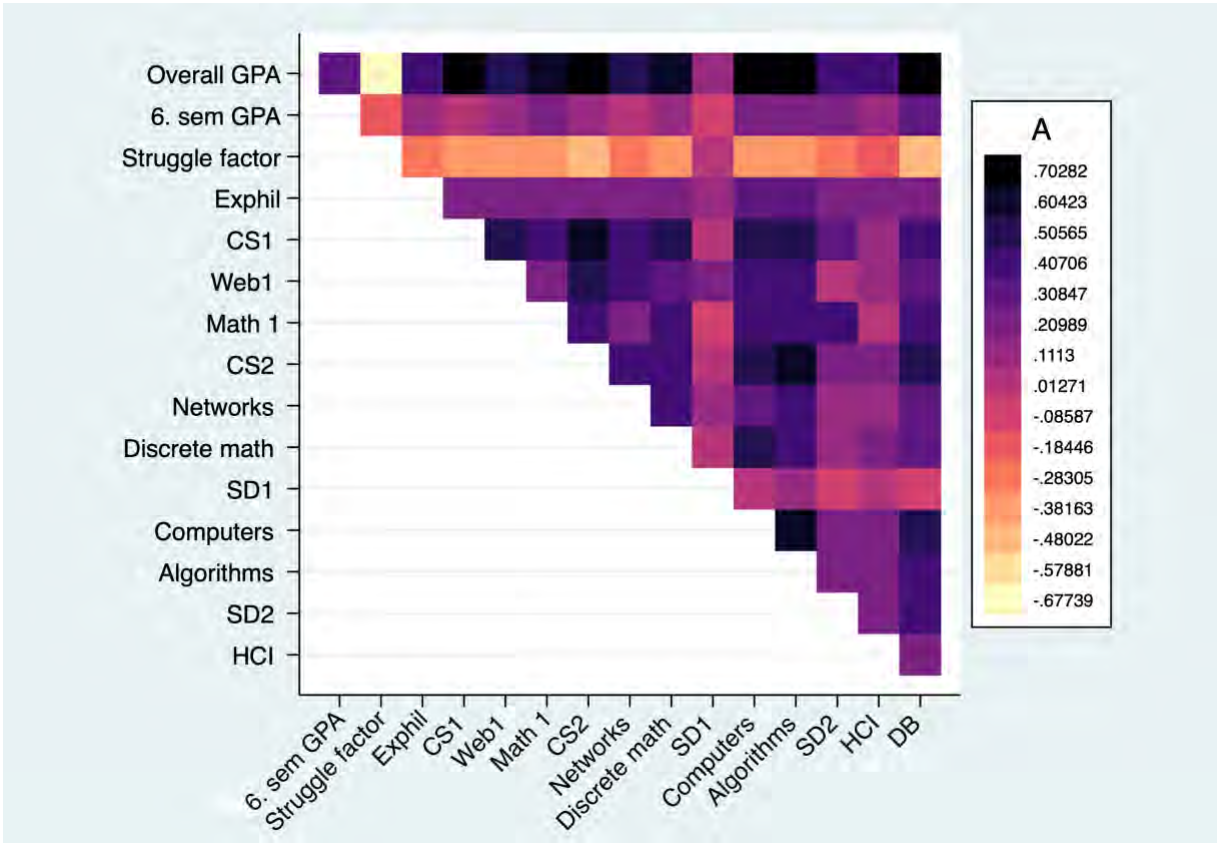
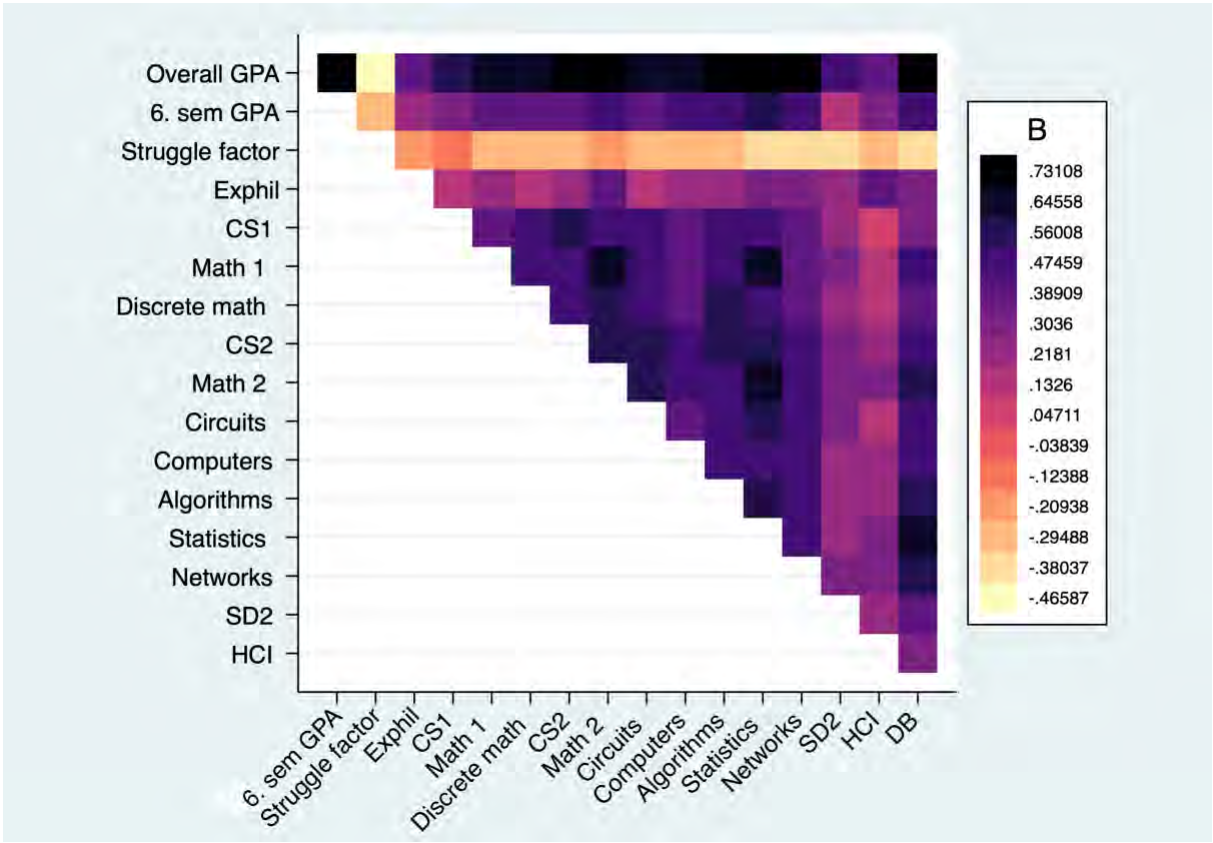


Figure 4: Heat map plot of the first two years for Program B. N=270



Struggle factor results

- There is a strong correlation between struggle factor and overall GPA, as indicated by the strong light color of the correlation box (second from the left, top row).
- The correlation to 6th semester GPA is not as strong as indicated by the less light correlation box below.
- The correlation to the different courses gradually becomes stronger as the students progress through the program as indicated by the gradually lighter correlation boxes (more distinct for program B).

Course types and sets results

Using the heat map to look at correlations between courses, we can learn about how various course types and sets relate or not. This is done by finding the intersecting correlation box for the two courses in question. In addition, one can investigate obvious outliers, that is areas which are lighter or darker than the surrounding areas. In the following, we have extracted the expected and not so expected correlations from this analysis.

Expected strong correlations:

- CS1 and CS2 have a high correlation.
- The different mathematics courses have a high correlation.
- Discrete mathematics and algorithms have a high correlation.

Notable weak correlations:

- Philosophy does not strongly relate to any other courses.
- SD1, SD2 and HCI do not strongly relate to other courses, GPAs or struggle factors. These can be seen as lighter columns/rows.

Discussion and related work

The overall research objective of this study was to investigate how general institutional data can be used to assess student performance and aspects of the educational design. We have explored variants of GPA, developed a potential new struggle factor and looked at how course types and sets relate, using exam data gathered from FS. In the following sections, we will discuss the application and implications of the results presented above, the methods we have used and the related research.

Using exam data to assess performance

To answer the research question of *what aspects of student performance can be evaluated using FS data*, we have examined overall GPA, 6th semester GPA, the struggle factor as well as course types and sets. Considering the two GPA variants, the overall GPA seems to be a better indicator than 6th semester GPA. This is based on the fact that overall GPA produces higher correlations to the courses and seemed to be more consistent. The use of GPA to indicate academic performance and success is common in educational research, although as the literature review done by York et al. found it does not always measure learning or growth in cognitive abilities [9]. In the current study, however, measuring performance or success with GPA is not directly looking at the summative learning growth. We have used GPA to indicate academic progression as a journey through a study program, where the actual grade in itself is not assessed, but the relation of a student's GPA to other grades. Previous work on performance in computing education research has also used GPA. In some cases to measure the effectiveness of certain educational approaches, teaching technologies or study behaviors [4, 5, 7]. In other cases, grades or GPA was used to differentiate students into high and low performers [4, 7], which was not the goal of our approach.

The struggle factor was a new approach to assessing student performance in a program with mixed results. To the authors' knowledge, similar approaches have not been reported on. On the one hand, the struggle factor seems to be a valid indicator of performance since the correlations to GPA and other courses is consistently high. On the other hand, it is difficult to interpret these results because the struggle factor is somewhat unprecise. It does not differentiate between retaking an exam because of a failed previous exam, tactical retakes and improving a grade. Nevertheless, the process of calculating and analyzing this variable provided some useful insights. In retrospect, the authors have discussed only including retakes based on a failed grade or calculating the time between the first attempt and the first passing grade. An additional aspect of the struggle factor is considering if it should be interpreted as a linear relationship. A possible perspective would be to square the number of attempts, hence emphasizing the students who have many retakes, which would, in fact, indicate struggle more. Lastly, this process sparked the idea of creating a struggle factor for courses as well as for students.

When it comes to course types and sets the heat map analysis provided some very interesting results. We would expect all courses to correlate to GPA as a program is designed with the intention of building the students' knowledge and skills over time. Therefore, the noticeable discrepancies found for philosophy, SD1, SD2 and HCI are striking. There are several possible explanations for these inconsistencies. In the case of philosophy, it is a course that does not "fit in" to a CS program, and it is therefore understandable if students treat that course differently and thus perform differently. In future work, it would be interesting to examine the performance of students in these *other* courses intended to broaden the knowledge of students. The results for SD1, SD2 and HCI, on the other hand, are harder to explain. One possibility is a divergence in content and assessment regime; in other words, a lack of constructive alignment [1]. Another possible explanation could be the structure of the course and assessment. We know that these courses are largely based on project and/or group work, in contrast to the other courses, which are based on individual assignments. Nevertheless, these explanations are based on tacit knowledge about the courses and programs. The key takeaway here is that this process of analyzing a program can highlight courses that need further investigation.

Using exam data from FS

A second aspect of the current research was *how FS data could be extracted and managed in an ethical and practical way*. We have found that by anonymizing the exam data from FS using a random identifier, the data can be analyzed in an ethical way. However, managing and analyzing the data in a practical and useful way provided more challenging. The fact that data goes back over many years is very useful, but many things change over time, which provided some challenges. Firstly, programs change over time and tracking these changes is hard. The changes can be due to course alterations, for instance, names or scope. An example of this is that there has been an introductory mathematics course in the first semester of both these programs for at least ten years. However, the name of this course has changed, as has the number of credits. Identifying and accounting for changes like this is time-consuming, but very important for such analysis'.

Limitations

The main challenge for this study is that the data set is so vast that the number of conceivable variables and methods of analyzation is very large. It is very possible that other tools would provide interesting results and answers to our research questions. Another limitation is that exam data in this form is an aggregated and summative indicator of very complex phenomena. Researchers should be cautious when drawing conclusions just based on the exam data. In our

case, the fact that both authors are involved in the study programs analyzed can be viewed as a strength because it informs the analysis.

Conclusion

This paper has summarized an attempt to use available exam data to find new ways to evaluate and assess study programs and student performance. We have looked at two variants of GPA and found that the overall GPA seems to be the better indicator of performance throughout a program. In addition, we have proposed looking at the relation between exam attempts and courses taken as a struggle factor. However, this variable might need some fine-tuning for future work. Lastly, we have found that examining course types and sets provides useful information about the design of a program.

In conclusion, the use of correlations visualized by heat maps was found to be very informative and we aim to further explore how this approach can be used in the future. An important motivation behind this work was to find useful tools for study program leaders and educational managers, tools that can inform decisions about study program design and enlighten possible inconsistencies between courses. What can we actually change and where are the possible rooms for actions are important future questions. At the student level, identifying and predicting challenging courses or transitions could help educators to implement impactful changes.

References

[1] Biggs, J. 1996. Enhancing teaching through constructive alignment. *Higher Education*. 32, 3 (Oct. 1996), 347–364. DOI:https://doi.org/10.1007/BF00138871.

[2] Cohen, J. 1988. *Statistical Power Analysis for the Behavioral Sciences (2nd Edition)*. Routledge.

[3] Cohen, L. et al. 2002. *Research Methods in Education*. Routledge.

[4] Edwards, S.H. et al. 2009. Comparing Effective and Ineffective Behaviors of Student Programmers. *Proceedings of the Fifth International Workshop on Computing Education Research Workshop* (New York, NY, USA, 2009), 3–14.

[5] Goold, A. and Rimmer, R. 2000. Factors Affecting Performance in First-year Computing. *SIGCSE Bull.* 32, 2 (Jun. 2000), 39–43. DOI:https://doi.org/10.1145/355354.355369.

[6] Johnson, B. and Christensen, L. 2012. *Educational Research: Quantitative, Qualitative, and Mixed Approaches*. SAGE.

[7] Liao, S.N. et al. 2019. Behaviors of Higher and Lower Performing Students in CS1. *Proceedings of the 2019 ACM Conference on Innovation and Technology in Computer Science Education* (New York, NY, USA, 2019), 196–202.

[8] Lorås, M. et al. 2018. First year computer science education in Norway. *Proceedings from the annual NOKOBIT conference 2018* (2018).

[9] York, T.T. et al. 2015. Defining and Measuring Academic Success. *Practical Assessment, Research & Evaluation*. 20, 5 (Mar. 2015).

VisAST: Generic AST Visualiser for Software Language Education

Ragnhild Aalvik*
Bekk Consulting, Norway

Jaakko Järvi†
University of Bergen, Norway

Abstract

Structural concepts such as *abstract syntax trees* (ASTs) are often best explained through visual representations. Students may, however, struggle with connecting such visual representations with the corresponding program text. To bridge this gap, we developed visAST, a tool for easily visualising ASTs of small languages written in Haskell. To assess the benefits and usability of visAST we conducted a user study in the context of students implementing interpreters. Students reported liking visAST and it being beneficial for learning. The experiment's results were not conclusive, but hint at visAST use improving students' performance.

1 Introduction

This paper presents *visAST*, a tool for visualising *abstract syntax trees* (ASTs) of programs. The purpose of visAST is to help students understand how ASTs and expression evaluation work. Its development was motivated by observing that students of programming language classes often struggle with how the textual representation of a program corresponds to its representation as a syntax tree.

When learning about interpreters, type checkers, and other language processors, students are often presented with ASTs as the internal representations of programs in visual form on the blackboard. When programming assignments on their own, however, they work with textual representations of these abstractions. Our conjecture is that a dynamic visual view of a program's AST helps the students see the connection between the textual and tree representation of a program, and help students learn essential aspects about interpreters and expression evaluation.

It would be unrealistic to expect that students, while learning about ASTs, interpreters, etc., simultaneously implement graphical views of their programs' internal representations. To be practical, such study aids must be at the student's disposal with minimal effort and complexity. VisAST adapts to any (Haskell) data type that represents ASTs, typically without students having to write any additional code for visualisation. To achieve this, we take advantage of Haskell's data type generic programming features.

We investigated the impact of visAST on students' learning and how students perceive its use in an experiment conducted as part of a class on Functional

*ragnhild.aalvik@bekk.no
†jaakko.jarvi@ii.uib.no

This paper was presented at the UDIT-2019 conference; see http://www.nik.no/.

Programming. Students' view of visAST was favorable, and the results of the experiment hint at visAST impacting students' performance positively.

2 Background and related work

Learning styles can be categorized to visual, auditory, and kinesthetic [1]. Visual learners remember best information that they can see, e.g., in pictures, animations, graphs, or films. Auditory learners remember best what they can hear, and they learn better from verbal, rather than visual explanations. Kinesthetic learners learn best through touch, taste, and smell.

Most students are visual learners [2, 3], yet most teaching is verbal. In a traditional lecture, the instructor talks to the class, often also writing textual content on the blackboard, which our brain processes similarly to spoken words [2]. Traditional teaching seems thus most beneficial for auditory learners.

Of course instructors also show visualisations in class, drawings or graphs on the blackboard, videos or animations, and some also using visualisation tools. Visualisation tools for teaching sorting algorithms, curiously, seem to be especially popular [4, 5, 6, 7]. This may be because sorting is somehow a concept that is easy to visualise or because as a topic it is a stable in the curriculum, justifying the effort of developing visualisation tools. There are also many tools for drawing graphs online [8, 9, 10], but such tools seem not to be widely used in teaching.

In the context of programming languages, popular languages have good tools for visualising program executions, but their use in teaching is not mainstream. One example tool is Online Python Tutor [11] (which is not limited to Python). It is essentially a debugger targeted for teaching and collaborative studying. Users can step forwards and backwards in an execution and observe the program state. Another example is the Ohm online editor [12] that can visualise expressions from a user-specified grammar. ExtendJ Explorer [13] lets users explore ASTs built by ExtendJ, relating the ASTs to the Java source code they are generated from.

Multiple studies report students in introductory programming courses liking visualisation tools. VIP, a visual interpreter for C++, was overall rated positively by students [14]. Feedback from students on VIPER, a tool for visualising Pascal code, was positive, particularly among beginners [15]. Daly [16] studied the opinions and impact of Alice, which visualises concepts like loops, objects, and recursion via 3D animations. She reported students liking using Alice and achieving better results and higher self-efficacy than students in a control group. Students with less prior programming experience found Alice significantly more useful than did students with more experience. Kaila et al. [17] investigated the use of the program visualisation tool ViLLE in an introductory programming course, and reported that novice programmers had the most positive views and largest increase in grades.

Ben-Bassat Levy et al. [18] reported a significant improvement in grades of students who used Jeliot, a tool for visualising Java code. In their study, average students benefited the most from the tool: better students did not need it and the weakest students were overwhelmed by it. This is an interesting contrast to our finding, reported in Section 4, that weaker students were the ones who used visAST the most and also perceived it the most useful. These differences might stem from the difficulty of the visualised concept and the usability of the visualisation tool.

Vegdahl [19] reported on using visualisations in a compiler course. Students could visualise various steps of the compilation process of Java programs, including

the AST. Students reported using the AST visualiser a lot, evaluating the tool from "very helpful" to "absolutely vital" for understanding and debugging their programs, and for gaining an intuition of how executable code relates to source code. This was also a motivation for our project, to help students relate their source code to the visual representations often presented in class.

Mernik and Zumer [20] investigated the impact of LISA that provides animated visualisations of different parts of a compiler. The students viewed the tool use positively; 76 % reported the tool as "important" in understanding compilers better.

Blythe et al. [21] developed LLparse and LRparse, two interactive tools for visualising examples of LL(1) and LR(1) parsing. Students of a programming course, with a focus on automata theory, used the tools for solving and error checking their assignments. Blythe et al. observed that students had fewer errors in their assignments after using the tools.

VAST is an educational tool for visualising *syntax trees* (STs), targeted for compiler or language processing classes. In an educational evaluation in a language processors class, the instructors observed that students were enthusiastic about VAST. Students perceived the tool as positive, easy to use, and that it supported them in their learning. [22]

Naser [23] compared students' exam results with the use of a visualisation tool in an AI searching algorithms course. Students who had used the tool performed better in the exams. Further, there was a strong correlation between students evaluating the visualisation tools as beneficial and their grade improvement in the exams.

Naps et al. [24] reviewed multiple experimental studies conducted on the effectiveness of visualisation tools. They divided the tools into *passive observation*, which includes observing different representations of the learning material through animations, and *active engagement* which includes learner involvement, e.g., by making the users construct their own data sets or program the algorithm to be visualised. This meta-analysis found that tools in the active engagement category were consistently more effective for learning than passive observation tools. Of the tools that required active engagement, 83 % produced significant positive results. Our visAST tool is clearly in the active engagement category: it visualises the evaluation steps of a program that a student writes.

In summary, past studies show that students like using visualisation tools, and that these tools can improve learning. Further, different types of students benefit differently from visualisations, but whether it is the stronger or weaker students that benefit the most varies.

3 visAST

VisAST is a web application, available at https://vis-ast.netlify.com. It can visualise ASTs of expressions generated by an arbitrary abstract grammar, and step through a sequence of ASTs, e.g., the evaluation steps produced by an interpreter.

VisAST offers two modes. The *Get familiar* mode visualises expressions of a simple, fixed, expression language. This mode is meant to make the tool easy to approach. It presents the user with a grammar and a field to type in expressions. VisAST parses the expression with a built-in parser, records all the evaluation steps of a built-in evaluator, draws the initial AST (see Figure 1), and lets the user step forwards and backwards through the steps. The interpreter implements small-step operational semantics, which makes the intermediate steps of expression evaluation

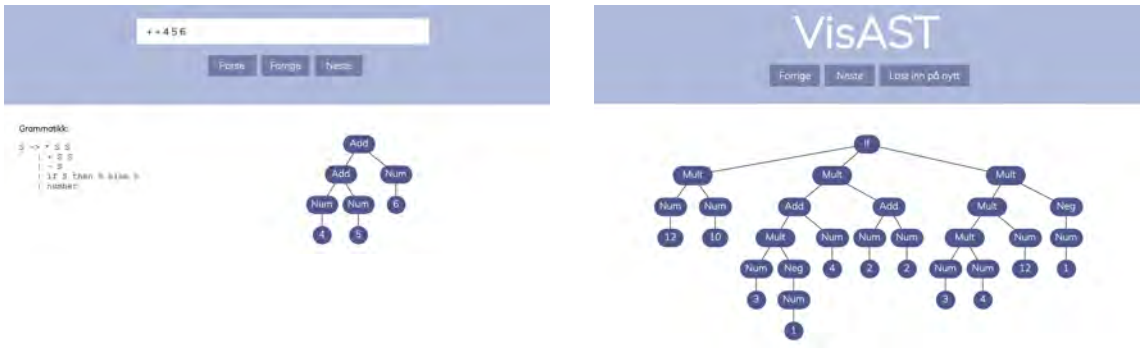


Figure 1: Two use modes of VisAST: *Get familiar* on the left shows the view of the initial AST after parsing an expression and *Advanced* on the right shows an AST saved and retrieved for a given lookup key.

explicit [25]. The *Advanced* mode lets the user visualise her own code. The user implements her own language, which means defining the abstract syntax and a small-step evaluator, and possibly a parser. VisAST's client library provides a function for sending the evaluation steps to the VisAST server, together with a lookup key that identifies the user. When the key is entered on the VisAST website, VisAST retrieves the ASTs saved for that key and visualises them as in the Get familiar mode. Figure 1 displays a stored and retrieved expression in the Advanced mode; no grammar is displayed because the grammar is defined in the user's source code.

The frontend of visAST is written in Elm and the backend server in Haskell. The server runs on Heroku and stores its data in a PostgreSQL database. The source code of the implementation is available as three GitHub repositories under https://github.com/aaalvik: visast-frontend and visast-backend are the frontend and backend implementations, visast-client-example implements a client that demonstrates how to use visAST in the Advanced mode.

The frontend speaks with the backend through a RESTful API. In the *Get familiar* mode, expressions typed in by the user are sent to the backend as a POST request, and parsed and evaluated on the server. As a response, the server sends back a list of ASTs. In the *Advanced* mode, the user code passes a list of ASTs to the visualise function defined in the VisAST client library. This function serializes the ASTs and sends them together with a user-defined key to be stored on the server, and opens the visAST web interface for the user.

Aalvik's thesis [26] gives a detailed description of the implementation and how a student uses visAST from her code. Here we show just enough code to demonstrate that visualising ASTs of a language that a student creates herself requires practically no programming. Listing 1 shows parts of an implementation of a simple language that a student might write. The data type Expr defines the abstract syntax. The parse function parses program text to ASTs of type Expr. The evaluate function then reduces the AST to a normal form, returning all reduction steps, as type [Expr]. These functions are written by the student, and not shown here. In order to visualise programs, nothing but a small amount of boilerplate is needed: 1. visAST client library must be imported (line 1); 2. the AST type must derive from Generic (line 3); 3. the AST type must be declared an instance of Generalise (line 4); and 4. parsed and evaluated ASTs must be sent to be visualised using the visualise function (line 5).

The user can invoke visualise with practically any AST types. Such generality is


```
import Client −− imports visualise, Generalise, etc.
data Expr = Num Int | Add Expr Expr | Mult Expr Expr | Neg Expr | If Expr Expr Expr
          deriving (Eq, Show, Generic)
instance Generalise Expr
main = visualise (evaluate (parse "(4 + 5) + 6"))
```

Listing 1: Using the visualise function to send ASTs to visAST.

attained with Haskell's Generics features that support writing algorithms over any type. The generics features serve for a similar purpose than Reflection in, say, Java. In particular, we inspect the user-defined AST data type to extract the names of each different AST node and the number of their arguments, and construct a general purpose AST representation that can be easily serialized and then visualised.

The trees are drawn by generating SVG, to be rendered by the browser. The generation is obtained with a top-down recursive traversal over the tree. For placing the nodes, visAST works hard to produce a visually pleasing outcome. The layout algorithm is described in Aalvik's thesis [26].

The reasons to implement the visualisation on the server, rather than entirely on the web client, was primarily to be able to collect data on how the students interact with visAST, but also to make the adoption as easy as possible and without the need to install additional software. VisAST records every request to the server. Concretely, it logs an event when a user parses an expression in the *Get familiar* mode, requests the ASTs of a specific lookup key in the Advanced mode, or invokes visualise. We could in principle also track when the user flips back and forth through the reduction steps, but we did not do that in our experiments.

4 Methods and results

In order to assess the usefulness of visAST, we conducted a user study. The subjects were students in the third semester undergraduate class on Functional Programming (INF122) at University of Bergen (UiB). The study took place in the context of an obligatory assignment, in which students implemented a simple expression language. They were asked to use visAST to visualise ASTs of their language, and later to fill out a questionnaire about how they perceived visAST.

We also wanted to measure whether the use of visAST had an impact on students' performance. Due to time constraints and the need to treat all students equally (the course is compulsory for most of the students), a controlled study was not possible—all students received the same assignment and the same instructions on using visAST. We can, however, draw some conclusions by comparing the students' answers in the questionnaire and their performance in the assignment.

The assignment

The focus of the assignment was parsing and evaluating expressions, which is one notable component of INF122. Students were given a grammar of a simple expression language consisting of arithmetic expressions formed with `*`, `+`, and unary $-$, along with if then else-expressions:

$$\langle S \rangle ::= \text{`}\texttt{*}\text{'}\ \langle S \rangle\ \langle S \rangle \mid \text{`}\texttt{+}\text{'}\ \langle S \rangle\ \langle S \rangle \mid \text{`}\texttt{-}\text{'}\ \langle S \rangle \mid \text{`}\texttt{if}\text{'}\ \langle S \rangle\ \text{`}\texttt{then}\text{'}\ \langle S \rangle\ \text{`}\texttt{else}\text{'}\ \langle S \rangle \mid \langle digit \rangle^{+}$$

Based on this language description the students were asked to (1.1) implement a parser (program text → AST), (1.2) pretty-printer (AST → program text), and (1.3) small-step evaluator for the language; (1.4) write a function that can run user commands, such as running the pretty printer; (2.1) implement the command for visualising ASTs using visAST; (2.2) fill out a questionnaire about visAST; and (3.1) a task unrelated to visAST. Before starting, students were instructed to download the visAST client and check that it worked on their machine.

As a voluntary subtask of 1.1, students were asked to test their parser using visAST, concretely to call `visualise` with their parsed AST. In Task 1.3, students could use visAST to show their interpreter's intermediate ASTs. Task 1.4 was to create the command that launched the visualisation, essentially to write the code that passes the list of ASTs to `visualise`. The questionnaire in Task 2.2 was given out as an online form; the questions are described below.

Participants and procedures

The subjects were all students in the INF122 class. There were originally 214 students signed up, 93 turned in the assignment, and 79 answered the questionnaire. 48 students answered Question 8 in the questionnaire; this question asked for free-form feedback. The questionnaire was part of the mandatory assignment, but students were informed that it had no effect on the assignment grade. The assignment was to be completed individually and handed in in two weeks' time; some collaboration was allowed, except for in the questionnaire.

Standard curricula at UiB do not include learning about parsing and interpreters prior to INF122. Most students thus knew little about these topics before this course, but during the course they had already been discussed to an extent. None of the participants had used visAST before this experiment.

The questionnaire was in Norwegian. Translated to English, the questions were:

- Q2 How well did you understand ASTs before this assignment? (1=understood little, 5=good understanding)
- Q3 How much did you use visAST in this assignment? (1=as little as possible, 5=a lot)
- Q4 How useful was visAST for your understanding of ASTs in this assignment? (1=not useful, 5=very useful)
- Q5 How useful do you think visAST would have been for your understanding the first time you were introduced to ASTs? (1=not useful, 5=very useful)
- Q6 Did visAST help you understand parsing better? (yes/no)
- Q7 Did visAST help you understand evaluation of expressions better? (yes/no)
- Q8 We would like your thoughts about visAST, in your own words. What was useful/not useful? What was easy/not easy? Any suggestions for improvements? Other thoughts?

Q1 was for identifying information we needed for aligning students' answers, visAST usage logs, and assignment and exam grades.

Research questions

To understand benefits of visualisation tools for learning about language processors, we were interested in how students evaluate their knowledge of ASTs and related topics, and their experiences with visAST. We formulated two research questions for the experiment: 1. *Do students find visAST helpful in understanding ASTs*

better? 2. *Do students find visAST helpful in understanding parsing and expression evaluation better?*

Q2, Q3, and Q4 deal with the students' previous understanding and how much they used the tool in the experiment. These are all relevant for answering the first research question. Q2, Q3, Q6, and Q7 are relevant for answering the second research question. Q6 and Q7 address it directly, and Q3 is also relevant for it has to do with how much the subjects used the tool. Q2 is relevant because an understanding of ASTs is a prerequisite to understand parsing and evaluation.

Q5 was included because of the timing of the experiment: students had already learned some about ASTs in the class, possibly rendering visAST less useful.

The purpose of Question 8 was to get feedback for improving visAST.

Results

We analysed the results first for all students, then divided into three groups based on their final exam score, graded from 0 to 105 points, as follows: *Stronger students* (80 points or more), *Average students* (50–80 points), and *Weaker students* (less than 50 points). The stronger group had 24 students, average 30, and weaker 15, amounting to 69 students in total. We excluded from the groups ten students who did not take the final exam. The reason for the grouping was to investigate how different types of students experienced visAST and how much they benefited from it. As discussed in Section 2, many studies show that visualisation tools benefit different types of students differently.

Questions Q2 through Q5 had an ordinal scale with labels on the lower (1) and upper (5) bound of the scale. To compute averages of this ordinal scale, we converted it to its numerical equivalent. In general, one should be careful when converting an ordinal scale to numerical [27]. Our data is clearly single peaked and we have no basis for any other assumption than that the data is approximately normally distributed. Under these assumptions, the risk of misanalysis is low [27].

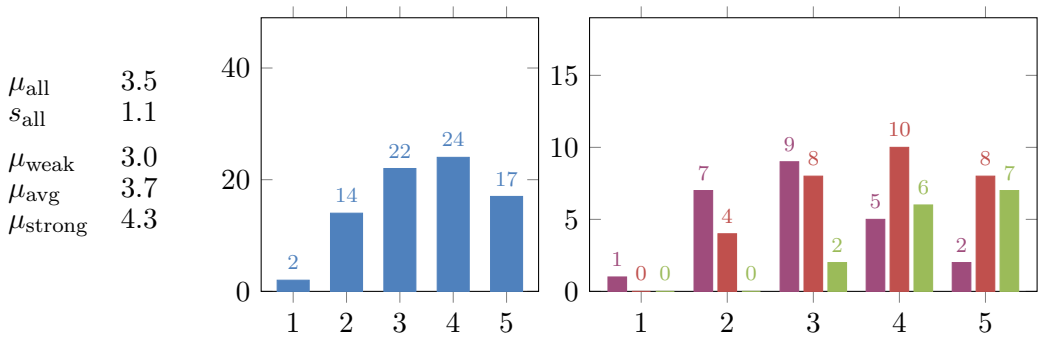
Figure 2 shows the distributions of the answers to the questions Q2–Q5, jointly for all students and separately for each group, as bar charts. It also lists the averages in the numeric scale and standard deviation (for the "all students" case).

The mean of 3.5 of answers to Q2 with standard deviation 1.1 suggests a varying but moderately good understanding of ASTs prior to the assignment. There were sixteen students who answered that they had very little or little understanding, all of which belonged in the weaker or average group. Even the weaker group, however, had a mean of 3.0, indicating a moderately good understanding on average.

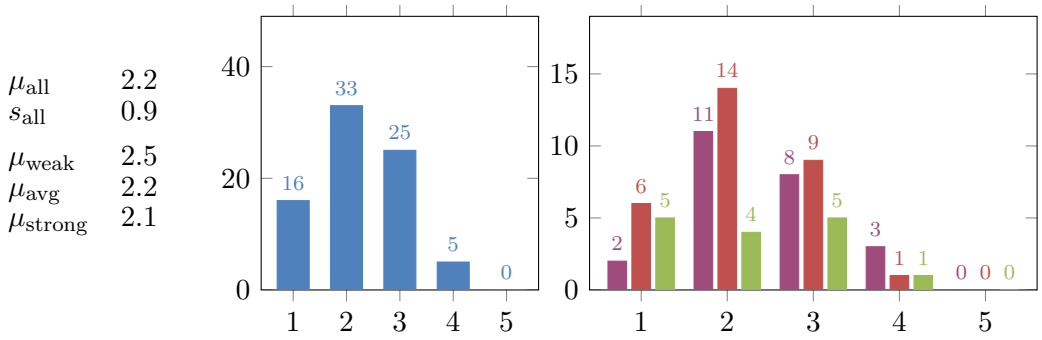
In Q3, no student reported using visAST "a lot" in the assignment, but that a substantial number of students reported moderate or more use. The weaker group used the tool somewhat more than the average and stronger groups.

In Q4, a majority (58%) of all students found visAST moderately or more useful for learning about ASTs, confirming our first research question above. The weaker students found visAST the most helpful—not a single student in this group answered "not useful". On average the stronger group found the tool the least useful, but there were many in that group too who found visAST either "useful" or "very useful".

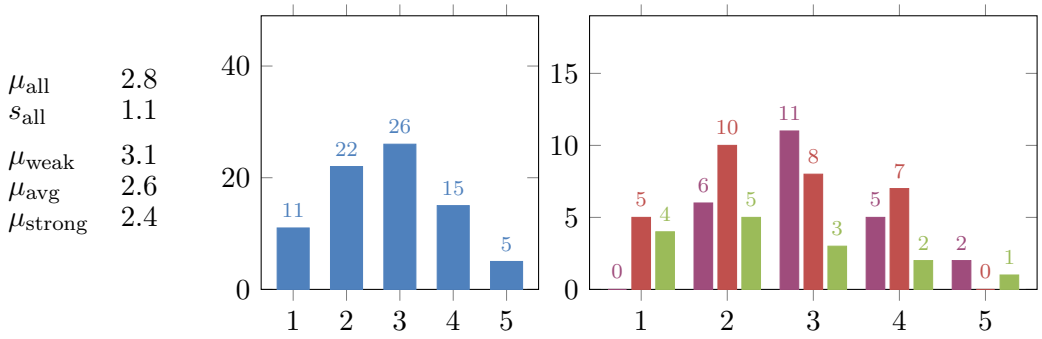
Answers to Q5 confirm our suspicion that we introduced visAST too late in the topic sequence of the class. All groups overwhelmingly agreed with this: 54.4% of the the students answered that visAST would have been "very useful" if introduced when ASTs were first introduced, and no students answered "not useful".



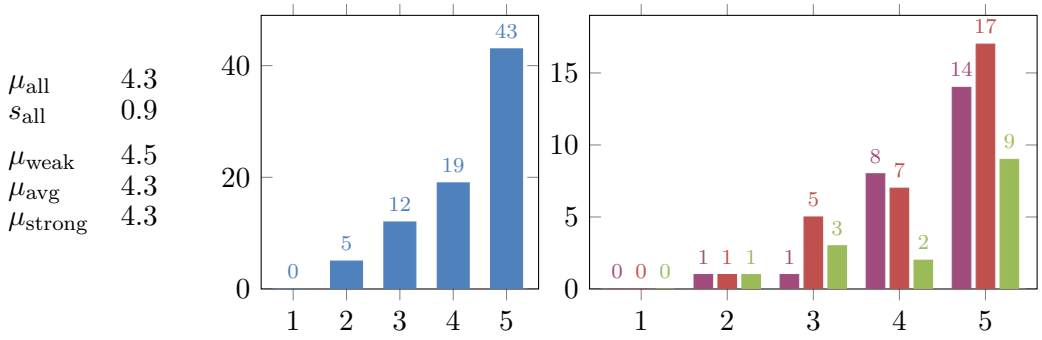
Q2: How well did you understand ASTs before this assignment? 1=understood little, 5=good understanding.



Q3: How much did you use visAST in this assignment? 1=as little as possible, 5=a lot.



Q4: How useful was visAST for your understanding of ASTs in this assignment? 1=not useful, 5=very useful.



Q5: How useful do you think visAST would have been for your understanding the first time you were introduced to ASTs? 1=not useful, 5=very useful.

Figure 2: Questions Q2–Q5. The leftmost charts show the distribution for all students and the rightmost for each group according to weak average strong. The x-axis is the number of answers, y-axis their ordinal scale. The columns on the left list the averages μ, and the standard deviation s for all students.

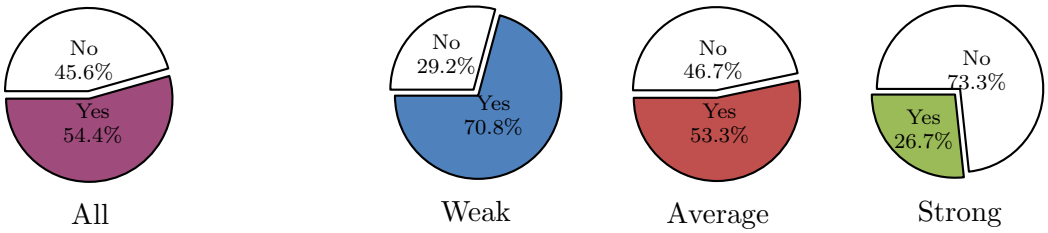


Figure 3: Q6: Did visAST help you understand parsing better?

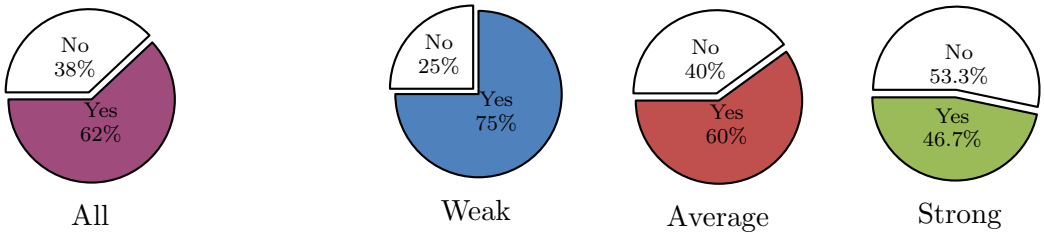


Figure 4: Q7: Did visAST help you understand evaluation of expressions better?

Figure 3 shows that a majority thought that visAST helped them understand parsing better, which answers part of our second research question above. However, quite a few students did not agree. The difference between groups was rather stark: only 27 % of the stronger students agreed, compared with 71% of the weaker ones.

Figure 4 shows that students found visAST even more helpful for understanding expression evaluation: a larger majority, 62 %, agreed with Q7, which answers the second half of our second research question. Again, of the three groups the weaker students found visAST most helpful, with 75 % agreeing, but almost half (47 %) of the stronger students agreed too.

Of the answers to the free-form question Q8, we identified four common themes. First, several students (28) commented on the timing of when the tool was introduced, reiterating that it would have been more useful earlier in the class. Second, students commented on the tool's usability positively, finding the design simple and vsualisations neat. Three students mentioned that visualising their own code helped them verify its correctness. Third, several students had good suggestions for future features or improvements for visAST, such as animating transitions between the trees to help focus on what changed in a reduction step. Fourth, there were a handful of negative responses, practically all about the installation of external Haskell libraries that visAST's Advanced mode requires.

In addition to gathering students' thoughts on visAST, we wanted to measure whether the use of visAST had an impact on students' learning. Concretely, we formulated an additional hypothesis H_a: *the use of visAST improved students' performance in assignment A2.*

As discussed in Section 4, we had no control group—all students were instructed to use visAST. We therefore used the students' performance in other assignments/exams as a reference point, as the expected performance in assignment A2. Prior to any comparison, we normalised the scores to standard normal distribution. We compared the score in A2, in which visAST was used, to this reference point and correlated the difference in scores to how much the students used visAST in A2. A positive correlation of the difference with how much visAST

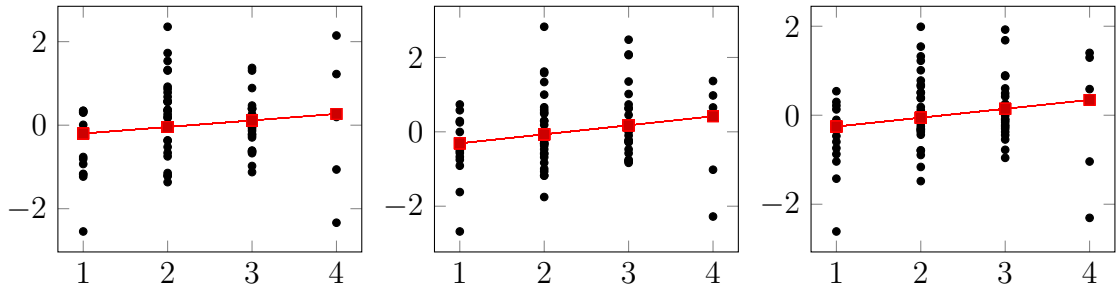


Figure 5: The difference of A2 normalized score and the reference score, A2-A1 on the left, A2-Exam in the middle, and A2-Rest on the right. The x-axis in all charts is from 1–4, since no student answered 5. The red lines show the linear regressions.

	Estimate	Std. error	t-value	$\Pr(>\|t\|)$
Δ_{A2-A1} (intercept)	-0.3566	0.3262	-1.093	0.278
visAST-use (slope)	0.1567	0.1343	1.167	0.247
$\Delta_{A2-Exam}$ (intercept)	-0.5520	0.3530	-1.564	0.1226
visAST-use (slope)	0.2426	0.1454	1.669	0.0998
$\Delta_{A2-Rest}$ (intercept)	-0.4543	0.3018	-1.505	0.1370
visAST-use (slope)	0.1997	0.1243	1.607	0.1130

Figure 6: Regressions between the difference in score and the use of visAST.

was used supports our hypothesis. We performed this analysis three times: A2 against the assignment A1, A2 against the exam, and A2 against the combined score (using equal weights) of A1 and the exam; we call these cases *A2-A1*, *A2-Exam*, and *A2-Rest*. Assignment A1 preceded A2, so visAST use did not effect A1 scores. The exam came after A2 and thus its score could in theory have been impacted by visAST use. No exam question, however, was directly related to ASTs, parsing, or evaluation and we thus expect any effect to be negligible.

For the A2-A1 comparison we discarded three students, A2-Exam four, and A2-Rest seven students who did not submit A1 or take the Exam. Figure 5 shows the difference in the normalised score between A2 and, respectively, A1, Exam, and the combined A1/Exam score, compared to how much the student used visAST in A2.

A summary of the regression analyses is shown in Figure 6. In the A2-A1 case, correlation with the score difference and visAST use was weak, 0.1411, and the p-values for the coefficients high, 0.278 and 0.247: we cannot reject the null hypothesis of no correlation. In the A2-Exam case, correlation was 0.1998; generally 0.2 is considered a threshold between weak and moderate positive correlation. The p-values for the coefficients (0.1226 and 0.0998) were smaller but do not indicate statistical significance. In the A2-Rest case, correlation was 0.1926 which is slightly lower than the A2-Exam case. The p-values were 0.1370 and 0.1130, which do not indicate statistical significance either. We can at best interpret the results as hinting towards that more use of visAST correlates with a higher A2 score.

5 Discussion and conclusion

Results from the questionnaire showed that students liked to use visAST as a learning tool. Part of the positive attitude may stem from the good usability of

visAST; we put a lot of effort to node layout for a visually pleasing outcome, and for making it possible to create visualisations with ease, directly from the students' program code. As discussed in Section 2, several other studies have reported students to have positive views toward using visualisation tools.

A notable characteristic of VisAST is that it can visualise ASTs of any language that a student implements, without requiring the student to write (practically) any code for the visualisation. Though we omitted many details of the implementation, it bears mentioning that Haskell's data-type generic features were key in our solution. We suspect that with some effort similar ease of use could be achieved in other languages, e.g., in Java with the help of the reflection features or in JavaScript by inspecting objects' properties dynamically.

Besides usability, students evaluated visAST's usefulness for learning. The majority considered visAST to be useful for understanding ASTs, parsing, and expression evaluation. These views were most dominant in the weaker student group; it was also this group that reported the highest use of visAST.

The regression analyses we performed hint towards that visAST can improve learning results, but we say this with reservations. More accurate results could be obtained with a better experiment design. As we had no control group, we used students' performance in other tasks as controls, which we expected to be noisy. Ideally, we would also control for the time invested on the assignment as assignment grades can correlate with effort, not only with knowledge, and more use of a tool may also mean more time invested on an assignment.

In sum, our work adds to the evidence that active engagement visualisation tools, which visAST is, are beneficial for learning programming concepts.

References

[1] Walter Burke Barbe, Michael N Milone, and Raymond H Swassing. *Teaching through modality strengths: Concepts and practices*. Zaner-Bloser, 1988.

[2] R. J. Kapadia. Teaching and learning styles in engineering education. In *38th Annual Frontiers in Education Conference*, pages T4B–1–T4B–4, Oct 2008.

[3] G. Gray and D. Duffin. SIF: Learning styles theme final report: Project report 2007-2009. Unpublished report, obtained by request from author, 2011.

[4] Steven Halim. Visualgo–visualising data structures and algorithms through animation. *Olympiads in Informatics*, page 243, 2015.

[5] Kanasz Robert. Visualization and comparison of sorting algorithms in C#. www.codeproject.com/Articles/132757/Visualization-and-Comparison-of-sorting-algorithms, December 2010. Accessed: 2019-05-02.

[6] Callum Macrae. Sorting algorithms visualised. http://macr.ae/article/sorting-algorithms.html, December 2016. Accessed: 2019-05-02.

[7] Carlo Zapponi. Sorting. http://sorting.at/, 2014. Accessed: 2019-05-02.

[8] Graph editor. https://csacademy.com/app/graph_editor/. Accessed: 2019-05-02.

[9] Graph online. https://graphonline.ru/en/, 2015. Accessed: 2019-05-02.

[10] Graphtea. http://www.graphtheorysoftware.com/. Accessed: 2019-05-02.

[11] Philip J. Guo. Online Python Tutor: Embeddable Web-based program visualization for CS education. In *Proceedings of the 44th ACM Technical Symposium on Computer Science Education*, SIGCSE '13, pages 579–584, New York, NY, USA, 2013. ACM.

[12] Patrick Dubroy, Saketh Kasibatla, Meixian Li, Marko Röder, and Alex Warth. Language hacking in a live programming environment. In *LIVE 2016 at ECOOP 2016*, USA, 2016. https://ohmlang.github.io/pubs/live2016/.

[13] Torbjörn Ekman. ExtendJ Exploring Tool. https://extendj.org/visualizer.html. Accessed: 2019-03-12.

[14] A.T. Virtanen, E. Lahtinen, and H.-M. Järvinen. VIP, a visual interpreter for learning introductory programming with C++. In *Kolin Kolistelut—Koli Calling 2005 Conference on Computer Science Education*, pages 125–130, 2005.

[15] Michał Adamaszek, Piotr Chrzastowski-Wachtel, and Anna Niewiarowska. VIPER, a student-friendly visual interpreter of pascal. In Roland T. Mittermeir and Maciej M. Sysło, editors, *Informatics Education—Supporting Computational Thinking*, pages 192–203. Springer Berlin Heidelberg, 2008.

[16] Tebring Daly. *Influence of Alice 3: Reducing the Hurdles to Success in a Cs1 Programming Course*. CreateSpace Independent Publishing Platform, 2013.

[17] E. Kaila, T. Rajala, M.-J. Laakso, and T. Salakoski. Effects, experiences and feedback from studies of a program visualization tool. *Informatics in Education*, 8:17–34, 2009.

[18] Ronit Ben-Bassat Levy, Mordechai Ben-Ari, and Pekka Uronen. The Jeliot 2000 program animation system. *Computers & Education*, 40:1–15, 01 2003.

[19] Steven R. Vegdahl. Using visualization tools to teach compiler design. In *Proceedings of the Fourteenth Annual Consortium on Small Colleges Southeastern Conference*, CCSC '00, pages 72–83, USA, 2000. Consortium for Computing Sciences in Colleges.

[20] M. Mernik and V. Zumer. An educational tool for teaching compiler construction. *IEEE Transactions on Education*, 46(1):61–68, Feb 2003.

[21] Stephen A. Blythe, Michael C. James, and Susan H. Rodger. LLparse and LRparse: Visual and interactive tools for parsing. *SIGCSE Bull.*, 26(1):208–212, March 1994.

[22] F. J. Almeida-Martinez and J. Urquiza-Fuentes. Syntax trees visualization in language processing courses. In *2009 Ninth IEEE International Conference on Advanced Learning Technologies*, pages 597–601, July 2009.

[23] Samy S. Abu Naser. Developing visualization tool for teaching AI searching algorithms. *Information Technology Journal*, 7(2):350–355, 2008.

[24] Thomas L. Naps, Guido Rößling, Vicki Almstrum, Wanda Dann, Rudolf Fleischer, Chris Hundhausen, Ari Korhonen, Lauri Malmi, Myles McNally, Susan Rodger, and J. Ángel Velázquez-Iturbide. Exploring the role of visualization and engagement in computer science education. *SIGCSE Bull.*, 35(2):131–152, jun 2002.

[25] Gordon D. Plotkin. A structural approach to operational semantics. *J. Log. Algebr. Program.*, 60-61:17–139, 2004.

[26] Ragnhild Aalvik. VisAST: Generic AST visualiser for software language education. Master's thesis, The University of Bergen, 2019. http://hdl.handle.net/1956/19781.

[27] Barbara A. Kitchenham and Shari L. Pfleeger. Personal opinion surveys. In *Guide to Advanced Empirical Software Engineering*, chapter 3, page 90. Springer-Verlag London, London, 2008.

Mandatory coursework in higher Norwegian IT education

Lauvås jr, Per
Kristiania University College
School of Economics, Innovation, and Technology

Sandnes, Tomas
Kristiania University College
School of Economics, Innovation, and Technology

Abstract

Obligatory exercise, mandatory activity and work requirement are all examples of terms describing the same phenomenon in higher Norwegian education: Something a student needs to pass in order to get access to an exam. In this paper we call them mandatory coursework in alignment with relevant existing research. Some argue that mandatory coursework assignments can, and should, be eliminated. Before we can discuss this within Norwegian IT education, we need to know to what extent mandatory coursework is in use.

A course description should describe any mandatory coursework within a course. In this paper we present extracted data from course descriptions from 12 institutions delivering IT education in Norway. The data tells us the frequency in which mandatory coursework is in use, the different types used, how many there are, in what stages within a study programme they are most commonly in use and the variation between the 12 institutions.

The results tell us that mandatory coursework to a large extent is in use in Norwegian IT education, although there are significant variations among the different institutions. The most common coursework are labs, assignments and submissions, but participation is also quite common. Mandatory coursework is in use in both bachelor and master programmes with year one in a study as the most coursework intensive.

This paper was presented at the NIK-2019 conference; see http://www.nik.no/.

1 Introduction

Educators may find students procrastinating student work [1] when a final exam is months away. In order to *force* the students to work continuously through the semester, a possible solution may be to introduce a steady flow of exercises or tasks that a student should deliver. The exercises may be assessed, and if so, using formative or summative assessment. Multiple definitions of formative and summative assessment exist. A distinction between the two can be described with the primary purpose of the assessment [2]. For summative assessment, the primary purpose is assessment *of* learning. Formative assessment is assessment *for* learning.

The assessment of mandatory coursework is an intriguing hybrid between summative and formative assessment. It is summative as the work must be assessed to passed in order for the student to get access to the final exam. It may be formative as the student can receive feedback on her work. The feedback may be as simple as the outcome of the assessment (pass or fail), but it may also be accompanied with a more thorough feedback helping the student in her learning process towards the final exam.

Norwegian authorities do not encourage or discourage the use of mandatory coursework in higher education. It is left to the educational organizations to handle access to the exams. Universities and university colleges act, section 3.9, part 7 (translated into English) [7]:

> "*The board itself provides regulations on the preparation and conduct of examinations and tests, including conditions for taking the exam (...)*"

We therefore find the definitions of mandatory coursework in the organizations' regulations. The terminology for the mandatory coursework itself varies, such as "Mandatory activities", "Mandatory teaching activities", "Work requirements" and "Mandatory requirements". But the outcome is the same for all definitions: it is something that must be passed in order for the student to get access to the exam. In this paper, we use the terminology "Mandatory coursework" in alignment with previous research on the subject.

2 Background

Haugan et al. explicitly argue that "Mandatory coursework assignments can be, and should be, eliminated!". In their case study [5,6], they describe how mandatory coursework assignments were replaced by formative assessment in five courses within an engineering bachelor degree programme. A total of 28 mandatory coursework assignments were removed and replaced with voluntary peer-assessment sessions. The findings in the case study are promising. The students performed very well on the examinations. Perhaps more surprisingly: The average number of study hours increased in all the courses.

The authors describe an *engineering* education where mandatory coursework assignments are heavily in use, at least at their own institution. Engineering and IT are both STEM disciplines and share some common ground. If we want to discuss whether mandatory coursework should be eliminated from IT education or not, we first need to know to what extent mandatory coursework is in use. We have, to the best of our knowledge, not found any study describing the use of mandatory coursework in IT education. This leads us the following four research questions:

- **RQ1**: To what extent is mandatory coursework in use in Norwegian IT education?
- **RQ2**: What types of mandatory coursework is in use in Norwegian IT education?
- **RQ3**: Does the use of mandatory coursework differ between the organizations delivering IT education in Norway?
- **RQ4**: Are there certain stages in an IT education where mandatory coursework is more common than others?

3 Method

When mandatory coursework is used in a course, it should be described in the course description. An initial attempt was made to harvest information regarding mandatory coursework through a web crawler. The task turned out to be more difficult than expected, so we decided to perform manual data extraction from publicly available course descriptions. We used studiebarometeret.no as a starting point for limiting which institutions to include. Using filter with subject group: "Information and computer technology" resulted in 96 study programmes delivered by 12 different institutions. The initial plan was to extract data from within these 96 study programmes, but we discovered that information in Studiebarometeret is not completely up to date, and also includes historical data. As an example, Nord University is in Studiebarometeret listed as one of the 12 institutions because they are listed as delivering the bachelor: "Games and Experience Technology" (G&ET). Visiting the Nord University web site, we find that the programme is no longer accepting students. But they are accepting students to a bachelor within information technology and computer science called "Digital economy and organization"(DE&O).

We defined the institutional scope to be the 12 institutions providing "Information and computer technology" education, as defined by Studiebarometeret, but used the institution's own web sites to find IT study programmes offered to new students starting fall 2019. Following the example from Nord University, DE&O is kept as part of the scope, but G&ET is not as it is no longer accepting students. We further limited the scope to only include full time studies at a Bachelor or Master level. Some institutions provide filtering where IT is included within a broader category. In that case, we performed a further subjective filtering to find what we considered to be IT programmes.

Table 1 displays the institutional scope and the programme filter used at the institution web sites.

Institution	Abbreviation	Programme filter
Norwegian University of Science and Technology	NTNU	Information Technology and Informatics
The Arctic University of Norway	UiT	Engineering, natural sciences, vocational and technical subjects[1]
Western Norway University of Applied Sciences	HVL	Engineering and maritime studies[1]
University of Bergen	UiB	Technology or engineering[1]
University of Agder	UiA	Computer Science
University of Stavanger	UiS	Engineer and civil engineer[1]
University of Oslo	UiO	Information technology and informatics
Kristiania University College	HK	Institute of technology
Østfold University College	Hiof	Information technology
Oslo Metropolitan University	OsloMet	Institute of information technology
Nord University	Nord	Information technology and informatics
University of South-Eastern Norway	USN	Engineering, Technology and IT[1]

Table 1: Scope within Norwegian educational institutions

We applied the following rules when extracting data from a study program:

- Exclude courses where the course description is not publicly available.
- If a programme has specializations, extract all IT specializations.
- If a course description is missing for an upcoming course, look for earlier versions of the same course (based on course code) and extract data if found.
- Extract all mandatory courses in the programme. If a student may choose courses from a list of *recommended* courses, extract all recommended courses.
- If a course is present in more than one study program, extract it only once.
- If a course may be completed at several stages within a programme (i.e. 2. or 4. semester) use average when extracting semester number information. If a course is completed in different stages in different programmes, use only the first value found.
- Extract all mandatory activities found, even though they may not be explicitly described as a requirement for exam access.
- If a mandatory coursework describes a repeating activity (i.e. a mandatory delivery to each lecture), use a qualified guess to count the number of mandatory activities.

[1]Further subjective filtering was performed due to broad categorization at the institution.

- If the course description describes a use of mandatory coursework, but no further information may be retrieved, look for a publicly available course site. If further information is still not found set the coursework count to *unknown, but existing* (as opposed to 0).
- If *one* mandatory coursework consists of a specific number of separate activities (i.e the coursework is three separate deliveries), count the number of separate activities.
- If a mandatory coursework consists of an unspecified number of tasks (i.e obligatory assignments must be approved), count it as one coursework.
- Exclude courses that have been replaced by a new course. Extract the new course if found.
- If a course description describes a maximum number of mandatory coursework, extract the maximum.
- If a course description describes activities that may or may not be mandatory, exclude the course.
- If no information regarding mandatory coursework can be found, set the number of mandatory coursework to 0.

4 Findings

We defined 45 Bachelor and 37 Master programmes in 12 institutions to be within our scope. Table 2 displays the scope and the number of programmes and courses where we have extracted data. We extracted mandatory coursework data from a total of 668 course descriptions within the scope.

	Bachelor programmes			Master programmes		
Institution	F	E	C	F	E	C
NTNU	8	2	39	5	2	36
UiT	2	1	20	3	1	18
HVL	2	1	25	1	1	6
UiB	9	1	19	4	1	9
UiA	3	1	26	4	1	12
UiS	1	1	19	2	1	12
UiO	7	2	30	11	4	51
HK	1	1	53	4	4	23
Hiof	4	4	43	1	1	9
OsloMet	3	3	43	1	1	26
Nord	1	1	25	0	0	0
USN	4	4	108	1	1	16
Total	45	22	450	37	18	218

Table 2: Course description data extraction by institution. F = Found programmes, E = Extracted programmes, C = Course descriptions extracted.

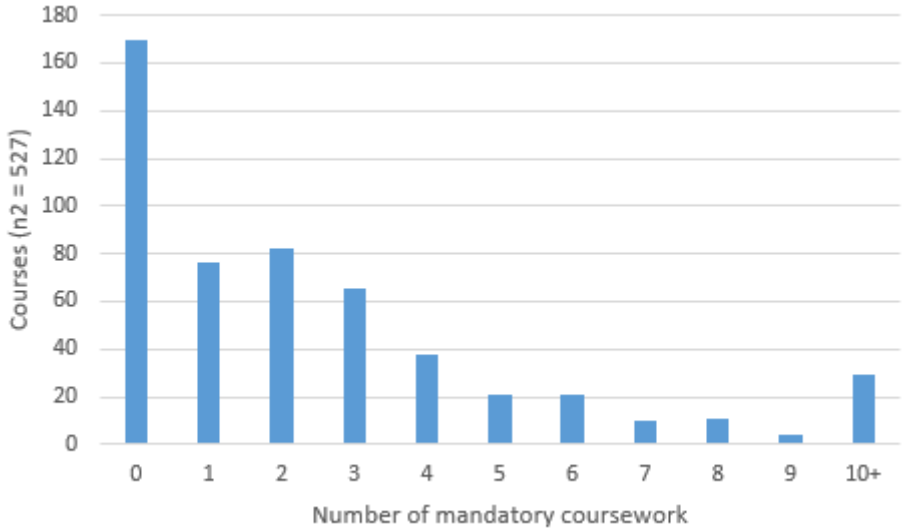
RQ1: To what extent is mandatory coursework in use in Norwegian IT education?

The course descriptions found at the different institutions all followed a pattern with common traits. They provide a short summary of the course and describe learning goals, structure, prerequisites, teaching methods, how the students are evaluated, mandatory coursework, etc.. Although mandatory coursework is present within the course description template, the level of detail of how they are described differs significantly. Some only provide a generic description of whether or not mandatory coursework is in use in the specific course. We therefore provide two types of answers to the research question. First we present an overview of courses with or without mandatory coursework. Secondly we provide data on the amount of mandatory coursework within the courses where the number of mandatory coursework could be retrieved from the course description.

From our pool of extracted course descriptions (n1=668), 75% of the courses use mandatory coursework.

We were able to retrieve the mandatory coursework count within a course from 527 courses. These were found directly in the course description or by looking at the course site (as described in chapter 3:Method). We found the mandatory coursework count to be 1362 within these 527 courses. Figure 1 displays the distribution of coursework in these courses.

Figure 1: Mandatory coursework count per course



From our pool of courses where the number of coursework could be determined (n2=527), the average number of coursework were 2,6.

The largest amount of coursework we found in a single course was 22: Obligatory attendance at the first lecture, 15 obligatory lab sessions and six submissions.

Counting coursework is not always easy. Some courses use specific rules on what needs to be achieved in order to take the exam. Here is an example from UiO:

> "*The course includes the following mandatory elements:*
>
> - *4 compulsory submissions*
> - *Lab work*

- *Computational essay with presentation*

Participation in at least 75% of the group sessions exempts one compulsory submission.

Participation in both introductory and final mapping exams exempt one compulsory submission."

Rules, such as the one above, and a variable level of detail in the course descriptions make it hard to give a precise number of coursework for each course. The numbers we present are our best effort when applying the rules described in chapter three: Method.

RQ2: What types of mandatory coursework is in use in Norwegian IT education?

When describing coursework type information, we use the same pool of courses (n2=527) as for the previous research question where details of the coursework could be found. As there is a difference in the level of detail for coursework descriptions, some types will be more general than others. After first extracting the terms used in the course descriptions, we translated and grouped them. Table 3 displays the type groups found in the pool.

Type	n	%	Example types
Lab	336	24,7	Lab, lab exercise, øving (N)
Assignment	316	23,2	Assignment, oppgave (N)
Submission	172	12,6	Innlevering (N)
Participation	122	9,0	Attendance, Participation
Project	79	5,8	Group work, project, project deliveries
Presentation	74	5,4	Group or individual presentations
Other	49	3,6	Blog, video, article
Generic	49	3,6	Work, activity
Meeting	38	2,8	Supervisor meeting, meeting notes
Report	37	2,7	Project, progress and status reports
Test	37	2,7	Test, quiz, exam
Assessment	36	2,6	Self evaluation, peer review, opponent
Contract	17	1,2	Group, supervisor, customer contracts
Total	1362	100	

Table 3: Groups of mandatory coursework type. N = Norwegian term

The top three type groups are similar as they are all activities where students need to apply acquired skills and knowledge to solve assignments within our outside lab sessions and submit a result to be assessed. Combined, they account for over 60% of the coursework within our pool of courses. Motivation for providing assignments throughout a course can sometimes be found in the course description, such as this example from IN1900 at UiO:

"*Programming is a subject that requires training, and in IN1900 there are compulsory assignments every week. You do not have to submit all the assignments, but you have to obtain a sufficient number of points to be able to take the exam.*"

A project is also an activity where students need to apply acquired skills and knowledge, but the task at hand is normally larger and is solved over a longer time period. There is usually a group of students involved in a project, but not always. Participation, on the other hand, is something very different. The participation type group includes:

- Registration (normally obligatory registration at the first lecture or session)
- Attendance at lectures, labs or special occasions such as camps or exhibitions.
- Participation.

Attendance is often set to a specific level of acceptance, such as attending at least 70% of all lectures. Participation is different. The student is not only required to attend but to actively take part in the activity, such as this example from a masters course, IN5420 at UiO: "Active participation in lessons and discussions is required".

Another coursework type we would like to mention specifically is "Assessment". In the previously mentioned case study [5, 6], voluntary peer-assessment sessions replaced mandatory coursework assignments. Currently, assessments are not widely used as mandatory coursework. Of all the mandatory coursework we extracted, assessment only accounted for 2.6 percent.

The most peculiar coursework we found (categorized within the "Other" group) was in ITF15019, Introduction to computer security at Hiof:

> "*Using the knowledge and techniques learned in the course against fellow students, staff or the school's infrastructure without this being agreed upon in advance will mean that the student will not be allowed to take the exam.*"

RQ3: Does the use of mandatory coursework differ between the organizations delivering IT education in Norway?

When presenting variation among the institutions, we provide data on whether mandatory coursework is in use or not, and (if available) how many. This is displayed in table 4. As the number of courses involved differs within the institutions, we also provide the total number of courses by institution (n1) and the number of courses where details of amount and type could be retrieved (n2).

Institution	n1	MCW%	n2	Avg
NTNU	75	73	48	1,8
UiT	38	74	29	2,6
HVL	31	100	29	4,2
UiB	28	96	10	2,4
UiA	38	82	23	1,3
UiS	31	65	29	3,3
UiO	81	91	67	4,0
HK	76	32	57	0,1
Hiof	52	73	52	3,6
OsloMet	69	67	69	2,7
Nord	25	48	14	0,1
USN	124	90	100	2,7

Table 4: Mandatory coursework by institution. n1 = Courses extracted, MCW% = % of courses using mandatory coursework. n2 = Courses where amount and type could be retrieved. Avg = Average mandatory coursework count.

We see that there is significant variation among the different institutions. At HK, only 1/3 of the courses use mandatory coursework with an average of 0,1 coursework per course in courses where the number of coursework could be retrieved. At HVL, we found mandatory coursework in all courses, with an average count of 4,2.

RQ4: Are there certain stages in an IT education where mandatory coursework are more common than others?

Table 5 displays the frequency in which a course uses mandatory coursework and the average number of coursework during the five years of a bachelor and master programme.

Year	n1	MCW%	n2	Avg
1	115	86	87	3,3
2	150	71	118	2,8
3	185	75	148	2,7
4	174	70	136	2
5	44	73	38	1,7

Table 5: Mandatory coursework by programme year. Bachelor 1-3, Master 4-5.

The numbers tell us that mandatory coursework is common in both bachelor and master programmes. IT students experience more coursework in the first year than later in their study.

5 Discussion

There is a big variation in the amount of mandatory coursework in use among the organizations, from an average mandatory coursework count of 0.1 (HK and Nord) to 4.2 (HVL). While there are differences from course to course within the organizations, there seems to be a general understanding within an organization

of the "right" amount of mandatory coursework (with a value that differs from organization to organization). At HK (the authors' organization), there has been a recommendation to have a zero-to-low amount of mandatory coursework, leading to a lot of courses eliminating earlier used mandatory coursework. A higher amount of mandatory coursework could likewise come from internal recommendations. At HVL, the bachelor education where we extracted data had exactly four mandatory coursework in 12 out of its 25 courses. This could come from recommendations from the organization itself, or it could be an internal fight for the student's attention. If one course in a given semester has more mandatory coursework than the others, one could argue that the other courses would aim for the same amount of mandatory coursework to even out the student's attention.

Hattie and Timperley [4] describe feedback as a way of "reducing the discrepancy between current and desired understanding". Within their feedback model they define four levels of feedback focus:

- About the task.
- About the processing of the task.
- About self-regulation.
- About the self as a person.

A "pass" on a mandatory coursework will say something about the task; it is "good enough" - passed. But, from the course descriptions, we do not know if there is additional feedback (or "feed forward") on possible improvements that could help the student moving forward. And we do not know if there is any feedback on process or self-regulation.

The type of coursework does say something about feedback, or the potential for feedback. For some types, such as labs, assignments, submissions and projects, there is reason to believe that more feedback than just the verdict is presented to the student. Additional feedback could help the student moving forward in order to reach the stated learning goals for the course.

But for other types, it is difficult to find any reason to provide any additional feedback at all. The most obvious case is the participation type, and especially attendance. In the context of constructive alignment [3], where teaching and learning activities, as well as assessment tasks, should align with the intended learning outcomes, it is hard to see how attendance fits in. Participation is the fourth most popular choice for mandatory coursework (covering 9 percent of the total mandatory coursework used). While most of the coursework types require the students to somehow display their knowledge within the relevant subject, Participation simply requires the student to be present and possibly actively participate. The coursework types "Presentation" and "Meeting" might fit the same categorization. If we add these, the amount of mandatory coursework that requires students to be present is 17.2 percent out of all mandatory coursework. Participation could represent an activity that has an *indirect* effect on learning outcome as it may contribute to the learning environment. As an example, a masters course where students read and discuss articles may depend on student attendance and participation. It is highly unlikely that there is a dedicated attendance learning goal in the course, but the students achieve other learning goals from reading and discussing the articles. It

could also be that lecturers believe that the general student improves his or her knowledge by being forced to attend in lectures and/or lab exercises, and thus use these types of mandatory coursework. They may also find support for mandatory attendance in existing research, such as Marburger [8] who found that "enforced mandatory attendance policy significantly reduces absenteeism and improves exam performance" in a macroeconomics course.

The test type group includes tests, quizzes and exams. The latter could be an indication that mandatory coursework and exam can be confusing terms. An example can be found at one of the course sites:

> "*The Exam will count 70% or 100% - depending on what gives the best result for the student – Oblig 10+10+10 % + Exam 70% as suggested earlier for the course - or Exam 100% (It means that the grades for the Oblig only can help to improve your overall grade – it will not be able to reduce the overall grade). Due to the fact that Obligs normally do not work in the same way as Home exams. Properly our Obligs with grading (typically good group results between A and C) should then have been made as home exams – as this did not happen – we provide now a solution for the students of (...) for the 2019 spring semester – to take advantage of the best of this.*"

An interesting finding within the test group was a course were tests were used in order to find students who struggled, so that these students could receive some needed help:

> "*Up to 6 tests will be given during the semester. If a test is not passed, the student must do a set of exercises related to the theme of the test, as well as participate in colloquium groups organized by the student assistants in the course.*"

Regarding average mandatory coursework usage per year, we see that there is a decrease each year, with the largest drop coming between the third (end of bachelor) and fourth (start of master) year. The largest drop in MCW% is from year one to year 2. One can assume the earlier years have a higher cost and time investment for usage of mandatory coursework than later years due to larger class sizes and student drop out. So saving expenses can not be the motivation for more mandatory coursework at the start of an education. A possible answer could be that lecturers believe there is a higher need for mandatory coursework early in the education.

6 Conclusion

The course descriptions have told us about the amount of coursework distributed across different institutions and different stages in Norwegian IT education. Mandatory coursework is common practice in Norwegian IT education, although there are big differences within the relevant organizations. Many different type of coursework exist, but the majority of coursework involve a task where the student displays skills and knowledge achieved within a specific topic. Mandatory coursework is found in both bachelor and master programmes, but it is most common in the early stages of a study.

What the course descriptions don't say much about is the formative aspect of the assessment. We do not know what type of feedback the students receive other than hopefully a "pass" and access to the exam.

7 Further work

The purpose of this study was not to discuss if mandatory coursework in Norwegian IT education should be eliminated or not. The purpose was to investigate the use of mandatory coursework to see if such a discussion is necessary. After concluding that mandatory coursework is, at least in some institutions, heavily in use, further work should follow.

A natural next step could be to investigate how students and educators experience mandatory coursework. Are there certain types of coursework that are higher regarded than others? Or maybe there are certain ways of using coursework that are well accepted? And if mandatory coursework should be eliminated, do we need to replace it with something else, such as voluntary peer review activities?

8 Limitations

We cannot know with certainty that mandatory coursework listed in a course description are implemented. Likewise, we assume that all mandatory coursework in use are listed in their relevant course descriptions. When we see course descriptions that do not mention mandatory coursework at all, we assume the course in question does not have any mandatory coursework. Our data rely on course descriptions, but we do not know how they refer to the actual implementation when a course is delivered.

Finally, the exact design and wording of course descriptions differs from organization to organization. There is no set format for how different types of mandatory coursework should be named or described. We have compared and categorized mandatory coursework across different organizations with our best efforts, but some mandatory coursework descriptions could be argued to be either of one type or another.

References

[1] David S. Ackerman and Barbara L. Gross. My instructor made me do it: Task characteristics of procrastination. *Journal of Marketing Education*, 27(1):5–13, 2005.

[2] Randy Elliot Bennett. Formative assessment: a critical review. *Assessment in Education: Principles, Policy & Practice*, 18(1):5–25, 2011.

[3] John Biggs. Enhancing teaching through constructive alignment. *Higher Education*, 32(3):347–364, Oct 1996.

[4] John Hattie and Helen Timperley. The power of feedback. *Review of Educational Research*, 77(1):81–112, 2007.

[5] John Haugan and Marius Lysebo. Hvorfor antall arbeidskrav bør reduseres. *Uniped*, 41(3):347–360, 2018.

[6] John Haugan, Marius Lysebo, and Per Lauvas. Mandatory coursework assignments can be, and should be, eliminated! *European Journal of Engineering Education*, 42(6):1408–1421, 2017.

[7] Lov om universiteter og høyskoler (universitets- og høyskoleloven). https://lovdata.no/lov/2005-04-01-15/§3-9. Accessed: 2019-07-03.

[8] Daniel R. Marburger. Does mandatory attendance improve student performance? *The Journal of Economic Education*, 37(2):148–155, 2006.

Students' mental models of references in Python*

Kristin Marie Rørnes Ragnhild Kobro Runde
Siri Moe Jensen

Department of Informatics
University of Oslo

Abstract

This paper reports on a study exploring students' understanding of references and reference assignments. Students in an introductory programming course in Python were interviewed with respect to what happens during execution of reference assignment statements and function calls involving references. Previous research on Java has identified two types of mental models related to reference assignment, which in this paper is referred to as the "Copy value model" and the "Copy reference model".

An important result in this paper is that each of these models can be divided into two sub-models, giving a total of four different models where only one of them is valid. In addition, we have identified three types of mental models related to references in function calls. This gives valuable insight into students' thinking, which can then be addressed by teachers both in class and in formative assessments.

Furthermore, students in two introductory courses were asked to participate in a survey with multiple choice questions asking the students to identify the correct results of executing code examples involving references. The patterns of answers were analyzed based on the mental models identified in interviews. It was found that the identified mental models explained the most common patterns in the student responses.

1 Introduction

In standard introductory programming courses, variables and assignments are among the first programming concepts encountered by the students. A number of studies have identified typical mistakes made by students regarding what actually happens even in the simplest examples of variable assignments, and that some of these persist well into the first or even second year of programming

*This paper is based on the master thesis of Kristin Marie Rørnes: "Mental Models in Programming: Students' understanding of references in Python" [1].

This paper was presented at the NIK-2019 conference; see http://www.nik.no/.

education [2, 3, 4]. When references are introduced, students seem to find both reference types and reference assignments an even more difficult concept [5, 6, 7].

In [5, 8], Ma et al established the importance of valid mental models of references, and showed that as much as 83% of the students participating in the study held invalid mental models of reference assignment. A mental model is *an explanation of someone's thought process about how something works in the real world* [9], and [5] defines a mental model as valid when the following two conditions are met:

1. The mental model has to match with the model of how a programming concept actually works (appropriate).

2. The mental model "always" has to match with the actual model (consistent).

[5] identified different invalid mental models of references and reference assignments in Java. With an increasing number of introductory programming courses now using Python [10], it is important to establish whether or not the mental models held by the students are the same in Python as was found for Java. Building on the work in [5], this paper presents mental models of references in Python found among students in IN1000[1], the main introductory programming course at the Department of Informatics, University of Oslo. Results from a survey among IN1000-students are compared with results from a similar survey distributed among students in BIOS1100[2].

By establishing why students make certain errors, it is possible to create lectures and exercises that purposefully provoke certain ways of thinking. [4] recommends that research on computer science education should focus more on how students form their understanding of programming concepts, instead of documenting their misconceptions. They suggest that researchers should focus on how students are changing their mental models, in order to develop better learning tools and teaching strategies.

The rest of this paper is structured as follows: An introduction to references in Python is found in Section 2. In Section 3 we present the method used in this survey, before presenting the results in Section 4. Finally, in Section 5 we compare our results with those of Ma et al in [5], and discuss their implications for computer science teachers and further research in this area.

2 References in Python

This paper is concerned with the mental models the students create of the programming concept *references*, and not the details of programming specifics. It is, however, necessary to understand how a reference behaves in order to determine what a valid mental model of references is in Python.

References and reference assignments are treated slightly different in different programming languages. In Python, which is the scope of this paper, the operator = is used for reference assignments, wheres names (or reference variables) are

[1]Introduction to Object-oriented Programming, `https://www.uio.no/studier/emner/matnat/ifi/IN1000/`

[2]Introduction to Computational Modelling in the Biosciences, `https://www.uio.no/studier/emner/matnat/ibv/BIOS1100/index.html`, a new course where students learn introductory programming in Python in order to understand more about biology.

assigned reference values. These values represent memory addresses, which denote where the objects are stored on the heap. Reference assigments can be visualised in the same way for all Python objects, regardless of whether they are mutable (may be changed, e.g. lists) or immutable (may not be changed, e.g. integers and strings). A simple example is shown in Figure 1.

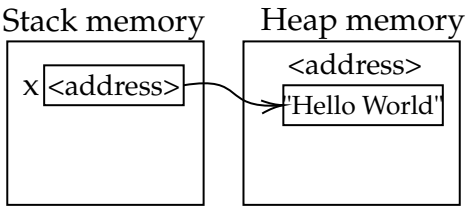


Figure 1: Visualisation of the result of the assignment `x = "Hello World"` in Python

In this paper we follow the terminology used in the syllabus of IN1000, where a reference denotes the memory location of an object, and a variable that contains the memory location to an object is *referring* to that object [11].

3 Method

The study reported in this paper was conducted at the Unversity of Oslo in Autumn 2018. Following the advice of Goldmann et al in [6], interviews were used to get a better insight into students' mental models of references. Nine IN1000-students were recruited to participate in individual interviews. The interviews were semi-structured and centered around code snippets with 4–6 lines of Python code. The participants were asked to think aloud and explain how the code would be executed. Throughout the interviews, the participants were able to speak freely and ask questions along the way. The interviewer guided them through the tasks and continued to ask questions about their thought process. Occasionally they were asked to draw what they thought would happen and they mostly used the conceptual model with boxes and arrows. At times, the interviewer also used the box-and-arrow model to make sure the participants' explanation was understood.

To supplement the data from the interviews, we also distributed an online survey among students in IN1000 and BIOS1100. The questionnaire consisted of ten multiple-choice questions partly inspired by those of [5]. The form was shared in the lab sessions to increase the possibility that students would answer and decrease the risk of students answering multiple times. This meant, however, that not all students would have a chance to answer the questionnaire. In total, we received answers from 76 students in IN1000 and 70 students in BIOS1100.[3] At the time of the survey, the students had learnt about the most basic concepts in progamming, like different data types (including lists), loops, and functions, but not classes and objects.

An important difference between IN1000 and BIOS1100, resulting in two different questionnaires being used for the two student groups, is that BIOS1100

[3]In comparison, 134 students were qualified for final exam in BIOS1100 fall 2018 (students retaking the exam omitted). The corresponding number for IN1000 is 472.

Listing 1: Hei verden

```
1 a = "Hello World"
2 b = "Hei Verden"
3 a = b
4 print(a)
```

teach that functions must have an explicit return statement, whereas IN1000 distinguishes between functions, which have explicit return statements, and procedures, which do not have explicit return statement.

Both questionnaires contained the code in Listing 1 as a control question. Students answering this question wrong are likely to have an incorrect understanding of basic assignments and the equal sign (=), which would influence their answers also to the remaining questions regarding references. 5 of 76 students from IN1000 and 5 of 70 students from BIOS1100 answered this control question wrong and were removed from the data set.

The questionnaires were anonymous. Each question contained 4–6 lines of Python code and asked what would be printed after the code was executed. The students had the possibility of choosing one of three given alternatives or writing a free-text answer if they thought the code would result in another answer than the alternatives. The questions were given in a random order to prevent students from collaborating. The alternatives, however, were not given in a random order but followed the same structure for each question to minimise any confusion. All of the code snippets and alternatives used in the questionnaires may be found in [1].

4 Results

In this section, we first present the identified mental models for references and reference assignment not involving functions, and then the identified mental models for references combined with functions. The mental models are then applied to the questions from the two questionnaires, and compared with the students' answers to see if the identified mental models could explain the patterns seen in the students' answers.

Mental models of references

The mental models presented here are based on explanations, drawings and patterns found in the transcribed interviews. There are two types of models that were clearly seen in the interviews:

Copy reference models: An understanding where the students explain `a=b` as meaning that `a` and `b` are referring to the same memory location (or "pointing" to the same object), which means that these variables have the same value (which is a reference). There are two variants of copy reference models, one invalid and one valid as will be demonstrated below.

Copy value models: A (mis-)understanding where the students explain `a=b` as meaning that `a` becomes a copy of the value that `b` has been assigned to.

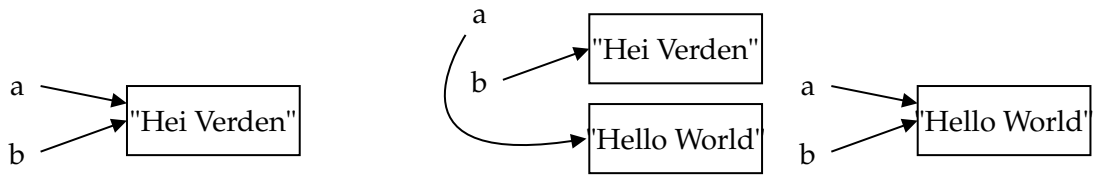
The copy value models are considered invalid, as there are cases where they give an incorrect result.

To illustrate the different mental models as found in the interviews, the code snippet in Listing 2 will be used.

Listing 2: Hei verden

```
1  a = "Hei Verden"
2  b = a
3  a = "Hello World"
4  print(b)
```

Figure 2 visualizes two different "Copy reference models" applied to the code in Listing 2. In both models, the assignment `a=b` results in `a` and `b` referring to the same text object as seen in Fig 2a. The difference between the two models can be seen in Figure 2b and Figure 2c. In the valid model MREF1, `a` is *reassigned* to a new memory location after code line 3 is executed, but the object that `b` refers to is not changed. In the invalid model MREF2, the value in `a` is changed *at the memory location* that `a` refers to after code line 3 has been executed, which means that the object `b` refers to also has changed. MREF2 may seem illogical, in that the assignment statement is interpreted differently depending on whether the right side is another variable (in line 2) or a value. However, many of the students seem to use this line of reasoning in a consistent manner.



(a) MREF1 and MREF2 after line 2 is executed

(b) MREF1 after line 3 is executed

(c) MREF2 after line 3 is executed

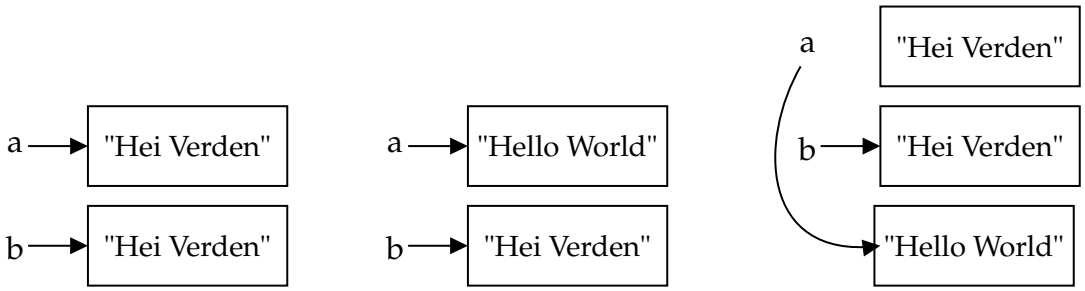
Figure 2: MREF1 and MREF2 for Listing 2

Figure 3 visualizes two possible "Copy value models". Although in this case, both result in the text `"Hei Verden"` being printed in line 4 (as in the valid model MREF1), these models are not considered valid since there are other cases where the result is not correct, as will soon be illustrated.

As can be seen in Figure 3a, the copy value models differ from the copy reference models in that here, `a` and `b` are understood to refer to (or point to) two different objects that contain the string object `"Hei Verden"`.

In Figure 3b the version called MVAL1 is presented. Here, the value in `a` is changed *at the memory location* that `a` refers to after code line 3 has been executed. In Figure 3c the version called MVAL2 is presented. In this version, `a` is *reassigned* to a new memory location after code line 3 is executed. This is a subtle difference, but important e.g. to distinguish MVAL2 from the valid model MREF1, which both explains code line 3 as a reassigment of `a`.

A case where MVAL1 and MVAL2 both give an incorrect result, is provided in Listing 3. Here, the correct result of the print statement in line 4 is `[10,2,3]`, while



(a) MVAL1 and MVAL2 after line 2 is executed

(b) MVAL1 after line 3 is executed

(c) MVAL2 after line 3 is executed

Figure 3: MVAL1 and MVAL2 for Listing 2

Listing 3: studentID

```
1 personID = [1,2,3]
2 studentID = personID
3 personID[0] = 10
4 print(studentID)
```

students using MVAL1 or MVAL2 would say that the result is `[1,2,3]` since they believe that `studentID` is a *copy* of `personID`, meaning that changes to `personID` are not reflected in `studentID`.

The questionnaires for IN1000 and BIOS1100 consisted of respectively four and five simple code snippets where the four models identified here could be applied directly. As can be seen from Table 1, only 38 of the 136 students answered consistently in accordance with the valid model MREF1. For almost 70% (68 of 90) of the remaning students, their answer patterns could be explained using one of the identified invalid models. However, the surveys did not include any questions to distinguish between the invalid models MVAL1 and MVAL2, as the difference between these had not been identified at the time of the surveys.

Model	**Notes**	**IN1000 (N=71)**	**BIOS1100 (N=65)**	**Total (N=136)**
MREF1	Copy reference (valid)	17	21	38
MREF2	Copy reference (invalid)	9	11	20
MVAL1/2	Copy value (invalid)	34	14	48
(undefined)	(invalid)	11	19	30

Table 1: Correlation between mental models and students' answers on questions without functions

Mental models of references combined with function calls

Introducing functions with references as parameters, adds to the complexity as students now also have to understand how parameter passing works, and if and when changes to the parameter inside the function body affects the rest of the program. Although having worked with functions and parameters in the course, none of the interviewed students could correctly explain what happens when a

function is called. Instead, most of their explanations were found to fall into one of two categories: 1) A copy of the argument's value is sent to and stored in the parameter of the function (the model FCOP), and 2) The argument is the actual object which is sent to the function as a parameter (the model FOBJ). To illustrate the different models, we will use the code example given in Listing 4.

Listing 4: language

```
1 language = ["English","Norwegian","German"]
2 programmingLanguage = ["Python","Java","Simula"]
3 def changeLanguage(array1,array2):
4   array1 = array2
5 changeLanguage(language,programmingLanguage)
6 print(language)
```

Before presenting the invalid models found in the interviews, we present a valid mental model (the model FREF) based on how this is taught in IN1000 and how Python actually works. When a function with parameters is called, local variables are created with the parameter names. These variables become references to the same objects as the arguments pased to the function are referring to. In Figure 4, the function call to `changeLanguage` in line 5 means that the parameters `array1` and `array2` become local variables that refer to the same objects as the arguments `language` and `programmingLanguage` refer to in the global scope.

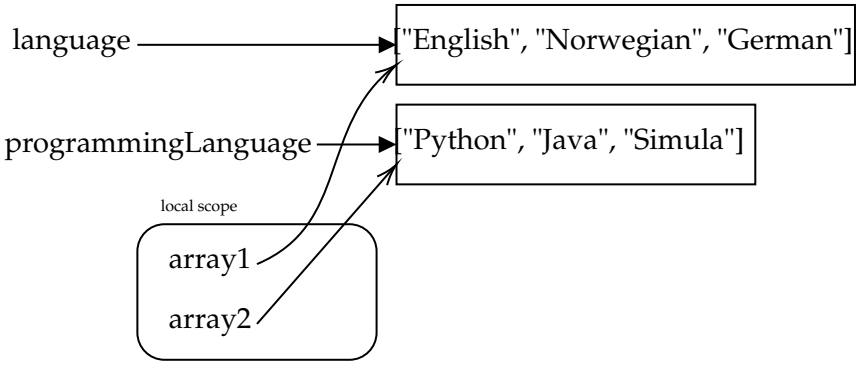


Figure 4: FREF after line 5 (i.e. before execution of the function body in line 4) in Listing 4

For the function body in line 4 in Listing 4, the combination of FREF with the valid model MREF1 for reference assignment is illustrated in Figure 5, which shows that the result of the `print`-statement in line 6 will be the list `["English","Norwegian","German"]`.

As explained above, some students have an invalid model FCOP where the parameters of a function becomes *copies* of the values referenced by the passed arguments, as illustrated in Figure 6. The parameters are treated as local variables and any changes inside the function will only affect the local copies. Combining FCOP with one of the four models for reference assignment (MREF1/2 and MVAL1/2), would result in a different understanding of what happens inside the function body, but as this would not have any effect outside the function, the effect of the print statement would be the same in all cases.

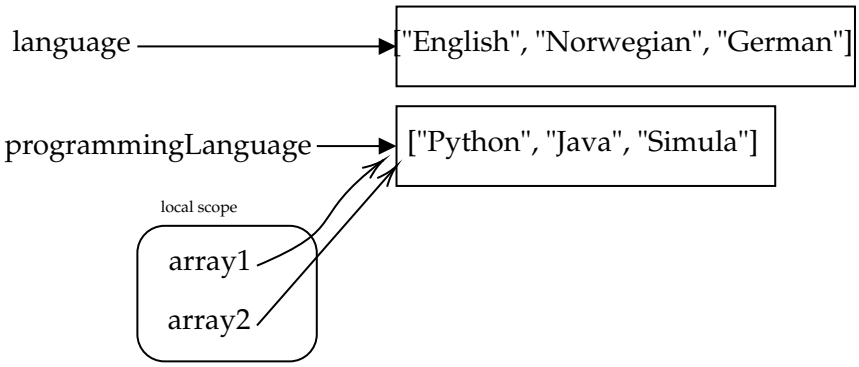


Figure 5: FREF+MREF1 at the end of line 4 in Listing 4

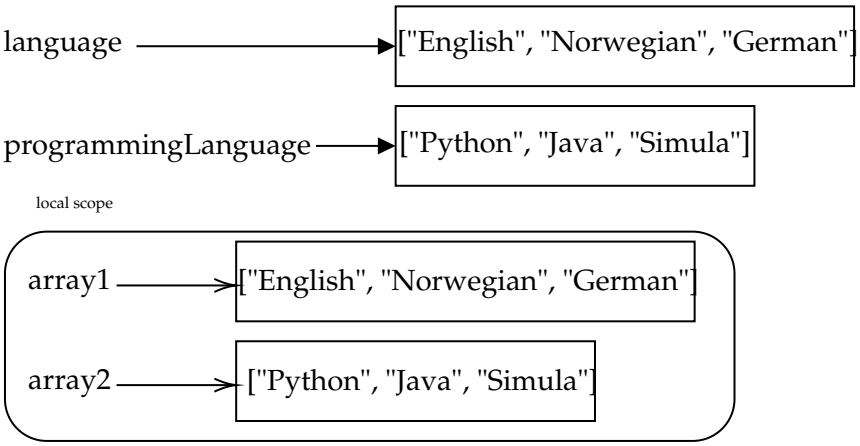


Figure 6: FCOP after line 5 in Listing 4

For references combined with functions, the other invalid model identified in the interviews is the model FOBJ where the parameters are believed to be set to the *actual* objects passed to the function. Figure 7 show that in this case, `array1` is identified with `language` and `array2` is identified with `programmingLanguage`. All the changes made to the local variables inside the function will affect the references and objects in the global scope.

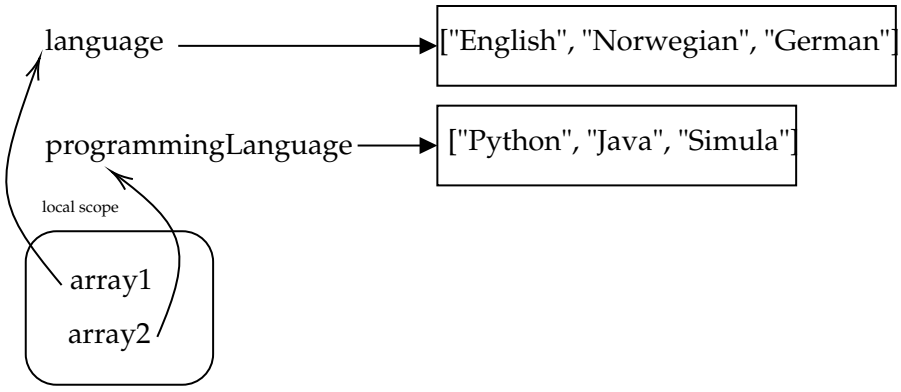


Figure 7: FOBJ after line 5 in Listing 4

Combining the invalid model FOBJ with the invalid copy value models MVAL1/2 results in an incorrect result as illustrated in Figure 8. As seen, the value of the object referred to by `programmingLanguage` is copied to the object

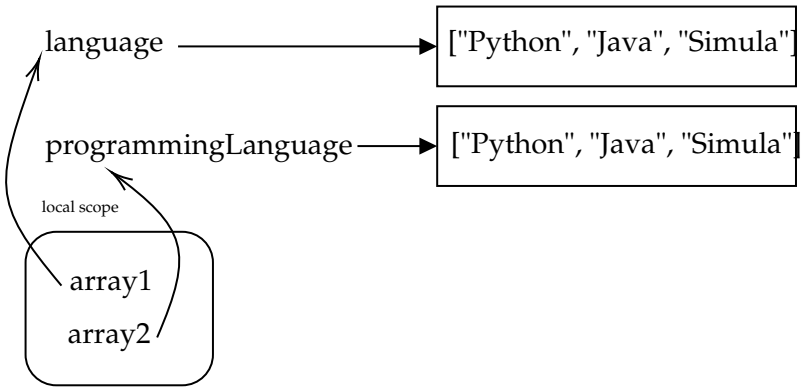


Figure 8: FOBJ+MVAL1/2 at the end of line 4 in Listing 4

that `language` is referring to.

The combination of FOBJ with the copy reference models MREF1 (valid) or MREF2 (invalid) is illustrated in Figure 9. As shown, `language` is now referring to the *same* object as `programmeringLanguage` is assigned to. As FOBJ is an invalid model, combining it with MREF1 will often give an incorrect result (as in this case), even though MREF1 in isolation is a valid model.

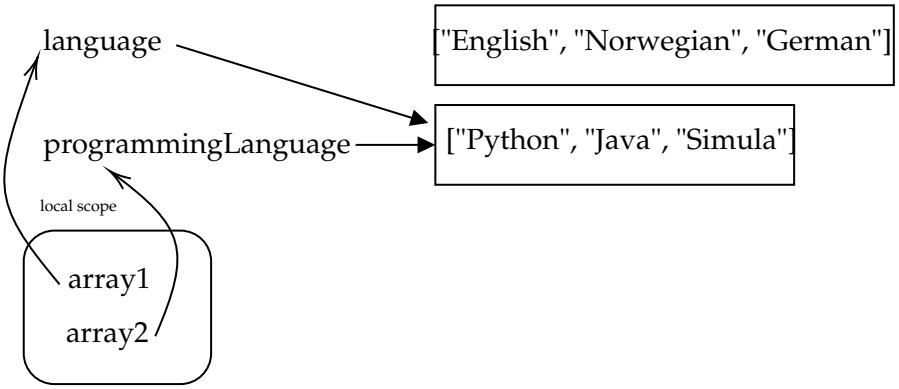


Figure 9: FOBJ+MREF1/2 at the end of line 4 in Listing 4

The questionnaires for IN1000 and BIOS1100 consisted of respecively two and four simple code snippets relevant for the identified mental model of references combined with functions.[4] As can be seen from Table 2, as little as 14% (19 of 136) answered in accordance with the valid model FREF, meaning that the remaining 86% either have an invalid/inconsistent model or have restorted to guessing. Most of the answers from the IN1000 students could be explained by one of the identified models, but only less than half of the answers from the BIOS1000 students.

Finally, we have compared the students' answers to the questions with and without functions and how they fit with the identified models. The results for IN1000 is presented in Table 3.[5] As can be seen, only 4 students (6%) have

[4]The questionnaire for IN1000 also included four other questions concerning functions, but during the analysis it was found that wrong answers here were most likely due to misunderstanding other Python specific concepts and not references as such. They are therefore not included in the results presented here.

[5]The results for BIOS1100 may be found in [1].

Model	Notes	IN1000 (N=71)	BIOS1100 (N=65)	Total (N=136)
FREF	valid	8	11	19
FCOP	invalid	22	7	29
FOBJ	invalid	34	10	44
(undefined)	(invalid)	7	37	44

Table 2: Correlation between mental models and students' answers on questions with functions

results consistent with the valid models MREF1 and FREF. For the remaining students potententially using the valid model MREF1, the invalid model FOBJ is the most common, indicating that while they correctly understand that reference assignment means reassigning the variable, they incorrectly believe that reference parameters is identified with the actual parameter names, and not their content.

For the invalid models MREF2 and MVAL1/2, most students seem to also use one of the invalid models FCOP and FOBJ, with the students almost evenly split between the two.

	FREF	FCOP	FOBJ	Other
MREF1	4	2	10	1
MREF2	0	3	5	1
MVAL1/2	2	14	14	4
Other	2	3	5	1

Table 3: Comparing the results from IN1000-students on questions with and without functions (N=71)

5 Conclusions and future work

Based on the results from student interviews, we have in this paper presented four mental models for reference assignment, and three mental models for functions with references as parameters. Student questionnaires from two introductory courses in programming show that very few students have valid mental models of references more than halfway through the course, and there are strong indications that the most common patterns in the student answers may be the result of using one of the identified models.

For reference assignment, the "Copy reference models" is similar to the "Component assignment model" in [5] and the "copy value models" is similar to the "Set equal model" in [5]. In this paper, we have explained these two types of models in more detail, and found that each type may be divided into two sub-models, all found among the students interviewed as part of this study.

Assuming that the students did not answer the questionnaires simply by guessing or answering too quickly, the results show that some students have answer patterns not consistent with any of the identified models. As this applies to students from BIOS1100 more than students from IN1000, this could be the result of teaching differences not captured in the interviews with only IN1000-students. A natural follow-up study would be to interview more students from

different programming courses.

At the Department of Informatics, University of Oslo, Java is used as the main programming language in the programming courses following IN1000. In principle, the models presented in this paper should be applicable to any standard object-oriented programming language. To learn more about students' understanding of references, and how this understanding develop over time and with different programming languages, it would be interesting to conduct a similar study with interviews and multiple-choice questions in one or more advanced programmering courses, and preferrably also a more longitudinal study following the same students over time.

One of the questions that emerge from these findings is how to present references to students who learn Python, and in particular how this may done in a way that is applicable or at least easily transferable to Java and other programming languages. This is ongoing work at the department, and preliminary results indicate that targeted interventions at least to some extent increase the number of students with a valid mental model of references and reference assignment.

We believe that being aware of how difficult the concept of references is for students, and the existence of the different invalid mental models, may help teachers pay more attention to this both in lectures and when helping individual students. As part of future research, a set of test questions should be developed to be able to clearly distinguish between the different models identified in our study. A precise set of questions will also be useful as a diagnostic tool for teachers and/or the students themselves. As we have seen, invalid mental models may give a correct answer in many cases, which may make it more difficult for the students to realize that a particular programming error may be due to a lack of understanding of references.

References

[1] K. M. Rørnes. Mental models in programming: Students' understanding of references in python. Master's thesis, Department of Informatics, University of Oslo, Norway, 2019.

[2] B. D. Boulay. Some difficulties of learning to program. *Journal of Educational Computing Research*, 2(1):57–73, 1986.

[3] Simon. Assignment and sequence: Why some students can't recognise a simple swap. In *Proc. of the 11th Koli Calling International Conference on Computing Education Research*, Koli Calling '11, pages 10–15. ACM, 2011.

[4] Y. Qian and J. Lehman. Students' misconceptions and other difficulties in introductory programming: A literature review. *ACM Transactions on Computing Education*, 18(1):1:1–1:24, 2017.

[5] L. Ma, J. Ferguson, M. Roper, and M. Wood. Investigating the viability of mental models held by novice programmers. In *Proc. 38th SIGCSE Technical Symposium on Computer Science Education*, SIGCSE '07, pages 499–503. ACM, 2007.

[6] K. Goldman, P. Gross, C. Heeren, G. Herman, L. Kaczmarczyk, M. C. Loui, and C. Zilles. Identifying important and difficult concepts in introductory computing courses using a delphi process. In *Proc. 39th SIGCSE Technical Symposium on Computer Science Education*, SIGCSE '08, pages 256–260. ACM, 2008.

[7] L. C. Kaczmarczyk, E. R. Petrick, J. P. East, and G. L. Herman. Identifying student misconceptions of programming. In *Proc. 41st ACM Technical Symposium on Computer Science Education*, SIGCSE '10, pages 107–111. ACM, 2010.

[8] L. Ma, J. Ferguson, M. Roper, and M. Wood. Investigating and improving the models of programming concepts held by novice programmers. *Computer Science Education*, 21(1):57–80, 2011.

[9] Wikipedia contributors. Mental model — Wikipedia, the free encyclopedia. `https://en.wikipedia.org/w/index.php?title=Mental_model&oldid=914045188`, 2019. [Online; accessed 7-September-2019].

[10] R. M. Siegfried, D. Liporace, and K. G. Herbert-Berger. What can the Reid list of first programming languages teach us about teaching CS1? In *Proc. of the 50th ACM Technical Symposium on Computer Science Education, SIGCSE 2019*, pages 1256–1257, 2019.

[11] C. S. Horstmann. *Python for everyone*. Wiley, 2nd edition, 2016.

Evaluating accessibility testing in automated software build processes

Aleksander Bai, Rannveig A. Skjerve, Till Halbach, Kristin Fuglerud

Abstract

Today, most software projects utilize an automated build process with unit tests, code quality checks, end-to-end integration tests, and more. To make software usable by as many people as possible, regardless of capabilities, accessibility testing must be integrated into the build process. The goal is highly accessible software where issues are found early in the development lifecycle. In this work, we have investigated the most common accessibility testing tools suitable for integration in an automated build process and split the tools into two categories based on their rulesets. The rulesets are evaluated against a well known demonstration site for calculation of the rulesets' precision, recall, and F_1 scores. Finally, we discuss the implications of their scores and how accessibility testing tools can be integrated into an automated build process.

Accessibility, Software Development, Testing, Automated Build Process.

1 INTRODUCTION

The aim of accessibility testing is to evaluate whether the final solution will be accessible for a wide range of people, including people with various types of impairments, such as limitations in vision, hearing, cognition, movement and speech. Accessibility testing is an important part of the process to achieve universally designed solutions (1).

Software development is a complex process where many people are involved. During the last 20 years, the profession has improved the quality and complexity to a level where it is not easy for a single person to have complete overview over all functionality. To distribute the burden of fixing bugs and avoid regression, the industry have introduced techniques that were uncommon 20 years ago; unit testing, faster and smaller iterations, automated building of source code, automated execution of integration tests, and so on. However, accessibility testing is still mostly performed by humans at the end of software cycles (2). Even though late accessibility testing is costly (3), projects seem to ignore this fact in the heat of the moment. Research also suggest that developers need easy methods of implementing accessibility in order to increase the overall accessibility of digital solutions (4; 5).

*All authors were with the Department of Applied Research, Norwegian Computing Center, Oslo, Norway.

This paper was presented at the NIK-2019 conference; see http://www.nik.no/.

To reduce the costs of accessibility testing, it seems wise to follow an automated approach which is well integrated in the software development process. Furthermore, it has been argued that automated accessibility testing avoids repetitive and manual testing, which allows teams to focus on delivering flawless and highly usable software (6).

In this paper, we try to answer the research question in how far automatic accessibility checkers can in fact add value to today's continuous-deployment projects. Doing so, we focus on testing tools for web development, since this is the primary medium for most software projects today. The main contributions of our work are the overview and assessment of available automatic tools, and the analysis of the underlying rulesets.

The remainder of the paper is organized as follows: After the discussion of related work in Section 2, we give an overview over the most common automated accessibility testing tools in Section 3. Our methodology is presented in Section 4. We present results from the evaluation of tools in Section 5, before different strategies for integration in build processes are discussed in Section 6. Finally, we summarize and highlight future research directions in Section 7.

2 RELATED WORK

Multiple studies have evaluated automated accessibility tools (7; 8; 9). The tools find different accessibility issues, but there is no single tool to find all the issues (7; 9). Even when results from the tools are combined, almost 1/3 of the issues are not found (10). Often, these tools must be invoked manually. Some of them are prone to poor usability and lack the adequate functionality for web developers (11; 9). Therefore, web pages should be tested for accessibility in at least three different ways: automated testing for code compliance and compatibility with assistive technology, manual testing by experts, and user testing involving persons with disabilities (12).

It is very cost-efficient to automatically uncover (and fix) accessibility problems as early as possible in the development process (3). In that way, testing that involves humans can concentrate on issues that cannot be checked automatically. Therefore, although the use of automated accessibility tools has limitations, using such tools can be an important step towards achieving more accessible solutions. Also, in order to ensure and maintain the achieved accessibility level, one is dependent on routine testing and the use of automated evaluation tools (13; 14).

Most studies focus on automated tools that must be operated by a user, either in the browser or using a native application. Some researchers discuss various accessibility testing tools to be integrated into a build process (15; 16), but the tools are not compared with each other, and only with other accessibility evaluation methods.

3 ACCESSIBILITY TESTING TOOLS

There is no global overview over all the possible tools for accessibility testing, and in particular not for tools that can be integrated in a build process. The most complete overview over available testing tools for web is W3C's *Web Accessibility Evaluation Tools List* (17). However, the list is slightly outdated and also missing popular alternatives, such as *Automated Accessibility Testing Tool* (AATT) from Paypal (18). Therefore, we conducted an extensive online search and seeks on software

project sites like Github and Bitbucket to identify accessibility testing tools possible to integrate in a build process.

According to the W3C, most existing tools cannot be run from the command line (17), and very few provide an API that enables their integration into a build process. Only tools and libraries that were possible to integrate fully into an automatic build process for web development were considered, and thus many popular accessibility checkers (like SiteImprove) were disregarded in our study.

We identified 11 tools that can be integrated in a build process for web development, along with the ruleset the tools is based on, enlisted in Table 1. A ruleset is the collection of a limited number of tests to carry out.

Table 1: Overview of automatic testing tools. A grey background marks active development

#	*Tool*	*Ruleset*	*Active*	*Envir.*
1	aXe	aXe-core	yes	Node
2	pa11y	HTML-CS	yes	Node
3	GADT	self-defined	no	Node
4	AATT	aXe-core, HTML-CS	yes	Node
5	accessibilityjs	accessibility.js	no	Node
6	The A11y Machine	HTML-CS	no	Node
7	Access Continuum	self-defined	yes	Java
8	Asqatasun-Runner	self-defined	yes	Java
10	Webhint	aXe-core	yes	Node
11	Google Lighthouse	aXe-core	yes	Node

Some tools (like tenon.io and WAVE's stand-alone API) are possible to integrate in a build process, but they cost money. They are accessible as a web service which hurts performance by introducing a network roundtrip for every test. This can be significant even for a small project if the code is built and tested often. Service tools were therefore not include in the evaluation.

To compare the different tools, we identified 32 criteria we deemed relevant from a development and integration perspective. We used already established criteria (19; 9; 20) before trimming the list down to 23 criteria that was essential for evaluating all the different tools.

We also removed redundant criteria because they were already covered by the ruleset. Criteria that covered setup complexity and very technical elements like DOM coverage were ignored. We will not discuss the 23 criteria here, but focus on a smaller subset that were important when selecting the correct tool.

Three criteria proved to be more significant when choosing tools for further investigation: ruleset, active development and environment. It is critical with an active development to make sure that bugs are fixed, new rules are added, and that the documentation is maintained. It is also important that a Node (JavaScript) environment is supported, since this is the standard environment for modern web development and build processes (21; 22).

If we eliminate all tools that do not have an active development or Node as environment, we can reduce the list to five tools, number 1, 2, 4, 10 and 11, marked with a grey background in Table 1. However, we can further trim the list down to the different rulesets deployed, since there are only two major rulesets: aXe-core (23) and HTML-CodeSniffer (HTML-CS) (24). They are designed to test not only

the accessibility of entire websites and pages, but also markup snippets and other HTML-based interfaces. Both are available on Github. This choice boils our 11 automatic tools down to essentially two types; those that use the aXe-core or the HTML-CS ruleset. This finding is confirmed by an extensive search which returned one or both of these tools in almost all search results.

Until more popular rulesets emerge, it is probably a wise choice to select a testing tool that is based on either aXe-core or HTML-CS. We have also seen that existing tools that are not based on one of these rulesets (but rather use their own) are starting to adopt them. E.g., Google turned down development of their GADT tools based on a self-defined ruleset and created the Lighthouse tool instead which is based on aXe-core.

Even though we focus on the rulesets used by the tools in this paper, there are of course many differences between the actual tools. Which specific tools that is the best fit for a project depends on many factors, and we suggest to use the Appendix A as a guidelines for deciding on a particular tool. However, as we will discuss in the next sections, it's important to select the best ruleset first.

4 METHODOLOGY

To evaluate the rulesets, they were tested against a site with defined accessibility problems. We used the W3C *before and after demonstration* site (25). This site has four pages in an inaccessible and an accessible version, with errors defined by persons within the Web Accessibility Initiative (26). This makes it possible to check how many issues a ruleset is able to discover in the inaccessible version, and how many false positives a ruleset reports in the accessible version.

We also checked which WCAG 2.1 guidelines (27) the rulesets claim to detect and which WCAG Level (A/AA/AAA) they apply to, and the results are shown in Table 2. We used the latest version of both rulesets (versions as listed in the table), and we used the aXe tool (28) when checking the aXe-core ruleset and the pa11y tool (29) when testing the HTML-CS ruleset. Both rulesets have multiple rules for a single WCAG guideline, and they appear to be quite similar in terms of number of rules; aXe-core has 56 rules in total while HTML-CS has 61 rules in total.

Both rulesets distinguish between WCAG levels. Furthermore, HTML-CS operates with levels of found issues; issues classified as errors are detectable by the tool alone, issues that are classified as notices and warnings requires manual inspection. Only rules that classify as errors were considered in this study.

The aXe-core ruleset does not contain any rules that requires manual inspection, but rules that are classified as "best practice" were excluded since they are not a part of the WCAG specification.

5 RESULTS

Table 2 gives an overview of the rulesets and their WCAG 2.1 coverage. A success criteria was defined as "covered" if one or more rules was present in the ruleset. This does not mean that a tool checks for every possible violation of the criteria, but that the tool has checks for at least one violation of the criteria.

In WCAG 2.1 (27), there are 30 guidelines for level A, 20 guidelines for level AA, and 28 guidelines for level AAA; a total of 78. Thus, the best ruleset covers only 31% (32% in WCAG 2.0) of the WCAG criteria, as indicated in Table 2. On the

Table 2: Overview of ruleset and their WCAG 2.1 coverage

Tool	*Level A*	*Level AA*	*Level AAA*	*Total*
aXe-core (3.2.2)	16 (53%)	6(30%)	2(7%)	31%
HTML-CS (2.2.0)	10(33%)	2(10%)	4(14%)	21%

positive side, the best ruleset covers 53% of the level A criteria, but this illustrates that even the best ruleset / automatic tool has a lot of potential for improvement. This is as expected and also reported by other studies (7).

In our evaluation we focused on three key parameters: True Positives (TP), False Positives (FP), and False Negatives (FN). TP is where a ruleset flags an issue, which really is an issue, in the inaccessible version. FP is where a ruleset reports a WCAG error that is not defined as an issue in the accessible version. FN is when a ruleset does not find a defined issue in the inaccessible version. We also conducted an expert evaluation if a ruleset reports a FP, since it potentially could be an error in the accessible version.

Based on these parameters, we can calculate a tool's precision (or correctness), and recall (or completeness) according to Eq. 1 and Eq. 2. Both these metrics are important to analyze since precision tells us how many of the reported issues are actually correct, and recall quantifies the degree of correctly identified issues (30). The range for both precision and recall is from 0 to 1, where 0 is the worst result and 1 is the best result. The number can be multiplied with 100 to give a percentage.

$$Precision = \frac{TP}{TP + FP} \tag{1}$$

$$Recall = \frac{TP}{TP + FN} \tag{2}$$

The results from the aXe-core ruleset are found in Table 3. Here, the ruleset is evaluated against the inaccessible version where the W3C has deliberately planted 110 issues that violate WCAG guidelines. The true number of errors is higher, as many guidelines are violated by multiple errors. We have counted the violations of WCAG guidelines and not the total number of errors.

As Table 3 shows, the aXe-core ruleset does not find many of the planted WCAG violations, with a high number of FN and a low recall for all pages. However, aXe-core does not report any FP and achieves a perfect precision once it detects a violation.

Table 3: Ruleset results for aXe-core from the inaccessible version

Page	*TP*	*FP*	*FN*	*Precision*	*Recall*
Home	8	0	19	1.0	0.30
News	7	0	20	1.0	0.26
Ticket	8	0	20	1.0	0.29
Survey	7	0	21	1.0	0.25

For the HTML-CS ruleset, the results from the inaccessible version are found in Table 4. The same pattern is found here, with a low recall and perfect precision. However, the HTML-CS ruleset has a worse recall than aXe-core.

Table 4: Ruleset results for HTML-CS from the inaccessible version

Page	*TP*	*FP*	*FN*	*Precision*	*Recall*
Home	5	0	22	1.0	0.19
News	5	0	22	1.0	0.19
Ticket	5	0	23	1.0	0.18
Survey	5	0	23	1.0	0.18

The aXe-core ruleset finds almost all WCAG guideline violations that HTML-CS finds, but HTML-CS does not find the following WCAG violations: 2.4.1 (Bypass Blocks), which is level A, 2.4.4 (Link Purpose), which is level A, and 3.3.2 (Labels or Instructions), which is level A. Since all are level A, it is a little surprising that the HTML-CS ruleset does not find them, considering that they should be programmatically easy to detect.

Both rulesets finds these WCAG guideline violations: 1.1.1 (Non-text Content), which is level A, 1.3.1 (Info and Relationships), which is level A, 1.4.3 Contrast (Minimum), which is level AA, 3.1.1 (Language of Page), which is level A, and 4.1.2 (Name, Role, Value), which is level A. It is not very surprising, though, that both rulesets find violations for the mentioned WCAG guidelines as these have well defined rules.

The results from the accessible version are shown in Table 5. According to the W3C demo site (25), there should not be a single guideline violation, and we have only listed FP in the table. Only the HTML-CS ruleset reports a WCAG violation.

Table 5: Number of reported FP in the accessible version

Page	*aXe-Core FP*	*HTML-CS FP*
Home	0	0
News	0	0
Ticket	0	1
Survey	0	1

It should be mentioned that the false positive reported by HTML-CS is a difficult case. It is claimed to be a violation of WCAG guideline 1.3.1, and HTML-CS reported the violation with the description *Incorrect headers attribute on this td element*. This is, however, incorrect as the td element uses the correct header references (31), even though one of the labels is not strictly above the column. This illustrates how difficult it is to interpret the guidelines.

Based on the total number (sums) of TP, FP, and FN, we can calculate the total precision and recall as shown in Table 6. We also calculate the $F_1 score$ according to Equation 3. This metric is a harmonic average of both precision and recall (30) and makes it easier to compare the rulesets against each other. The $F_1 score$ reaches its best value at 1 (perfect precision and recall) and worst at 0.

$$F_1 score = 2 \times \frac{precision \times recall}{precision + recall} \tag{3}$$

As Table 6 show aXe-core has a decent $F_1 score$ with 0.43, while HTML-CS is slightly worse with 0.30. Even though aXe-core has a significant higher $F_1 score$ than HTML-CS, both rulesets achieves reasonable results considering they can only check for programmatic violations.

Table 6: Total results from all pages and both versions

Ruleset	*TP*	*FP*	*FN*	*Precision*	*Recall*	*F1 score*
aXe-core	30	0	80	1.00	0.272	0.429
HTML-CS	20	2	90	0.909	0.181	0.303

The aXe-core tool both finds more violations (TP) and is more reliable (no FP) as compared to HTML-CS. So, given the version 3.2.2 of aXe-core and version 2.2.0 of HTML-CS, aXe-core is clearly the better option. However, as both tools are under active development, our tests should be repeated from time to time to mirror the improvements of the development team's latest patches.

6 DISCUSSION

In our evaluation, both rulesets achieved low recall scores (0.43 and 0.30), which is not unexpected since automated accessibility checkers are far from enough to find all accessibility issues on a webpage. They struggle in particular to find violations of WCAG guidelines that requires deduction about context and layout (32).

However, the high precision scores (1.00 and 0.91) show that the rulesets can be trusted in what they report. This is reassuring and indicates that they are safe to integrate in an automated build process. A high number of FPs causes unnecessary noise during the build process where humans manually have to inspect if reported violations are actual errors or not.

Unless a test breaks, code is not compilable, or something unexpected happens, most software teams want the build process to operate in the background. This strategy is actually imperative in continuous deployment where several hundred or even thousands of builds happens every day (33; 34).

It is not obvious what the best integration approach for accessibility testing tools is. For example, should an accessibility violation break the whole build similar to a unit test failure? Or should only a violation of WCAG Level A guidelines interrupt deployment? Should particular violations be tolerated in prototyping, and/or in development, or even production?

We know from other studies that accessibility testing is often considered a burden and therefore not prioritized (35). This means that a possible consequence of a noisy accessibility testing tool is that teams might decide not to integrate accessibility tools altogether. However, it is not advisable to go down that road, as not having accessibility testing as part of an automated build process is considered to be an anti-pattern (36), and accessibility testing should therefore be fully integrated in an automated build process for cost effectiveness.

An argument against this, however, is that automated accessibility testing does not find all accessibility issues. While this is true to some extent, it is not a good reason not to take control of the issues that they do find. A high degree of automation will also likely ease the burden of manual accessibility testing by removing accessibility issues that are easy to find programmatically.

Another argument against integrating accessibility testing tools in the build process is that it takes time to fix the accessibility violations. This concern is debunked by the benefits of providing accessible software (37; 38), and because more and more countries are introducing accessibility laws (39; 40).

To change the current software testing approach and mindset, accessibility

testing tools thus need to be integrated fully into modern build processes. The above arguments also advocate for following the approach that, if an accessibility violation is found, this event should break the build in the same manner as unit tests. It will cause a little overhead in the beginning for teams that are not used to work with accessibility, but once the team members understand why the build is failing, they are anticipated to start focusing on how to avoid introducing those bugs.

Having accessibility testing tools as part of the build process will then also provide a fail-safe for web development teams. A common misconception is that once a webpage has been made accessible, the work required to keep it accessible is done. This is grossly incorrect since a webpage is in continuous development as teams change, features are altered or removed, new features are added, and since the software is constantly evolving. This means that a webpage that once was accessible can suddenly become inaccessible if no attention is paid to accessibility.

7 CONCLUSION

Based on an overview of the most common automatic accessibility tools for software build integration, we discovered that the most significant factor was the ruleset. There were two primary rulesets among all the active tools: aXe-core (23) and HTML-CS (24). Our investigation revealed moderate differences in quality, and aXe-core outperformed HTML-CS particularly in terms of $F_1 score$. However, both tools still have considerable potential for improvement when it comes to WCAG coverage and recall. This could be remedied by adding more tests to the rulesets, by converting tests from manual to automatical execution, and by improving the quality of existing tests.

We argue that the result presented here is independent of tools we might have missed, as such tools are likely to be based on either aXe-core or HTML-CS, and we have based our study on those rulesets and not the actual testing tool. It is, however, nevertheless important to ensure that the selected testing tool uses the latest version of the rulesets to mirror the latest code improvements.

Underlying rulesets and automatic tools are critical enablers for having accessibility testing fully integrated in an automated build process. They are also a prerequisite if teams want accessibility violations to interrupt the deploy process. Their successful integration is thus a promising strategy to achieve a continuous focus on accessibility in software development.

We conclude that some automatic accessibility checkers in fact have the potential to add value to today's continuous-deployment projects if integrated properly. Added value also implies, though, that the right strategy for handling accessibility errors is chosen. The tools' two underlying rulesets have different quality, and our analysis shows that the use of aXe-core is currently preferable over HTML-CS.

Two final comments. First, we have focused in this work on accessibility testing tools for web development because the area has various tools available to check accessibility issues, and because web is the dominating medium in the industry. Our method can, however, be extended to typical desktop and native software development, assuming the sufficient ruleset and tools are developed. Second, to our knowledge, there is no resource other than W3C's *before and after demonstration* site that has a quality-assured and controlled accessible and inaccessible version of a set of webpages. We urge the accessibility community to create a new and modern

webpage resource that can benefit the benchmarking of accessibility testing tools in the future. Such a resource should use typical scenarios like shopping, finding content and other complex interactions which are commonly used. In addition, an upgrade to WCAG 2.1 is needed, since the W3C's *before and after demonstration* site only covers WCAG 2.0.

8 ACKNOWLEDGMENT

This work has been supported by the ITIK project funded by the Norwegian Directorate for Children, Youth and Family Affairs in the UnIKT program, grant number 62127.

A RULESET CRITERIA

Table A contains all the 22 criteria that were used in evaluating the different accessibility testing tools. The most significant criteria are marked with a grey background.

Table 7: Overview of criteria

#	*Criteria*	*Description*
1	Automatic testing	Is it possible to run tests automatically from a script?
2	Build integration	Can be integrated in the development/build process
3	Local tests	Can be installed and run locally without external API calls
4	Mature	Is the tool considered stable or in beta?
5	License	License type and name
6	Active development	Is the tool under active development?
7	Community activity	Is the community active?
8	Documentation quality	Is documentation available? What is the quality of the documentation?
9	Flexible integration	Can the tool be integrated with more than one framework or technology?
10	Requirements	Which frameworks and technologies are required?
11	Ruleset	Which rulset(s) are used?
12	Run configuration	Is it possible to include/exclude rules?
13	Output severity options	Is it possible to include/exclude results with varying degree of severity?
14	Output formating options	Is it possible to specify different output formats?
15	Login capabilties	Is it possible to test pages that require login?
16	Complicated setup	How much work is required to setup the tool?
17	Performance	How long does it take to perform a test on a webpage with average complexity?
18	Supports testing of frames	Is it possible ot perform test of frames and it's content?
19	Disrupts process	Will an error in the test stop the process it is run within?
20	Expandable rule set	Is it possible to code and add new rules?
21	CSS coverage	Does the tool test for possible errors in css files?
22	Code inspection	Does the tool provide indication of where issues are located in the source code?
23	Environment	What environment is supported?

References

[1] K. S. Fuglerud, *Inclusive design of ICT: The challenge of diversity.* Dissertation for the Degree of PhD, University of Oslo, Faculty of Humanitites, 2014.

[2] D. Swallow, C. Power, H. Petrie, A. Bramwell-Dicks, L. Buykx, C. A. Velasco, A. Parr, and J. O. Connor, "Speaking the language of web developers: Evaluation of a web accessibility information resource (WebAIR)," in *Lecture Notes in Computer Science*, vol. 8547 LNCS, 2014.

[3] S. Horton and D. Sloan, "Accessibility in Practice: A Process-Driven Approach to Accessibility," in *Inclusive Designing.* Cham: Springer International Publishing, 2014, pp. 105–115.

[4] M. Baez and F. Casati, "Agile development for vulnerable populations: Lessons learned and recommendations," in *Proceedings of the 40th International Conference on Software Engineering: Software Engineering in Society.* ACM, 2018, pp. 33–36.

[5] M. Crabb, M. J. Heron, R. Jones, M. Armstrong, H. Reid, and A. Wilson, "Developing accessible services: understanding current knowledge and areas for future support." 2019.

[6] C. Putnam, M. Dahman, E. Rose, J. Cheng, and G. Bradford, "Best practices for teaching accessibility in university classrooms: cultivating awareness, understanding, and appreciation for diverse users," *ACM Transactions on Accessible Computing (TACCESS)*, vol. 8, no. 4, p. 13, 2016.

[7] T. Halbach and W. Lyszkiewicz, "Accessibility checkers for the web: How reliable are they, actually?" in *Proceedings of the 14th International Conference WWW/Internet*, vol. 2015, 2015, pp. 3–10.

[8] G. Brajnik, "Comparing accessibility evaluation tools: a method for tool effectiveness," *Universal Access in the Information Society*, vol. 3, no. 3-4, pp. 252–263, 2004.

[9] S. Trewin, B. Cragun, C. Swart, J. Brezin, and J. Richards, "Accessibility challenges and tool features: an ibm web developer perspective," in *Proceedings of the 2010 international cross disciplinary conference on web accessibility (W4A)*. ACM, 2010, p. 32.

[10] Gov.uk. How do automated accessibility checkers compare? [Online]. Available: https://alphagov.github.io/accessibility-tool-audit/

[11] H. Petrie, N. King, C. Velasco, H. Gappa, and G. Nordbrock, "The usability of accessibility evaluation tools," in *International Conference on Universal Access in Human-Computer Interaction*. Springer, 2007, pp. 124–132.

[12] C. L. Vera, in *The 2018 ICT Accessibility Testing Symposium: Mobile Testing, 508 Revision, and Beyond*, Arlington, VA, USA.

[13] M.-L. Leitner, C. Strauss, and C. Stummer, "Web accessibility implementation in private sector organizations: motivations and business impact," *Universal Access in the Information Society*, vol. 15, pp. 249–260, 2016.

[14] M. L. Sánchez-Gordón and L. Moreno, "Toward an integration of web accessibility into testing processes," in *Procedia Computer Science*, vol. 27, 2013.

[15] A. Bai, K. Fuglerud, R. Skjerve, and T. Halbach, "Categorization and comparison of accessibility testing methods for software development." *Studies in health technology and informatics*, vol. 256, pp. 821–831, 2018.

[16] H. L. Antonelli, L. Sensiate, W. M. Watanabe, and R. P. de Mattos Fortes, "Challenges of automatically evaluating rich internet applications accessibility," in *Proceedings of the 37th ACM International Conference on the Design of Communication*. ACM, 2019, p. 32.

[17] Web accessibility evaluation tools list. [Online]. Available: https://www.w3.org/WAI/ER/tools/

[18] Automated accessibility testing tool (aatt). [Online]. Available: https://github.com/paypal/AATT

[19] Selecting web accessibility evaluation tools. [Online]. Available: https://www.w3.org/WAI/test-evaluate/tools/selecting/

[20] G. Brajnik, "Automatic web usability evaluation: what needs to be done," in *Proc. Human Factors and the Web, 6th Conference*, 2000.

[21] D. Herron, *Node. js web development: server-side development with Node 10 made easy*. Packt Publishing Ltd, 2018.

[22] Measuring web framework popularity so you can find interesting frameworks to check out. [Online]. Available: https://hotframeworks.com/

[23] Deque. axe-core ruleset. [Online]. Available: https://github.com/dequelabs/axe-core

[24] Squiz. Html codesniffer. [Online]. Available: https://squizlabs.github.io/HTML_CodeSniffer/

[25] Before and after demonstration. [Online]. Available: https://www.w3.org/WAI/demos/bad/Overview.html

[26] W3C. Web accessibility initiative. [Online]. Available: https://www.w3.org/WAI/

[27] ——. Wcag 2.1. [Online]. Available: https://www.w3.org/TR/WCAG21/

[28] Deque. Axe-core. [Online]. Available: https://axe-core.org/

[29] T. Pa11y. pa11y. [Online]. Available: http://pa11y.org/

[30] D. L. Olson and D. Delen, *Advanced data mining techniques*. Springer Science & Business Media, 2008.

[31] W3C. Using id and headers attributes to associate data cells with header cells in data tables. [Online]. Available: https://www.w3.org/TR/WCAG20-TECHS/H43.html

[32] M. Tollefsen and T. Ausland, "A practitioner's approach to using wcag evaluation tools," in *2017 6th International Conference on Information and Communication Technology and Accessibility (ICTA)*. IEEE, 2017, pp. 1–5.

[33] M. Shahin, M. A. Babar, and L. Zhu, "Continuous integration, delivery and deployment: a systematic review on approaches, tools, challenges and practices," *IEEE Access*, vol. 5, pp. 3909–3943, 2017.

[34] T. Savor, M. Douglas, M. Gentili, L. Williams, K. Beck, and M. Stumm, "Continuous deployment at facebook and oanda," in *2016 IEEE/ACM 38th International Conference on Software Engineering Companion (ICSE-C)*. IEEE, 2016, pp. 21–30.

[35] T. Halbach and K. S. Fuglerud, "Reflections on cost-benefit analyses concerning universal design of ICT solutions," in *Proceedings of 10th International Conference on Interfaces and Human Computer Interaction*. IADIS, 2016.

[36] V. Garousi and B. Küçük, "Smells in software test code: A survey of knowledge in industry and academia," *Journal of systems and software*, vol. 138, pp. 52–81, 2018.

[37] W. T. Andrade, R. G. d. Branco, M. I. Cagnin, and D. M. B. Paiva, "Incorporating accessibility elements to the software engineering process," *Advances in Human-Computer Interaction*, vol. 2018, 2018.

[38] S. Schmutz, A. Sonderegger, and J. Sauer, "Implementing Recommendations from Web Accessibility Guidelines: A Comparative Study of Nondisabled Users and Users with Visual Impairments," *Human Factors*, vol. 59, no. 6, pp. 956–972, 2017.

[39] C. M. Law, P. D'Intino, J. Romanowski, and R. Sinclair, "Priorities for the field: Management and Implementation of Testing within Accessibility Programs," in *The 2018 ICT Accessibility Testing Symposium: Mobile Testing, 508 Revision, and Beyond*. Arlington, VA, USA: Accessibility Track Consulting, LLC, 2018, pp. 141–153.

[40] W3C. Web accessibility laws policies. [Online]. Available: http://www.w3.org/WAI/policies/

Autonomous Vehicle Control: End-to-end Learning in Simulated Environments

Hege Haavaldsen* Max Aasbø* Håkon Hukkelås
Frank Lindseth †

Abstract

This paper examines end-to-end learning for autonomous vehicles in diverse, simulated environments containing other vehicles, traffic lights, and traffic signs; in weather conditions ranging from sunny to heavy rain. The paper proposes an architecture combing a traditional Convolutional Neural Network with a recurrent layer to facilitate the learning of both spatial and temporal relationships. Furthermore, the paper suggests a model that supports navigational input from the user to facilitate the use of a global route planner to achieve a more comprehensive system.

The paper also explores some of the uncertainties regarding the implementation of end-to-end systems. Specifically, how a system's overall performance is affected by the size of the training dataset, the allowed prediction frequency, and the number of hidden states in the system's recurrent module. The proposed system is trained using expert driving data captured in various simulated settings and evaluated by its real-time driving performance in unseen simulated environments.

The results of the paper indicate that end-to-end systems can operate autonomously in simulated environments, in a range of different weather conditions. Additionally, it was found that using ten hidden states for the system's recurrent module was optimal. The results further show that the system was sensitive to small reductions in dataset size and that a prediction frequency of 15 Hz was required for the system to perform at its full potential.

1 Introduction

In recent years, substantial progress has been made towards a vehicle's ability to operate autonomously. Primarily, two different approaches have emerged; module-based and end-to-end based methods. The prevailing state of the art approach is to divide the problem into a number of sub-problems and solve them by combining techniques like mapping and localization, perception and prediction, and path planning and control. This approach requires expert knowledge in several domains and often results in complex solutions, consisting of several cooperating modules.

On the other side of the spectrum, we have end-to-end approaches. End-to-end models are designed to utilize minimal expert domain knowledge and instead relies on machine

*These authors contributed equally to this work.

†Corresponding author.

This paper was presented at the NIK-2019 conference; see http://www.nik.no/.

learning to learn a mapping from the perceived environment into a sequence of actions. End-to-end models are, by design, black box methods, making them hard to explain and untrustworthy. However, with the rapid advances in deep learning and the increasing abundance of open-source datasets, end-to-end solutions are showing great potential.

Traditionally, end-to-end models are trained in a supervised manner, studying data from an expert driver, and trying to imitate the experts' actions. Even though the end-to-end model is able to perceive the current environment, it is not able to make the correct navigational decisions solely based on its perception. Therefore, it is necessary to include a user's intent in situations that require a decision (e.g., when approaching an intersection). Hence, we desire that the end-to-end model is able to receive and adapt to navigational commands.

In this work, we investigate end-to-end models' ability to drive autonomously in complex simulated environments. We adapt the model from Haavaldsen *et al.* [1] to manage difficult environments, including complex driving scenarios with multiple agents in the environment, and challenging weather conditions. We present several ablation experiments to quantify the effect of specific design choices typically made when implementing an end-to-end system for autonomous vehicle control: including the number of frames to consider when predicting an action, the importance of the training set size, the number of hidden states in the LSTM module, and the value of prediction frequency. Finally, we perform extensive real-time testing on our best model. [1]

The rest of this paper is organized as follows. Section 2 presents related work, while 3 describes the problem definition of this paper. Section 4 addresses the environment in which the data were collected and where the experiments were conducted. Section 5 reviews the collection and preprocessing of the data. The model architectures are presented in Section 6. Section 7 and 8 covers the experimental setup and results, while section 9 discusses the results. Finally, Section 10 concludes the paper.

2 Related Work

Haavaldsen *et al.* [1] explores end-to-end systems capability to operate in simulated urban environments. Their model is based on the DAVE-2 architecture [2] and is trained in a supervised manner using expert driving data from simulated environments. Their empirical results indicate that the DAVE-2 architecture can significantly improve its autonomous driving capability by including temporal information, with the use of Long-Short-Term-Memory (LSTM) cells. However, their experiments are focused on simple urban environments with little to no variance in weather condition. In this work, we adapt their model. Specifically, we explore the models ability to act autonomously in more challenging environments; highways with several agents, complex urban environments, and a large variance in weather conditions. Furthermore, we perform several ablation experiments to improve upon their proposed model.

End-to-end learning for autonomous vehicles has become increasingly popular over the last decades. Pomerleau *et al.* [3] proposed the first end-to-end model back in 1989, which was a fully-connected neural network. Later on, the first promising proof-of-concept project emerged in 2003, known as DAVE [4]. DAVE was able to fully autonomously drive around a junk-filled alley while avoiding obstacles. Three years later NVIDIA developed DAVE-2 [2], a framework with the objective to make real vehicles drive reliably on public roads. DAVE-2 is the basis for most end-to-end approaches seen

[1] Examples of our model driving can be seen here: `www.tinyurl.com/e2e-example`

today [5, 6]. The project used a CNN to predict a vehicle's steering commands. Their model was able to operate on roads with or without lane markings and other vehicles, as well as parking lots and unpaved roads.

Codevilla *et al.* [5] further explored NVIDIA's architecture by utilizing *Conditional Imitation Learning*, where navigational commands are given as input during training, allowing the model to function in scenarios that require decisions. This is expressed by Equation 1.

$$\underset{\theta}{\text{minimize}} \quad \sum_i L(F(o_i, c_i; \theta), a_i) \tag{1}$$

Here, F is a deep neural network, o_i is an observation, and c_i represents the user's intent. Given the input o_i and c_i, the model predicts an action $\hat{a}_i = F(o_i, c_i; \theta)$. The objective is to optimize a set of parameters θ, such that the difference between the estimated action, $\hat{a}_i$, and the expert's action, a_i, is minimized. By adding c_i, the model should be able to handle more scenarios than before and provide the ability for real-time control of the model. The authors proposed two network architectures: a *branched network* and a *command input network*. The *branched network* used the navigational input as a switch between a CNN and three fully connected networks, each specialized to a single intersection action, while the *command input network* concatenated the navigational command with the output of the CNN, connected to a single fully connected network.

Hubschneider *et al.* [6] proposed using turn signals as control commands to incorporate the steering commands into the network. Furthermore, they proposed a modified network architecture to improve driving accuracy. They used a CNN that receives an image and a turn indicator as input such that the model could be controlled in real-time. To handle sharp turns and obstacles along the road, the authors proposed using images recorded several meters back to obtain a spatial history of the environment. Images captured 4 and 8 meters behind the current position were added as an input to make up for the limited vision from a single centered camera.

Hesham *et al.* [7] proposed a model including temporal dependencies to predict an action. They combine a CNN with an LSTM network to utilize temporal dependencies. Additionally, the authors proposed formulating the steering angle prediction as a classification task, thus imposing a spatial relationship between the output layer neurons in the network. This was done by encoding the output as a sine wave and letting the steering angle correspond to the phase shift. This encoding is given in Equation 2.

$$Y_i = \sin\left(\frac{2\pi(i-1)}{N-1} - \frac{\phi\pi}{2\phi_{max}}\right), \quad 1 \leq i \leq N \tag{2}$$

Y_i is the activation of neuron i, and N is the number of classes of the output layer. They use $N = 95$ neurons and a max steering angle of $\phi_{max} = 190°$. Gradual changes in steering angle will then lead to gradual changes in the output layer's activations. , and their empirical results shows that a C-LSTM improves the angle and stability root mean square error by 35% and 87% on the comma.ai dataset [8] , respectively.

3 Problem Definition

This paper proposes a revised version of the conditional imitation learning from Codevilla *et al.* [5]. To account for both the user's intent and additional world information, the model takes an observation o_i, a user's intent h_i, and external state information s_i as

input in the decision making. h_i is a one-hot encoded vector representing the user's navigational intent, hereby referred to as the navigational input. The navigational input can either be: *Turn Left at Next Intersection*, *Turn Right at Next Intersection*, *Continue Straight at Next Intersection*, or *Follow Lane*. Moreover, additional information about the world, such as the current speed limit, the vehicle's speed, and the current traffic light state, are introduced by s_i, hereby referred to as the external state information. The revised imitation learning technique is expressed in Equation 3, where a model, F, learns a mapping between the inputs (i.e., o_i, h_i, s_i) and the executed action a_i, by fitting the learnable parameters, θ, to minimize the loss, L.

$$\underset{\theta}{\text{minimize}} \quad \sum_i L(F(o_i, h_i, s_i; \theta), a_i) \tag{3}$$

4 Environment

For this paper, the CARLA simulator [9] was used to gather training data and to evaluate the proposed model. CARLA is an open-source simulator built for autonomous driving research and provides an urban driving environment populated with buildings, vehicles, pedestrians, and intersections. The simulator allows customization and control of non-player agents and provides flexible setup of environments: such as type of sensors, number of vehicles, and weather conditions. CARLA comes with a range of different urban and suburban layouts out-of-the-box, differing in size and complexity.

By using a simulator, we could effectively set up a variety of corner cases needed for training, validation, and testing; while removing any safety risks and material costs. However, it is important to recognize that a simulation is only an imitation of a real-world system, and a model trained on only simulated data may not be able to function reliably in the real world. Nonetheless, a model's performance in a photo-realistic simulator serves as a good indication of its real-world performance and may help to illuminate weaknesses early on. Additionally, a simulator serves well for benchmarking different models as you are able to assure that any external factors are constant throughout the tests.

5 Data Generation

When performing imitation learning, the selection-process of training data plays a significant role in a model's ability to perform reliably in different conditions. We collect data from the CARLA simulator, following the collection technique proposed in Haavaldsen *et al.* [1].

Two different datasets were gathered, one from Town 1 and one from Town 4. All data were captured in environments without pedestrians, alongside a varying number of non-player vehicles. Some of the training data were captured entirely without any other vehicles, while some of the data were captured with a randomly generated amount (100-200) of other vehicles.

Data were gathered in seven different weather conditions, at noon and sunset. The different weather conditions consisted of clear sky, cloudy sky, soft rain, medium rain, hard rain, clear wet, and cloudy wet; resulting in a total of 14 different weather/lightning combinations.

The data was captured using a capture frequency of 10 observations each second. In total, 3.4 hours of training data were captured, 2.4 hours in Town 1, and 1.0 hour in Town 4. Table 1 summarizes the gathered datasets.

Town	No. of observations	Size	Duration
1	87 893	39.6 GB	2.4 h
4	33 757	14.1 GB	1.0 h

Table 1: The collected training data. An observation contains the captured data from a single rendered frame in the simulator. Data were captured with a frequency of 10 Hz.

Data augmentation is crucial for our models ability to generalize. Specifically, it is possible to simulate several of the varying environmental conditions using different image transformations. For each observation in the training set, we generate a new augmented sample by using one of the desirable transformations picked at random. These transformations includes: a random change in brightness, random changes in hue, the addition of Gaussian blur, the addition of simulating shadows, and the addition of simulated rain.

Another vital factor in end-to-end learning is the balance of target values within a dataset. A model trained on unbalanced data may incorrectly bias some actions over others. For example, when typically driving, a majority of the observations are the vehicle driving straight. To counteract this, we use a fraction of the examples with a small steering angle. Also, we increase the number of observations with a large steering angle by upsampling. Additionally, the observations corresponding to the different intersection decisions (i.e., turn left, turn right, or straight ahead) were balanced by analyzing the observations' *Navigational Input*-property and downsampling the over-represented choices. Finally, most of the observations where the vehicle was not moving (e.g., waiting for a red light) were downsampled.

6 Model Architecture

We propose a system consisting of two modules, a *Steer Predictor* and a *Throttle-Brake Predictor*. Each module takes a sequence of forward-facing images, navigational inputs, and external state information as input. The images are sent as input to a CNN, where each image is processed independently to extract important features from the images. The output from the CNN is sent to an LSTM alongside navigational inputs and external state information to also learn temporal dependencies between the subsequent images. The system overview is illustrated in Figure 1.

In the *Steer Predictor*-module, the output of the LSTM is passed through a fully connected layer, producing a steer angle prediction encoded as points forming a sine wave. Each of the neurons in the connected layer corresponds to a point on the predicted sine wave. To train the model, we compute the loss between the predicted steer sine wave and the encoded ground truth with the root mean square error (RMSE), and optimize the model with the adam optimizer [10]. In inference time, the steer prediction is decoded back into a steer angle by taking the phase shift of the predicted sine wave. Figure 2 illustrates the training and deployment phase of the steering angle prediction.

In the *Throttle-Brake Predictor*-module, the output of the LSTM is passed through a fully connected layer to predict a throttle and brake value. During training, the loss between the predicted values and the ground truth, are calculated using an MSE loss function, and optimized with the same optimizer as the steer predictor. During inference time, the output from the predictor yields the final throttle and brake predictions.

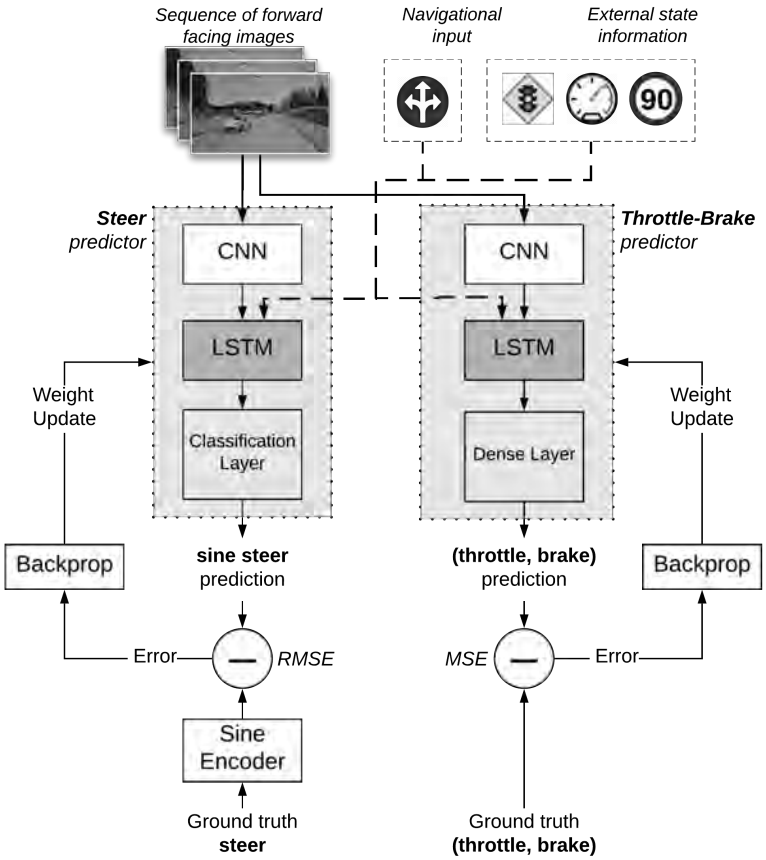


Figure 1: An overview of the propsed system in its training phase.

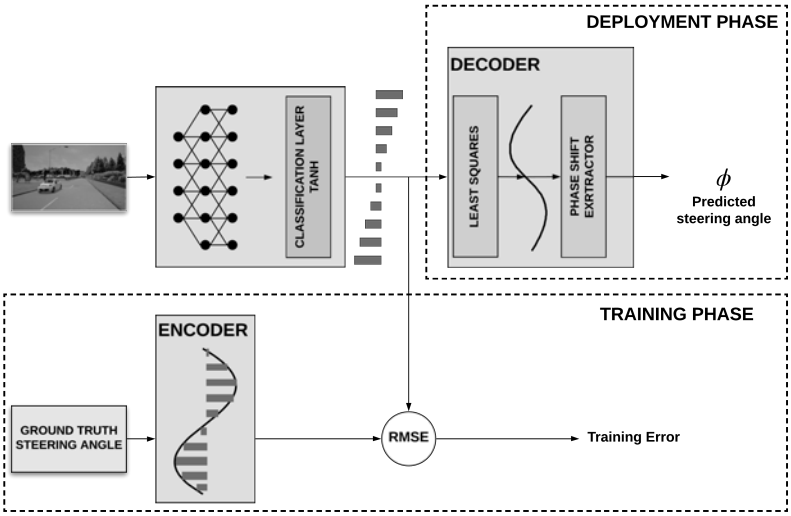


Figure 2: The training and deployment procedure of the steer angle prediction. The procedure follows the same setup as Hesham *et al*. [7]

CNN

Our proposed model uses a CNN to extract useful features from the input image. The architecture is inspired by the architecture used in NVIDIA's DAVE-2 system [2]. The modified network takes a 160x350x3 image as input, followed by a cropping layer and a normalization layer. The cropping layer removes the top 50 pixels from the image, while the normalization layer scales the pixel values between -0.5 and 0.5. Next follows six convolutional layers, all using a ReLU activation function. The first three convolutional layers use a 5x5 filter, while the last three use a 3x3 filter. The first four convolutional layers use a stride of 2, while the last two use a stride of 1. The output of the last convolutional layer is flattened, resulting in a one-dimensional feature vector with 1024 elements.

LSTM

The output from the CNN is concatenated with the current navigational input and external state information, producing a vector of length 1033. The concatenated vector is connected to an LSTM layer with ten hidden states, that uses a sequence of feature extractions over time to produce a control signal. For each time step in the sequence, the LSTM layer sends its output to itself. Finally, at the last time step, the output is forwarded to a dense layer.

Steering Angle Classification

Following the same setup as Hesham *et al.* [7], the *Steer Predictor*-module used a classification layer containing ten neurons. Furthermore, a *tanh* activation is applied to the classification layer allowing the neurons to shape a sine wave with an amplitude of 1. The original steering angle corresponds to the phase shift, ϕ, of the sine wave. During training, the ground truth steering angle is encoded as a sine wave using Equation 2. Y_i is the encoded target value for output neuron i, ϕ is the raw steering angle, and ϕ_{max} is the maximum possible raw steering angle. The loss of the prediction is calculated as the RMSE between the predicted waveform and the encoded ground truth waveform. During deployment, the classification layer's output is decoded back to a steering angle. The decoding is done by fitting the classification layer's output to a sine function and returning its phase shift.

7 Experimental Setup

Real-time tests in CARLA We perform real-time tests by letting a model control a simulated vehicle in completely unseen environments. That is, the model is continuously fed with images from the vehicle's vantage point, while the model's predictions are applied as control signals for the vehicle. The model is to drive five different predefined routes, in five diverse weather conditions comprised of *Clear Noon*, *Clear Sunset*, *Heavy Rain Noon*, *Soft Rain Sunset*, and *Wet Sunset*. Route 1-3, positioned in *Town 2*, tests the model's ability to operate in urban environments containing traffic, intersections, traffic lights and speed limits. Route 4 and 5, positioned in *Town 4*, tests the model's ability to operate on highways and high-speed arias.

Experiment 1: Sequence Length The goal of experiment 1 was to find out how the number of hidden states in the LSTM cells affects the system's performance. Five models

were trained using 1, 5, 10, and 20 hidden states in the LSTM cells. Each trained model was evaluated by their performance from a single run of the real-time tests in CARLA.

Experiment 2: Size of Dataset The goal of experiment 2 was to clarify how the size of the training dataset affects the system's performance. Four models were trained using datasets of varying size. The different datasets all originated from the final, augmented and balanced dataset, but with an increasingly larger portion of the dataset randomly dropped. The models were trained on 20 %, 40 %, 60 %, and 80 % of the original training set. Each trained model was evaluated by their average route completion on a single run of the real-time tests in CARLA.

Experiment 3: Prediction Frequency The goal of experiment 3 was to determine how the model's prediction frequency affects the performance in the real-time tests. The prediction frequency limits how many control-signals the model can produce each second, thus restricting how fast the model is able to react to changes in the perceived environment. The model was exposed to the real-time tests in CARLA five times. For each run, the model's allowed prediction frequency was set to 30 Hz, 15 Hz, 10 Hz, 5 Hz, and 1 Hz, respectively. The models were evaluated by their average route completion.

Experiment 4: Extensive Real-time Tests The goal of experiment 4 was to evaluate the performance of the best model from *Experiment 1* extensively. The model was evaluated by its average performance on ten runs of the real-time tests in CARLA.

8 Experimental Results

Experiment 1: Sequence Length Table 2 shows the model's average route completions for four different sequence lengths in the model's recurrent module. Using a sequence length of 1, 5, 10, and 20, lead to an average route completion of 79.9%, 89.7%, 92.5%, and 76.8%, respectively.

Experiment 2: Size of Dataset Table 3 lists the average route completions for a model using four different training dataset sizes. The model using the original dataset had a route completion of 97.1%. Keeping 80% of the dataset lead to 74.1% average route completion, while further dropping the dataset by 20% lead to a 59.1% average route completion. After that, the performance decrease was less noticeable. Keeping 40% and 20% of the dataset lead to 60.0% and 55.1% average route completions, respectively.

Experiment 3: Prediction Frequency Table 4 shows the average route completions for the model tested with different prediction frequencies. The models using a 30 Hz or 15 Hz prediction frequency yielded the best results with an average route completion of 97.1%. The model with a 10 Hz prediction frequency had a slightly lower route completion of 95.1%. Using a prediction frequency of 5 Hz, 3 Hz, and 1 Hz, resulted in an average route completion of 86.1%, 76.2%, and 27.8%, respectively.

Seq. Length	Avg. Route Compl.
1	79.9%
5	89.7%
10	**92.5%**
20	76.8%

Table 2: The average route completions after using different sequence lengths in the model's recurrent module.

Dataset Size	Avg. Route Compl.
100%	**97.1%**
80%	74.1%
60%	59.1%
40%	60.0%
20%	55.1%

Table 3: The average route after dropping increasingly larger portions of the dataset.

Pred. Freq. [Hz]	Avg. Route Compl.
30	**97.1%**
15	**97.1%**
10	95.1%
5	86.1%
3	76.2%
1	27.8%

Table 4: The average route completions after reducing the model's allowed prediction frequency.

Town	Clear Noon	Clear Sunset	Heavy Rain Noon	Soft Rain Noon	Clear Wet Sunset	Avg.
Town 2	100%	96.5%	80.1%	100%	100%	95.3%
Town 4	100%	100%	83.8%	94.9%	41.3%	84.0%

Table 5: The average route completions in each weather condition from Experiment 4.

Experiment 4: Extensive Real-time Tests Table 5 shows the average route completions in each weather condition over ten real-time tests. The model had an average route completion of 95.3% and 84.0% in Town 2 and Town 4, respectively. Table 6 lists the number of failures per traveled km in Town 2 and Town 4. We divided the failures into three categories: minor, moderate, and severe. Lane touches and sidewalk touches were minor failures, while object collisions, navigational input violation, and rear-end vehicle collisions were considered moderate failures. Severe failures were comprised of front-end vehicle collisions, entering the oncoming lane, and traffic light violations.

In Town 2, the model had the highest number of failures in all categories in *Heavy Rain Noon*, while in Town *Clear Wet Sunset* was the weather condition with the highest frequency of failures.

Parts of experiment 4 were recorded to create a video[2] previewing the system's performance.

	Failures					
Weather Condition	Town 2			Town 4		
	Minor	Moderate	Severe	Minor	Moderate	Severe
Clear Noon	0	0.2	0.2	0.68	0.15	0
Clear Sunset	0.41	0	0.21	0.61	0.68	0
Heavy Rain Noon	**3.49**	**3.99**	**2.00**	0.63	0.54	0
Soft Rain Noon	1.2	0.8	0	0.24	0.08	0.08
Clear Wet Sunset	1.8	1.6	0	**1.47**	**1.28**	**0.73**
Total	1.30	1.21	0.42	0.63	0.44	0.09

Table 6: Experiment 4 – Number of failures per traveled km in Town 2 and Town 4.

[2]A preview of the overall performance can be seen here: `www.tinyurl.com/e2e-example`

9 Discussion

Experiment 1: Sequence Length

The results from *Experiement 1* showed that a small sequence length could lead to faster reactions. However, the network could only learn short term temporal dependencies and was consequently more susceptible for individually misclassified frames. A longer sequence length, on the other hand, led to smoother behavior, and thus more stable steering predictions. Still, it could also increase the probability of learning wrong long-term temporal dependencies. Overall, the system performed most reliably when using a sequence length of 10.

Experiment 2: Size of Dataset

The results from the second experiment showed that a small reduction of dataset size impacted the performance negatively. Reducing the dataset by 20% and 40% resulted in a performance decrease of 24% and 39% respectively. However, reducing the dataset further did not result in any drastic performance decrease. The average route completion stayed approximately the same after reducing the dataset by 40%, 60%, and 80%.

The results also suggest that the system were more sensitive to changes in dataset size in complex weather conditions. The average route completion in *Clear Noon* and *Clear Sunset* were only decreased by 22.5% after reducing the dataset by 80%. In *Sof Rain Noon* and *Clear Wet Sunset*, the same reduction resulted in a decrease in average route completion of 43.5%. In *Heavy Rain*, the average route completion was decreased by 92.3%.

Experiment 3: Prediction Frequency

The experimental results revealed that the reduction of prediction frequency from 30 Hz to 15 Hz did not result in any performance decrease. Both models were able to achieve an average of 97.1% route completion. When further reducing the prediction frequency to 10 Hz, a small performance reduction of 2% was observed. However, reducing the frequency to less than 10 Hz resulted in a significant performance decreases. Using a prediction frequency of 5 Hz, 3 Hz, and 1 Hz respectively, resulted in an 11%, 20.9%, and 69.3% worse performance than the best model. Furthermore, the results showed that the system is more sensitive to reduction of prediction frequencies in complex weather conditions.

Experiment 4: Extensive Real-time Test

Town 2 - Urban Environment Regardless of weather conditions, the system always stopped at red lights, never ignored a navigational input, and followed the current speed limit. Moreover, it performed stable lane following and always tried to position itself in the center of the lane. When drifting out of the lane, the system corrected its trajectory both quickly and smoothly, never experiencing any oscillating steering predictions. In situations where the system had to decrease its speed, it tried to let go of the throttle first. If that was insufficient, the system applied the brakes as well.

The results indicate that *Hard Rain Noon* was the most challenging weather condition. The system displayed a smooth driving behavior on straight stretches, but the steering was uneven in turns. The system had trouble with detecting cars in front of it, mainly black and dark grey cars. The light conditions were poor in *Hard Rain Noon*, which made the dark-colored cars blend more in with the environment.

Town 4 - Highway Regardless of weather conditions, the system always stopped at red lights and followed the current speed limit. The system adjusted its distance to the car in front of it by adjusting the throttle and was able to handle traffic congestions in both low-speed and high-speed areas.

Clear Wet Sunset turned out to be the most challenging weather condition in *Town 4*, and the system had its lowest average route completion in this weather condition. The sunlight reflected off the wet parts in the road, making it especially challenging to detect the lane lines. As a result, the system had more trouble with staying in the lane and drove out of the road four times throughout the experiment. Additionally, the system experienced four object collisions, two rear-endings, and one route deviation.

10 Conclusion

In this paper, we explore an end-to-end system's ability to drive autonomously in diverse, simulated environments. We propose a system architecture consisting of two modules: a *Steer Predictor* and a *Throttle-Brake Predictor*. Each module combines a traditional CNN, inspired by NVIDIA's DAVE-2 system [2], with an LSTM layer to facilitate the learning of both spatial and temporal relationships. The system is trained using expert driving data from various simulated settings and we evaluate our proposed model by presenting extensive real-time experiments in unseen simulated environments.

Our proposed model is able to operate successfully in urban environments containing other vehicles, speed limits, and traffic-light regulated intersections. The system always stopped at red lights, always tried to execute the navigational input, and followed the current speed limit. On highways, the real-time tests demonstrated that the system was able to perform stable lane following in most weather conditions and that it was capable of handling traffic congestions in both low-speed and high-speed areas.

The system is able to operate in a range of different weather conditions. It excelled in dry weather during the daytime, making few mistakes. In more challenging weather conditions, like soft rain, the system was more susceptible to failures, which could lead to the occasional rear-end collision. In even harsher weather conditions, such as heavy rain, the system was more likely to perform severe failures, such as entering the oncoming lane or front-end collisions. Nonetheless, the system was able to navigate remarkably well in harsh weather conditions where even humans could struggle.

In this paper, we also aimed to explore some of the uncertainties regarding the implementation of end-to-end systems. The experimental results showed that the system was sensitive to small reductions in dataset size. However, the system was able to operate somewhat successfully even with a severely reduced dataset. Furthermore, the results suggest that the system performed to its full potential when using a prediction frequency of 15 Hz or higher. Finally, it was found that using ten hidden states for the system's recurrent module was optimal for the overall performance.

Even though the system experienced several severe failures during testing, our real-time tests reflects great potential in using end-to-end systems to achieve fully autonomous vehicles.

References

[1] Hege Haavaldsen, Max Aasboe, and Frank Lindseth. Autonomous vehicle control: End-to-end learning in simulated urban environments, 2019.

[2] Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D. Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, Xin Zhang, Jake Zhao, and Karol Zieba. End to end learning for self-driving cars. *CoRR*, abs/1604.07316, 2016.

[3] Dean A. Pomerleau. Advances in neural information processing systems 1. chapter ALVINN: An Autonomous Land Vehicle in a Neural Network, pages 305–313. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1989.

[4] Y. Lecun, E. Cosatto, J. Ben, U. Muller, and B. Flepp. Dave: Autonomous off-road vehicle control using end-to-end learning. Technical Report DARPA-IPTO Final Report, Courant Institute/CBLL, http://www.cs.nyu.edu/˜yann/research/dave/index.html, 2004.

[5] Felipe Codevilla, Matthias Müller, Alexey Dosovitskiy, Antonio López, and Vladlen Koltun. End-to-end driving via conditional imitation learning. *CoRR*, abs/1710.02410, 2017.

[6] C. Hubschneider, A. Bauer, M. Weber, and J. M. Zöllner. Adding navigation to the equation: Turning decisions for end-to-end vehicle control. In *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*, pages 1–8, Oct 2017.

[7] Hesham M. Eraqi, Mohamed N. Moustafa, and Jens Honer. End-to-end deep learning for steering autonomous vehicles considering temporal dependencies. *CoRR*, abs/1710.03804, 2017.

[8] Eder Santana and George Hotz. Learning a driving simulator. *CoRR*, abs/1608.01230, 2016.

[9] Alexey Dosovitskiy, Germán Ros, Felipe Codevilla, Antonio López, and Vladlen Koltun. CARLA: an open urban driving simulator. *CoRR*, abs/1711.03938, 2017.

[10] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.

Blending functions based on trigonometric and polynomial approximations of the Fabius function

Hans Olofsen
UiT The Arctic University of Norway in Narvik
hans.olofsen@uit.no

Abstract

Most simple blending functions are polynomials, while more advanced blending functions are, for example, rational or expo-rational fractions. The Fabius function has the required properties of a blending function, but is a nowhere analytic function and cannot be calculated exactly everywhere on the required domain. We present a new set of trigonometric and polynomial blending functions with the shape and other properties similar to the Fabius function. They consist of combinations of trigonometric polynomials and piecewise polynomials. The main advanced of these are that they are easy to implement, have low processing costs and have simple derivatives. This makes them very suitable for the calculation of splines. Due to the self-differential property of the Fabius function, scaled versions of these functions can even be used to approximate their own derivatives.

1 Introduction

In computer graphics, interpolation and approximation of points, curves and surfaces have always been an important topic. Many of the techniques used involve blending. A blending function is used to blend data or functions, and smoothness of the result increases with the order of the blending function.

For decades, trigonometric polynomials have been used for approximation. This has especially been the case for approximating periodic functions using Fourier series. Yet there has been little focus on using them use as blending functions, where the domain is $[0,1]$. This might be due to the fact that using a limited number of coefficients from the Fourier series in trigonometric polynomials does not result into a valid blending function where the endpoints have exact values of 0 and 1.

The Fabius function has the properties of blending functions, is C^∞-smooth and is therefore an interesting candidate for the use as a blending function. Unfortunately it is nowhere analytic and cannot be calculated exactly on the fully required domain. Using trigonometric series, the Fabius function can be approximated while still adhering to the properties of blending functions. The self-differentiability of the Fabius function can be used to find derivatives of the approximate functions, but also to find better approximate functions.

We will now first look at blending functions and the Fabius function, before we start looking at the approximate functions.

This paper was presented at the NIK-2019 conference; see http://www.nik.no/.

2 Preliminaries

Blending functions

Blending functions, or B-functions, are used for blending two other functions, e.g. for the interpolation of curves and surfaces. How exactly this is done is beyond the scope of this paper. For most applications, a (point) symmetric blending function is required. For simplicity, whenever a blending function is called symmetric, it implies that the function is point symmetric about the center point $\left(\frac{1}{2},\frac{1}{2}\right)$. As a consequence, all odd-order derivatives are line symmetric about the vertical center line $t=\frac{1}{2}$, while all even-order derivatives are point symmetric over the middle point $\left(\frac{1}{2},\frac{1}{2}\right)$.

Definition 2.1. A symmetric B-function has the properties, described in [1]:

- $B(t)\colon\ [0,1]\to[0,1]$ (P1)
- Has fixed endpoints: $B(0)=0$ and $B(1)=1$ (P2)
- Is continuous and monotone: $B'(t)\geq 0$ (P3)
- Is (point) symmetric: $B(1-t)=1-B(t)$ (P4)
- Has a Hermite order $\mathsf{S}\geq 0: B^{(j)}(0)=B^{(j)}(1)=0\,, j=1,\ldots,\mathsf{S}$ (P5)

Besides blending the values of the two functions, blending the derivatives of the two function is important for smooth results. The Hermite order determines up to which order derivatives the blending is interpolated. The simplest function $B(t)=t$ can be used to give a continuous result, but does not provide any smoothness at the border: $\mathsf{S}=0$.

The Fabius function

The Fabius function[2] is an infinitely differentiable, but nowhere analytic function that is self-differential:

$$\mathfrak{F}'(x)=2\,\mathfrak{F}(2x)\ ,\ x\in[0,\infty) \tag{2}$$

Figure 1 shows the Fabius function and its first 3 derivatives.

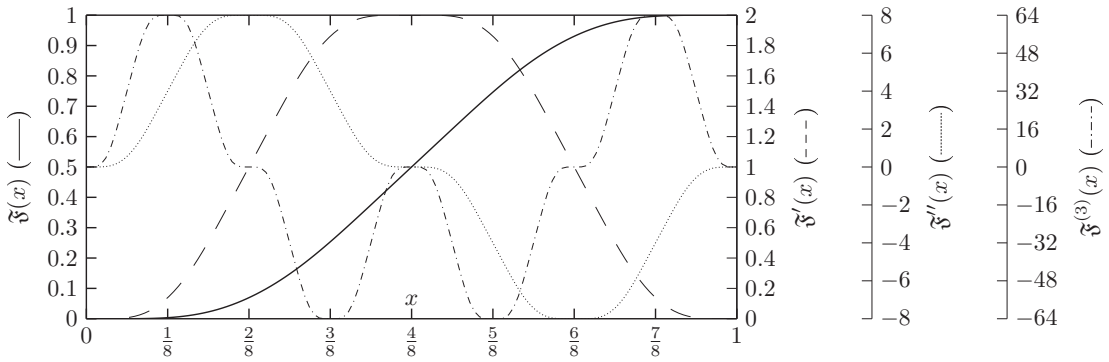


Figure 1: The Fabius function (——) and its first 3 derivatives (– – , ····· and – · –). Note the ranges of the four different y-axes.

On the domain $x\in[0,1]$, the Fabius function is point symmetric about $\left(\frac{1}{2},\frac{1}{2}\right)$. This makes it possible to limit the domain for evaluation of the Fabius to $x\in[0,1]$, by using the

"unextended definition", which is the piecewise definition

$$\mathfrak{F}(x) = \begin{cases} \int_0^{2x} \mathfrak{F}(\tau)\ \mathrm{d}\tau\ , & \text{if } x \in \left[0, \frac{1}{2}\right]\ , \quad (3a) \\ \int_{2x-1}^{1} \mathfrak{F}(\tau)\ \mathrm{d}\tau\ +\ 2x{-}1\ , & \text{if } x \in \left(\frac{1}{2}, 1\right]\ . \quad (3b) \end{cases}$$

In the domain $[0,1]$, the Fabius function has the basic properties for symmetric blending functions (P1) – (P5) and therefore it can be used as a blending function. It is infinitely smooth C^∞ and when used for blending, the resulting curve or surface will be C^∞-smooth as well. However, it cannot be calculated exactly (except for positive dyadic rational numbers, numbers of the form $\frac{m}{2^n}$, [3]) and therefore needs to be approximated.

3 Approximating the Fabius function

In order to find approximations $\mathcal{F}$ of the Fabius function, we have several options. Since the purpose of these approximate functions is the use as blending functions, they should adhere to the basic properties for symmetric blending functions (P1) – (P5). We will therefore use t as the argument of these functions. We can find function with a shape that is similar to the shape of the Fabius function. We can then also adjust the shape of the function such that its derivatives are similar to the derivatives of the Fabius function. We will show how this can be done with trigonometric polynomials.

We will also show how an approximation $\mathcal{F}(t)$ can be transformed into a better approximation $\widehat{\mathcal{F}}(t)$, and this process can be repeated in order to get better approximations, though at a computational cost.

Approximating with a sum of weighted cosines

A sum of weighted cosines can approximate the Fabius function on the domain $[0,2]$, similar to [4, eq.(30)], by the following trigonometric polynomials:

$$\mathfrak{F}(t) \approx \mathcal{F}_{\mathsf{M}}(t) = \frac{1}{2} - \sum_{m=1}^{\mathsf{M}} c_m \cos(\pi m t) \quad (4)$$

Due to point symmetry (P4), it follows that $c_m = 0$ for m is even. We decide that M has only values that are rounded up to the nearest even number, and we do so consistently such that we can later use the value of M to calculate the Hermite order. M is then twice the number of non-zero c_m coefficients which is equal to the number of constraints.

From (P2) (endpoints) it follows that $\sum c_m = \frac{1}{2}$ which is the first constraint. As a consequence, the most simple trigonometric approximation, with one non-zero c_m is:

$$\mathcal{F}_{\mathsf{M}=2}(t) = \frac{1}{2} - \frac{1}{2}\cos(\pi t) \quad (5)$$

Increasing the Hermite order

The Hermite order (P5) of (5) is then $\mathsf{S} = 1$. With $\mathsf{S} = \Delta\mathsf{S} - 1$, the Hermite order can be increased by $\Delta\mathsf{S}$ in steps of 2 by solving equations such that one additional derivative at

the endpoints vanishes. This results in a larger number of non-zero values for c_m. The following sets of equations need to be solved for $\Delta\mathsf{S}=2$ and $\Delta\mathsf{S}=4$:

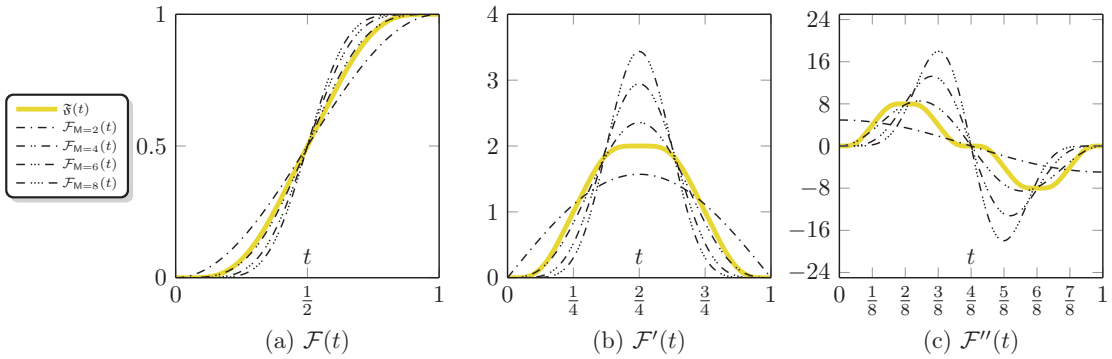
$$(\Delta\mathsf{S}=2)\quad c_{1,3}:\begin{cases} c_1+\quad c_3=\frac{1}{2} \\ 1^2c_1+3^2c_3=0 \end{cases};\tag{6}$$

$$(\Delta\mathsf{S}=4)\quad c_{1,3,5}:\begin{cases} c_1+\quad c_3+\quad c_5=\frac{1}{2} \\ 1^2c_1+3^2c_3+5^2c_5=0 \\ 1^4c_1+3^4c_3+5^4c_5=0 \end{cases}.\tag{7}$$

For $\Delta\mathsf{S}=2$, the solutions are $c_1=\frac{9}{16}$ and $c_3=-\frac{1}{16}$. We then get the following approximate function

$$\mathcal{F}_{\mathsf{M}=4}(t)=\frac{1}{2}-\left(\frac{9}{16}\cos(\pi t)+-\frac{1}{16}\cos(3\pi t)\right).\tag{8}$$

In Figure 2 we see that both $\Delta\mathsf{S}=2$ and $\Delta\mathsf{S}=4$ clearly give a better approximation, but higher values cause a convergence to a step function (e.g. Heaviside: $\mathcal{H}(t-\frac{1}{2})$) rather than to the Fabius function.



(a) $\mathcal{F}(t)$ (b) $\mathcal{F}'(t)$ (c) $\mathcal{F}''(t)$

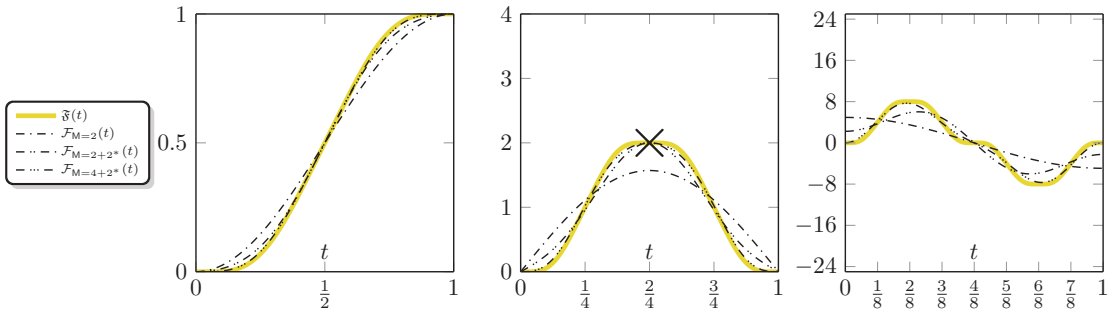
Figure 2: Increasing hermite order S by increasing the number of c_m with constraints. A higher $\Delta\mathsf{S}$ does not necessarily make a better approximation.

Other constraints

Increasing the Hermite order might be desired, but this alone does not improve the approximation. This can be solved by adding constraints for the derivatives at certain (dyadic rational) positions. These (and possibly other constraints) increase M, and this will be written as: $\mathsf{M}=\Delta\mathsf{S}+\mathsf{M}^*$, where M^* does not increase the Hermite order.
In the center of Figure 3 we can see that the second derivate also vanishes, but the third does not. This could be improved by adding more constraints, but requires even more values for c_m which requires more calculations. For $t\in\left[0,\frac{1}{2}\right]$, we see that $\mathfrak{F}''(t)$ is symmetric about $t=\frac{1}{4}$. However, the approximations with (4) are not. This is solved by using the antiderivatives as explained below.

Approximating by using the antiderivatives

So far we have focused on values of the Fabius function and its derivatives, which is, essentially, interpolating those. However, we can also focus on the main property of the



(a) $\mathcal{F}(t)$ (b) $\mathcal{F}'(t)$ (c) $\mathcal{F}''(t)$

Figure 3: A better approximation can be achieved by finding values for c_m such that $\mathcal{F}'(\frac{1}{2}) = \sum c_m \pi (-1)^{\frac{m-1}{2}}$ $m = 2$ (marked with $\times$).

Fabius function that defines its shape, namely the self-differential property. Suppose $\mathcal{F}(t)$ is an approximation to the Fabius function $\mathfrak{F}(t)$, with error $\mathcal{E}(t)$:

$$\mathcal{F}(t) = \mathfrak{F}(t) + \mathcal{E}(t)\,, \tag{9}$$

then, similar to (3), we define $\widehat{\mathcal{F}}(t)$ as a "new" approximation with error $\widehat{\mathcal{E}}(t)$:

$$\widehat{\mathcal{F}}(t) = \begin{cases} \displaystyle\int_0^{2t} \mathcal{F}(\tau)\,\mathrm{d}\tau\,, & \text{if } t \in \left[0, \frac{1}{2}\right]; \\ \displaystyle\int_{2t-1}^{1} \mathcal{F}(\tau)\,\mathrm{d}\tau + 2t - 1\,, & \text{if } t \in \left[\frac{1}{2}, 1\right], \end{cases} \tag{10a}$$

$$= \mathfrak{F}(t) + \begin{cases} \displaystyle\int_0^{2t} \mathcal{E}(\tau)\,\mathrm{d}\tau\,, & \text{if } t \in \left[0, \frac{1}{2}\right]; \\ \displaystyle\int_{2t-1}^{1} \mathcal{E}(\tau)\,\mathrm{d}\tau\,, & \text{if } t \in \left[\frac{1}{2}, 1\right], \end{cases} \tag{10b}$$

$$= \mathfrak{F}(t) + \widehat{\mathcal{E}}(t)\,, \tag{10c}$$

where $\widehat{\mathcal{E}}(t) = -\widehat{\mathcal{E}}(1-t)$ is the error of the new approximation:

$$\widehat{\mathcal{E}}(t) = \begin{cases} \displaystyle\int_0^{2t} \mathcal{E}(\tau)\,\mathrm{d}\tau\,, & \text{if } t \in \left[0, \frac{1}{2}\right]; \\ \displaystyle\int_{2t-1}^{1} \mathcal{E}(\tau)\,\mathrm{d}\tau\,, & \text{if } t \in \left[\frac{1}{2}, 1\right]. \end{cases} \tag{11}$$

In Figure 4 and 5 the error $\mathcal{E}(t)$, its definite integral $\int_0^t \mathcal{E}(\tau)\,\mathrm{d}\tau$ and the new error $\widehat{\mathcal{E}}(t)$ are shown. We can see that the new approximation is better, since both the average and the maximum of the error become smaller. This is true because the width of the domain $[0,1]$

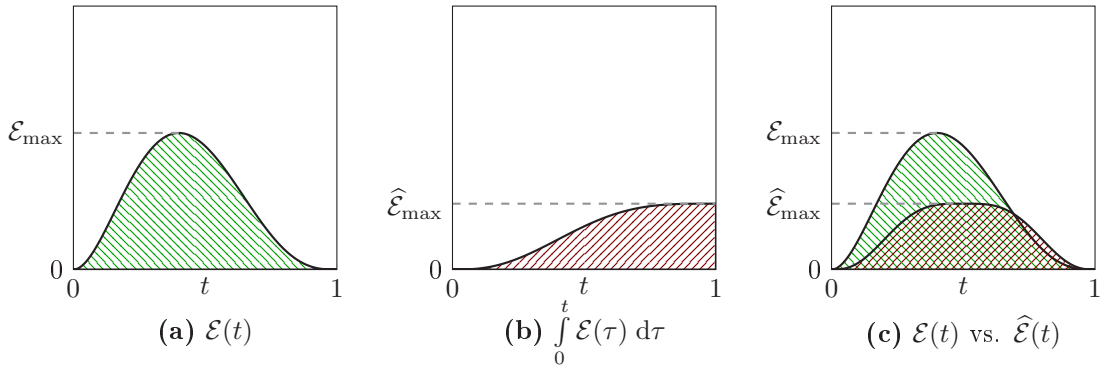


Figure 4: An example where the error of an approximation to the Fabius function doesn't change sign on the domain $[0,1]$. For $\mathsf{P} > 1$ it shows that $\widehat{\mathcal{E}}_{max}$ is about twice ass small as $\mathcal{E}_{\max}$.

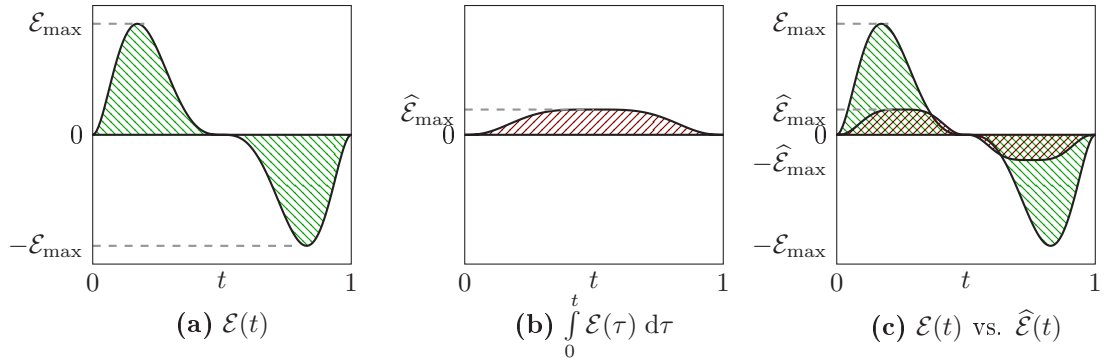


Figure 5: An example where the error of an approximation to the Fabius function changes sign at $t = \frac{1}{2}$. This is the case for all approximations discussed in this paper. For $\mathsf{P} > 1$ it shows that $\widehat{\mathcal{E}}_{max}$ is about 4 times smaller than $\mathcal{E}_{\max}$.

is not more than 1. Summarized this is the following inequation:

$$\mathcal{E}_{\max} = \max_{0<t<1} \mathcal{E}(t) > \int_0^1 \mathcal{E}(\tau)\,\mathrm{d}\tau = \widehat{\mathcal{E}}(1) = \widehat{\mathcal{E}}_{max} \tag{12}$$

In order to get a better approximation, we can then replace $\mathcal{F}$ by $\widehat{\mathcal{F}}$ and repeat this process. This process can be repeated in order to get an even better approximation, where P is the number of times:

$$\mathcal{F}_{\mathsf{P}}(t) = \widehat{\mathcal{F}}_{\mathsf{P}-1}(t)\ ,\ \text{for}\ \mathsf{P} > 1\ . \tag{13}$$

It should be noted that (12) is not true when $\mathcal{E}(t)$ is constant. However if the process is repeated, it will be true as $\widehat{\mathcal{E}}(t)$ won't be constant.

The cost of each improvement is more complex functions due to piecewise definitions. It might seem that each time, the number of sub-functions should double, but fortunately this is not always the case as we will see later. Now that we have a way to make better approximations, we should find a way to start with a first approximation. It should be noted that, in order to find a valid blending function, it is not necessary to start with a valid blending function as long as the process is repeated until it results in a valid blending function. The only requirement are that $\int_0^1 \mathcal{F}(\tau)\,\mathrm{d}\tau = \frac{1}{2}$ and that the antiderivative is chosen such that $\widehat{\mathcal{F}}(0) = 0$ and $\widehat{\mathcal{F}}(t)$ is continuous. For example, it is possible to start

with

$$\mathcal{F}_{\mathsf{P}=0}(t) = \frac{1}{2}\,, \tag{14}$$

which clearly is not a valid blending function since it does not satisfy (P2), and, one could say in this context, has "Hermite order -1". However, after one improvement step, we get the following function which is a valid blending function:

$$\mathcal{F}_{\substack{\mathsf{P}=1\\ \mathsf{M}=0}}(t) = \begin{cases} t\,, & \text{if } t \in \left[0, \frac{1}{2}\right]; \\ 1 - t \;+\; 2t - 1\,, & \text{if } t \in \left[\frac{1}{2}, 1\right], \end{cases} = t\,. \tag{15}$$

We see that the two sub-functions are that same, and therefore the number of sub-function doesn't double in this case.

In Figure 6 the approximations for $\mathsf{P} = 1, \ldots, 5$ are shown, as well as the maximum errors $\mathcal{E}_{\max}$ for $\mathsf{P} = 1, \ldots, 6$. After several iterations of using the proper antiderivative, the resulting piecewise polynomial shape approximates the Fabius function quite well. Figure 6 (b) shows the two sub-functions, separated by bullets, of the approximation

$$\mathcal{F}_{\mathsf{P}=2}(t) = \begin{cases} 2t^2\,, & \text{if } t \in \left[0, \frac{1}{2}\right]; \\ 1 - 2(1-t)^2\,, & \text{if } t \in \left[\frac{1}{2}, 1\right], \end{cases} \tag{16}$$

They are are not the same, and therefore here the number of sub-function doubles. As a consequence, the function is no longer C^∞-smooth as summarized in (23).

$$\mathcal{F}_{\mathsf{P}=3}(t) = \begin{cases} \displaystyle\int_0^{2t} 2\tau^2 \,\mathrm{d}\tau\,, & \text{if } t \in \left[0, \frac{1}{4}\right]; \\ \displaystyle\int_0^{1/2} 2\tau^2 \,\mathrm{d}\tau + \int_{1/2}^{2t} 1 - 2(1-\tau)^2 \,\mathrm{d}\tau\,, & \text{if } t \in \left[\frac{1}{4}, \frac{1}{2}\right]; \\ \displaystyle\int_{2t-1}^{1/2} 2\tau^2 \,\mathrm{d}\tau + \int_{1/2}^{1} 1 - 2(1-\tau)^2 \,\mathrm{d}\tau \;+\; 2t - 1\,, & \text{if } t \in \left[\frac{1}{2}, \frac{3}{4}\right]; \\ \displaystyle\int_{2t-1}^{1} 1 - 2(1-\tau)^2 \,\mathrm{d}\tau \;+\; 2t - 1\,, & \text{if } t \in \left[\frac{3}{4}, 1\right], \end{cases} \tag{17a}$$

but, again, the to sub-functions in the middle are the same:

$$\cdots = \begin{cases} \frac{16}{3} t^3\,, & \text{if } t \in \left[0, \frac{1}{4}\right]; \\ \frac{16}{3}\left(\frac{1}{2} - t\right)^3 - \frac{1}{2} + 2t\,, & \text{if } t \in \left[\frac{1}{4}, \frac{3}{4}\right]; \\ 1 - \frac{16}{3}(1-t)^3\,, & \text{if } t \in \left[\frac{3}{4}, 1\right], \end{cases} \tag{17b}$$

The sub-functions are slightly rewritten such that they show the symmetry. Figure 6 (c) shows these three sub-functions.

Due to point symmetry, it is clear that the sub-functions in the middle are the same for odd values of P, which is the case for odd polynomials. The number of pieces for $\mathsf{P} = 0, 1, 2, \ldots$ are $1, 1, 2, 3, 6, 11, 22, 43, \ldots$ which is sequence A005578 in the *On-Line Encyclopedia of Integer Sequences*[5]. One can verify that the number of pieces converges to $\frac{2^{\mathsf{P}}}{3}$.

The combination of these

A better approximation can be achieved by starting first with (4) and then use the antiderivatives. This way one gets a combination, namely the sum of a polynomial and a trigonometric polynomial. The result is a non-periodic function, however we are only interested in the domain $[0, 1]$.

The same way as in (15), since $\cos(\pi m t)$ is symmetric in $t = 1$, we get two sub-functions that are the same. As a consequence the functions $\mathcal{F}_{\substack{\mathsf{P}=1\\\mathsf{M}}}(t)$ simplify easily:

$$\mathcal{F}_{\substack{\mathsf{P}=1\\\mathsf{M}}}(t) = \mathcal{F}_{\substack{\mathsf{P}=1\\\mathsf{M}=0}}(t) - \begin{cases} \displaystyle\int_0^{2t} \sum_{m=1}^{\mathsf{M}} c_m \cos(\pi m \tau)\,\mathrm{d}\tau, & \text{if } t \in \left[0, \frac{1}{2}\right]; \\ \displaystyle\int_{2t-1}^{1} \sum_{m=1}^{\mathsf{M}} c_m \cos(\pi m \tau)\,\mathrm{d}\tau, & \text{if } t \in \left[\frac{1}{2}, 1\right], \end{cases} \tag{18}$$

$$= t - \sum_{m=1}^{\mathsf{M}} \frac{c_m}{\pi m} \sin(2\pi m t) \tag{19}$$

Figure 7 shows $\mathsf{P} = 0, \ldots, 3$ for increased Hermite order $\mathsf{M} = 2, 4$. Figure 8 shows $\mathsf{P} = 0, \ldots, 3$ with the additional $\mathcal{F}'(\frac{1}{2}) = 2$ constraint for $\mathsf{M} = 2 + 2^*, 4 + 2^*$.

4 Results

We now have several approximations of the Fabius function available that can be used for blending. Table 1 shows the maximum difference between these functions and the Fabius function. One can see that every increment of P causes an up to about 4 times lower maximum error. For blending in practice, we do not have to minimize this difference, and we can choose approximation that are good enough. For $\mathsf{P} > 0$ this maximum difference is at $t = \frac{1}{4}$ where $\mathfrak{F}\left(\frac{1}{4}\right) = \frac{5}{72} \approx 6.9 \times 10^{-2}$ as evaluated in [3]. Take notice of $\mathcal{F}_{\substack{\mathsf{P}=1\\\mathsf{M}=0}}(t) = t$ which is the simplest B-function.

The Hermite order of the functions presented here is:

$$\mathsf{S} = \mathsf{P} + \Delta\mathsf{S} - 1. \tag{20}$$

When a high Hermite order is not required, and speed and simplicity are priorities, some of the simplest are the C^∞-smooth single-piece functions

$$\mathcal{F}_{\substack{\mathsf{P}=0\\\mathsf{M}=2}}(t) = \frac{1}{2} - \frac{1}{2}\cos(\pi t) \qquad \text{(5 revisited)}$$

and

$$\mathcal{F}_{\substack{\mathsf{P}=1\\\mathsf{M}=2}}(t) = t - \frac{\sin(2\pi t)}{2\pi} \tag{21}$$

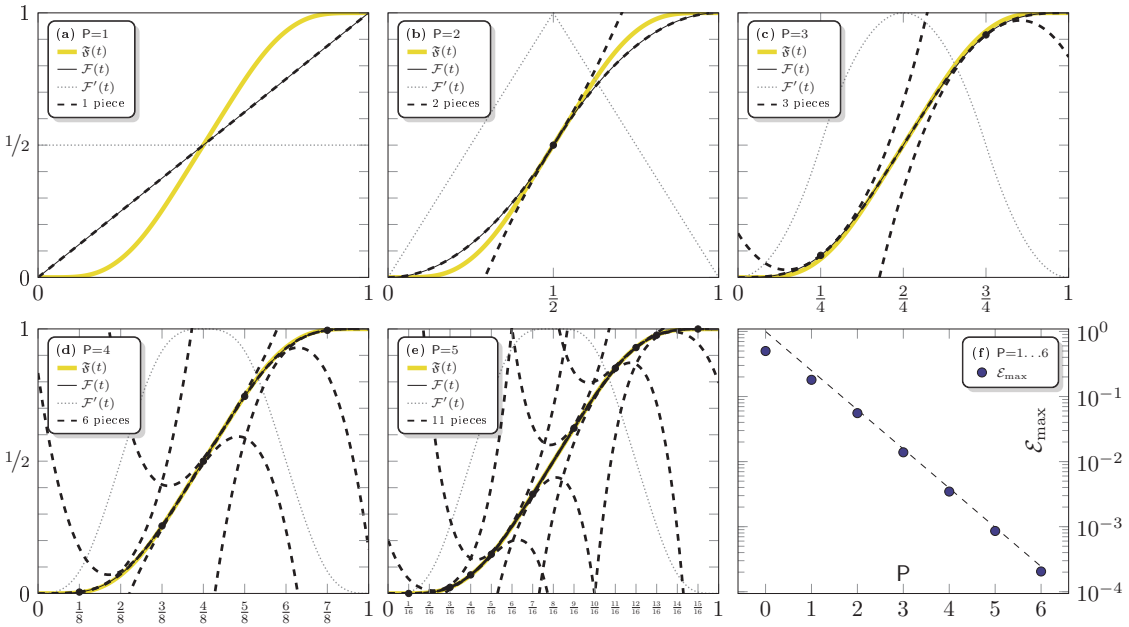


Figure 6: (a)-(e): Approximations for $\mathsf{P} = 1,\ldots,5$ (f): the maximum errors $\mathcal{E}_{\max}$ for $\mathsf{P} = 1,\ldots,6$.

In this figure and in Figure 7 and 8, the sub-functions (- - -) of the piecewise-defined functions are plotted for the whole domain; the first derivatives (·····) are scaled by a factor of 2.

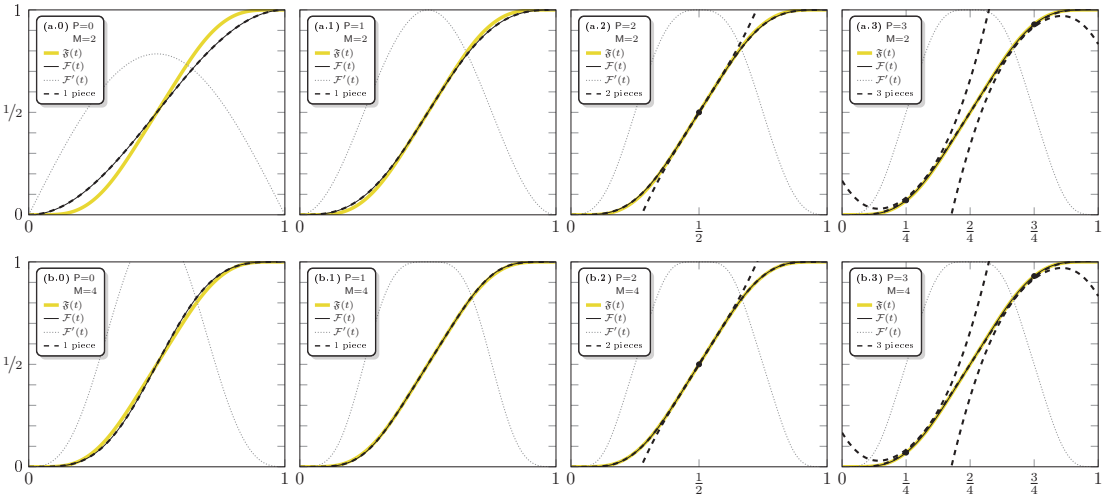


Figure 7: Approximations for $\mathsf{P} = 0,\ldots,3$; $\mathsf{M} = 2,4$

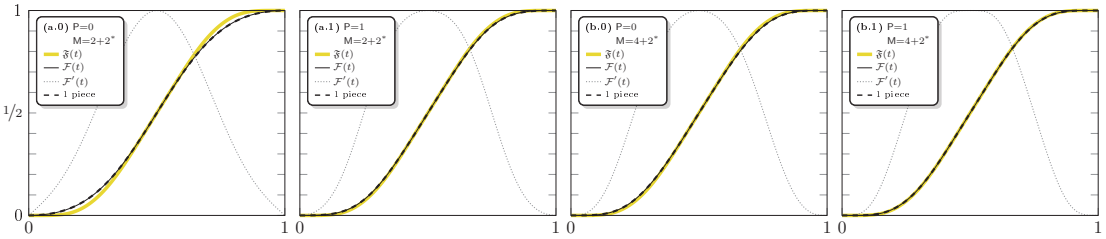


Figure 8: Approximations for $\mathsf{P} = 0,1$; $\mathsf{M} = 2+2^*,4+2^*$

P	M = 0	M = 2	M = 4	M = 2+2*	M = 4+2*
0	(not a valid B-function)	$7.8{\cdot}10^{-2}$	$2.6{\cdot}10^{-2}$	$2.9{\cdot}10^{-2}$	$9.1{\cdot}10^{-3}$
1	$1.8{\cdot}10^{-1}$	$2.1{\cdot}10^{-2}$	$5.1{\cdot}10^{-3}$	$6.9{\cdot}10^{-3}$	$4.3{\cdot}10^{-3}$
2	$5.6{\cdot}10^{-2}$	$4.9{\cdot}10^{-3}$	$7.4{\cdot}10^{-4}$	$1.8{\cdot}10^{-3}$	$4.8{\cdot}10^{-4}$
3	$1.4{\cdot}10^{-2}$	$1.2{\cdot}10^{-3}$	$1.9{\cdot}10^{-4}$	$4.4{\cdot}10^{-4}$	$1.1{\cdot}10^{-4}$
4	$3.5{\cdot}10^{-3}$	(not calculated)	(not calculated)	(not calculated)	(not calculated)
5	$8.6{\cdot}10^{-4}$	(not calculated)	(not calculated)	(not calculated)	(not calculated)
6	$2.0{\cdot}10^{-4}$	(not calculated)	(not calculated)	(not calculated)	(not calculated)

Table 1: Maximum absolute difference $\mathcal{E}_{\max}$ between a few approximated Fabius functions and the Fabius function. In the top row: $\mathsf{M} = \Delta\mathsf{S} + \mathsf{M}^*$

with $\mathsf{S} = 1$ and $\mathsf{S} = 2$, as well as the piecewise-defined polynomial function

$$\mathcal{F}_{\substack{\mathsf{P}=2\\ \mathsf{M}=0}}(t) = \begin{cases} 2x^2, & \text{if } t \in \left[0, \frac{1}{2}\right]; \\ 1-2(1-x)^2, & \text{if } t \in \left[\frac{1}{2}, 1\right], \end{cases} \tag{22}$$

which is only C^1 smooth, because its first derivative is not differentiable at $t = \frac{1}{2}$. Other approximations can be chosen depending on requirements for the Hermite order, smoothness class, as well as implementation specific calculation optimizations such as parallelization.

Smoothness The smoothness class of the functions depends on P, since for $\mathsf{P} > 1$ piecewise defined functions are required and is:

$$\mathsf{C} = \begin{cases} \mathsf{C}^\infty, & \text{if } \mathsf{P} \leq 1; \\ \mathsf{C}^{\mathsf{P}-1}, & \text{if } \mathsf{P} > 1. \end{cases} \tag{23}$$

The sub-functions that define the pieces of the piecewise functions are always C^∞-smooth.

5 Applications

To illustrate how easy it is to implement all these functions, all figures so far in this paper have been calculated in TikZ, instead of using external tools or a precalculated data file.

While we haven't done any comparative benchmarking on these function, it is clear that the implementation is straight forward. Both functions (5) and (22) require only three basic arithmetic operations and one trigonometric function call/lookup that most platforms have implemented in hardware. No further interpolation is required. For functions with $\mathsf{P} > 1$ the speed will depend on the platform's branching mechanism.

An application: Fabius B-splines

The (approximations of) the Fabius function can be used in B-splines. In this case the linear translation and scaling function

$$w_{d,i}(t) = \begin{cases} \dfrac{t - t_i}{t_{i+d} - t_i}, & \text{if } t_i \leq t < t_{i+d} \\ 0, & \text{otherwise} \end{cases} \tag{24}$$

is replaced by $B \circ w_{d,i}(t)$ where the B-function B in this case is the Fabius function or one of the approximate functions $\mathcal{F}$.

Figure 9 shows the Cubic B-spline basis functions for normal polynomial B-splines, for Fabius B-splines where the Fabius function is used as the B-function B, as well as approximations (5) and (21). The curves made with these splines have a high smoothness, increased by the Hermite order S of the function $\mathcal{F}$, and even C^∞-smooth in case of Fabius B-splines.

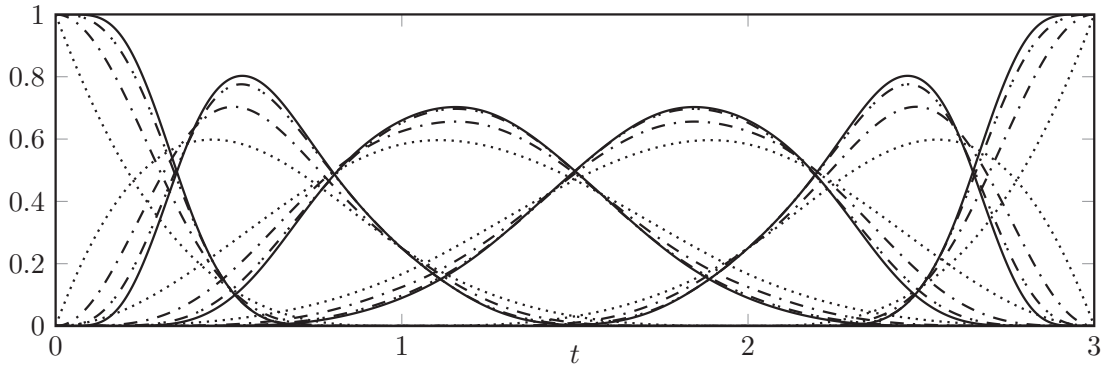


Figure 9: The Cubic B-spline basis functions for four different B-functions on knot vector $\vec{t} = \{0,0,0,0,1,2,3,3,3,3\}$.
····· Normal polynomial B-splines: $B(t) = t$
– · – B-splines using $B(t) = \mathcal{F}_{\substack{\mathsf{P}=0\\ \mathsf{M}=2}}(t) = \frac{1}{2} - \frac{1}{2}\cos(\pi t)$ from (5).
– - – B-splines using $B(t) = \mathcal{F}_{\substack{\mathsf{P}=1\\ \mathsf{M}=2}}(t) = t - \frac{\sin(2\pi t)}{2\pi}$ from (21).
—— Fabius B-splines: $B(t) = \mathfrak{F}(t)$.

Approximation of the derivatives

In most applications the derivatives on the blending function are needed. The smoothness of the approximate functions is determined by their differentiability. The derivative of a C^k-smooth function is C^{k-1}-smooth and as such the Hermite order of the derivative of a blending function is also lower then the Hermite order of that blending function. The derivative of an approximate Fabius function is in general easy to calculate, as it consists of only polynomials and trigonometric polynomials. But it is also a (piecewise) function where the sub-functions are scaled and mirrored versions of the approximate function with $\mathsf{P}-1$. As an example, compare the $\wedge$-shape of the derivative $\mathcal{F}'_{\mathsf{P}=2}(t)$ in Figure 6 (b) to the /-shape of $\mathcal{F}_{\mathsf{P}=1}(t)$ in Figure 6 (a). This means that the derivatives are worse approximations than the approximate functions.

However, since the functions are only approximations, the true derivatives of the functions are not so interesting. To approximate the derivatives, we can use the self-differentiability of the Fabius function. We define the *Fabius-derivative* $\widetilde{\mathfrak{D}}$ of a function $f(x)$ on the domain $[0,1]$ as:

$$\widetilde{\mathfrak{D}}\, f(x) = \begin{cases} 2f(2x), & \text{if } x \in \left[0, \frac{1}{2}\right]; \\ 2f(2-2x), & \text{if } x \in \left[\frac{1}{2}, 1\right]. \end{cases} \tag{25}$$

If f is C^a-smooth and has Hermite order $\mathsf{S} = b$ (at $x = 1$), then $\overset{\smile}{\mathfrak{D}} f(x)$ is $\mathsf{C}^{\min(a,b)}$-smooth and has Hermite order $\mathsf{S} = b$. It follows that higher order Fabius-derivatives have the same smoothness and Hermite order as the first order Fabius-derivative.

If one can accept that derivatives are also approximated, one can say that if a blending function $\mathcal{F}(t)$ approximates the Fabius function, its Fabius-derivative approximates the derivative of the Fabius function and $\mathcal{F}(t)$ is approximately C^∞-smooth and has approximately Hermite order $\mathsf{S} = \infty$. For the presented approximate functions this means that

$$\overset{\smile}{\mathfrak{D}}\, \mathcal{F}_{\mathsf{P}}(t) = \mathcal{F}'_{\mathsf{P}+1}(t). \tag{26}$$

Mathematically seen this can be considered as cheating or simply incorrect. However for practical applications this trick can be acceptable and quite useful.

6 Conclusion

We have presented new types of trigonometric and polynomial blending functions with the shape and properties similar to the Fabius function. We did this by showing how to find trigonometric polynomial blending functions with shapes that approximate the Fabius function, including some of their derivatives.

Then, using the self-differentiability of the Fabius function, we have shown how an approximation of the Fabius function can be improved iteratively, while retaining the blending function properties. This resulted in a set of blending functions that can be evaluated fast, because they can be calculated directly without needing further interpolation.

The blending functions have been successfully used in extended B-splines, resulting in smoother splines than normal B-splines.

References

[1] Arne Lakså. Beta-function B-splines and subdivision schemes, a preliminary study. In Ivan Lirkov and Svetozar Margenov, editors, *Large-Scale Scientific Computing*, pages 592–599, Cham, 2018. Springer International Publishing.

[2] J. Fabius. A probabilistic example of a nowhere analytic C^∞-function. *Zeitschrift für Wahrscheinlichkeitstheorie und Verwandte Gebiete*, 5:173–174, 1966.

[3] J. K. Haugland. Evaluating the Fabius function. *ArXiv e-prints*, September 2016.

[4] J. Arias de Reyna Martínez. Definition and study of an infinitely differentiable function with compact support. *Rev. Real Acad. Cienc. Exact. Fís. Natur. Madrid*, 76(1):21–38, 1982.

[5] OEIS Foundation Inc. The On-Line encyclopedia of integer sequences. A005578, 2019. [Online; accessed 2019-08-01].

Improved color edge preserving smoothing

Ali Alsam and Hans Jakob Rivertz
Norwegian University of Science and Technology
Trondheim, Norway

Abstract

An edge preserving smoothing algorithm is presented. To prevent smoothing over edges, the algorithm requires that diffusion in a direction should not result in an increase in neighbouring gradients. Intuitively, we think that smoothing reduces gradients. This is, however, not true in the vicinity of strong edges where smoothing over the edge results in surface deformation. The possibility of reducing the calculation cost is explored by reducing the frequency of calculating the edge strength.

1 Introduction

The theory of image formation states that a scene recorded by a camera is noise free. Noise is added during acquisitions due to the limitations in the electronics, optics and the intensity of the light that can be focused on the surface of the sensor. Filtering noise from an image is, thus, a fundamental operation that is incorporated in all digital cameras and image processing software.

Real images are formally defined as the sum of two functions: one is camera noise, the other is an idealised noise free surface. Based on this definition, the challenge is to devise algorithms that filter the real image to remove noise without degrading the underlying original image.

Filtering is the process of replacing the value of an image at a certain location with the result of applying a function to the values in the same neighbourhood. Examples of noise reduction filters that comply to this definition are the mean, median, Gaussian and the Weiner filter [4, 6].

The introduction of partial differential equations to cast image noise filtering as an anisotropic diffusion process where flow from one location to another is allowed along edges but not across them, [7] led to a wealth of follow up methods with landmarks algorithms such as the use of level set methods [9] and total variation (TV) [8]. Although these methods are effective and mathematically elegant, they are criticised for being iterative, slow and inefficient.

The literature on image denoising is vast, including important methods such as bilateral filtering [10], method-guided filter [5] and non-local means [3]. The focus in this paper, however, is on iterative diffusion methods. Specifically, we wish to examine improving the efficiency an algorithm by Alsam and Rivertz [2]. We present a reformulation the method to reduce complexity, resolve color artefacts. Further we

This paper was presented at the NIK-2019 conference; see http://www.nik.no/.

explore reducing the per-iteration calculation of the edge strength which is a prerequisite to the diffusion process.

The Alsam and Rivertz method [2], requires that blurring should always result in decreased gradients. Further, edge degradation is shown to be the result of increasing gradients in the vicinity of an edge. This observation that can readily be understood by applying an averaging filter to a block function where the result would be an increase as well as a decrease in gradients. To stop gradient increase, diffusion from a given pixel to its neighbour is permitted *if and only if* it does not result in increasing the neighbouring gradients. In other words, diffusion in the north direction is permitted if the result does not cause an increase in the south, east, west and diagonal gradients.

We start by presenting an efficient formulation of our previous work [2], we then run experiments on images to show results of the algorithm. Having done that, we limit the admissibility check to a limited number of iteration. Instead of checking the increase in the gradient in each iteration, we check at given intervals.

Not surprisingly, calculating the diffusion direction only once results in a very fast algorithm. Experiments show, however, that better noise reduction is achieved when the edge direction is calculated at a higher frequency. In search for deeper insight, we allowed the diffusion direction to change and tracked the number of directions that were removed or added from the first iteration. This investigation showed that the change drops fast with increased iteration number with more directions removed than added. We found that when the number of iterations is high, e.g. 100, the results of checking the gradient condition every second or fifth iteration resulted in clear noise reduction whilst the quality of the edges were not diminished.

2 Method

A color image is a rectangular array of RGB-values. The RGB-value at the point (x,y) is a vector $\mathbf{I}(x,y) = (R(x,y), G(x,y), B(x,y))$ consisting of the red, green and blue color components. At each point (x,y) we consider a 3-by-3-group of points centred around the point (x,y), which we will call the center point of that group. The remaining 8 points are called neighbours of (x,y). We denote the pixel values in the 3-by-3-group by

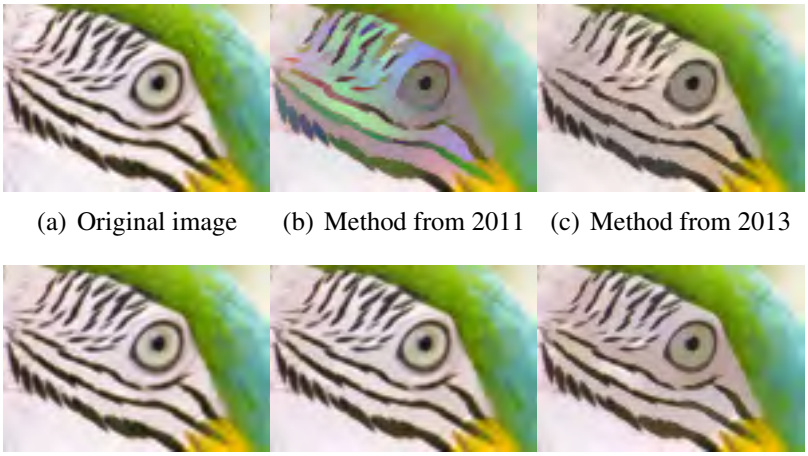
$$\begin{array}{lllllllll} \mathbf{I}_3 & = & \mathbf{I}(i-1,j-1) & \mathbf{I}_2 & = & \mathbf{I}(i,j-1) & \mathbf{I}_1 & = & \mathbf{I}(i+1,j-1) \\ \mathbf{I}_4 & = & \mathbf{I}(i-1,j) & \mathbf{I}_8 & = & \mathbf{I}(i,j) & \mathbf{I}_0 & = & \mathbf{I}(i+1,j) \\ \mathbf{I}_5 & = & \mathbf{I}(i-1,j+1) & \mathbf{I}_6 & = & \mathbf{I}(i,j+1) & \mathbf{I}_7 & = & \mathbf{I}(i+1,j+1) \end{array} . \quad (1)$$

The color differences between the eight neighbours and the center are $\mathbf{P}_j = \mathbf{I}_j - \mathbf{I}_8$, $j = 0,1,\ldots,7$.

The proposed diffusion process is iterative, at each level, $\mathbf{I}_8$ is replaced by $\mathbf{I}'_8 = \mathbf{I}_8 + \sum_{j=0}^{7} d_j \mathbf{P}_j$. The diffusion level d_j is non-zero only if the diffusion in j'th direction is admissible. Admissibility is determined by a test diffusion in a single direction. Let $\mathbf{I}_8^{(j)} = \mathbf{I}_8 + s\mathbf{P}_j$ be the test diffusion in the j'th direction. We let the size of the test diffusion level s be 0.2. The entire idea is that the test diffusion should not increase the differences $\mathbf{P}_i$ in any direction $i = 0,1,\ldots,7$. After a test diffusion, the differences are $\mathbf{P}_i^{(j)} = \mathbf{I}_i - \mathbf{I}_8^{(j)} = \mathbf{P}_i - s\mathbf{P}_j$. We require that diffusion in the j'th direction is admissible if and only if $|\mathbf{P}_i^{(j)}|^2 - |\mathbf{P}_i|^2 < s\alpha$ for all $i = 0,1,\ldots,7$. $|\mathbf{P}|$ is the 2-norm of $\mathbf{P}$ and α is a threshold.

Calculations gives the following equivalent condition for admissibility.

$$s|\mathbf{P}_j|^2 - 2\mathbf{P}_j \cdot \mathbf{P}_i < \alpha,$$



(a) Original image (b) Method from 2011 (c) Method from 2013

(d) New method $\alpha = 5$ (e) New method $\alpha = 10$ (f) New method $\alpha = 25$

Figure 1: The figure shows the results after 1000 iterations with different methods. The color artifacts with the method in [1] disappeared in the method from 2013 [2] and in the new method. The images (d-f) shows that the present method is very stable even after 1000 iterations. Higher value of α results in more smoothing. Edges are preserved in (d-f)

for all $i = 0, 1, \ldots, 7$. If diffusion is admissible in a vertical or horizontal direction, we let $d_j = d$ for the respective j. Otherwise, we let $d_j = d/\sqrt{2}$.

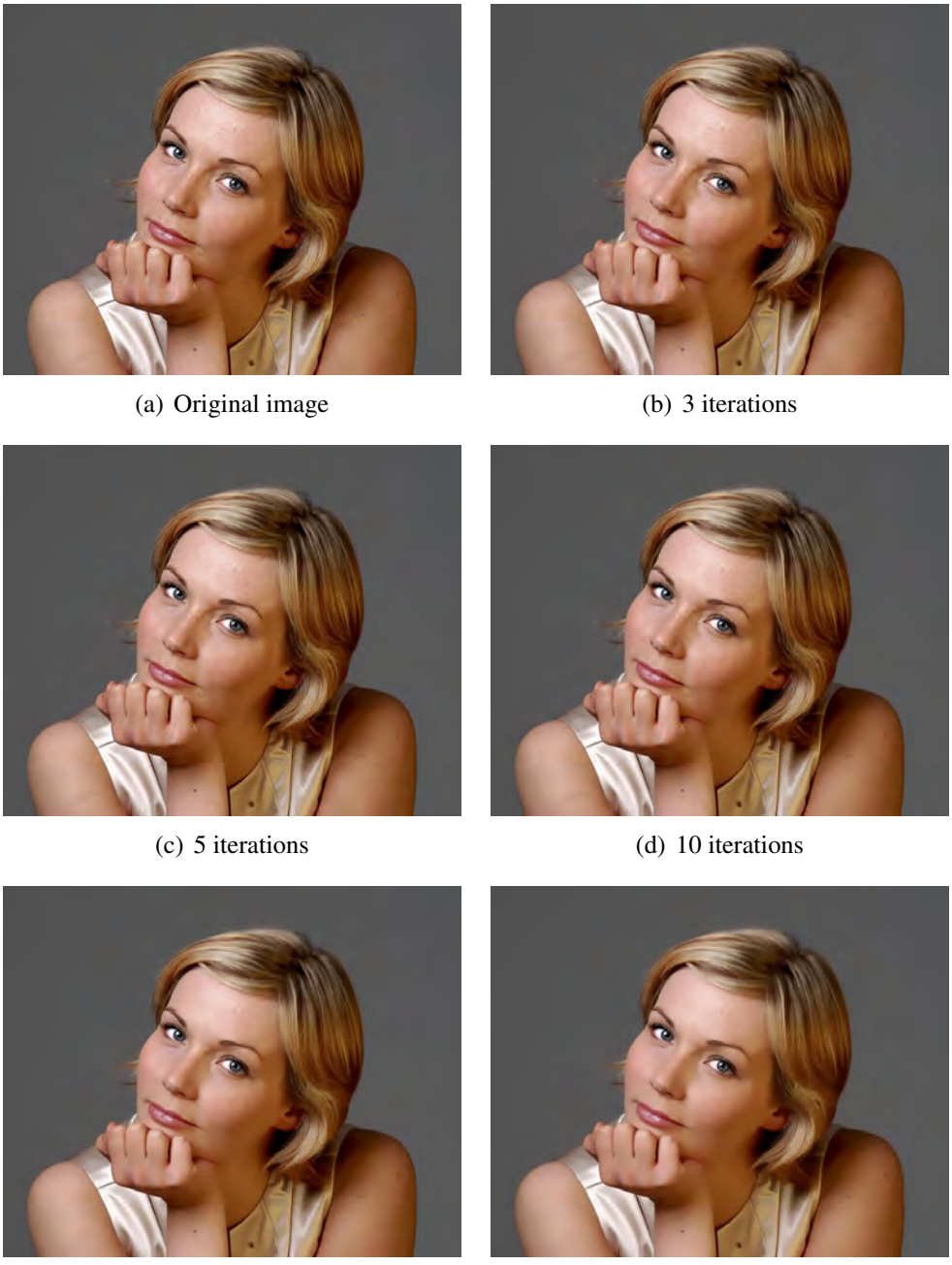
3 Results

We start the results section by a visual comparison between the current formulation and our previous work. In figure 1, we show the results based on one thousand iterations. Here we note that the method from 2011 had clear colour bleeding issues which were resolved in the later versions of the algorithm. In the 2013 work, the reduction of color smear was achieved by means of projections along the hue lines, i.e more calculation. As is evident from the previous section no projection is used in the current work.

In figures 2 and 3, we present the results of the algorithm on two images with 5, 10, 50 and 100. Here we observes that the images are smoothed, fine noise is removed and edges are preserved. These results vary with the chosen value of the threshold where more smoothing can be achieved with a higher value.

The dynamics of the iteration process

Experiments on the original photo in figure 4 shows that the number of obstructed diffusions differs throughout the diffusion process. Table 1 shows that the there are between 2 million and 2.5 million obstructions in each step. That is between 63% and 75% of all possible diffusions. We observe that there is a certain dynamics process where some obstructions are added in each step, but more are removed. Experiments were also done with 100 and 1000 iteration steps. There was a steady decrease of obstructions as well as the dynamics of the obstructions. An experiment where the obstructions from the first iteration were used throughout showed much less noise reduction. Figure 4 compare the results from the experiment with the dynamically changing constraints and static constraints. A visual inspection shows that the edges are equally preserved.



(a) Original image (b) 3 iterations

(c) 5 iterations (d) 10 iterations

(e) 50 iterations (f) 100 iterations

Figure 2: The figures show the original picture (a) and several numbers of iterations in (b-f). The diffusion level in each step is 0.1 up and down and 0.0707 in the diagonal directions. The threshold is set to $\alpha = 25$.

(a) Original image (b) 3 iterations

(c) 5 iterations (d) 10 iterations

(e) 50 iterations (f) 100 iterations

Figure 3: The figures show the original picture (a) and several numbers of iterations in (b-f). The diffusion level in each step is 0.1 up and down and 0.0707 in the diagonal directions. The threshold is set to $\alpha = 10$.

(a) Original image

(b) Static rule. 10 iterations

(c) Dynamic rule. 10 iterations

(d) Static rule. 100 iterations

(e) Dynamic rule. 1000 iterations

Figure 4: The above images show (a) the original picture of the parrots, (b) the result with a static constraint calculated in the first iteration, and (c) the result where the constraint is calculated in every iteration. The images (d) and (e) shows the results with 100 respectively 1000 iterations and with dynamic constraints. The diffusion level in each step is 0.1 up and down and 0.0707 in the diagonal directions. The threshold is set to $\alpha = 5$.

Iteration	Total	Added	Removed
1	2557435	2557435	0
2	2497360	194811	254886
3	2393305	97082	201137
4	2311478	59030	140857
5	2254723	41341	98096
6	2215126	31762	71359
7	2186976	26519	54669
8	2164778	22750	44948
9	2146259	20190	38709
10	2130197	18093	34155

Table 1: The table shows how many constraints that are added and removed in each step when processing the parrot picture. The value of α was five and the size of the image is 797×531.

4 Discussion

An algorithm to remove image noise without smearing the edges is presented. The core idea is a condition that allows diffusion in certain directions only if it does not cause and increase in the local gradients. Results after many iterations show clear edge color preservation. Experiments that need to be expanded show that it is possible to iteratively diffuse an image without having to calculate the edge strength at each step. This idea needs to be studied more with a larger set of images. Building upon those experiments, further exploration of the dynamics of admissibility as well as comparison with other diffusion methods are what is needed to be done in future work.

References

[1] ALSAM, A., AND RIVERTZ, H. J. Fast edge preserving smoothing algorithm. In *Proceedings of the 13th IASTED International Conference on Signal and Image Processing* (2011), ACTA Press, pp. 8–12.

[2] ALSAM, A., AND RIVERTZ, H. J. Color edge preserving smoothing. In *Advances in Visual Computing, 9th International Symposium, ISVC 2013* (2013), Springer-Verlag Berlin Heidelberg, pp. 79–86.

[3] BUADES, A., COLL, B., AND MOREL, J.-M. A review of image denoising algorithms, with a new one. *Multiscale Modeling & Simulation 4*, 2 (2005), 490–530.

[4] GONZALEZ, R. C., AND WOODS, R. E. Digital image processing, chapter 11, representation and description, 2006.

[5] HE, K., SUN, J., AND TANG, X. Guided image filtering. *IEEE transactions on pattern analysis and machine intelligence 35*, 6 (2013), 1397–1409.

[6] JAIN, A. K. *Fundamentals of digital image processing*. Englewood Cliffs, NJ: Prentice Hall, 1989.

[7] PERONA, P., AND MALIK, J. Scale-space and edge detection using anisotropic diffusion. *IEEE Transactions on Pattern Analysis and Machine Intelligence 12* (1990), 629–639.

[8] RUDIN, L. I., OSHER, S., AND FATEMI, E. Nonlinear total variation based noise removal algorithms. *Physica D: nonlinear phenomena 60*, 1-4 (1992), 259–268.

[9] SETHIAN, J. A. *Level set methods and fast marching methods: evolving interfaces in computational geometry, fluid mechanics, computer vision, and materials science*, vol. 3. Cambridge university press, 1999.

[10] TOMASI, C., AND MANDUCHI, R. Bilateral filtering for gray and color images. In *Iccv* (1998), vol. 98, p. 2.

Cloud-based Implementation and Validation of a Predictive Fire Risk Indication Model

Sindre Stokkenes[1], Lars M. Kristensen[2], and Torgrim Log[3]

[1,2]Department of Computer science, Electrical engineering and Mathematical sciences
[3]Department of Safety, Chemistry, and Biomedical laboratory sciences
Western Norway University of Applied Sciences

Abstract

The high representation of wooden houses in Norwegian cities combined with periods of dry and cold climate during the winter time often results in a high risk of severe fires. This makes it important for public authorities and fire departments to have an accurate estimate of the current fire risk in order to take proper precautions. We report on the implementation of a predictive mathematical model based on first order principles which exploits cloud-provided measurements from weather stations and weather forecasts from the Norwegian Meteorological Institute to predict the current and future fire risk at a given geographical location. We have experimentally validated the model during the winter 2018-2019 at selected geographical locations, and by considering weather data from the time of several historical fires. Our results show that our cloud and web-based implementation is both time and storage efficient, and capable of being able to accurately predict the fire risk measured in terms of the estimated time to flashover. The paper demonstrates that our methodology in the near future may become a valuable risk predicting tool for Norwegian fire brigades.

1 Introduction

The large amount of forest in Norway has provided the foundation for a very long tradition of constructing houses using wood as the main material. These houses can be extremely susceptible to fire under weather conditions that typically occur during the winter time when there are long periods of dry and cold weather. When the air gets colder and drier, the water concentration in the wood decreases [17], meaning there is a high probability of the wood to catch fire. As identified by Log [12], it is the period during December - January when the weather is usually cold that have the highest frequency of fires in Norway. The very severe fire in Lærdalsøyri January 18-19 2014, resulting in the loss of more buildings in a single conflagration since 1923, may serve as an example [5, 11].

This paper was presented at the NIK-2019 conference; see http://www.nik.no/.

The long-term aim of the work presented in this paper is to address the research question as to whether there is a way to reduce the impact of fires by means of an early warning system that can for example warn the local fire brigade when a high fire risk is expected in the coming days. This would enable them to stay alert and be better prepared in case a fire incident occurs [14]

The first contribution of this paper is to investigate whether the predictive model developed by Log [12, 13] for estimating the relative humidity in wooden structures based on first order principles can be used in combination with weather data measurements and forecast data to obtain a reliable and useful fire risk prediction. In this work, we use the estimated time to complete flashover as an indication of fire risk. The time to flashover is in practice in the order of minutes. To be useful, our predictions should be within similar order of magnitude in terms of time and be consistent with observed weather conditions [10]. A second contribution is to show how the fire risk prediction can be provided as a cloud-service and be implemented based on cloud-services. We have implemented the fire risk indication prototype as a distributed application based on the architectural principles of micro-services [3] and REST [16]. For the implementation of the fire risk prediction service we have used the Spark/Java framework [18], and deployed the application on the Amazon EC2 platform. For data storage, we rely on the MongoDB noSQL database deployed in the Azure cloud platform. As the data source for meteorological data we rely on the Frost[15] and MET [9] REST web-services of the Norwegian Meteorological Institute (MET) providing data from high-end weather stations and weather forecast data. In addition to these professional meteorological services, we have also investigated the consumption of meteorological data from Netatmo consumer-grade weather stations.

The reminder of this paper is organised as follows. In Section 2 we briefly outline the predictive fire risk indication model which has served as a basis for our investigations. Section 3 explains how we have implemented a software prototype by aggregating data from external cloud-services to obtain the input data required for the predicative fire risk indication model. In Section 4 we present selected results of our experimental evaluation. Finally, in Section 5 we provide the conclusions and discuss directions for future work. The paper is based on the master's thesis [19].

2 Predictive Fire Risk Indication Model

The predictive fire risk indication (FRI) model [12] is based on computing the relative indoor humidity of a wooden structure using the measured and/or predicted indicators for the outdoor climate [13]. Obtaining the relative indoor humidity makes it possible to determine the concentration of water in the wood which in turn makes it possible to estimate the time to complete flashover in case of a fire.

The basic observation underlying the model is that as the air gets dryer the wood releases moisture and when the relative humidity increases the wood will absorb moisture. In addition to this, the predictive model takes into account decay periods related to the transport of water in and out of the wood. The decay period gives a delayed effect in terms of fire risk. Due to space limitations, we only outline the theoretical foundation of the FRI model below. Details can be found in [12, 13].

Outdoor Climate. The FRI model requires the outdoor air temperature, outdoor relative humidity, and the outdoor water concentration of the air. The first two

elements are typically measured. The outdoor water concentration can be estimated by calculating the water saturation vapour pressure as:

$$P_{sat} = 610.78 * e^{\frac{17.2694*T_c}{T_c*237.3}} \tag{1}$$

where T_C ($^\circ C$) is the the outdoor air temperature. The outdoor water concentration can now be obtained as:

$$C_{wa} = RH_{out} * (\frac{P_{sat}(T_c) * M_w}{R * (T_c + K)}) \tag{2}$$

where RH_{out} is the outdoor relative humidity, $M_w(0.01801528)(kg/mol)$ is the molecular mass of water, $R(8.314J/Kmol)$ is the molar gas constant, and K is the absolute temperature $273.15K$.

Indoor Climate. The indoor climate is dependent on the outdoor climate [13] in addition to factors such as the humidity released by people, plants and animals inside the house, and also the air change rate. Log found [12, 13] that it is reasonable to assume that 1 kg of moisture is released daily and that older wooden houses have a lower air change rate compared to newer houses. Based on the investigations of Log [12, 13], the concentration of water in the air inside a house with forced ventilation can be computed iteratively using the following formulas:

$$C_{in,0} = RH_{inside} * C_{sat,in} \tag{3}$$

$$C_{in,i} = ((1-\beta) * C_{in,i-1} + \beta * C_{wa} * (\frac{T_{out}}{T_{in}}))) + \frac{m_{wall,loss}}{v} + ms * \frac{\Delta t}{v} \tag{4}$$

where RH_{inside} is a base relative humidity set to 40% as a starting point; $C_{sat,in}$ is found using Eq. 2 without RH and T_c is set to $22^\circ C$; β accounts for the ventilation (air changes per hour), T_{out} and T_{in} is the absolute outdoor and indoor temperature where we set the indoor temperature to an estimated 22 $^\circ C$; m_{wall} is the sum loss of water concentration in the walls, $volume$ is the volume of the room, ms is the moisture supply which is $\frac{1}{24*3600} kg/s$ and Δt is the calculation interval set to 720 s.

The second order partial differential diffusion equation, i.e. analogous to the heat equation, is then solved for the wooden wall layers involved in the moisture transport. The innermost layer is for simplicity treated as a mathematical reflection surface. This is well-founded based on moisture diffusion barrier requirements in Norwegian houses preventing rot formation as a result of wall and thermal insulation cold weather temperature gradients. The Bernoulli equation based air change rate as recommended by Log [13] was used in the modeling. It should be noted that the methodology relies on basic physics, with no empirical constants. Parameters such as the moisture diffusuin coefficient is taken from the literature [1].

With the indoors concentration of water calculated, the relative humidity can be computed using the following equation:

$$RH_{inside} = \frac{C_{in}}{C_{sat,in}} \tag{5}$$

where C_{in} is then obtained using Eq. 4.

Fuel Moisture Content. Based on the computed indoor climate figures, it is possible to estimate the concentration of water in each of the layers of wood, and based on this in turn estimate the *fuel moisture content* (FMC) of the wood at a given moment of time. Having computed FMC, the time to flashover t_{FO} can be computed as [10]:

$$t_{FO} = 2 * e^{0.16*FMC} \tag{6}$$

The time to flashover is in practice in the order of minutes where the lower the time to flashover, the higher the fire risk. It should be noted in the current version of the model, we do not take into account further risk elements such as wind-speed and wind-direction which may contribute to a high risk of a fire spreading.

The advantage of the FRI model is that it can estimate fire risk based on outdoor climate elements. Measurements and predictions of outdoor climate elements are publicly available as measurements and forecasts covering all of Norway, whereas it is not yet realistic to assume that all houses would be equipped with sensors making indoor climate elements publicly available.

3 Cloud- and Microservice-based Implementation

We have made an implementation of the FRI model capable of computing fire risk indications and providing the indications as a REST web service. The application and the web service has been designed such that it can be used both for our experimental evaluation, and also be consumed by clients in general that need to access to fire risk indications for a given geographical location.

Figure 3 shows the overall software architecture of our prototype system which is divided into several smaller components following a microservice-oriented approach. The Fire Risk Prediction Model (FRP) component implements the FRI model, and the data harvesting and collection component (DHC) deals with collecting weather data, both forecast and historical weather data (measurements). The fire risk web service acts as a controller service that handles communication from clients to the other two components. All components are to be deployed on a cloud platform where we additionally store collected weather forecasts and measurements in order to be able to run the controlled experiments required to validate the FRI model.

The implementation is able to compute fire risk indication based on historical data in the form of measurements from meteorological (weather) stations, forecast data, and a combination of the two. The latter is highly relevant in practice as one often want to compute the indications based on measurements for the last 1-7 days and forecast for the coming 1-7 days. Figure 2 illustrates the interaction between the components of the application when providing a fire risk indication based on a combination of measurements (historical data) and forecast data.

Part of our evaluation goal is to compare the fire risk computed using different weather data sources. In order to have consistent data sets throughout the evaluation, the weather data from all external services are stored in databases. This allows the model to use a consistent data set when testing occurs at a later date for instance as the fire risk model is being refined. The data is stored using the original format as it was retrieved from the external services in order to have the complete data available. Since external services provide data in a JSON or XML representation, we are using a noSQL database [6] for storing the weather data.

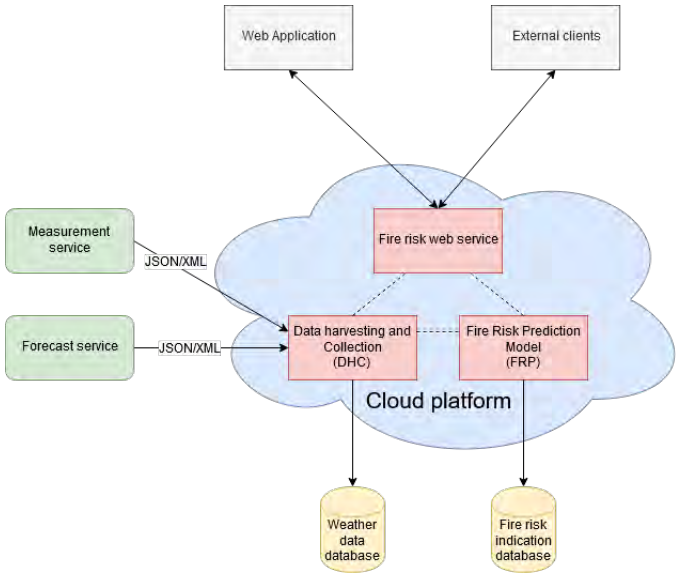


Figure 1: Software architecture of the FRI prototype application

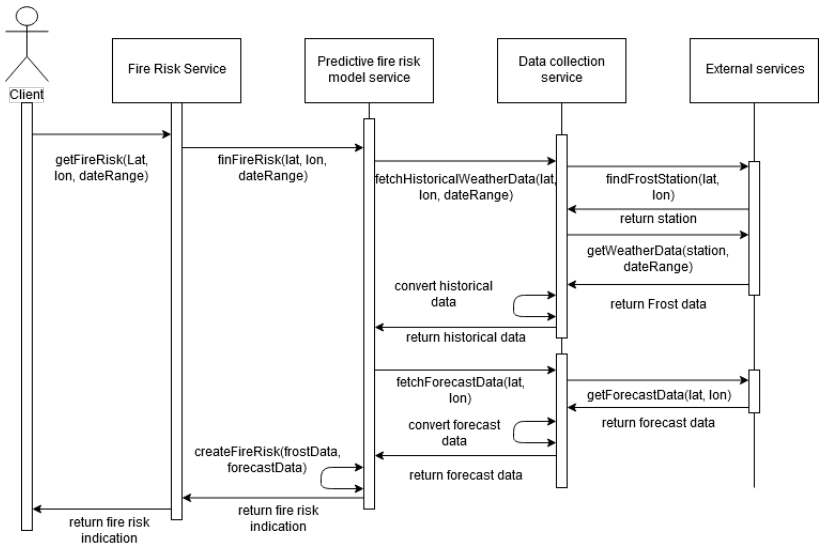


Figure 2: Sequence diagram for fire risk indication with historical and forecast data

The FRI application services and components have been deployed on the Amazon EC2 platform and implemented using the Spark/Java microservice framework. The data storage uses a MongoDB database deployed on the Azure cloud platform. The FRI application uses three external web services to obtain weather data from several sources. Two of the sources provide historical weather data while the third one provides forecast weather data to predict the fire risk in the coming days.

Frost. The Frost API [15] is a REST web service that provides historical weather data recorded by MET. Consumers of the service must provide the locations of where it shall retrieve weather data. This can be done by providing the identity of the source (station), or by giving the longitude and latitude of a position and the service will then find the nearest station. The service gives access to all the stored data that MET has recorded. The Frost API gives access to resources about locations, weather records, observations, lighting, sources (weather station

metadata), elements (weather elements), climate normals, and frequencies. The FRI application uses location, observation, and meta-data about the stations.

Netatmo. The Netatmo service [2] deals with the same type of weather data as the Frost service, but relies on consumer grade weather stations typically installed in private homes. The consumers publish their weather data into a cloud-based server. Through this cloud-based server, it is possible to retrieve the measured weather data, that can then be used in the FRI application. The Netatmo API offers different services based on the types of Netatmo product. In the case of the FRI application, only temperature and humidity is used. The meta-data for the stations contains the identification of the indoor and outdoor module, and general information about the stations such as location which is used by the FRI application.

MET. The MET API [9] provides predictive analysis of the weather in terms of forecast data. It offers resources that estimate how the weather will be in the near future, as well as current weather data such as lowest and highest temperatures over a certain period. The service is able to return the weather data for predictions of the weather for a nine day period into the future. The first three and a half days are provided as hourly measures. The next five and a half days are provided at six hour intervals. The forecast data is offered in XML and JSON format.

4 Experimental Evaluation

We have collected weather data in the winter period 2018 - 2019 at four selected locations in Bergen, Haugesund, Gjøvik, and Lærdal. The reason for choosing these locations were due to the varied climate. At the west coast it is more humid in the winter than in some of the inland locations which in turn are also generally a lot colder during the winter. The collected data includes weather data from Netatmo stations, placed in Bergen and Gjøvik, MET stations, and weather forecasts from Frost. Part of the evaluation has been to investigate whether the data source used impacts the fire risk indication, and to validate the fire risk indication model in terms of being able to provide plausible indications. We also investigated the difference in computing fire risk indications based only on (historical) measurements versus using forecast data or a combination of the two. We also considered historical fires to see how the FRI model implementation would have indicated the fire risk at the time of fire, and in the days leading up to the fire. Finally, we have evaluated the computation-time and storage efficiency of our implementation.

Historical Weather Data. Figure 3 shows the average fire risk indications based on measurements collected in the winter period 2018 - 2019 (December (12) until May (05)). From the graph it can be seen that the FRI model generally indicates an expected higher risk of fire (shorter time to complete flashover) at the colder inland locations. This is also shown in Table 1 which summarises key figures for the complete period. It should be noted that at 50% and 60% indoor relative humidity, the time to flashover is around 9 and 11 minutes, respectively.

Historical and Forecast Weather Data. Being able to predict the fire risk within the next coming days is a main objective. We therefore explore the

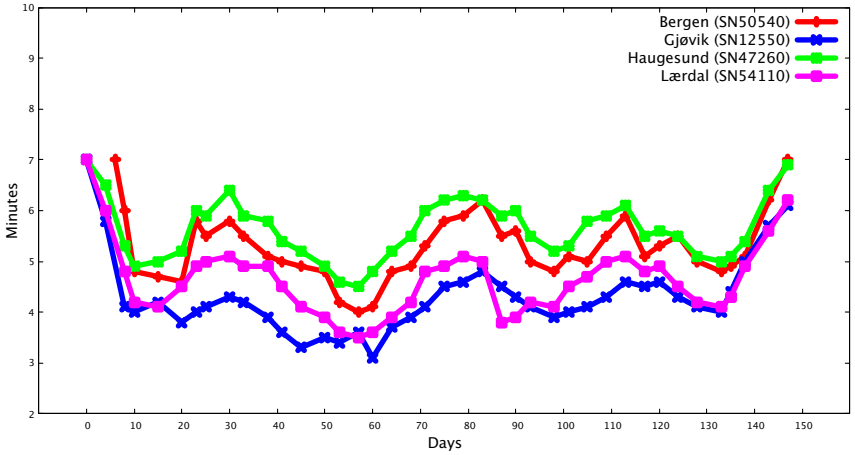


Figure 3: Estimated time to flashover for the four selected locations

Location	Average	Std. dev	Max	Min
Bergen	5.50	0.67	7.64	4.13
Haugesund	5.70	0.63	7.59	4.32
Gjøvik	4.48	0.90	8.02	3.32
Lærdal	4.77	0.74	7.57	3.56

Table 1: Key figures for fire risk indicates from the four locations

combination of measurements (historical data) and forecast data. An aspect to consider is that the FRI model requires a few days of self-calibration before it can accurately begin to indicate the fire risk. To investigate this we ran the FRI model using only weather forecast data (no calibration) and using a combination of historical data (for calibration) and forecast data. Figure 4(left) shows the results for the January-February period in Bergen without the use of historical data for calibration. The corresponding fire risk indication obtained using only historical data is also visualised as a reference. In Figure 4(right) historical weather data (measurements) is added on to the forecast data, which is used for the calibration. As can bee seen from the figures, the fire risk based on forecast follows roughly the same curve as the fire risk indication based on historical data in the first three and a half days. In this period of time, the forecast works with weather data at hourly measures. After that period it starts giving forecast data at a six hour interval. When it starts the six hour interval, the FRI model has less data to work with and predict the fire risk until the next measure in six hours.

Table 2 provides the key figures. It can be seen that the average difference between using only historical and forecast fire risk indication when additional data has not been added for self-calibration is estimated to around 0.26 minutes. The standard deviation for the difference is at 0.24. The maximum difference between forecast and historical when calibrated is 0.58 and when not calibrated is 0.62. The results demonstrates that the use of historical data for calibration results in a more accurate prediction. In practice this is also how we expect the FRI model to be used. When computing a fire risk prediction at a given day, historical data from the past days will be used in combination with the forecast data.

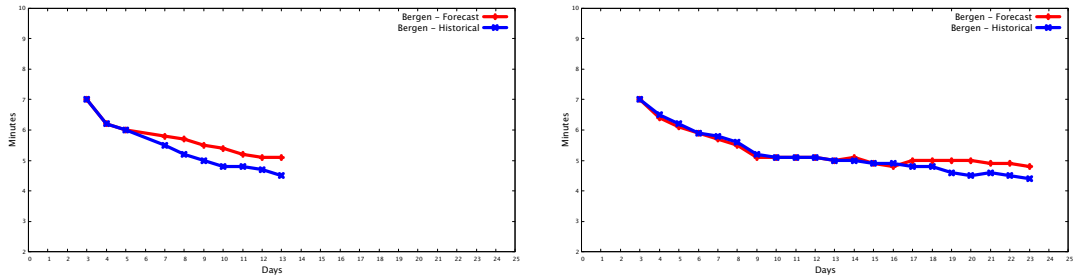


Figure 4: Time to flashover for forecast and historical weather data - without calibration (left) and with calibration (right)

Location	Avg. Diff	Std. Div	Max. diff
Bergen (no calibration)	0.26	0.24	0.63
Bergen (calibration)	0.12	0.18	0.58

Table 2: Fire risk information from four locations

MET stations and Netatmo stations. The results above were based on using MET data from high-end measuring stations. We also investigated the use of low-end measuring stations in the form of Netatmo stations. These stations are typically placed closer to the houses than that of the MET stations. Figure 4 shows the difference between using a a Netatmo station and a MET station for two locations.

As can be seen in Figure 4 (right), the fire risk indication based on the Netatmo station in Gjøvik follows almost the exact same curve as the one based on the MET station. In Figure 4 (left), the Netatmo fire risk indication from Bergen is not exactly the same as the one from MET. The curve itself follows almost to the point of what the MET-based fire risk indicates. This may be due to the fact that the Netatmo station is not correctly calibrated and measures temperatures higher then the MET, station, or that the Netatmo station in Bergen is placed around $1.76km$ north east of the MET station. The MET station in Bergen is located close to the water with an open field surrounding the station whereas the Netatmo station was place in the inner city with surrounding houses.

Table 3 summarises the numerical differences between the use of Netatmo stations and MET stations when computing the fire risk. At Gjøvik, the difference between the Netatmo stations and the Frost stations is almost negligible, whereas the result from Netatmo and Frost Bergen is more varied for certain periods. Still the overall difference is 0.5 minutes on average, and at most around 1 minute.

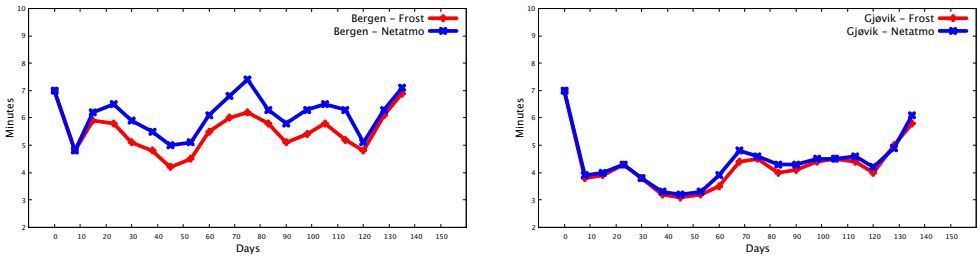


Figure 5: Time to flashover for input data based on the MET and Netatmo stations in Bergen (left) and Gjøvik (right)

Location	avg.diff	std.dev	Max diff	Max	Min
Bergen	0.53	0.28	1.06	7.09	4.13
Gjøvik	0.07	0.07	0.19	6.14	3.32

Table 3: Difference between MET stations and Netatmo stations

Fire Risk for Historical Fires. Another aspect in terms of validating the FRI model, is to consider fires in the past. This way it is possible to determine how the fire risk was at the time of the fire, and the period leading up to the fire. A recent fire that will serve as an example, is the one in Lærdalsøyri on 18th January 2014 [4, 5, 8, 11]. This is a place with many old wooden buildings and at a location that gets very dry during the winter period. The fire risk estimated for that time is visualised in Figure 6(left) with day 0 being the day of the fire. During a period of around 12 days before the fire, the temperature started dropping which results in the climate getting drier. In this dry period, the wood inside the houses released humidity to the indoor air, and was gradually ventilated out of the houses. At the time of the fire at around 22:50 the FRI model indicates that it would take around 3.8 minutes until complete flashover. The fire department learnt about the fire at 22:53pm, and the fire fire truck was on scene at 22:59pm [8]. At this time it was reported that the house was in complete flashover. Since the FRI model indicate a complete flashover in 3.8 minutes, the fire department did not have sufficient time to put out the fire. It should also be noted that at the time, there was storm strength winds in the area [8] which also contributed to higher risk of fire conflagration.

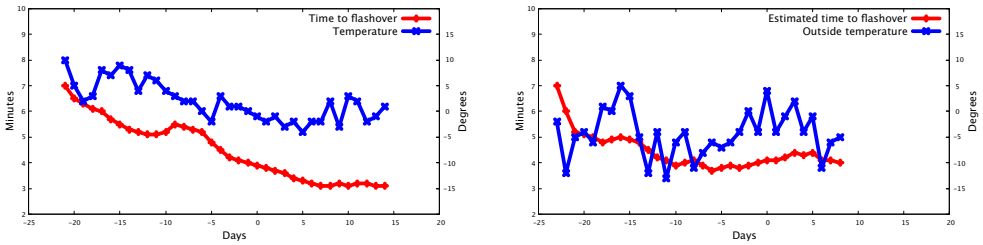


Figure 6: Time to flashover for the fire in Lærdal 2014 (left) and at a home care center in Kongsberg 2017 (right)

Another fire that we considered was the one in Kongsberg, at 24th December 2017, at a home care center[7]. The fire resulted in the loss of life. The fire risk indication for this period is visualised in Figure 6(right). During the December month of that year, the time to flashover averaged around 4.2 minutes. Since this is a home care center, the fire department must conform to response time given by the Norwegian law which that states the required response time is to be 10 minutes or less. Our results indicates the same as the fire risk from the fire in Lærdal of 2014, that the time it takes for a complete flashover is considerably lower than that of the required response time from the fire department.

Given these results, the FRI model could have warned the fire department to be readily available. It also looks like the required response time does not take into consideration dry indoor conditions. Based on these results, the FRI model could have predicted that the fire department could not properly handle a fire that would flashover that quickly. The current minimum response time that the fire department is expected to comply to hence seems to be too high for the dry periods of winter.

Storage and Computation Time. During the 2018-2019 winter period, the FRI application collected approximately five months worth of weather data. This includes forecast data and historical data from four locations, and weather data from two Netatmo stations. The FRI application performs continuous harvest of weather data. Every 24 hours, the FRI application fetches historical weather data for the previous day, from MET and Netatmo, and forecasts for the next nine and a half days. Whenever the FRI application fetches new historical weather data, it will take the previously calculated fire risk indication and create an augmented fire risk indication for the new weather data, and add it to the back of the previous one. By doing it this way, the storage efficiency depends not on the weather data, but only on how many fire risk indications are stored.

Each weather forecast stored in the database had a list of 87 objects containing weather information, such as temperature and humidity. The total amount of storage that these forecasts use, amounts to 12.5Mb, with an average of around 25.4 kb. per forecasts. The weather data from the Frost stations were stored in 24 hour intervals and contains hourly recorded weather elements, mostly the same types as the forecast. The collection in the database that stores historical data ended up containing 634 documents, each of these documents covering 24 hours worth of weather data. The total amount of storage used was 5.6 Mb with an average of 9.0 kb per document. The measurements from the Netatmo stations were stored in the same way as the Frost stations, where each document contains 24 hours worth of weather data. The total number of documents containing weather data from Netatmo totalled at 336.7 kb of storage with a average of 1.3 kb per document.

A fire risk indication for a 24 hour period at one location uses 61.6 kb of storage. Given this, it is possible to calculate how much storage is needed when doing continuous fire risk calculations for several locations. For instance, with continuous fire risk calculations for 10 locations this will amount to 616 kb of fire risk indications every day. For a whole year this will require 224.84 mb of storage. With 100 locations, each with a separate weather station as source, the total amount of storage for a whole year would be 2.24 Gb which is a modest amount of space.

With regard to runtime efficiency, it took 0.07 seconds to compute fire risk indications for a full year. Note that this excludes the time it takes to retrieve the weather data from the external services and the time for converting the data. If everything is included for creating a fire risk for a whole year, the time is 4.1 seconds to retrieve the weather data, another 0.2 seconds to convert it. Then it is passed on to the FRI component which add another 0.6 seconds for conversions and it takes 0.07 seconds to compute the fire risk. The total time elapsed for creating a fire risk indication with weather data for a full years amounts to 5 seconds. If the same was done for half a year, the time 2.5 seconds of which 2.36 seconds is used to fetch the data, and 0.04 seconds used for converting and computations. The rest of the time is spent communicating between the components. This shows that fire risk indications can be computed and stored in both a space and time efficient way.

5 Conclusions and Future Work

We have implemented an innovative and science-based predictive fire risk indication in a cloud-service context where external data services provided by MET and Netatmo have been used to obtain the weather data required for the computation. The results indicate that we are able to obtain a reasonably accurate fire risk

prediction in terms of the estimated time to flashover. In particular given the result of the fires in Lærdal and Kongsberg, we conclude that the FRI model gives accurate fire risk indications. Furthermore, information gathered from the fire department in Bergen, stated that they had a minimum requirement of 10 minutes response time to certain critical buildings. This included hospitals, nursing homes, historical buildings and shopping centres. With the result regarding the Lærdal and Kongsberg fires, many of the fire departments around Norway would not have sufficient time to respond to a fire during the winter period - even if they formally conform to current regulations during other periods of the year.

Our results also shows that it is feasible to use a combination of measurement data and forecast data in order to compute a useful fire risk indication. In fact, our results demonstrate that the best option is to combine the two using measurement data to properly calibrate the FRI model. With regard to storage efficiency, the FRI application requires relatively little storage, and we can conclude that the software architecture has adequate storage efficiency. Furthermore it has been shown that it does not accumulate large amount of weather data. With regard to the run-time efficiency of creating fire risk indications, most of the time was spent fetching data from the external services. The time for computing a fire risk indication was negligible. This indicates that the implementation of the FRI model is adequate regarding the run-time efficiency. The most time consuming internal operation of the FRI application was conversion of weather data.

On the implementation side, we have not yet considered end-user clients. In addition to this, future work may also include further improving the implementation of the FRI model for fire risk indication aimed at making it more efficient. However, the time it takes to compute the fire risk itself is very low, and for that reason it would be more relevant to consider the time it takes to retrieve data from external services possibly by using background fetch of the data.

Based on the result of indicating the fire risk of the four locations, we can identify a period from the start of January to the middle of February as the period with the highest fire risk. After this period the fire risk reduces steadily with a few periods where it goes up again before it starts to decrease from the middle of April. It is at this point the weather climate gets warmer, but that does not necessarily mean that the fire risk will be lower. The reason for this is because of, e.g,. heathlands and forests that can easily catch fire. This may also have an impact on the fire risk for later periods of the year. The current FRI model considers only colder climate conditions and would have to be mathematically refined in order to cover also the warmer periods of the year. Another aspect that may be improved in the future is to take into account more parameters when calculating the fire risk. This could be done by weather elements such as wind-speed and direction to get a more accurate result regarding conflagration risk. There is also the possibility of combining the wind with a parameter that may indicate how densely the houses are located.

Acknowledgement. This study was partly funded by the Research Council of Norway, grant no 298993 *Reducing fire disaster risk through dynamic risk assessment and management (DYNAMIC).*

References

[1] R. Baronas, F. Ivanauskas, I. Juodeikiene, and A. Kajalavicius. Modelling of moisture movement in wood during outdoor storage. *Nonlinear Anal. Model.*

Control, 6(2):3–14, 2001.

[2] Netatmo Connect. Weather Data. `api.netatmo.com/api`.

[3] N. Dragoni, S. Giallorenzo, A. Lafuente, M. Mazzara, F. Montesi, and R. Mustafin. Microservices: Yesterday, Today, and Tomorrow. In *Present and Ulterior Software Engineering*, pages 195–216. Springer, 2017.

[4] DSB. Brannene i lærdal, flatanger og p frøya vinteren 2014. Technical report, Norwegian Directorate for Civil Protection, 2014. In Norwegian.

[5] S. Hansen et.al. Evaluation of fire spread in the large lrdal fire. In *14th Int. Fire and Materials Conf. and Exhib.*, page 10141024, January 2015.

[6] Jing Han, Haihong E, Guan Le, and Jian Du. Survey on NoSQL database. In *Proc. of Intl. Conf. on Pervasive Computing and Applications*. IEEE, 2011.

[7] C. Hunshamar, I. Rønold, G. Andersen, and M. Holmes. Brann i Kongsberg: En person funnet død. `vg.no`, 2017.

[8] F. Ighoubah and S. Solheim. Slik var de første meldingene om Lærdalsbrannen. `nrk.no`, 2014.

[9] Meteorologisk Institutt. Weather Forecast. `api.met.no`.

[10] A. Kraaijeveld, A. Gunnarshaug, B. Schei, and T. Log. Burning rate and time to flashover in wooden 1/4 scale compartments as a function of fuel moisture content. In *8th Int. Fire Science and Eng. Conf., Interflam*, page 553558, 2016.

[11] T. Log. Cold climate fire risk; a case study of the lrdalsyri fire. *Fire Techn*, 52:1815–1843, January 2014.

[12] T. Log. Indoor Relative Humidity as a Fire Risk Indicator. *Building and Environment*, 111:238–248, 2017.

[13] T. Log. Indoor relative humidity and wood moisture content as a proxy for wooden home fire risk. *Sensors*, 2019. Submitted.

[14] M. Metallinou and T. Log. Cold climate structural fire danger rating system challenges. *Challenges*, 9(12):1–15, 2018.

[15] Meteorologisk Institutt. Historical Weather Data. `frost.met.no`.

[16] C. Pautasso, O. Zimmermann, and F. Leymann. Restful Web Services vs. Big Web Services: Making the Right Architectural Decision. In *Proc. of Intl. Conf. on World Wide Web*, pages 805–814. ACM, 2008.

[17] A.R. Pirsko and W.L. Fons. Frequency of urban building fires as related to daily weather conditions. Technical Report 866, US Dep. of Agriculture, 1956.

[18] Spark/Java: Microservice Framework. `www.sparkjava.com`.

[19] S. Stokkenes. Implementation and Evaluation of a Fire Risk Indication Model. Master's thesis, Western Norway University of Applied Sciences, 2019.

Escape Local Minima with Improved Particle Swarm Optimization Algorithm

K. Darshana Abeyrathna[a] and Chawalit Jeenanunta[b]
Department of Information and Communication Technology,
University of Agder, Grimstad, Norway[a].
School of Management Technology, Sirindhorn International Institute of Technology, Thammasat University, Pathumthani, Thailand[b].
darshana.abeyrathna@uia.no[a], chawalit@siit.tu.ac.th[b]

Abstract

Particle Swarm Optimization (PSO) is a powerful meta-heuristic technique which has been maneuvered to solve numerous complex optimization problems. However, due to its characteristics, there is a possibility to trap all particles in a local minimum in the solution space and then they cannot find the way out from the trap on their own. Therefore, we modify the traditional PSO algorithm by adding an extra step so that it helps PSO to find a better solution than the local minimum that they undesirably found. We perturb all the particles by adjusting parameter values in the traditional algorithm when there is no improvement of the objective value over the training iterations, assuming that particles have stuck in a local minimum. In this research, we mainly focus on adjusting the learning factors. However, the parameter values have to be used in an effective way to perturb the particles. The behavior of the proposed modification and its parameter adjustments are studied using a function which has a large number of local minima - Schwefel's function. Results show that 2 out of 3 PSO attempts trap in local minimum and slight changes on learning factors do not help them to get out from the traps. However, perturbances made with large learning factors can find better solutions than the local minima that they stuck in and help to find the global minimum eventually.

1. Introduction

Particle Swarm Optimization (PSO) is a global optimization technique where it is used to solve complex optimization problem with highly non-linear objective functions. It has been successfully applied by researchers and engineers in number of domains to solve complex optimization problems, including weights optimization of Artificial Neural Networks [1, 2].

In most of the applications, we try to minimize the objective value of the optimization problems. In many of those cases, it can be adjusting parameters to find the minimum distance, minimum cost, minimum time required, minimum error, etc. However, with PSO, a large number of solutions for the objective function are created by particles in the PSO at one generation. Generation by generation, all particles evolve to an optimum which provide the optimum parameter values.

Once the objective function is ready with a set of constraints, fitness of each particle is calculated. This is the objective output for the selected particle. At end of the fitness value calculation of all the particles, personal best (pbest) and the global best (gbest) solutions for the current population is updated. Each particle has a best solution that has been obtained so far. This best solution of each particle is called the pbest. However, gbest is the best particle that provides the best parameter values to the objective function out of all other particles from all generations. These updated pbest and gbest are used to calculate the positions and velocities of each particle at the next generation using the following equations.

$$V^i(t+1) = w \times V^i(t) + c1 \times rand1 \times (p - X^i(t)) + c2 \times rand2 \times (g - X^i(t)) \quad (1)$$

This paper was presented at the NIK 2019 conference. For more information see http://www.nik.no/

$$X^i(t+1) = X^i(t) + V^i(t+1) \quad (2)$$

The velocity and position of i^{th} particle at generation t is given by $V^i(t)$ and $X^i(t)$, respectively while pbest and gbest given with p and g in the above equations. The effect of current velocity for the next generation is decided by the inertia weight w. With a higher value for the inertia weight, current velocity has a higher impact from the previous velocity and vice versa. Random values, *rand1*, *rand2* lie between 0 and 1. Learning factors *c1* and *c2* have to be predetermined and different values of them alter the performance of the PSO algorithm significantly.

However, when the learning procedures of all the particles in PSO decide by the Eq. (1) and (2), parameter values can be adjusted so that, all particles closely follow the global best or they follow their own best to beat the global best solution separately. Both these procedures have their own advantages and disadvantages. The highlighting limitation of the first procedure is once the global best particle traps in a local minimum, all particles will follow that specific particle and trap in the same local minimum. Since they closely follow the global best particle, none of the other particles will find a way to get out from the trap. In the second procedure, each particle competes each other to becomes the best particle of the entire set. In that case, PSO requires large number of training cycles to find the global best solution and will stop at a random location where they find the best at that specific training iteration, otherwise.

Therefore, in this research, we propose a solution for the above limitations of the PSO algorithm combining the discussed properties of the individual procedures. The algorithm starts following the first procedure where all particles closely follow the global best particle. Once they trap in local minima, we switch the parameter values of the Eq. (1) so that they find a completely new solution set for the next generation. In that way, one or some of the particles can find better solutions than the current global best where it is actually a local best of the actual solution space. In this research, we manly focus on adjusting the learning factors *c1* and *c2*. We also conduct an experiment to find the best learning factor combination.

The rest of the paper is organized as follows. The works related to the proposed algorithm and their usage are discussed in the following section (Sec. 2). Steps of the proposed algorithm are presented in detail in the Methodology section (Sec. 3). A detailed description about the experiment setup is given in Sec.4. Results of the conducted experiment are presented and discussed in the Results and Discussion section (Sec. 5). The conclusion for the entire research is made at the end (Sec. 6).

2. Related Works

Finding the best hyper parameter combination for any algorithm is important and difficult. The selected parameters decide the performance and final outcome of the algorithm. This is critical for the PSO algorithm compared to others since it has collectively 5 hyper parameters to be adjusted (Eq. (1)). Adjusting them separately while keeping others in a fixed position will consume a lot of time and might not find the best combination. However, researchers have conducted different experiments to select the best hyper parameters for PSO [3, 4]. The effect of different inertia weights on the performance of PSO has been tested in [3]. The performance has been evaluated by varying inertia weight with an additional constraint call "maximum velocity", V_{max}. According to the empirical results, they identify, when the chosen maximum velocity is small (≤ 2), inertia weight should be fixed approximately at 1. In contrast to the above,

if the maximum velocity is not small (≥ 3), 0.8 for the inertia weight is a better choice. According to the experiment setup in [4], two sets of parameters are selected to test the PSO performance based on the selected parameters. In both sets, learning factors are considered as qual (*c1 = c2*). In the parameter set 1, the inertia weight is 0.6 and learning factors are 1.7 each. In the parameter set 2, the inertia weight is 0.729 and learning factors are 1.494 each. Test results reveal better performance of parameter set 1 while showing the increase of performance parallel to the increase of the number of particles in a generation. However, still it's a comparison of results between selected sets, but it is not the best way to find the best parameter set when the best parameter set is not included in the testing sets.

In addition to findings of best hyper parameters, some researchers have tried to modify the algorithm in order to get better results. A cooperative approach to the PSO algorithm has been presented in [5]. In this proposed approach, different sections of the solution vector are optimized using different individual swarms in PSO. The proposed algorithm is used to solve a benchmark optimization problem and compared against the traditional PSO approach to show the value of the proposed approach. Similar to the above approach, [6] presents a modified version of PSO called Quantum Delta-Potential-Well-based Particle Swarm Optimization (QDPSO) algorithm. In their approach, they identify the quantum behavior of particles. Then they put a higher attention on the particles which are far from the global best. According to their argument, those particles have a higher potential to find better solutions than particles which closely following the global best. Another approach call Adaptive Particle Swarm Optimization (APSO) is presented in [7]. The proposed approach has following steps: fitness of each particle is calculated, four evolutionary states are identified based on the population distribution – 1. exploration, 2. exploitation, 3. convergence, and 4. jumping out. Based on these states, hyper parameters of Eq. (1) are automatically controlled. Then next generation is created based on the adjusted parameter values.

The performance of the algorithm has been improved not only by modifying the algorithm parameters, but also by combing it with some well know algorithms such as Genetic Algorithm (GA) [8]. The procedure in [8] is similar to the procedure in this paper. However, it is complex in nature as it uses GA operations once they stuck in local minima. In [9], a selection mechanism from evolutionary computation has been used to improve the PSO performances. A score for a particle is calculated based on its current position and positions of the other particles. Once the scores of all particles are calculated, they are sorted based on the calculated score. Then the positions and velocities of worst 50% of particles are replaced with the remaining 50% best particles. At each training iteration, this additional procedure is added to the algorithm. In the testing phase, the proposed algorithm shows better performances on three out of four testing functions. A similar approach to enhance the PSO performance is given in [10] by combining PSO and Chaos theory. According to the simulation results, chaotic PSO (CPSO) outperforms standard PSO and some other meta-heuristics algorithms.

The concept of the proposed algorithm in this research is quite similar to the methods in [8, 10]. Compared to [8], the proposed method is easy and computationally inexpensive. Compared to [10], the proposed algorithm starts with traditional PSO and let all particles (moving closer to the global best) to reach an optimum point (global or local) which is faster compared to their research. Then, only when it's required, particles are perturbed to get out from the trap in case if they have stuck in local minima. This proposed algorithm is discussed in detail in the next section.

3. Methodology

The proposed algorithm starts similar way to the traditional PSO approach. Parameters in the objective function collectively form the particles in PSO. Each particle makes a solution to the objective function and all particles collectively moves towards the global optimum during the training phase to find the optimum parameter set.

The algorithm starts creating *n* random solutions to the objective function and *n* is equal to the number of particles in a population. The fitness value of each particle is calculated using the objective function. The particle with the best fitness value is considered as the global best and they also update their personal bests at every training iteration as explained in the introduction section. Likewise, with the aid of Eq. (1) and (2), generation by generation they reach to the global optimum in the solution space. However, it is not possible to say whether particles have reach to the global optimum or traps in a local optimum when it shows minor or no improvements at some adjacent training iterations. However, with the traditional approach, they cannot get out from the local minimum in case if they trap in one. Therefore, in this algorithm, we add an additional step to perturb the particles if there is no or les improvement in the global best for 10 consecutive training iterations (10 stall iterations) by changing learning factors, *c1* and *c2*. However, we also save the current personal best and global best solutions in case if they have already reach to the global solution. This step is applied two times to the traditional approach and assume that it's enough for them to escape local minima and reaches to the global minimum. Then the training procedure stops when it reaches to the maximum training iterations. The states of the proposed algorithm can be well learned by using the following flowchart.

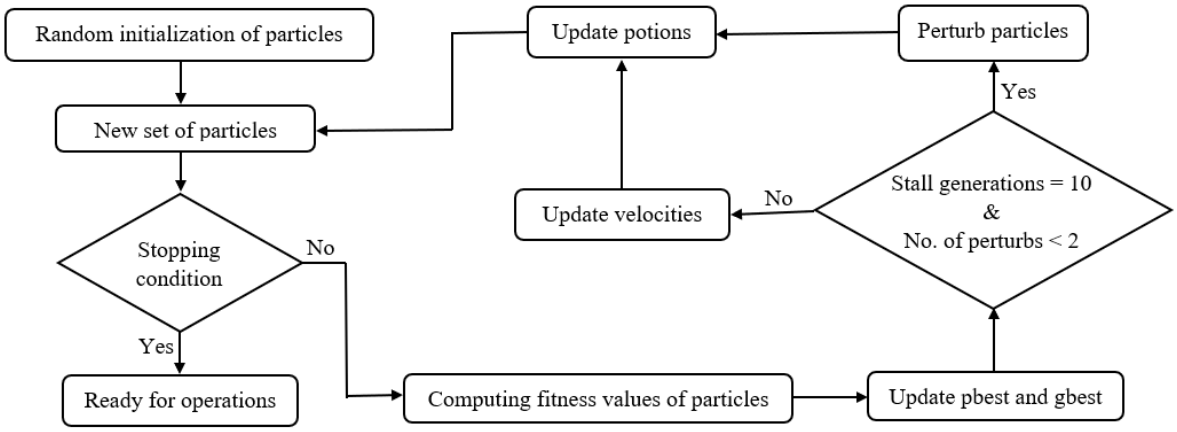


Figure 1 Flowchart of the improved PSO algorithm

However, the best learning-factor combination, *c1* and *c2*, to perturb particles are still unknown. Therefore, to find the best combination of learning factors and to check the performance of the proposed algorithm, an artificial dataset is created. Detail of the experiment setup is given in the next section.

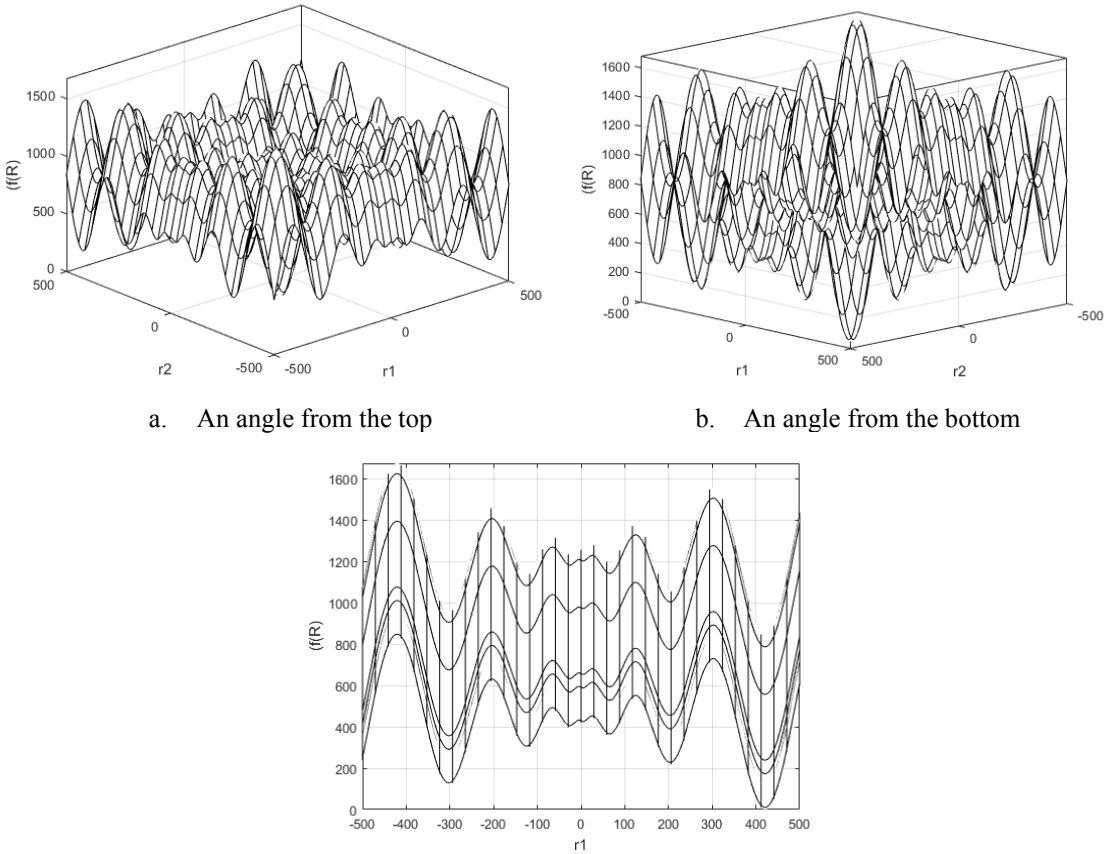
4. Experimental Setup

The objective of the proposed algorithm is to improve the traditional PSO algorithm which, sometimes, shows poor convergence ability with local minima in the solution space. Therefore, to demonstrate the optimization qualities of the proposed algorithm, a

function with multiple local minima, which is famous in terms of its toughness to find the global optimum [11] is selected: Schwefel's function. While Eq. (3) gives the Schwefel's function, Figure 2 shows some angles when it is plotted for two variables.

$$f(R) = \quad 418.9829 \times d - \sum_{i=1}^{d} r_i \times \sin(\sqrt{|r_i|}) \tag{3}$$

In this equation, to reach the global optimum, 0, we set a constraint where the maximum and minimum limits of each variable r_i are 500 and -500, respectively. Then, with any number of variables, d the objective value $f(R)$ reaches to 0 at the optimum R^* [$R^* = (\ r_1^*,\ r_2^*, \ldots..\ r_d^*\ ) = (420.9687, 420.9687, \ldots.. 420.9687)$]. However, the following figure shows different angles of the Schwefel's function when it is drawn with two variables ($d = 2$).



a. An angle from the top

b. An angle from the bottom

c. Side view from r_1 direction

Figure 2 Schwefel's function for two variables

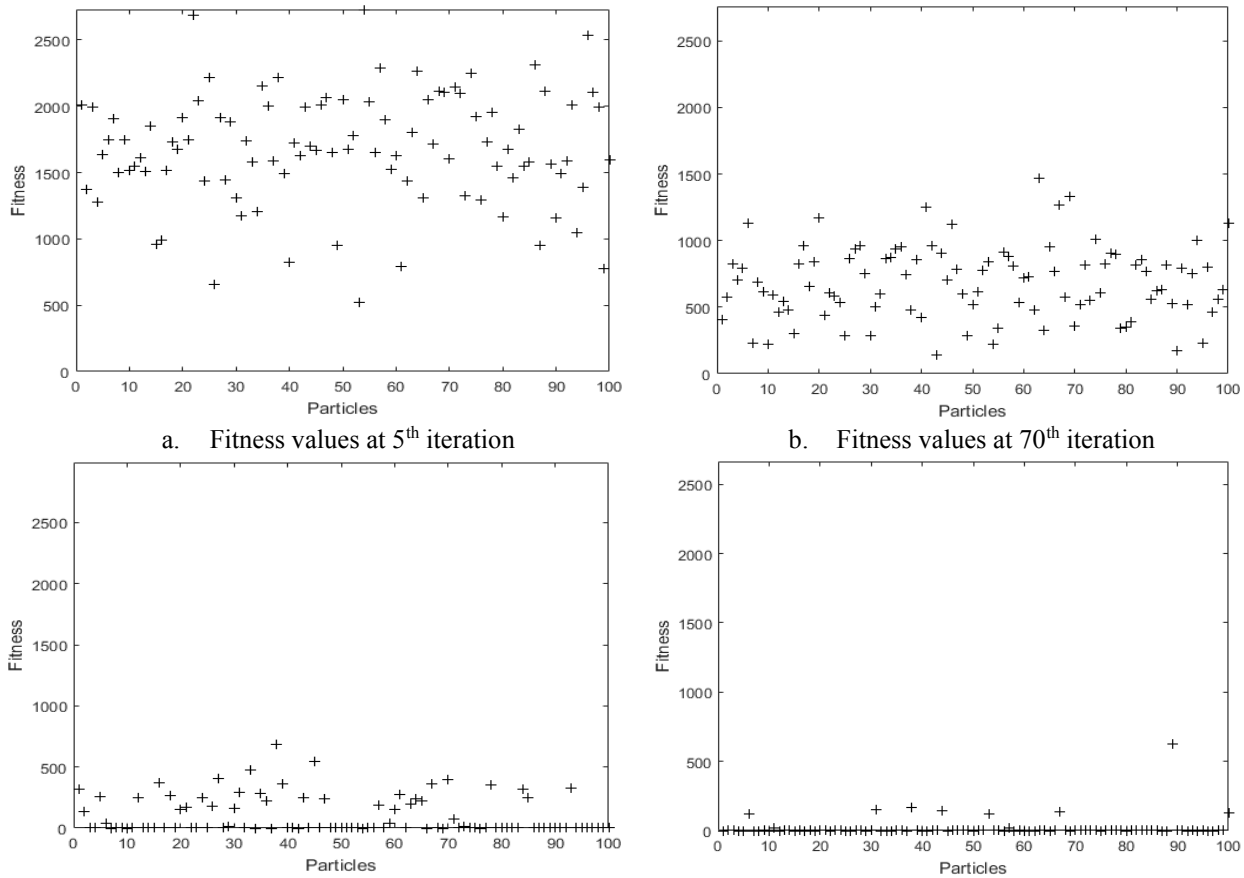
Figure 2 reveals the difficulty of finding the global minimum of the Schwefel's function since it has many local minima. However, in the experiment, we make it more difficult adding two more variables ($d = 4$) to find the global optimum with both the proposed and traditional PSO algorithms.

The proposed algorithm is similar to the traditional PSO at the beginning. To initiate with, 100 particles are created randomly with the considered constraint for the variables ($r_i \in$ [-500, 500]). Fitness values of those particles are calculated using the Schwefel's

function. Based on the fitness values, pbest and gbest are updated. Velocities and positions for the next generation are calculated using Eq. (1) and (2), respectively. Considering the best parameter values identified at [4], we fix the inertia weight, *w* and learning factors, (*c1* = *c2*) at 0.6 and 1.7, respectively. The traditional PSO algorithm runs 500 training iterations (generations), if does not meet 10 continuous stall generations, without converting to the proposed algorithm. Once it meets 10 continuous stall generations, all the particles in the current generation are perturbed to create the next generation. However, in this research, particles are perturbed by only adjusting learning factors. Therefore, these values are changed starting from 2 (*c1* = *c2* = 2, 2 is slightly higher than the current value 1.7). The other values selected for learning factors are, 10, 20, 50, 100, 500, and 1000. The results obtained with these changes are presented and discussed in the next section.

5. Results and Discussion

The fitness values of each particle at each training iteration is studied separately. At the beginning, fitness values of all particles are higher and generation by generation they all reach to a minimum point (global or local). The evolution of these particles of a successful PSO attempt are captured at four different training iterations and resented in Figure 3.



a. Fitness values at 5th iteration

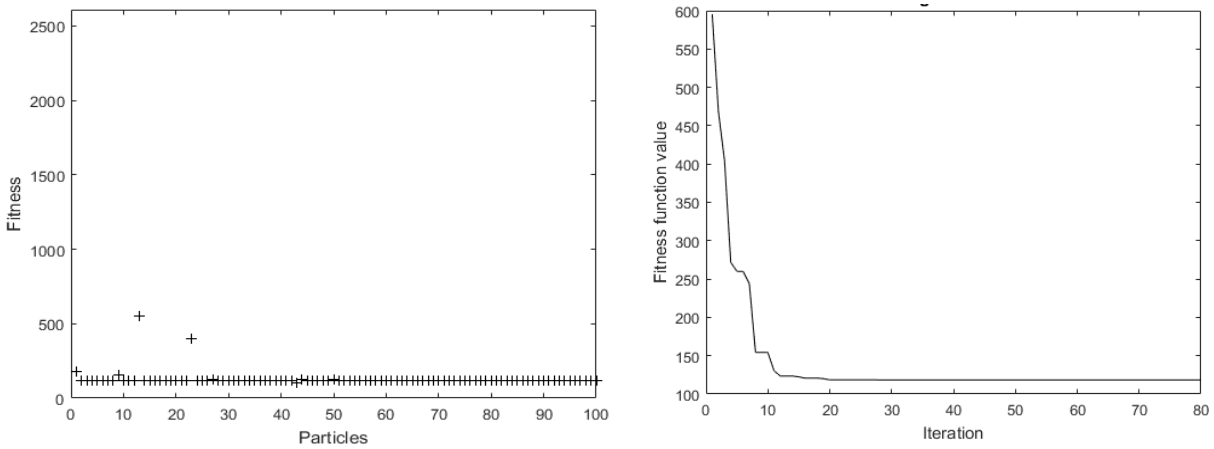
b. Fitness values at 70th iteration

c. Fitness values at 150th iteration

d. Fitness values at 170th iteration

Figure 3 Evolution of particles at different training iterations

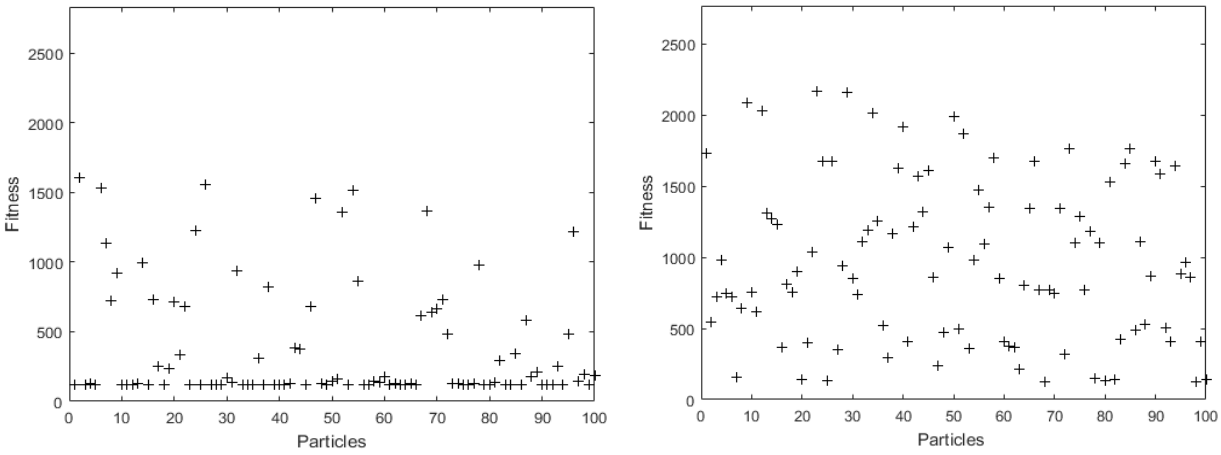
However, when the global best moves towards to a local minimum, all particles follow it and trap in the same local minimum (in case the other particles unable to find a better solution) as shown by Figure 4. Figure 4.a shows that almost all particles have reached to the same local minimum at end of the training iterations and Figure 4.b shows the evolution of the fitness value of the global best over the training iterations.



a. Fitness values of all particles at 80th iteration b. Global best fitness over the iterations

Figure 4 Evolution of particles towards a local minimum

We perturb all particles when they stick in these kinds of traps. However, different values for the learning factors (*c1* and *c2*) perturb particles in the Figure 4.a in different ways. For instance, variation of fitness values of particles, when learning factors during the perturbance process are equal to 10 and 500, are given in Figure 5.a and 5.b, respectively.



a. Perturb with $c1 = c2 = 10$ b. Perturb with $c1 = c2 = 500$

Figure 5. Perturb particles when they trap in local minima

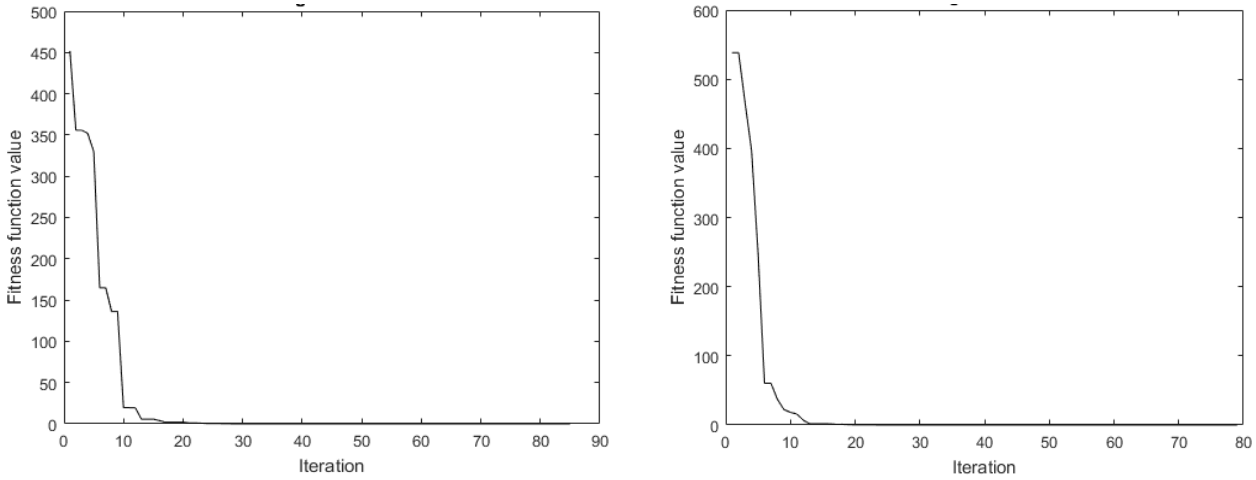
Therefore, the effect of the perturbances depends on the values of the used learning factors. The Table 1 provides the average minimum values of each training algorithm (traditional PSO and proposed algorithm with different learning factor values) obtained after 30 runs.

The table shows that 2 out of 3 PSO attempts trap in local minima and small perturbances do not help them to get out from those traps. With higher values for learning

factors, PSO starts finding new and better solutions than the local minima that they stuck. Figure 6, 7 and 8 provide selected examples for the cases where (Figure 6) PSO found the global optimum itself, (Figure 7) PSO stuck in local minima, and (Figure 8) how perturbances help PSO to escape local minima.

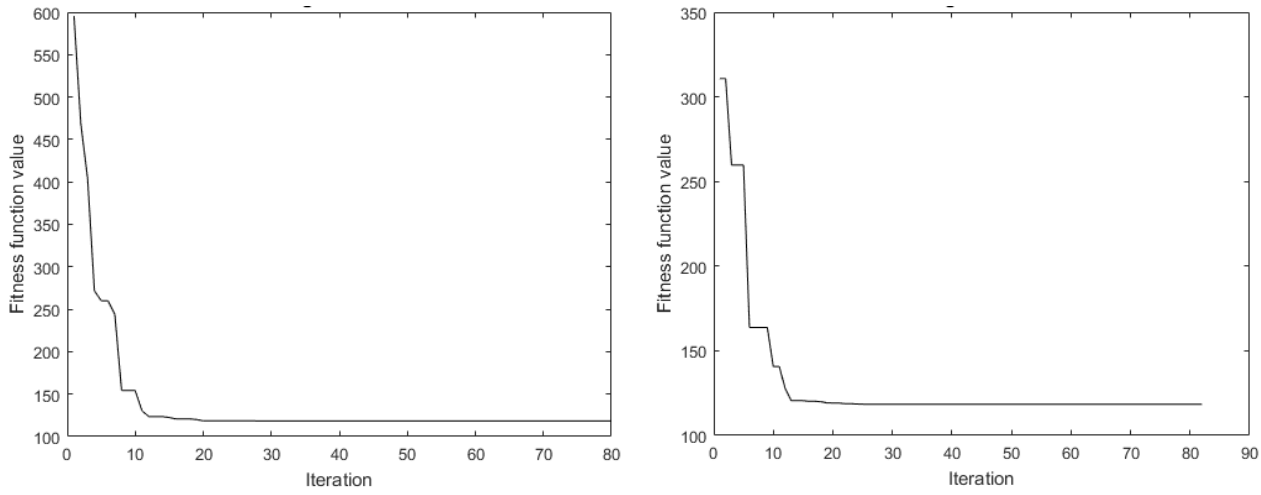
Table 1 Average of the minimum values given by different algorithms with different parameters

	PSO	**Proposed Algorithm with *c1* = *c2* =**						
		2	**10**	**20**	**50**	**100**	**500**	**1000**
No. of global minimum findings (out of 30)	10	9	10	10	12	14	19	21
Average of minimum values	93.87	97.80	90.07	101.47	85.73	77.87	50.73	35.40



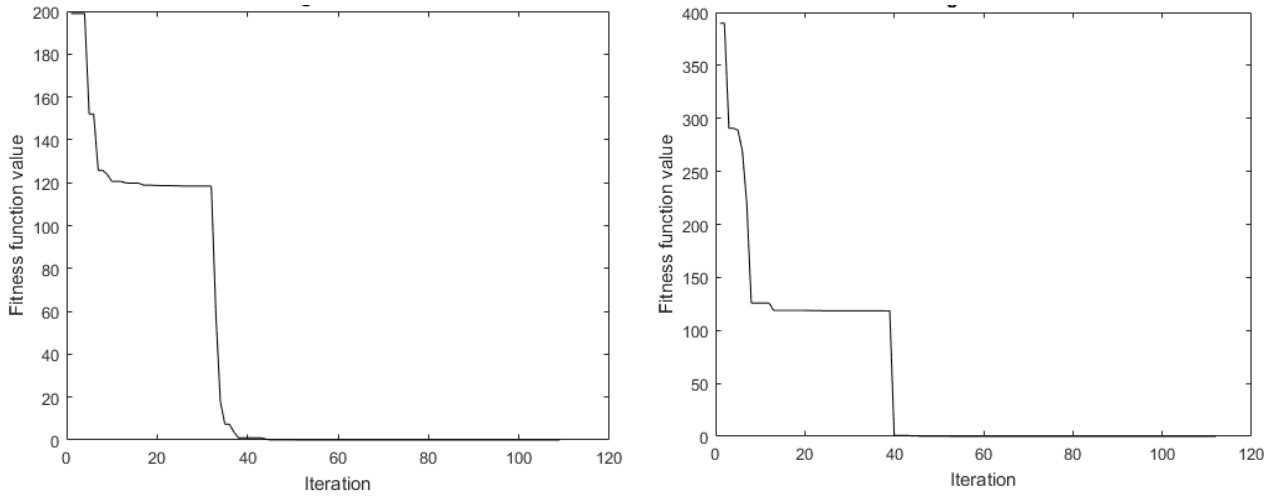
a. Example 1 b. Example 2

Figure 6 Examples that PSO found the global minimum without the help of perturbances



a. Example 1 b. Example 2

Figure 7. Examples that PSO stuck in local minima



a. Example 1 b. Example 2

Figure 8. Examples that PSO found the global minimum with the help of perturbances

Examples in both Figure 7 and 8 show that there is a local minimum around 120 fitness value in the solution space. Figure 6 shows that PSO can find the global minimum without the help of perturbance step. However, both Figure 7 and 8 and also the Table 1 show that still PSO can trap in local minima and therefore need perturbance step to escape them. The following section brings the conclusions of the research based on the hypothesis we made, the method we proposed, and the above results.

5. Conclusion

PSO is a powerful meta-heuristic technique which has been used to solve plenty of complex optimization problems. However, it has a limitation where when it traps in local minima in the solution space, it cannot get out from the trap on its own. Therefore, in this research, we slightly modify the algorithm by adding an extra step to perturb the particles when they trap in local minima. This modification to the algorithm bring a significant improvement to the results discussed in the previous section.

However, even though we identify that PSO likely to trap in local minima when it is used to solve functions which have many local minima and the proposed modifications to the algorithm can solve the problem, the parameter selection is still an issue. Therefore, similar to the other hyper parameters, this value of the learning factors during the perturbance stage have to be selected after a grid search. Values are subject to change with the difficulty of the problem. Therefore, a trial and error attempt to find the best learning factors to perturb the particles when they trap in local minima is necessary. These factors should be able to perturb the particles until at least they produce the initial fitness values.

References

1. Jeenanunta, C. and K.D. Abeyrathn, *Combine Particle Swarm Optimization with Artificial Neural Networks for Short-Term Load Forecasting.* ISJET, 2017. **8**: p. 25.

2. Daş, G.S., *Forecasting the energy demand of Turkey with a NN based on an improved Particle Swarm Optimization.* Neural Computing and Applications, 2017. **28**(1): p. 539-549.
3. Shi, Y. and R.C. Eberhart. *Parameter selection in particle swarm optimization*. 1998. Berlin, Heidelberg: Springer Berlin Heidelberg.
4. Trelea, I.C., *The particle swarm optimization algorithm: convergence analysis and parameter selection.* Information Processing Letters, 2003. **85**(6): p. 317-325.
5. Van den Bergh, F. and A.P. Engelbrecht, *A cooperative approach to particle swarm optimization.* IEEE transactions on evolutionary computation, 2004. **8**(3): p. 225-239.
6. Sun, J., B. Feng, and W. Xu. *Particle swarm optimization with particles having quantum behavior*. in *Evolutionary Computation, 2004. CEC2004. Congress on*. 2004. IEEE.
7. Zhan, Z.-H., et al., *Adaptive particle swarm optimization.* IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), 2009. **39**(6): p. 1362-1381.
8. Abeyrathna, K.D. and C. Jeenanunta, *Hybrid Particle Swarm Optimization With Genetic Algorithm to Train Artificial Neural Networks for Short-Term Load Forecasting.* International Journal of Swarm Intelligence Research (IJSIR), 2019. **10**(1): p. 1-14.
9. Angeline, P.J. *Using selection to improve particle swarm optimization*. in *Evolutionary Computation Proceedings, 1998. IEEE World Congress on Computational Intelligence., The 1998 IEEE International Conference on*. 1998. IEEE.
10. Liu, B., et al., *Improved particle swarm optimization combined with chaos.* Chaos, Solitons & Fractals, 2005. **25**(5): p. 1261-1271.
11. Whitley, D., V.S. Gordon, and K. Mathias. *Lamarckian evolution, the Baldwin effect and function optimization*. 1994. Berlin, Heidelberg: Springer Berlin Heidelberg.

Geological Multi-scenario Reasoning

Crystal Chang Din[a], Leif Harald Karlsen[a], Irina Pene[b], Oliver Stahl[a], Ingrid Chieh Yu[a], Thomas Østerlie[c]

[a]Department of Informatics, University of Oslo, Norway
[b]Department of Geosciences, University of Oslo, Norway
[c]Department of computer and information science, NTNU, Norway

Abstract

In the oil and gas industry, during exploration prospect assessment, explorationists rely on ad hoc manual work practices and tools for developing and communicating multiple hypothetical geological scenarios of the prospect. This leaves them with little efficient means to make the fullest use of state of the art digital technologies to communicate and systematically compare and assess different hypothetical geological scenarios before deciding which scenario to pursue. In this paper we present a formal framework for geological multi-scenario reasoning, a novel tool-based method for geologically oriented subsurface evaluation. The methodology applies formal methods and logic-based techniques to subsurface evaluation and expresses interpretive uncertainty as discrete scenarios with branches of potential alternatives. This framework consists of (i) a proto-scenario generator that takes user observations and geological evidence as input and generates semantically valid initial states based on formalized geological knowledge in first-order logic (ii) geological processes formalized as a rewrite theory that are executable in Maude. By applying geological rewrite rules onto the proto-scenarios, we are able to assist explorationists with multi-scenario generation and reasoning beyond human capacity.

1 Introduction

This paper presents a novel method and tool developed with the aim to assist geologists, in particular explorationists in their subsurface evaluation process and in their exploration for new subsurface petroleum resources. This framework is developed as part of a larger interdisciplinary research project on geological multi-scenario reasoning taking place at the SIRIUS research based innovation center.

A key challenge for exploration in particular, but also subsurface evaluation in general, is that geodata are "uncertain, intermittent, sparse, multiresolution, and multi-scale" [6]. Geodata underdetermine concrete models and interpretations [11].

This paper was presented at the NIK-2019 conference; see http://www.nik.no/.

This means that even though geodata may refute particular interpretations of the subsurface, they can never verify the correctness of a single interpretation. As such, it is common for different geoscientists to hold widely differing interpretations of available data [2, 3]. Advances in digital tool support for quantitatively-oriented geodata interpretation over the past decades have improved geoscientists' capacity to delineate closed structures and potential traps indicating possible hydrocarbon deposits in the subsurface [13]. However, available geodata provide limited information about critical factors for exploration; whether source rock is present and mature, the presence and quality of reservoir rock, the presence of an effective seal, and whether or not there are hydrocarbon accumulations in subsurface structures observed in the data. Geoscientists therefore supplement these quantitative methods with qualitatively oriented forms of reasoning we refer to as geological history reasoning. Geological history reasoning makes active use of data underdetermination as a source of knowledge to develop and assess multiple hypothetical scenarios of the subsurface to justify whether, where, and (importantly) why there can be hydrocarbon deposits in a prospect through many-faceted geological scenarios explaining available data in terms of sequences of interrelated geological processes [10]. Despite its centrality in exploring for new subsurface hydrocarbon resources, this qualitative form of reasoning remains almost completely unsupported in the digital tool set currently available for geoscientists [12]. Instead, they use pen and paper along with generic digital drawing and presentation tools for developing and communicating multiple hypothetical geological scenarios. With little to no efficient means to make the fullest use of state of the art digital technologies to systematically compare and assess different hypothetical geological scenarios before deciding which scenario to pursue when assessing exploration prospects, geoscientists often end up limiting themselves to only assessing a few possible scenarios due to the time constraints of commercial petroleum exploration.

Through geological multi-scenario reasoning we apply formal methods and logic-based techniques to express the space of possible interpretations as discrete scenarios with branches of potential alternatives. Through systematic representation, analysis, and comparison of different geological scenarios the goal is to support geoscientists to reason beyond human capacity. In this paper, we present the formal framework behind such tool-supported geological reasoning. This framework consists of (i) a formalization of geological domain knowledge [8] in first-order logic (FOL) that generates *proto-scenarios*, i.e., expanded semantically valid geological descriptions based on geologists' observations and geological evidence (ii) a formal specification of geological processes in rewriting logic (RL) [1, 9] that can be executed in Maude [1]. With RL, we capture geological entities, concurrent geological process, and distributed geological states. With FOL, we address complex geometrical relationships and relative timing. By applying geological rewrite theory that we have developed with geologists onto the proto-scenarios, we are able to assist explorationists with multi-scenario reasoning. The combination of these two reasoning methods having rigorous semantics allow us to explore, generate and analyse multi-scenarios to a depth that is otherwise impossible to achieve. Finally, the methodology and tool chain are designed in such a way that allows deployment on a scalable technology.

We have organized the remainder of the paper as follows. First we outline petroleum system analysis, a particular form of subsurface assessment that we have

used as use case for developing our framework. We then progress to outlining the formalization of geological processes in rewriting logic, how we represent geological knowledge, and the scalable infrastructure for running multiple geological scenarios. Finally, we outline the use case demonstration, before drawing the conclusion.

2 Petroleum System Analysis

A main objective for explorationists is to make decisions about whether and where to drill given limited and sparse subsurface data. This decision is made based on detailed petroleum system analysis of the area of interest.

Petroleum system is the framework used in substantiating the possibility of hydrocarbons accumulation in an area. It is defined as a unifying concept that

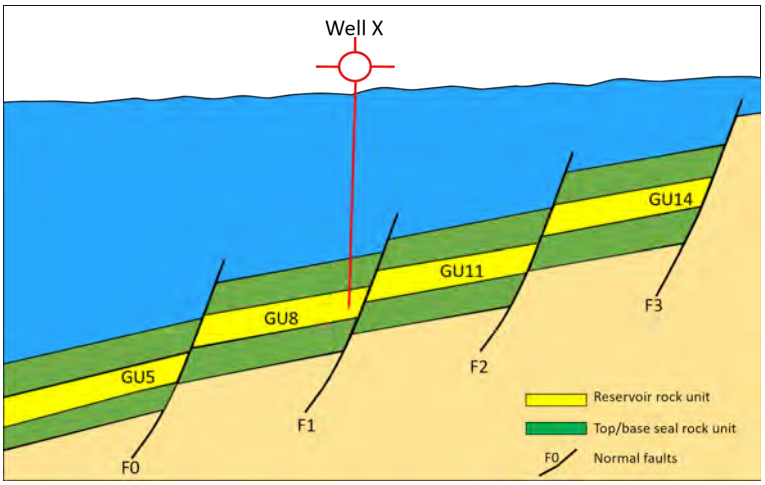


Figure 1: The geological setting for the use case, showing the four rotated fault blocks, with the three geological units that are making them up (reservoir rock units, top/base seal rock units) and the bounding faults

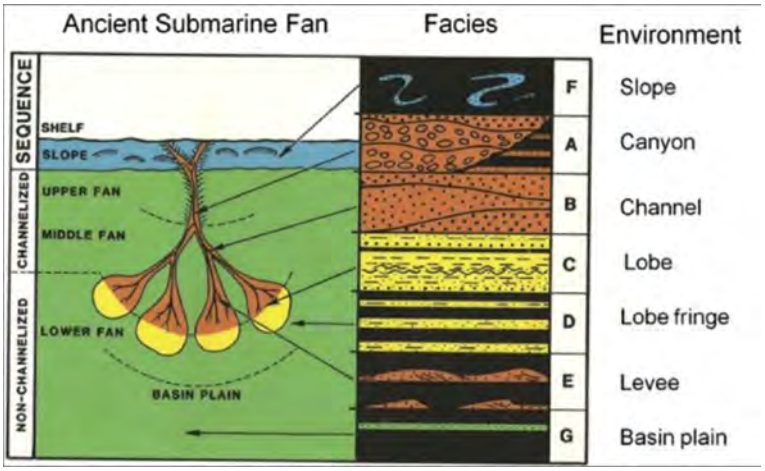


Figure 2: Ancient Submarine Fan, showing the main environments and their corresponding facies, in which the four fault blocks, representing the use case, have been deposited in (from proximal to distal: canyon, channel, levee, lobe, lobe fringe and basin plain). Canyon, channel and lobe represent the reservoir rock units, and basin plain represent the seal rock units.

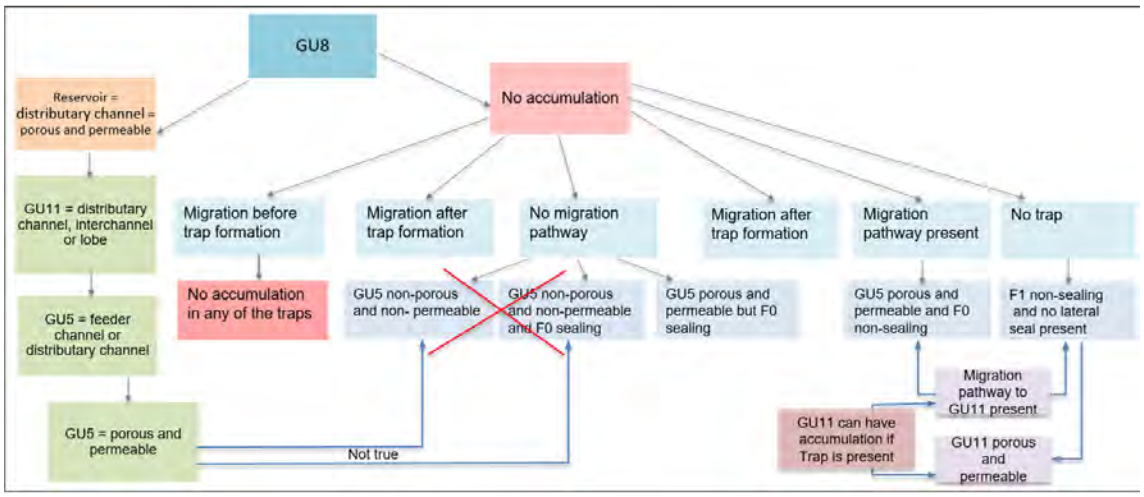


Figure 3: Manual reasoning of the use case

encompasses all of the disparate elements and processes of petroleum geology. The essential elements of a petroleum system include: source rock, reservoir rock, cap rock and overburden rock. Petroleum systems have two processes: trap formation and generation–migration–accumulation of hydrocarbons. If one of these elements or processes are not present than there is no petroleum system.

Use Case The use case chosen is addressed to explorationists in their process of *lead maturation*. The geological settings comprise a series of four rotated fault blocks, consisting of three geological units: base seal, reservoir and top seal (Figure 1), deposited in a marine environment, as a submarine fan (Figure 2). Source rock is present and has generated oil and gas, traps are fault-traps relying on faults being sealing or on lateral seal being present. Useful information is provided by the well X (marked by a red cross), drilled in the area of interest that did not encounter any hydrocarbons but the core and logs acquired tell us that the reservoir is present, of good quality and deposited in a distributary channel.

To demonstrate the necessity and usefulness of a tool for automated reasoning, a human reasoning example is shown in Figure 3. This example will answer the question: "Is it possible to have hydrocarbons accumulation in GU11, while having no hydrocarbons accumulation in GU8? The manual reasoning starts from analysing the data provided by the well, explaining why there is no hydrocarbons accumulation in GU8. After finding out the answer to this first question we proceed explaining under what circumstances there can be an accumulation in GU11. This might be deduced from this decision tree, however, the more complex the use case, the harder the reasoning gets. For certain, the manual reasoning is not always fail-free or complete.

3 Formalization of Geological Processes in Rewriting Logic

Rewriting logic is a computational logic and can be used as a semantic and logical framework. It was introduced in the 1990s and has been applied in various domains, such as hardware, software, logic systems, and computational systems. A rewrite theory $\mathcal{R} = (\Sigma, E, R)$ consists of a membership equational logic [5] theory (Σ, E)

and a set of (possibly conditional) rewrite rules R. As a semantic framework, (Σ,E) specifies the static aspects of distributed states. Σ is a set of declarations of sorts, subsorts, and function symbols. In Maude, an implementation of rewriting logic capable of executing rewriting logic theories, a function symbol f is declared with the syntax `op` $f : s_1 ... s_n \rightarrow s$, where $s_1 ... s_n$ are the sorts of its arguments, and s is its (value) sort. For example, we can define constructors of different environments in a submarine fan:

```
sort SubmarineFanEnv .
op   feederChannel       : -> SubmarineFanEnv .
op   distributaryChannel : -> SubmarineFanEnv .
op   interChannel        : -> SubmarineFanEnv .
op   lobe                : -> SubmarineFanEnv .
op   lobeFringe          : -> SubmarineFanEnv .
op   basinPlain          : -> SubmarineFanEnv .
```

E is a set of confluent and terminating (possibly conditional) equations. In Maude, equations are written with syntax `eq` $t = t'$ and `ceq` $t = t'$ `if cond`, where the latter expresses conditional equations. Rewrite rules R specify the dynamic aspects of a concurrent and distributed system, i.e. system's transition from a sub-distributed state to a new sub-distributed state. R is a set of (possibly conditional) labeled rewrite rules of the form `rl` $[l] : t \Rightarrow t'$ and `crl` $[l] : t \Rightarrow t'$ `if cond`, where the latter is a conditional rewrite rule. The rule specifies the system's transitions from an instance of t to the corresponding instance of t', where l is a label. Conditional rules apply only if their conditions hold. Operationally, a term is reduced to its normal form modulo a set of equational axioms before any rewrite rule is applied. This dynamic aspect of rewriting logic captures local concurrent transitions. With several local concurrent transitions performed at the same time, true concurrency is represented.

A *state* or *configuration* denotes a multi-set of objects and messages. The Maude simulation and execution starts with an initial state/configuration.

Rewriting Logic for Geology The evidence accumulated over the last twenty years strongly supports the claim that rewriting logic can rightfully be said to have "ϵ representational distance" [9] as a semantic and logical framework. That is, what is represented and its representation are often isomorphic structures. We exploit this property and together with the geologists, we apply rewriting logic in the domain of geology and define the geological rewriting theory $\mathcal{R}_{geo}$. Given a subsurface specified as a rewrite theory (Σ,E,R), rewriting logic then allows us to reason about the complex changes that are possible in the geological system, given the basic geological changes modeled as rewrite rules R. That is, we can then use (Σ,E,R) together with Maude and its supporting formal tools to simulate, study, and analyze geological dynamics. In particular, we can study in this way complex processes involving chains of geological changes and leading to multi-scenarios of hydrocarbon migration and accumulation.

In Figure 4 we show a conditional rewrite rule, which captures the geological process of hydrocarbon migration and accumulation through a fault. All the capitalized terms are variables. For example, GU_1 and GU_2 are placeholders for the identity numbers of geological units. RT is the placeholder for the type of


```
crl [migration-through-fault-fillToMaxClosure-accumulation-rule] :

⟨ GU1 : GeoUnit | Type: RT, Permeability: PMT1, Porosity: PRT1, SubmarineFan: SF1,
     DepositedIn: ENV, Hydrocarbon: HC1 ⟩
⟨ GU2 : GeoUnit | Type: sandstone, Permeability: PMT2, Porosity: PRT2,
    SubmarineFan: SF2, DepositedIn: ENV, Hydrocarbon: HC2 ⟩
⟨ N : Pathways | PType: fault(F, GU1, GU2) ⟩
trapformation(GU2, TPT, SPT, TFT)
accumulation(GU2, B)
⟹
⟨ GU1 : GeoUnit | Type: RT, Permeability: PMT1, Porosity: PRT1, SubmarineFan: SF1,
     DepositedIn: ENV, Hydrocarbon: HC1 ⟩
⟨ GU2 : GeoUnit | Type: sandstone, Permeability: PMT2, Porosity: PRT2,
    SubmarineFan: SF2, DepositedIn: ENV, Hydrocarbon: HC1 ⟩
⟨ N : Pathways | PType: fault(F, GU1, GU2) ⟩
trapformation(GU2, TPT, SPT, TFT)
accumulation(GU2, true)
if  HC1 =/= null and PMT2 == permeable and PRT2 == porous
    and (TPT == faultDependent-sealing or TPT == faultDependent-nonSealing)
    and SPT == fillToMaxClosure and isYounger(timeOf(Migration), TFT) .
```

Figure 4: A Rewrite Rule for Hydrocarbon Migration and Accumulation

geological unit such as shale or sandstone. PMT_1 and PMT_2 are the placeholders for the permeability of reservoirs such as permeable or non-permeable. PRT_1 and PRT_2 are placeholders for the porosity of reservoirs such as porous or non-porous. SF_1 and SF_2 are the placeholders for the environments of submarine fan such as feeder channel, distributary channel, inter channel, lobe, lobe fringe, or basin plain. ENV is the placeholder for the depositional environment. HC_1 and HC_2 are the placeholders for the identity numbers of hydrocarbon objects, i.e. **null** if hydrocarbon does not exist. F is a placeholder for the identity number of a fault. TPT is a placeholder capturing the type of traps. SPT is a placeholder capturing the maximum hydrocarbon accumulation level. TFT is the placeholder capturing the trap-formation time. B is a Boolean variable. This is a conditional rule. This rule can only be applied if GU_1 contains hydrocarbon, GU_2 is permeable and porous, the trap for GU_2 is fault-dependent and was formed before the hydrocarbon migration happened, and there is a migration pathway from GU_1 to GU_2 through fault F. After this rule is successfully applied, the state of the configuration expresses that GU_2 accumulates the same type of hydrocarbon as GU_1.

4 Knowledge Representation

In order to use rewriting logic for simulating these complex geological processes, we need to start with an initial state. In most cases, the end-user does not have complete knowledge about the original state of the system to simulate. This leads to an underdetermined description of the initial state, in which many initial states are possible. To determine which state is a possible state that makes sense for the given domain requires reasoning on a combination of the end-users knowledge about this specific case, and general domain knowledge about what is a meaningful state. However, this knowledge is purely static, i.e. it concerns only individual states, and not the evolution of one state to another. The latter is achieved by the runtime application of $\mathcal{R}_{geo}$.

To automate this task, we formalize both the general domain knowledge and


```
above(E1, E2) :- ontopof(E1, E2) .
above(E1, E2) :- ontopof(E1, E), above(E, E2) .
```

Figure 5: Example Prolog program defining the `above` relation.

the knowledge specific for this case, in such a way that a machine can reason on this knowledge and determine which state makes sense and which does not. Such formalization of knowledge is normally done using logic, typically first order logic, a modal logic, or a description logic (see e.g. [14] for an overview). For our purposes we have chosen to formalize the geological knowledge in Prolog [4]. Prolog can be viewed as both a declarative general-purpose programming language, and a first-order logical language for formalizing knowledge. In Prolog, knowledge is represented via facts and simple implications, both terminated by a dot. Facts have the form

$$\mathtt{H}(\vec{x}) \ .$$

and the implications have the form

$$\mathtt{H}(\vec{x}) \ \mathtt{:-}\ \mathtt{B}_1(\vec{y_1}), \mathtt{B}_2(\vec{y_2}), \ldots, \mathtt{B}_n(\vec{y_n}) \ .$$

where `H` and each $\mathtt{B}_i$ are relation names and $\vec{x}$ and each $\vec{y_i}$ are tuples of terms (constants or varibles). Furthermore, `:-` denotes left implication (i.e. $\leftarrow$) and comma is conjunction (i.e. $\wedge$). Negation (with negation-as-failure semantics [4]) is written $\backslash+$. Variables are denoted by a capital first letter, and everything else are constants. Figure 5 presents an example Prolog program that defines the `above` relation as the transitive closure of the `ontopof` relation, where `ontopof(A, B)` states that `A` is on top of `B`.

The Prolog formalization is responsible for taking the end-users knowledge about a scenario, and generate all possible initial states with respect to the temporal relationships between the elements. These temporal relationships can be determined from the objects' geometrical relationships. For example, a layer that is geometrically above another layer is also younger. For more complex relations, see the example below.

Our formalization could also have been done in a different logical language, such as OWL [7] (a type of description logic). OWL is more commonly used but less expressive. We plan to investigate the possibility to move (part of) the formalization to OWL in the future, to be able to better reuse already existing formalizations.

Example One important part of determining all possible proto-scenarios is to find the possible temporal relationships one can have between the events forming the different elements of a geological system. For example, we know that if a fault cuts a particular layer, then the layer is formed before the event that produced the fault. As noted above, a fault is younger than another if its cross-cutting topmost layer is above the cross-cutting topmost layer of the other. Accordingly, we define the approximate time of the formation of a fault by finding out the topmost layer it cuts. Both the Prolog-formalization of this relation between a fault and its topmost


```
top_layer(Fault, Layer) :-
    fault(Fault), geological_unit(Layer),
    goes_through(Fault, Layer),
    \+ (ontopof(OtherLayer, Layer),
        goes_through(Fault, OtherLayer)) .

younger_than(F1, F2) :-
    fault(F1), fault(F2),
    top_layer(F1, R1), top_layer(F2, R2),
    above(R1, R2) .
```

Figure 6: Prolog formalization of temporal knowledge.

```
geological_unit(layer1) .    fault(fault1) .
geological_unit(layer2) .    fault(fault2) .
geological_unit(layer3) .    goes_through(fault1, layer3) .
ontopof(layer1, layer2) .    goes_through(fault2, layer3) .
ontopof(layer2, layer3) .    goes_through(fault2, layer2) .
```

Figure 7: Prolog formalization of end-user knowledge.

layer and the temporal order of faults is presented as `top_layer` and `younger_than`, respectively, in Figure 6. The Prolog formalization of the former can be read as: "`Fault` has top-layer `Layer` if `Fault` is a fault, `Layer` is a geological unit, `Fault` goes through `Layer`, and there does not exist a layer that is on top of `Layer` which `Fault` goes through".

In Figure 7 we can see an example knowledge base describing some known facts input by the end-user. From these facts and the definitions of Figure 6 we can e.g. derive that `fault2`'s topmost layer is `layer2` and `fault2` is younger than `fault1`.

5 Scalable Infrastructure

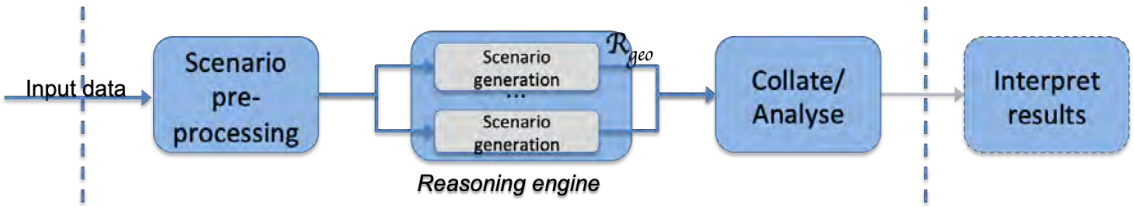


Figure 8: Work flow

In order to run the sequence of tools in a reliable and efficient way, we need to establish a pipeline, see Figure 8, that encapsulates and automates all the different steps of the process and at the same time is able to scale the simulations to shorten computing time. As a secondary priority the preparation and installation time on systems supposed to run this pipeline should be kept as small as possible.

The decision to use rewriting logic in the simulation part of the pipeline enables us to scale and run the simulations on different systems in parallel. This allows using affordable standard or cloud-based hardware while still keeping the required time per use case within reasonable bounds. The chosen approach is to establish a computation framework which consists of a controller and several worker nodes. The controller dictates and schedules the workflow while being agnostic of any domain knowledge. All computation work is done by the worker nodes which have the necessary domain knowledge and all required tools installed. Each worker node that is started automatically registers with its controller. The controller exposes a few webservices to allow user interactions such as starting a computation with a given set of data, collecting results afterwards, as well as giving access to some status information. The first step taken in the pipeline is the creation of proto-scenarios from the input data which was given into the system when submitting the use case. After this is successfully done, the number of simulations that have to be run is known to the system for the first time. According to this number, work packages for the proto-scenarios are prepared and queued. The controller then starts to asynchronously distribute these work packages to all available and currently idle worker nodes in set intervals. Each of the nodes then runs a simulation based on the proto-scenario it received with the work package from the controller and returns a final configuration as result after finishing the simulation. Each result/final configuration is reported back to the controller and also persisted into a database for later evaluation. After all work packages (all simulations) are completed, the resulting data can be collected by the user by calling the appropriate webservice.

The controller as well as the worker nodes are implemented as Java Spring applications and use webservices to communicate to each other and the outside world. The database currently used is PostgreSQL, but could basically be any relational database that can be used in connection with the Spring Framework and JPA (Java Persistence API). To increase portability and reduce the effort for installation and configuration, controller and worker nodes are encapsulated into Docker images (with the worker node image also containing everything necessary to create the proto-scenarios and run simulations). All worker nodes use the same implementation and Docker image, only being different in the configuration of the Docker container created when first started (namely container name and port claimed on the host machine, plus IP address in cases where different machines are used). This allows us to keep administrative work when deploying the pipeline environment to a minimum.

6 Use Case: multi-scenario analysis of hydrocarbon migration and accumulation

This section shows the analysis results of the use case presented in Section 2. Note that a geological unit GU defined in Section 2 is presented as GeoUnit in our tool. Based on the facts about depositional environment, the observation of geometrical relations, and the assumptions of sealing capacity, hydrocarbon migration time, and reservoir locations, our reasoning engine generates 3024 proto-scenarios for our use case as the input configurations (i.e., initial states) for the Maude engine. Multi-scenarios of geological processes are computed by Maude simulation. Currently, the total execution time, including the time for generating all 3024 proto-scenarios plus the complete simulation time, using 4 worker nodes in parallel on the same

Figure 9: An Example of Scenario Explanation

machine is about 77 minutes. We observe that the number of scenarios for such a small use case is already pretty high. This shows that the comprehension of the geologically oriented subsurface evaluation is likely beyond human capacity and requires digital support. Therefore, we design an interactive tool interface that provides users various options to constrain and aggregate the scenarios computed by Maude. The more constraints given, the less scenarios remained for investigation. Take an example of our use case, by giving three constraints: (i) GeoUnit 8 is in distributary channel, (ii) GeoUnit 8 does not have accumulation, and (iii) migration happened *after* the trap was formed, the number of scenarios remained is reduced from 3024 to 14. Based on the observations, evidence (i) and (ii), and assumption (iii), these 14 scenarios explain all the possible reasons of *why* GeoUnit 8 in our use case is a dry well. Figure 9 shows one of the scenario explanations by our engine among these 14 scenarios. The yellow part highlights the observation, which shows the location of the source rock and the type of the hydrocarbon. The green parts highlight the evidence (i) and (ii). The blue part highlights the assumption (iii), i.e. a trap is formed when the corresponding fault is ceased. In addition, the engine is capable of providing users the following information: the hydrocarbon migration pathway, the sealing capacity of faults, how a reservoir is trapped or why a reservoir cannot be trapped, whether a reservoir has hydrocarbon accumulation or not and the maximum accumulation volume if any, and the simulation history of hydrocarbon migration and accumulation. By continuing with the same approach of constraining scenarios, our tool shows not only that it is possible to find hydrocarbon in the surrounding area of the dry well GeoUnit 8 but also explains *why* it is possible and *how* the hydrocarbon can be accumulated. Figure 10 is a screenshot of the tool that shows under which constraints we may find accumulation in GeoUnit 11 and the number of scenarios is reduced from 14 to 8. The corresponding explanations of these 8 scenarios are provided by our tool and the format is similar to the one shown in Figure 9. Note that during the process of constraining scenarios, if none of the remaining scenarios satisfies the given constraint, the tool shows *"No cases are matched"*.

In order to assist the users during the process of constraining scenarios, our tool provides multiple options for selecting and seeing the variations of a specific


```
8 scenarios.

############## What you have chosen to keep in the search space: ##############
(1) GeoUnit 8 is in distributaryChannel.
(2) GeoUnit 8 does not have accumulation.
(3) Migration happened AFTER the chosen fault 1 was ceased.
(4) GeoUnit 11 has accumulation.
################################################################################
```

Figure 10: The tool shows that it is possible to find accumulation in GeoUnit 11

```
**********************************************
GeoUnit 8 was deposited in distributaryChannel.
GeoUnit 8 is permeable and porous.
**********************************************

1 variation.
```

```
**********************************************
GeoUnit 11 was deposited in distributaryChannel.
GeoUnit 11 is permeable and porous.
**********************************************

**********************************************
GeoUnit 11 was deposited in interChannel.
GeoUnit 11 is non-permeable and non-porous.
**********************************************

**********************************************
GeoUnit 11 was deposited in lobe.
GeoUnit 11 is permeable and porous.
**********************************************

3 variations.
```

Figure 11: An example of how the tool shows the variations of a target category

target category, for example the user can choose to see only the submarine fan environment and the corresponding permeability and porosity of a reservoir, or all the possible migration pathways up to a certain reservoir, or the sealing capacity of a specific fault, or the trap-formation of a reservoir, etc. These supplementary information pinpoint the evidence across scenarios based on the already given constraints. Figure 11 shows an example of how our tool presents these variations. While GeoUnit 8 is in the distributary channel, GeoUnit 11 can only be in either distributary channel, inter-channel or lobe.

7 Conclusion

In this paper we have shown how our method and tool can support geological reasoning through historical narratives. By exploring, explaining and constraining scenarios based on observations, evidence and assumptions, we are able to assist explorationists in communicate and systematically compare and assess different hypothetical geological scenarios before deciding which scenario to pursue when assessing exploration prospects. Our methodology provides a qualitative form of reasoning and addresses a gap that is currently existing between geologically oriented work processes and the digital tools available.

The framework offers a novel application of rewriting logic that holds the potential of extending the scope of formal methods beyond the conventional domain of computing.

Acknowledgment The authors would like to thank Hallgrim Ludvigsen from Schlumberger, Michael Heeremans from the Department of Geosciences at University of Oslo, Adnan Latif from SIRIUS, and Vegar Skaret for their fruitful discussions and constructive feedbacks.

References

[1] L. Bachmair, editor. *Rewriting Techniques and Applications, 11th International Conference, RTA 2000, Norwich, UK, July 10-12, 2000, Proceedings*, volume 1833 of *Lecture Notes in Computer Science*. Springer, 2000.

[2] C. E. Bond. Uncertainty in structural interpretation: Lessons to be learnt. *Journal of Structural Geology*, 74:185 – 200, 2015.

[3] C. E. Bond, A. Gibbs, Z. Shipton, and S. Jones. What do you think this is? "Conceptual uncertainty" In geoscience interpretation. *GSA Today*, 17(11):4–10, 12 2007.

[4] I. Bratko. *Prolog programming for artificial intelligence*. Pearson education, 2001.

[5] M. Clavel, F. Durán, S. Eker, P. Lincoln, N. Martí-Oliet, J. Meseguer, and C. L. Talcott, editors. *All About Maude - A High-Performance Logical Framework, How to Specify, Program and Verify Systems in Rewriting Logic*, volume 4350 of *Lecture Notes in Computer Science*. Springer, 2007.

[6] Y. Gil, S. A. Pierce, H. Babaie, A. Banerjee, K. Borne, G. Bust, M. Cheatham, I. Ebert-Uphoff, C. Gomes, M. Hill, J. Horel, L. Hsu, J. Kinter, C. Knoblock, D. Krum, V. Kumar, P. Lermusiaux, Y. Liu, C. North, V. Pankratius, S. Peters, B. Plale, A. Pope, S. Ravela, J. Restrepo, A. Ridley, H. Samet, and S. Shekhar. Intelligent systems for geosciences: An essential research agenda. *Commun. ACM*, 62(1):76–84, Dec. 2018.

[7] P. Hitzler, M. Krötzsch, B. Parsia, P. F. Patel-Schneider, and S. Rudolph. Owl 2 web ontology language primer. *W3C recommendation*, 27(1):123, 2009.

[8] L. Mastella, M. Perrin, Y. Aït-Ameur, M. Abel, and J.-F. Rainaud. Formalizing geological knowledge through ontologies and semantic annotation. 4:2473–2477, 06 2008.

[9] J. Meseguer. Twenty years of rewriting logic. *The Journal of Logic and Algebraic Programming*, 81(7-8):721–781, 2012.

[10] E. Monteiro, T. Østerlie, E. Parmiggiani, and M. Mikalsen. Quantifying quality: Towards a post-humanist perspective on sensemaking. pages 48–63, Cham, 2018. Springer International Publishing.

[11] N. Oreskes, K. Shrader-Frechette, and K. Belitz. Verification, validation, and confirmation of numerical models in the earth sciences. *Science*, 263(5147):641–646, 1994.

[12] T. Østerlie, E. Parmiggiani, and E. Monteiro. Information infrastructure in the face of irreducible uncertainty. In *In 5th Innovation in Information Infrastructures (III) Workshop, 7th-9th November, Rome.*, 11 2017.

[13] A. K. Turner and C. W. Gable. A review of geological modeling. 01 2007.

[14] F. Van Harmelen, V. Lifschitz, and B. Porter. *Handbook of knowledge representation*, volume 1. Elsevier, 2008.

Exploring future C++ features within a geometric modeling context

Jostein Bratlie and Rune Dalmo

UiT - the Arctic University of Norway
October 20, 2019

Abstract

The development of the C++ programming language and its standard library has undergone a renaissance since the emerge of the C++11 standard. Through modern features, expansions to the standard library and simplifications the language has become more relevant than ever before. Comparing past and future feature sets (C++17, C++20, ...) is somewhat similar to comparing different programming languages. In this article we address how new and upcoming features of the language can be utilized to ease the development of domain specific application areas through features such as *non-intrusive inheritance*, *semantic compile-time polymorphism* and type safety. We provide representative examples by application to differential geometry by modeling a hierarchical structure for parametric object evaluation.

1 Introduction

Differential geometry is a field within mathematics where the theory of differential manifolds [2], such as curves and surfaces, and Riemannian spaces [3] are central. The R&D group Simulations at UiT Narvik has conducted research within this field since the mid 1990s, with a special focus on modeling techniques of spatial objects and spline theory. Realized implementations of these mathematical and geometric concepts and constructions are considered as important tools for visual understanding and verification throughout the R&D work process. As a result, an in-house software library named GMlib [8], which aids in this aspect of the R&D work, has emerged over the years. GMlib is a fairly clean C++ library with only a few external dependencies besides the standard template library (STL). However, due to its age and maturity, the code base, written in C++98, consists of major parts of legacy code which now faces a set of challenges in the future. After an internal review in 2017 it was decided to construct a new prototype where the focus should be on statically descriptive tools for the differential geometric modeling aspects of our R&D work. In addition to historical reasons for continuing to use C++ we are fond of its closeness to hardware, optimization potential, no-overhead principles,

This paper was presented at the NIK-2019 conference; see http://www.nik.no/.

and the support for generating bindings to other programming languages, such as, for example, Python. The major focus points for the development process are the following:

- light-weight header-only library which follows the C++ core language guidelines [7] and targets the latest C++ standard (current: C++17 [6]),
- restrict library maintenance to key R&D activities, and
- optimize for rapid prototyping of domain-specific features and strong types.

After an analysis we decided that the dependency needs of the new code base was limited to three C++ libraries, besides the STL; namely, Blaze [4] for basic linear algebra, OpenMesh [1] for triangular- and polygonal meshing and topology, and Qt [12] for matters related to graphics, application framework and demo control. The libraries were selected due to their use of modern C++ features, which we depend on, their proven stability, and their ability to target a variety of architectures; from micro-controllers to smartphones and desktop- and backend systems. Both the old and new prototype libraries and demo applications can be found at the Nordic e-Infrastructure Collaboration (NeIC) software hub [14], under the project "gmlib".

2 Problem setting

The intended use of the new software library is as a tool to verify, very strictly, abstract mathematical concepts and constructions, where compile-time rather than runtime-rules are utilized to catch constructional errors. A way to realize this is by using strongly typed expressions. In C++ this translates into a subset of features which enhances the static type system.

This article focuses on certain new and future language features and how they can be utilized to enforce our R&D work. The article introduces a set of design techniques and discusses how they are beneficial in programming of geometric modeling concepts. The techniques are exemplified via building a small application programming interface (API) throughout the article. This API is used to model embedded spatial-hierarchical parameterized objects. Examples of such constructions can be

- a sphere, torus or Bézier surface, which all are two-dimensional objects that can be embedded in a three-dimensional space,
- or a multi-hierarchical object chain, such as the ones illustrated in fig. 1,

of which the latter example is illustrated in the left part of fig. 1. Such problems, where the concatenated and multi-hierarchical are relevant scenarios, quickly become complex and challenging to model programmatically with confidence.

Implementation wise, if considering these objects as part of an infinite set, the natural modeling technique used in C++ is by utilization of abstract interfaces and virtual functions. This in turn leads to the use of techniques which builds on dynamic casting and runtime type information (RTTI). On the other hand, by considering these hierarchical objects as parts of a finite set of objects we can use a different set of C++ modeling techniques and utilize much more of the static type system, which again alleviates the use of runtime-centric techniques.

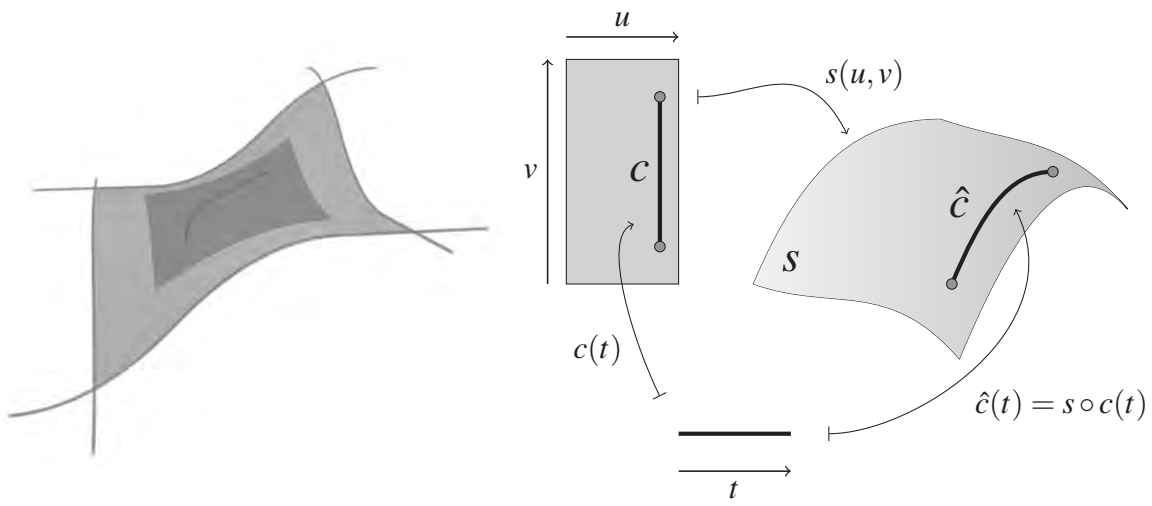


Figure 1: Left: a rendering of inheritance-hierarchical parametric objects; On the boundary we have b-spline curves ($\mathbb{R} \mapsto \mathbb{R}^3$), from these we have sub-curves ($\mathbb{R} \mapsto \mathbb{R}$) which again are input to a bi-linear Coons patch surface, ($\mathbb{R}^2 \mapsto \mathbb{R}^3$). From this Coons patch we have a sub-surface, plane ($\mathbb{R}^2 \mapsto \mathbb{R}^2$), and finally from this we have a sub-curve, 5^{th}-degree Hermite curve ($\mathbb{R} \mapsto \mathbb{R}^2 \mapsto \mathbb{R}^3$). Right: a mathematical sketch of spatial mappings: $\hat{c}(t) = s \circ c(t)$

The article is divided into three sections; *principal design techniques*, an exemplified *parametric object API* and a section which discusses pros and cons of these techniques and explores new and future language features. The considered new and future language features primarily enhances these principal design techniques as well as alleviates boiler plate code and therefore reduces the complexity and amount of programmatic errors.

3 Principal design techniques

C++ is a mature well behaving language possessing a set of features which are desirable for system design. One of these features is its well defined type system, which consists of a finite set of built-in types and a single user-defined type: `class` /`struct`. The user defined type is used to describe how objects are laid out in memory. Its associated member functions describe an object's semantics and related operations. As the language only consists of a single user-defined type, the inheritance model is equally well defined, and sub-objects with altered semantics are handled by the use of virtual tables. Furthermore, this yields a memory model that reflects the system memory for a target architecture and the requirement that all types are known at the point of compilation.

Due to its memory model, C++ supports casting between dynamic objects via runtime mechanisms. Dynamic casting is a powerful feature, but it comes with a possible high cost with respect to resources. The complexity of this and related mechanisms depend on the relationships between the dynamic objects and the hierarchical class structure. Additionally, that introduces a weak object model since an object's inheritance from a base type to a deduced type is subject to runtime checks. Nevertheless, this is the de facto method used to design object oriented problems, where the possible objects from a given class hierarchy represents an infinite set where there is a parent-child relationship.

We propose to take an alternative approach to such design problems where we consider our infinite set of potential sub-classes to be finite at compile time. This allows us to use other mechanisms besides virtual tables and for this purpose disable the RTTI. In the following sections this is exemplified through a small proposed API. We start by discussing a set of key C++ implementation techniques used to realize the static flavor of polymorphism, namely,

- non-intrusive inheritance,
- semantic compile-time polymorphism, and
- aggregated properties and transitive constructors.

Non-intrusive inheritance

The term non-intrusive inheritance refers to an inheritance model where a derived class can inherit a realized base class, possibly unknown at the time of design, where the realized base class adheres to a set of static rules such that the memory model is preserved at the time of compilation. For instance, the derived class is defined in a library, but the realized base class can be designed by the library's end-user. To facilitate non-intrusive inheritance one can utilize variadic class templates; `template <typename... Ts>`, which accepts one or more template parameters into a template pack. The ellipsis operator, `...`, is then used to unpack the template arguments:

```
struct SceneObject {}; struct Base {};

template <typename... Base_Ts> struct Object : Base_Ts... {};

using NonInheritingObject = Object<>;
using ObjectOfBase        = Object<Base>;
using SceneObjectOfBase   = Object<SceneObject,Base>;
```

Semantic compile-time polymorphism

The standard mechanisms used in C++ to implement polymorphism are function overloading and virtual functions. The latter enables runtime-dynamic types where the semantics of a derived object can be reimplemented for a given virtual function in a derived class and then be called for the derived object dynamically through a pointer of the base class type. By flattening the structure, using a middle layer *kernel* type, utilizing *non-intrusive inheritance*, and exploiting function overloading, we can mimic much of the same behaviour for semantic compile-time polymorphism. Consider the following type definitions:

```
                    struct Base {};
<typename Kernel_T> struct Object : Kernel_T {};

template <Base_T> struct KernelA : Base_T {
  auto evaluate( int par ) { return Vector{par}; }
  auto evaluate() { return double(0.5); } };

template <Base_T> struct KernelB : Base_T {
  auto evaluate() { return double(0.5); } };

template <typename Obj_T, typename... Params_Ts>
auto evaluate(Obj_T t, Param_Ts... params){
  return t.evaluate(params...); }
```

The `Object` class inherits a potential kernel which again inherits a common base.

```
using ObjectA = Object<KernelA<Base>>;
using ObjectB = Object<KernelB<Base>>;
```


The defined objects `ObjectA` and `ObjectB` share the same polymorphic API, where the return type is deduced by binding to the unconstrained placeholder type `auto`. Both object types can then be utilized in polymorphic contexts, as follows:

```
auto A = ObjectA; auto B = ObjectB;
auto a = evaluate(A,4); // Ok
auto b = evaluate(A);   // Ok
int  c = evaluate(A,4); // Compile error:    return type mismatch
auto d = evaluate(B,4); // Compile error: parameter type mismatch
```

Aggregated properties

Static properties can be utilized to hold compile-time types or sizes such as an object's spatial dimension or base unit type. These are properties which are changed for instance to increase computational precision (e.g. *long double* or *integer* over *floats*), or to specify the dimension of an embedding space (e.g. two-dimensional, $\mathbb{R}^2$, instead of three-dimensional, $\mathbb{R}^3$). Such internal properties can be defined using static constant expressions or `using` type definitions as part of *non-intrusive inheritance* (is-a relationship), or as internal aggregated type definitions (is-a or has-a relationship). Static properties are not part of an instantiated object by definition, but can be stored if needed. As such, this mechanism is powerful and inflicts no run-time penalty. However, such properties are not aggregated through inheritance and must therefore be explicitly defined, as follows:

```
struct StaticProperties {
  using                Unit      = int;
  static constexpr auto Dimension = 3ul; };

template <typename Props_T> struct Object {
  using                Unit      = typename Props_T::Unit;
  static constexpr auto Dimension = Props_T::Dimension; };
```

Transitive constructor inheritance

When utilizing a *non-intrusive inheritance model* we need a mechanism to transitively aggregate the future constructors of a class-type we do no yet know exists. This can be achieved using a combination of *templated constructors*, *variadic templates* and *perfect forwarding*. Let us exemplify it through an internal sub-object construction with a has-a relationship, as such:

```
template <typename SubObj_T> struct Object {
  SubObj_T  sub;
  template <typename... Ts> Object(Ts&&... ts)
    : sub(std::forward<Ts>(ts)...){} };
```

Given an object type `A` for an internal sub-object with the following constructors

```
struct A {
  explicit A(float f) {}
  explicit A(std::array<int,3> i) {} };
```

yields the following natural, but implicit aggregate, syntax:

```
auto obj_one Object<A>(2.3f);
auto obj_two Object<A>({1,2,3});
```

Simplifying API through type deduction

Finally, it is useful to simplify an API without introducing new types. This can be achieved by using type definition via the `using` directive. Contrary to the pre-C++11 standard's `typedef` directive, `using` directives can be templated. Consider this API for a parametric object

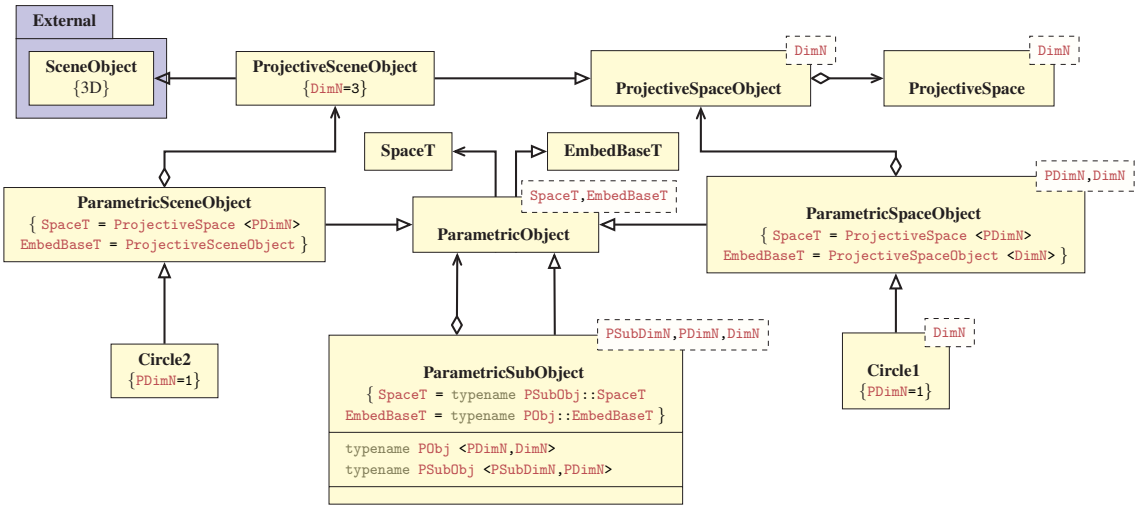


Figure 2: A templated model showing inheritance and semantic polymorphism.

```
template <size_t PSpaceDim, size_t EmbedDim> struct Object;
```

Specialized semantic APIs for curves and surfaces can then be constructed as follows:

```
template <size_t EmbedDim> using Curve   = Object<1ul,EmbedDim>;
template <size_t EmbedDim> using Surface = Object<2ul,EmbedDim>;
```

4 The parametric object API

Utilizing the above techniques, this section introduces a representative example API for the purpose of building parametric object spaces of the kind illustrated in fig. 1. A simplified API type diagram is shown in fig. 2.

Projective space

The projective space is common when working with visual parametric modeling. It is an affine mathematical space (see e.g. literature on differential- or Riemannian geometry [2, 3]), consisting of points and vectors with respect to a perspective projection such that parallel lines end up in a point at infinity, e.g., the horizon. The projective space meta-data description consists of its affine space containers for points and vectors in addition to a numerical unit data type and a space dimension. The space is usually represented as a frame in the shape of a homogeneous matrix. We model the meta-information of this mathematical space statically by using an *information only structure*:

```
template <typename Unit_T, size_t Dim_T> struct ProjectiveSpace {
  static constexpr auto Dim = Dim_T;   // Space dimension
  using Unit    = Unit_T;              // Storage unit type
  using Point   = Vector<Unit, Dim>;   // Affine space data types
  using Vector  = Vector<Unit, Dim>;
  using Frame   = Matrix<Unit, Dim>; };
```

From this, one can model a projective space object using a static *has-a* relationship.

```
template <typename EmbedSpace_T> struct ProjectiveSpaceObject {
  using EmbedSpace = EmbedSpace_T;         // Projective space
  using Unit = typename EmbedSpace::Unit; // Type aggregation
  /* ... */
  Frame frame = {/* identity */};          // Affine space data structure
  void translate( Vector ) { /*math*/ };  // Affine operations
  /* ... */ };
```


Parametric objects

A parametric object is a user defined type describing a differential mapping from an n-dimensional parameter space into a m-dimensional embed space, under certain restrictions. This can be modeled using the principles of *non-intrusive inheritance*, where a parametric object inherits a template kernel type, `Kernel_T`, realizing the actual parametric object.

```
template <typename Kernel_T>
struct ParametricObjectImpl : Kernel_T {
  // Context
  using Kernel = Kernel_T;
  // Embed Space
  using EmbedSpaceObj = typename Kernel::EmbedSpaceObj;
  using EmbedSpace    = typename Kernel::EmbedSpace;
  using Unit          = typename Kernel::Unit;
  /* ... */
  // Parametric space
  using PSpace        = typename Kernel::PSpace;
  using PSpaceUnit    = typename Kernel::PSpaceUnit;
  /* ... */
```

The specific space types are inherited from the kernel as aggregated properties and the member functions are inherited from the kernel according to the rules of inheritance. These can be used to construct chaining operations; such as transformations or evaluation with respect to a *parent* space (`frame`).

```
  auto evaluateParent(PSpacePoint par) {
    return frame * evaluate(par); }
  /* ... */
```

The kernel's constructor is inherited using *transitive constructor inheritance*, effectivly translating the kernel's constructor into the parametric object's own constructor. This can also be done for other *internal* kernel methods, such as private sub-methods.

```
  template <typename... Ts>
  explicit ParametricObjectImpl(Ts&&... ts)
    : Kernel(std::forward<Ts>(ts)...) { }
}; // END ParametricObjectImpl
```

A parametric object kernel

A specialized parametric object is then realized as a parametric object of a provided kernel. As an example, the parametric object kernel, `CircleKernel`, is constructed as a user defined type inheriting the projective space object as a template type, `EmbedSpaceObj_T`. Then as with the parametric object itself we use *aggregated properties* to inherit the static types. Additionally we define equivalent projective space types for the parameter space itself and extra utility types, prefixed `PSpace`. This is exemplified using a parameterized circle $C(t) = (x(t), y(t), z(t)) : \mathbb{R} \mapsto \mathbb{R}^3$.

```
template <typename  EmbedSpaceObj_T> struct CircleKernel
  : EmbedSpaceObj_T {
  using EmbedSpaceObj= EmbedSpaceObj_T;                  // Embed Space
  using EmbedSpace    = typename EmbedSpaceObj::EmbedSpace;
  /* ... */
  using PSpace        = space::ProjectiveSpace<1ul,Unit>; // PSpace
  using PSpaceUnit    = typename PSpace::Unit;
  static constexpr auto PSpaceDim = PSpace::Dim;
  /* ... */
  using PSpaceBoolArray = array<bool, PSpaceDim>;        // Utility
  using PSpaceSizeArray = array<size_t, PSpaceDim>;
```

The above code in fact represents boilerplate code for any parametric curve. The kernel-specifics for the parametric circle, such as data members, constructor and semantic polymorphic functions, can then be defined as follows:


```
  Unit r;                                          // Member: radius
  template <typename... Ts>                        // Transitive constructor
  CircleKernel(Unit radius = 3, Ts&&... ts)
    : Base(std::forward<Ts>(ts)...), r{radius} {}
                                                   // Semantic polymorphic functions
  auto            evaluate(PSpacePoint par) const {
     const auto& [t] = par;

     const auto x    = r * std::cos(t);
     const auto y    = r * std::sin(t);

     if      constexpr (Dim == 2) return Point{x, y};    // 2D
     else if constexpr (Dim == 3) return Point{x, y, 0}; // 3D
  }
  PSpaceBoolArray isClosed()       const { return {true}; }
  PSpacePoint     startParameter() const { return {0}; }
  PSpacePoint     endParameter()   const { return {2*M_PI}; }
}; // END CircleKernel
```

The template dependent *constexpr if* statement enables us to specify different return types for each supported embed space dimension.

Sub-space objects

The driving motivation to develop such an API has been to enable strong typing of hierarchical object structures such as the ones addressed in section 2. Contrary to the basic parametric object, a sub-object kernel is defined by a parametric object which is embedded in the parameter space of another parametric object. This can be achieved by defining the kernel and implementation of the sub-object slightly different to the parametric object type itself. However, as an invariant, the internal type-definitions should be kept identical to the ones of the parametric object, such that sub-object chains can be defined.

The kernel is defined from three types; the subbed parametric object, `ParametricObject_T`, a parametric object embedded in that parametric object's parametric space, `PSpaceObject_T`, and an embed space object, `EmbedSpaceObject_T`, which usually is the same as the embed space of the sub-object. This gives us a parametric object with three distinct spaces: the embed space of the parametric object, `EmbedSpace`, the parametric space of the sub-object, `PSpace`, and the parametric space of the parametric object, `ParametricObject_PSpace`.

```
template <typename PSpaceObject_T, typename ParametricObject_T,
          typename EmbedSpaceObject_T>
struct CurveInSurfaceKernel : EmbedSpaceObject_T {
  using PSpaceObject            = PSpaceObject_T;      // Context
  using ParametricObject        = ParametricObject_T;
  using EmbedSpaceObject        = EmbedSpaceObject_T;

  using EmbedSpace              = typename EmbedSpaceObject::EmbedSpace;
  using PSpace                  = typename PSpaceObject::PSpace;
  using ParametricObject_PSpace = typename ParametricObject::PSpace;

  using Unit                    = typename EmbedSpace::Unit;
  /* ... */
```

This leaves us with a similar boilerplate code as for the parametric circle kernel, but with an additional type set representing the common parametric space. Continuing, the sub-object's data members, constructor and polymorphic parametric object methods are defined as follows.

```
  PSpaceObject      pspace_object;      // Members
  ParametricObject* parametric_object;

  template <typename... Ts>             // Transitive constructor
  explicit SubCurveKernel(ParametricObject* obj, Ts&&... ts)
  : Base(), pspace_object(std::forward<Ts>(ts)...),
                          parametric_object{obj} {} };

  auto evaluate(PSpacePoint par) const  // Semantic polymorphic functions
  {
```



```
    const auto pspace_res = pspace_object.evaluate(par);
    const auto parametric_object_res =
      parametric_object->evaluate(pspace_res);
    return parametric_object_res;
  }
  auto isClosed() const { return pspace_object.isClosed(); }
  /* ... */
}; // END CurveInSurfaceKernel
```

Parametric sub-object implementation type

The implementation type for the parametric sub-object type is a small variation of the parametric object's type. Types are aggregated from the sub-object-in-object kernel which transitively calls the right constructor combination. Again, it is important that it retains the same type definition interface as the `ParametricObjectImpl` such that it also can be used in sub-object chain constructions.

```
template <typename Kernel_T> struct ParametricSubObjectImpl
  : ParametricObjectImpl<Kernel_T> {
  // Context
  using Kernel           = Kernel_T;
  using PSpaceObject     = typename Kernel::PSpaceObject;
  using ParametricObject = typename Kernel::ParametricObject;
  using EmbedSpaceObject = typename Kernel::EmbedSpaceObject;
  // Aggregated types
  using PSpace      = typename Kernel::PSpace;
  using EmbedSpace = typename EmbedSpaceObject::EmbedSpace;
  using ParametricObject_PSpace =
    typename Kernel::ParametricObject_PSpace;
```

Object creation is handled through a transitive constructor, which can be enforced by deleting the sub-objects default constructor.

```
  ParametricSubObjectImpl() = delete; // Delete default constructor
  template <typename... Ts>          // Transitive Constructor
  explicit ParametricSubObjectImpl(ParametricObject* obj,
                                   Ts&&... ts)
    : Base(obj, std::forward<Ts>(ts)...) {}
}; // END ParametricSubObjectImpl
```

User space API

C++ template code naturally results in low readability API's, however, as discussed in section 3, this can be mended utilizing `using` type-definitions:

```
// ParametricObject
template <template <typename> typename Kernel_T,
          typename EmbedSpaceObject_T>
using ParametricObject
  = ParametricObjectImpl<Kernel_T<EmbedSpaceObject_T>>;

// ParametricSubObject
template <template <typename> typename PSpaceKernel_T,
          template <typename, typename, typename>
          typename Kernel_T,
          typename ParametricObject_T>
using ParametricSubObject =
  ParametricSubObjectImpl<Kernel_T<
    ParametricObjectImpl<PSpaceKernel_T<ProjectiveSpaceObject<
      typename ParametricObject_T::PSpace>>>,
    ParametricObject_T,
    ProjectiveSpaceObject<
      typename ParametricObject_T::EmbedSpace>>>;
```

which leads to the following API usage:

```
using ProjSpace    = ProjectiveSpace<double,3ul>;
using ProjSpaceObj = ProjectiveSpaceObject<ProjSpace>;
using Circle       = ParametricObject    <CircleKernel, ProjSpaceObj>;
using Torus        = ParametricObject    <TorusKernel,  ProjSpaceObj>;
using SubCircle    = ParametricSubObject <CircleKernel, SubCurveKernel, Torus>;

auto circle          = Circle();
auto torus           = Torus();
auto torus_subcircle = SubCircle(&torus, 2.0);
```


5 Concluding remarks

Utilizing the principles of *non-intrusive inheritance* and *semantic compile time polymorphism* we can redefine the object space of our modeling problem from an infinite to a finite set of possible objects. This allows us to design our C++ programs using static compile time techniques, which in turn lets us, most notably; to *move viable logic from run-time to compile-time*, and to apply *stronger types*. Consequently, fewer potential runtime errors occur since they can be caught at compile time and we can dismiss entire type categories by definition. Furthermore, the paradigm enables us to apply static analysis tools to problems where we earlier had to perform dynamic analysis. This again reduces errors and potentially leads to faster-runtime code; for example by dismissing branching opportunities.

On the other hand template programming makes libraries more complex, which again puts more responsibility on library programmers to construct understandable APIs. One drawback of a templated codebase is increased compile times and compile time resources. If not handled with care the increase in compile time and resources can grow exponentially. However, this will be alleviated with the upcoming C++20 feature; *modules*. Modules will mitigate the compile time overhead endured from repeated recursive header includes as well as the header inclusion mangling itself. Another drawback, if not handled with care, is the combinatoric nature of templates. As all function candidates, types and branches must be known at compile time, all permutations of possible template combinations are computed and evaluated by the compiler. This can then make both compile times and size of executables grow exponentially.

6 Future work

Let us take a look at future language features which will further help the verification of our R&D work and ease devlopment of such hierarchical class structures. C++20 introduced a new set of features. This is the biggest jump in a standarized feature level since the emerge of C++11. In this section we argue what we see as the most prominent feature with respect to the current proposed API, namely *Concepts*. Furthermore we briefly touch on a set of future features which are in development and planned for inclusion in standardized C++ over the next decade.

Concepts

In today's codebase *concepts* [10] are represented by *type-traits* and enforced through the SFINAE-mechanism (substitution failure is not an error). SFINAE together with *type-traits* limits the type space a template variable can occupy, and is used to remove template function candidates, at compile time, if type deduction fails.

Concepts on the other hand builds on the placeholder type `auto`. The type `auto` is the value-equivalent of `typename`. It is deduced at compile time, upon definition. With

```
auto i = int{4};
```

the type of the variable `i` is deduced to integer, while in the following example the type is deduced to the return-type value of the factory function.

```
auto curve = circleFactory(3.0);
```

A concept type is a restricted placeholder type. Imagine an algorithm which finds the intersection between two parametric objects. In today's codebase a generic

intersect algorithm is represented as a templated function over two unconstrained typenames; returning some result:

```
template <typename ObjA_T, typename ObjB_T> auto intersect(ObjA_T, ObjB_T);
```

In theory, types `ObjA_T` and `ObjB_T` can take any type. The error checking is delayed to later in the code where the functionality and types are realized. This leads to obscure error messages often hidden inside a static template-recursion layer.

An intersect method written using concept mechanisms would look something like the following (using the work group's short-hand notation [5]):

```
template <> IntersectResult auto intersect(Curve auto, Surface auto);
```

where `Curve`, `Surface` and `IntersectResult` are concept placeholder types. In this scenario the compiler deduces the placeholder types at the time of binding and can give an error message based on the restricted concept types, for instance: "function parameter one is not a curve, because ...".

The improved error messages is an appreciated side effect, but the real benefit is that we can model concepts by their unique and shared properties rather then restricting their incidental side effects.

Reflection, meta classes and contracts

Reflection and metaclasses introduces compile time functionality, building on the C++20 feature `consteval`, to either reflect upon existing user defined source elements or inject program source fragments at *compile-time*. Using syntax rules from the proposal paper [11], we can define a parametric circle type as follows:

```
class(ParametricCurve) Circle {
  using Unit                = double;
  static constexpr auto Dim = 3ul;

  PSpacePoint     startParameter() const { /**/ };
  PSpacePoint     endParameter() const   { /**/ };
  PSpaceBoolArray isClosed() const       { /**/ }
  Point           evaluate(PSpacePoint par) const { /**/ } };
```

The feature then dictates that our user-defined type, `Circle`, needs to comply with the rules posted through the meta-class `ParametricCurve`. If so, a set of the code fragments, completing the parametric curve construction, will be statically generated. This does not inflict any implicit inheritance.

Contracts on the other hand is a run-time tooling feature which documents and enforces invariants. For instance in the parametric object construction we could apply the following value invariants (this example uses syntax from the proposal [9]).

```
Point evaluate(PSpacePoint t) const
  [[ expects:             t >= startParameters() and t <= endParameters() ]]
  [[ ensures default res: not std::isnan(res) ]]
{
  [[ assert: not std::isnan(t) ]]
}
```

The `expects` directive is applied to input parameters, the `ensures` directive can reflect constraints onto the return value, and the `assert` directive produces assertion conditions. Contracts provides a functionality to turn on restrictions during development.

7 Acknowledgments

The prototype library, GMlib2, is a work in progress; therefore the article is accompanied by a small source code base containing a proposed principal

implementation example of the discussed application programming interface (API), which can be obtained by contacting the authors. The authors acknowledge the equal contribution norm [13] associated with alphabetical listing of authors.

References

[1] Mario Botsch, Stefan Steinberg, Stefan Bischoff, and Leif Kobbelt. OpenMesh: A Generic and Efficient Polygon Mesh Data Structure. In *OpenSG Symposium 2002*, 2002.

[2] M.P. do Carmo. *Differential Geometry of Curves and Surfaces*. Prentice-Hall, 1976.

[3] M.P. do Carmo. *Riemannian Geometry*. Mathematics (Boston, Mass.). Birkhäuser, 1992.

[4] K. Iglberger, G. Hager, J. Treibig, and U. Rüde. Expression templates revisited: A performance analysis of current methodologies. *SIAM Journal on Scientific Computing*, 34(2):C42–C69, 2012.

[5] ISO. *ISO/IEC JTC! SC22 WG21 N4549 — Programming languages — C++ Extensions for Concepts*. 2015.

[6] ISO. *ISO/IEC 14882:2017 Information technology — Programming languages — C++*. Fifth edition, December 2017.

[7] IsoCpp. C++ core guidelines. https://github.com/isocpp/CppCoreGuidelines, 2019.

[8] Arne Lakså, Børre Bang, and Arnt Roald Kristoffersen. GM_lib, a C++ library for geometric modeling. Technical report, Narvik University College, Narvik, Norway, 2006.

[9] G. Dos Reis, J. D. Garica, J. Lakos, A. Meredith, N. Myers, and B. Stroustrup. Support for contract based programming in C++. Technical Report p0542r5, 2018.

[10] Aleksander Stepanov and Paul McJones. *Elements of Programming*. Semigroup Press, authors'/second edition, 2019.

[11] Herb Sutter. Metaclass functions: Generative C++. Technical Report p0707r4, 2019.

[12] The Qt Company. Qt - complete software development framework. https://www.qt.io, 2019.

[13] Teja Tscharntke, Michael E. Hochberg, Tatyana A. Rand, Vincent H. Resh, and Jochen Krauss. Author sequence and credit for contributions in multiauthored publications. *PLoS biology*, 5(1):e18, 2007.

[14] UiT - The Arctic University of Norway. GMlib/GMlib2 C++ geometric modeling library. (NeIC) https://source.coderefinery.org/gmlib, 2019.

An automatic image-based system for detecting wild and stocked fish

Espen Myrum*, Simen Andre Nørstebø*, Sony George*, Marius Pedersen*, and Jon Museth+
* Department of Computer Science, NTNU
+ Norwegian Institute for Nature Research

Abstract

Fish stocking is the method of raising fish in a hatchery and releasing them into a river or lake to sustain or increase an existing population or to create a population. This has been practised in many countries, including Norway. Before the fish are released, the adipose fin is commonly removed in order to identify that it is a stocked fish. Cameras have been mounted in several Norwegian rivers in order to monitor fish populations. Classification of fish from these cameras is today a manual task carried out by people. In this paper we propose an automatic classification method to separate wild fish from stocked fish using machine learning. Experiments on an image set of trouts (Salmo Trutta) show a very high accuracy of the proposed method.

1 Introduction

Many rivers and lakes are not able to sustain a population of fish or one would like to establish a fish population in a river or lake where there is not a fish population. Fish stocking has been a method to do this, where fish are raised in a hatchery and then released into the wild when they reach a certain size. Stocked fish normally have their adipose fin, a small fleshy fin found between the dorsal fin and the caudal fin, removed prior to being released in the wild, being the best way to identify stocked fish from wild fish (see Figure 1).

There is a need to monitor fish populations to contribute to the knowledge base for more sustainable management of rivers and lakes. A common way to monitor fish populations is by using an underwater camera, which is installed in a specially designed setup. In rivers this is commonly installed in a fish ladder, which is a structure on or around natural or man-made barriers (for example dams or waterfalls). This allows to monitor all fish migrating through the fish ladder. When a fish passes a sensor placed in the fish ladder, the camera starts to record a video. These videos can be used to determine the fish species of the fish passing the camera, but also to determine whether the fish is wild or stocked. Today, this work is commonly done manually by a human, which is time and resource demanding. Classification of whether a fish is raised in the wild or a hatchery is

This paper was presented at the NIK-2019 conference; see http://www.nik.no/.

important information that can benefit population count, quality control of fish and also monitoring the ecosystem. In this work we focus on automatic classification of fish in underwater images to decide whether it is wild or stocked. We will focus on the trout species (Salmo trutta), in images captured in freshwater.

Living fish classification is a challenging problem since the fish move freely, light conditions can vary significantly, different visibility in the water, and objects that are not desired can occur (non-fish objects, for example leaves, branches, etc.). Good quality video is required, especially when making a decision on whether or not a fish is wild or stocked, as it is only the adipose fin that can provide information on this. Figure 1 shows example images from an underwater camera, where Figure 1a shows a wild trout with the adipose fin intact while Figure 1b shows a stocked fish with the adipose fin removed. As these fish in the example images are larger fish, it is easier to see the adipose fin. Smaller fish makes it more difficult, and also requiring better video quality.

(a) Wild trout (b) Stocked trout

Figure 1: Example image of a wild trout and stocked trout. The adipose fin is marked in a red circle on the wild trout on the left, while the trout on the right does not have an adipose fin and is therefore a stocked trout.

The goal of this paper is to classify trout as wild or stocked fish from underwater natural images, and we will do this by using a deep neural network and transfer-learning.

This paper is organized as: first we present relevant background in Section 2, then we continue with materials and methods in Section 3, further results and discussion are presented in Section 4, at last we conclude and present future work in Section 5.

2 Relevant background

To the best of our knowledge automatic classification of wild and stocked fish has not been published before. However, there exist work on classification of fish species and the use of machine vision systems in aquaculture [1].

In underwater environments, many approaches for classification have been presented during the last decade. Ogunlana et al. [2] used a Support Vector Machine (SVM) based technique to solve binary fish classification based on 6 shape features with a very small training data. The results showed an accuracy of almost 80%. Chuang et al. [3] proposed a fully unsupervised clustering approach for multiclass

classification based on SVM, binary hierarchy and partial classification. Their results also show a high accuracy on fish image dataset.

Boom et al. [4] introduced an underwater camera surveillance system for monitoring fish. They used a hierarchical classifier based on color, contour and texture features for classification of fish. They stated to be able to recognize 15 different fish species, and showed an average recall of about 83%. Hossain et al. [5] classified different fish species using pyramid histogram of visual words features with an SVM classifier. They evaluated their method on low quality and high quality images showing an accuracy of 40.1% and 91.7%, respectively.

Villon et al. [6] introduced two methods for fish classification; the first based on a traditional two-step approach with extraction of histogram of gradient features and an SVM classifier, and the second method based on deep learning with the GoogLeNet architecture. The results from their experimentation showed the first method to give an F-measure of 0.49 and for second method an F-measure of 0.64.

Pengying et al. [7] proposed a Convolutional Neural Network (CNN) based classification method for trout and grayling. Their approach was based on the pre-trained Alexnet with stochastic gradient descent with momentum. Their experiments showed a very high accuracy above 99%, their method could also classify incomplete fish images with an accuracy of 98%. They also evaluated if Contrast-limited adaptive histogram equalization (CLAHE) would improve the classification accuracy, but it was shown to be decreased by pre-processing.

Pre-processing has also been introduced for underwater classification. Rizzini et al. [8] used CLAHE to compensate light attenuation and remove artefacts in the images before applying a multi-feature object detection algorithm. They state to have satisfactory precision and recall on three different datasets.

Zhao et al. [9] proposed an image-based solution for recognition of individual trouts in the wild. They used the spot pattern from a region from the head for recognition, as this region has shown to be enough for individual fish recognition [10, 11]. Their approach was based on a codebook, where Speeded Up Robust Features (SURF) were used to generate the codebook. Further, SVM was used to generate the final output descriptors. Their evaluation showed an accuracy of about 74% when it came to individual recognition.

Recently, CNNs and deep learning have also been applied to underwater fish classification. Rathi et al. [12] combined CNNs, deep learning and image processing to deal with background noise, image distortion, undesirable objects, occlusion and image quality. On a 23 fish species dataset they obtained 96.3% accuracy.

Machine vision has been used in aquaculture for monitoring fish. This has been driven by the advantage of being fast, cheap and noninvasive. Odone et al. [13] used image parameters to find a relationship between weight and shape, this was done through using SVM. De Verdal et al. [14] used image analysis for individual growth monitoring. Zion et al. [15] used fish area to estimate fish mass, but mentions that image quality needs to be high enough for accurate segmentation and shape analysis. Low quality images makes it difficult to separate between two fish species in their study. Bermejo [16] tested different support vector machine classifiers using a cod database for fish age classification, and showed that a combination of fish length, weight sex with morphological features gave an accuracy of about 75%.

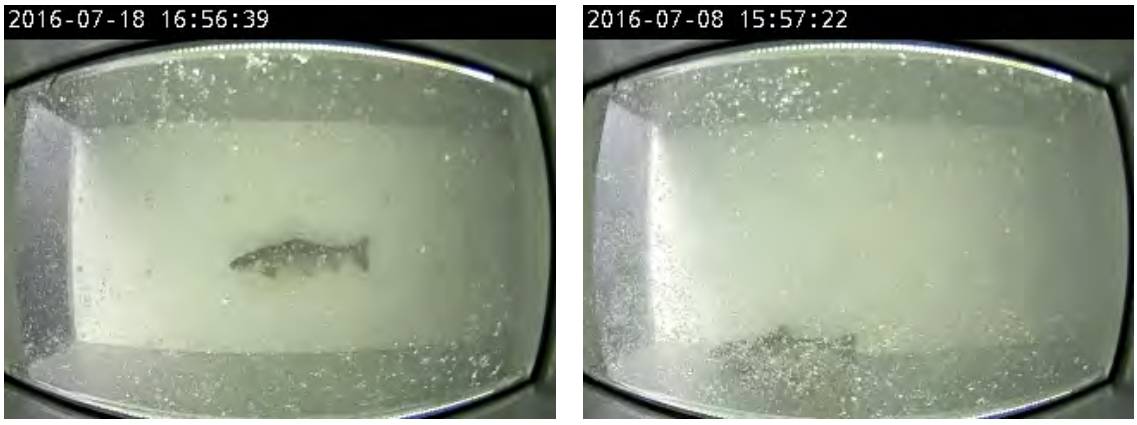
3 Materials and methods

We will first present the dataset used in this study before we introduce the proposed method for automated classification of wild and stocked trout.

Dataset

Our dataset contains 204 video clips, where 101 videos are of stocked fish and 103 videos of wild fish. Each video clip is 24 seconds with a resolution of 320×240 pixels. The quality of the videos are varying in terms of illumination level, illumination uniformity, as well as distortions as air bubbles (as shown in Figure 2 on the right side) and algae. Examples of a "good" image is shown in Figure 3a and an image that is not as sharp, with discoloration of the water and with more air bubbles is shown in Figure 3b. The videos contain fish of different sizes, from small to large trouts (Figures 3b and 2. These videos provide a diverse dataset with challenges. We expect smaller trouts to be more difficult to classify as their adipose fin is not as visible compared to larger trouts. Also the combination of air bubbles, discoloration and small fish would be difficult in terms of classification.

These videos have been converted into still images, where each frame is a still image. The format of the images is PNG.



(a) (b)

Figure 2: Example images. A smaller trout on the left without occluding air bubbles, and on the right a smaller trout with many occluding air bubbles. It is clear that air bubbles can influence the classification.

The images have been categorized into three classes by an expert serving as the ground truth; no fish (images with no fish), stocked fish and wild fish. Our dataset contains 5000 images of each class.

Data augmentation has been done through a horizontal flip of the images, doubling the dataset. A horizontal flip also makes sense as the fish can swim upstream and downstream through the fish ladder. We split the dataset in two parts for training and testing, namely 70% for training and 30% for testing.

Proposed method

Our starting point is the pretrained AlexNet convolutional neural network, which is trained on more than a million images from the ImageNet database [17]. Alexnet is eight layers deep, consisting of 5 convolutional layers and 3 fully connected layers,

(a) (b)

Figure 3: Example images. A higher quality image on the left and a lower quality image on the right. It is clear that discoloration can influence the classification.

and has been trained to classify images into 1000 object categories. Because of this Alexnet has learned many feature representations for a variety of images. Image input to the network is 227 by 227 pixels. Our approach is by using transfer learning and fine-tuning the network to the application of classifying wild and stocked fish. We refer to this approach as the pretrained Alexnet CNN hereafter.

For the pretrained Alexnet CNN we tested different optimizers, namely ADAM [18], stochastic gradient descent with momentum (SDGM), and RMSprop. We also tested if pre-processing the images with Contrast-limited adaptive histogram equalization (CLAHE) [19] would improve the classification. Our tests showed that SDGM gave the best results without CLAHE. Results shown hereafter is by using SDGM on the original images (without any pre-processing).

For the training we used a batch size of 32, 15 epochs, initial learning rate of 0.003, and we reduced the learning rate by a factor of 0.1 every 5 epochs. For the reported results we show the average results after training the network five times.

Another approach we will test is based on an Error correcting output codes (ECOC)-classifier. ECOC is an ensemble method specifically designed for multi-class classification. It uses binary classifiers to solve a multi-class problem. After transfer learning using the pretrained Alexnet CNN, it is run through the ECOC classifier. We refer to this approach as ECOC hereafter.

The implementation has been done in Matlab R2018B, using the "Deep Learning Toolbox 12.0" and the "Statistics and Machine Learning Toolbox 11.4".

4 Results and discussion

For the pretrained Alexnet CNN we obtain an average accuracy of 99.16% and for the ECOC approach we obtained an accuracy of 99.87%. Observations of the results indicate that the misclassification occurs when the trouts are small, which makes it more difficult to see the adipose fin. In some frames there are a significant amount of bubbles (distortions) which also makes classification difficult, and discoloration could also impact. These aspects are illustrated in Figure 2 and Figure 3. The findings that bubbles influence the classification is not surprising, and it is also found in the literature that noise [20, 21] (which is somewhat similar to bubbles) will decrease the performance of neural networks. Experts from NINA also confirmed that bubbles is

a problem when visually classifying if a trout is wild or stocked.

We also report the result for each video in the dataset, where we have calculated the average probability for the frames in each video for a given class. We will use 80% as a threshold, meaning that an average probability of the class above 80% is considered as good. The results can be shown in Table 1. We can see that for the pretrained Alexnet CNN for the wild fish 94 videos (out of 103) have an average probability of belonging to that class to be higher than 80%, 9 videoes are below but still correctly predicted and none are misclassified. The results for ECOC for wild fish are similar. When we see the results for stocked fish, the pretrained Alexnex CNN has 67 videos at a probability above 80%, 33 videos below 80% but still predicted as stocked fish, and one misclassified video. ECOC reports better numbers with 87 videos above 80%, 14 below 80% and none misclassified. These results might indicate better results for ECOC than the pretrained Alexnet CNN.

Method	Wild fish			Stocked fish		
	higher than 80%	less than 80%	Mis-classified	higher than 80%	less than 80%	Mis-classified
Pretrained Alexnet CNN	94	9	0	67	33	1
ECOC	90	13	0	87	14	0

Table 1: Classification results. We can see that both methods (pretrained Alexnet CNN and ECOC) provide good results. ECOC does not have any misclassified fish.

Visual investigation shows that videos with a lot of air bubbles are challenging, but also frames with low quality (blurred and/or with low contrast). An example of a challenging frame that is blurred and low contrast is shown in Figure 4. Blur and low contrast can be caused by different factors; motion blur occurs when fish swims very fast through the fish ladder, faster water flow and disturbance in the water will also impact the sharpness, and low illumination can influence both. One observation from the dataset is that some videos contain algae and debris in the frames, although this is not changing in the period that the fish is passing through the field of view, there is a significant difference between videos. Algae can be seen in Figure 4. We have not observed that this influenced our results, but it could potentially have some impact. Algae will slowly be introduced over time in these videos, but when the camera setup is cleaned it will introduce a significant difference.

Visually there are more air bubbles in the top and bottom part of the frame, and it seems like fish that passes through the center of the frame is classified with a higher accuracy than those passing at the bottom or top. An example of this aspect is shown in Figure 5. At last, frames where the fish is closer to the lens will results in the fish covering a larger area, which could also make the detection of the adipose fin easier. A larger dataset is required to verify many of the aspects discussed here.

We also tested how JPEG compression influenced the performance, where the network was trained on PNG images and tested on JPEG images. The performance was almost identical to that for testing on PNG images. This corresponds well to the findings of Dodge and Karam [20], where they found CNNs to be very resilient towards JPEG compression, as long as the compression is not at very low quality levels. There has also been proposed work to be more robust towards JPEG

Figure 4: Example of challenging frame that is blurred and with low contrast. Algae can also be seen in the frame.

(a) (b)

Figure 5: Example of how air bubbles influence the frames. The two frames are from the same video of the same trout. When the fish is at the top of the frame occluding air bubbles can influence the classification. Less occlusion is present when the fish is closer to the center.

compression and other distortions [22], which could be a direction for future work.

5 Conclusion and future work

We have investigated the use of deep learning for automatic classification of stocked and wild trouts (Salmo Trutta). Two different approaches; one based on transfer learning using a pretrained Alexnet convolutional neural network and the other being similar to the first but with an Error-Correcting Output Codes classifier. The results for both give very good validation accuracy; 99.16% and 99.87%, respectively. However, the dataset contains only 5000 images in each class, which is a limitation. Despite the limitations, this is to be of our knowledge the first work at trying to automate the classification of wild and stocked fish.

Our approach was based on still images, the work should be extended to work on video, as well as dealing with multiple fish in the same video frame. Additional evaluation should also be done on a larger dataset with extensive statistical evaluation. Other fish than trouts should also be considered.

References

[1] Mohammadmehdi Saberioon, Asa Gholizadeh, Petr Cisar, Aliaksandr Pautsina, and Jan Urban. Application of machine vision systems in aquaculture with emphasis on fish: state-of-the-art and key issues. *Reviews in Aquaculture*, 9(4):369–387, 2017. doi: 10.1111/raq.12143. URL `https://onlinelibrary.wiley.com/doi/abs/10.1111/raq.12143`.

[2] SO Ogunlana, O Olabode, SAA Oluwadare, and GB Iwasokun. Fish classification using support vector machine. *African Journal of Computing & ICT*, 8(2):75–82, 2015.

[3] M. Chuang, J. Hwang, and K. Williams. A feature learning and object recognition framework for underwater fish images. *IEEE Transactions on Image Processing*, 25(4):1862–1872, April 2016. ISSN 1057-7149. doi: 10.1109/TIP.2016.2535342.

[4] Bas Boom, P.X. Huang, Concetto Spampinato, S. Palazzo, Jiyin He, C. Beyan, Emmanuelle Beauxis-Aussalet, Jacco van Ossenbruggen, G. Nadarajan, Yun-Heh (Jessica) Chen-Burger, Daniela Giordano, Lynda Hardman, Fang-Pang Lin, and Robert Fisher. Long-term underwater camera surveillance for monitoring and analysis of fish populations. In *Workshop on Visual observation and Analysis of Animal and Insect Behavior 2012*, pages 1–4, jan 2012.

[5] Ekram Hossain, SM Shaiful Alam, Amin Ahsan Ali, and M Ashraful Amin. Fish activity tracking and species identification in underwater video. In *2016 5th International Conference on Informatics, Electronics and Vision (ICIEV)*, pages 62–66. IEEE, 2016.

[6] Sébastien Villon, Marc Chaumont, Gérard Subsol, Sébastien Villéger, Thomas Claverie, and David Mouillot. Coral reef fish detection and recognition in underwater videos by supervised machine learning: Comparison between deep learning and hog+ svm methods. In *International Conference on Advanced Concepts for Intelligent Vision Systems*, pages 160–171. Springer, 2016.

[7] Thitinun Pengying, Marius Pedersen, Jon Yngve Hardeberg, and Jon Museth. Underwater fish classification of trout and grayling. In *Under review*, 2019.

[8] Dario Lodi Rizzini, Fabjan Kallasi, Fabio Oleari, and Stefano Caselli. Investigation of vision-based underwater object detection with multiple datasets. *International Journal of Advanced Robotic Systems*, 12(6):77, 2015.

[9] Lingcong Zhao, Marius Pedersen, Jon Yngve Hardeberg, and Børre Dervo. Image-based recognition of individual trouts in the wild. In *8-th European Workshop on Visual Information Processing*, Oct 2019.

[10] Carlos Garcia de Leaniz, Neil Fraser, Victor Mikheev, and Felicity Huntingford. Individual recognition of juvenile salmonids using melanophore patterns. *Journal of Fish Biology*, 45(3):417–422, 1994.

[11] Caitlin M Gifford and David W Mayhood. Natural marks for identifying individual fish in small populations of at-risk westslope cutthroat trout. In

Wild trout symposium IX: sustaining wild trout in a changing world, pages 275–81, 2013.

[12] Dhruv Rathi, Sushant Jain, and S. Indu. Underwater fish species classification using convolutional neural network and deep learning. *2017 Ninth International Conference on Advances in Pattern Recognition (ICAPR)*, pages 1–6, 2017.

[13] Francesca Odone, Emanuele Trucco, and Alessandro Verri. A trainable system for grading fish from images. *Applied Artificial Intelligence*, 15(8):735–745, 2001.

[14] Hugues De Verdal, Marc Vandeputte, Elodie Pepey, Marie-Odile Vidal, and Béatrice Chatain. Individual growth monitoring of european sea bass larvae by image analysis and microsatellite genotyping. *Aquaculture*, 434:470–475, 2014.

[15] B Zion, A Shklyar, and I Karplus. Sorting fish by computer vision. *Computers and electronics in agriculture*, 23(3):175–187, 1999.

[16] S. Bermejo. Fish age classification based on length, weight, sex and otolith morphological features. *Fisheries Research*, 84:270–274, April 2007.

[17] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[18] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[19] Karel Zuiderveld. Contrast limited adaptive histogram equalization. In *Graphics gems IV*, pages 474–485. Academic Press Professional, Inc., 1994.

[20] Samuel Dodge and Lina Karam. Understanding how image quality affects deep neural networks. In *2016 eighth international conference on quality of multimedia experience (QoMEX)*, pages 1–6. IEEE, 2016.

[21] Samuel Dodge and Lina Karam. Quality resilient deep neural networks. *arXiv preprint arXiv:1703.08119*, 2017.

[22] Dan Hendrycks and Thomas Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. In *International Conference on Learning Representations*, 2019. URL `https://openreview.net/forum?id=HJz6tiCqYm`.

Evaluating Population Based Training on Small Datasets

Frode Tennebø and Marius Geitle
Østfold University College, Halden, Norway
{frodet, mariusge}@hiof.no

Abstract

Recently, there has been an increased interest in using artificial neural networks in the severely resource-constrained devices found in Internet-of-Things networks, in order to perform actions learned from the raw sensor data gathered by these devices. Unfortunately, training neural networks to achieve optimal prediction accuracy requires tuning multiple hyper-parameters, a process which has traditionally taken many times the computation time of a single training run of the neural network. In this paper, we empirically evaluate the Population Based Training algorithm, a method which simultaneously both trains and tunes a neural network, on datasets of similar size to what we might encounter in an IoT scenario. We determine that the population based training algorithm achieves prediction accuracy comparable to a traditional grid or random search on small datasets, and achieves state-of-the-art results for the *Biodeg* dataset.

1 Introduction

Over the last few years, we have seen a significant increase in research and development of novel Internet-of-Things (IoT) devices, and they are now used for controlling industrial machinery [1], monitoring crowds [2], managing energy in smart buildings [3], and in autonomous vehicles [4], to name a few.

Because IoT devices, especially in the edge, are typically severely constrained in terms of energy, computation, storage, and bandwidth resources, early approaches to using machine learning (ML) in IoT networks were often cloud-centric, in which predictions were made in the cloud. This approach required transferring all necessary data to the cloud, which prevented use-cases in which the necessary bandwidth and latency allowances were not available. Lately, prediction has mostly been moved to the nodes, where possible, but this move still requires the training to be done before deployment or in the cloud and therefore prevents using ML in situations where the data is not available before deployment, and the available bandwidth is insufficient for transfer to the cloud.

Among the most successful ML algorithms for learning from complex sensor signals are artificial neural networks (ANN); however, the majority of research into ANNs in the last decade has focused on improving the predictive performance on large datasets using an abundance of computational resources. This trend has resulted in a situation where even training a small state-of-the-art ANN on a small dataset can be too resource-demanding for an IoT node.

One of the reasons for the large resource requirements of training an ANN is that conventional approaches for tuning the hyper-parameters consist of an outer optimization loop based on strategies like random search [5], Bayesian optimization [6], and evolutionary computation [7]. The time required to tune the parameters is then the product of the training time and the number of hyper-parameter configuration evaluations. Recently, a promising new algorithm called Population Based Training (PBT) was proposed by Jaderberg et al. [8], which reduces the tuning time by both training and tuning the neural network simultaneously. However, the method was developed and tested on cases with an abundance of resources, and it is unclear whether it works well for small datasets.

This paper was presented at the NIK 2019 conference. For more information, see http://www.nik.no/.

In this paper, we empirically evaluate how well PBT performs using small datasets, and we have discovered that the PBT algorithm can achieve results similar to a simple grid-search approach, both using the gradient descent (GD) optimization algorithm to update the weights.

The remainder of this paper is structured as follows: in Section 2, we give a brief overview of related work, then describe the PBT implementation in Section 3. We go on to describe our two experiments in Section 4. Finally, we discuss the results and future work in Section 5.

2 Related work

Training the ML models outside of the cloud is a natural progression of the recent trend of moving services out of the cloud and closer to the consumers. An early example of this is the concept of edge computing, which involves processing the data closer to the user before transferring it to the cloud [9]. Training ML models on edge nodes can be difficult due to resource constraints, and doing so when the IoT network has only weaker nodes might require novel techniques such as federated learning [10], transfer learning [11] or online learning [12].

More recently, however, the concept of fog computing has been proposed [13]. Fog computing can be thought of as extending the concept of the cloud out and towards the edge by using more powerful processing gateways. With these more powerful nodes available, it is worth considering training ANNs locally, as these are powerful models, and prediction is typically fast.

Traditional approaches which involved automatically tuning the hyper-parameters of ANNs has so far involved either grid search [14], random search [5], Bayesian optimization [6], or evolutionary computation [7]. However, these approaches require independent trials of each hyper-parameter configuration being evaluated, causing a linear increase in the computation time required to train and tune a network. Additionally, optimizing schedules using these approaches requires manually specifying the model beforehand, often at the cost of increasing the number of hyper-parameters that must be optimized.

The PBT algorithm solves these problems by continually updating the parameters and hyper-parameters simultaneously, in addition to improving the accuracy of the model. However, it is unclear how well PBT will handle the higher noise in the evaluation of individual networks and the increased risk of overfitting caused by the smaller datasets.

3 Population based training

The PBT algorithm is a combined training and tuning algorithm proposed by Jaderberg et al. [8]. The outline of the algorithm can be seen in Algorithm 1.

It employs a population of N "cooperating" individuals, each exploring the problem space with randomly initialized hyperparameters. For the training algorithm, we use the standard *traingd* gradient descent with the backpropagation algorithm from MATLAB's *Deep Learning Toolbox* in a feed-forward neural network. For every 20 epochs, the entire population is evaluated on the validation set, and all individuals can have their weights and hyperparameters updated externally by two methods:


```
Initialize population P with random hyperparameters and weights
while not end of training do
  for p_i ∈ P do
    train p_i one step
  loop
  if ready for evaluation(P) then
    evaluate (P)
    for p_i ∈ P do
      p_i ← exploit (p_i, P)
      if exploited then
        p_i ← explore (p_i)
      end if
    loop
  end if
loop
```

Algorithm 1: High-level Population Based Training algorithm.

- *Exploit*: Replaces the weights and hyperparameters of the worst individuals from the bottom 20% quantile with randomly sampled individuals from the top 20% quantile. In their paper, Jaderberg et al. [8] named this *truncation selection* and noted that it, in general, gave better results than *binary tournament* and *T-test selection*.
- *Explore*: Proposes new hyperparameters to potentially better explore the problem space of an individual which has in the current evaluation already been a target for exploitation. Two methods were tried. We have only reported on *perturb*, in which each hyperparameter is independently and randomly scaled by a factor of either 0.8 or 1.2. We also implemented *resample*, in which each hyperparameter is resampled from the original prior distribution, but results from this test are not included as it did not give an improved performance in our tests.

The principles of *exploit* and *explore* very much resemble inheritance and mutation, respectively, as used in evolutionary computation (EC). Unlike EC, the crossover operation is not part of PBT. Instead, weights are inherited and hyperparameters are mutated. However, the rate of mutation is decoupled from the rate of learning, where learning is performed using standard training methods, in our case gradient descent. This process is similar to Lamarckian evolution with the Baldwin effect [15].

When the entire population is trained to completion, in our case either 600 or 2,000 epochs in total, the best ranked individual is selected and evaluated on the test set. This structure also makes the PBT algorithm well suited for parallel training.

4 Experiments

For our experiments, we wanted to see whether we could train models which achieve similar or improved predictive performance with PBT when compared to the traditional grid search with gradient descent (GSGD) approach for classification problems with a small number of examples. In order to make the comparison fair, we made sure that the total number of epochs was the same when using GSGD compared to a given population size N of PBT, e.g. for PBT with N=20 individuals which are trained for 600 epochs, the grid search was divided into 20 discrete values which are also trained for 600 epocs. Early stopping was not enabled in any of our two experiments to make sure

that the number of epochs was identical. The architecture was also the same for both experiments.

The first experiment was done with an implementation of PBT where only learning rate was tuned, to get an impression of how well it performed on various datasets. The second experiment expanded on this to tune two hyperparameters, learning rate and regularisation level, to see how it affected a small, unbalanced dataset.

While Jaderberg et al. [8] have used a variety of specialized measurements for measuring the performance of the trained models, we are interested in an automated approach for any classification dataset. While accuracy is one of the most used ranking metrics for classification performance, it can easily mischaracterize the predictive performance when used with unbalanced datasets [16]. Therefore, we opted to optimize the Area Under the Receiver Operating Characteristics Curve (AUC) for all datasets since AUC has been suggested as a better measure than accuracy [17].

4.1 Datasets

We have used four small, binary classification datasets retrieved from UCI[1], which contains between 1 055 and 11 055 instances. Additionally, we also used the *HIGGS* dataset, also from UCI, which contains 11 million instances, and are likely larger than what would be relevant when training the ANN in an IoT node. These sets are unbalanced to a variable degree, as shown in Table 1.

Dataset	# instances	# features	Minority share
Biodeg [17]	1,055	41	34 % (356)
HIGGS [18]	11,000,000	21/28[2]	47%
HIGGS subset	100,00	21	47%
Phishing websites [19]	11,055	30	44% (4,898)
Spambase [20]	4,601	57	39% (1,813)
Wilt [21]	4,839	6	5% (261)

Table 1: Overview of the datasets used, the number of samples, number of features, and minority share in percent (absolute numbers) for each dataset.

These datasets were selected because they were of small size, with a moderate number of features and without missing values. The *HIGGS* dataset was used as a reference as it is nearly balanced and much larger. However, only 100,000 instances and 21 features were used in our experiment due to run-time constraints. Each dataset was divided into three parts: 20% for the test set, 16% for the validation set, and 64% for the training set.

4.2 Basic PBT experiment

We ran the initial PBT experiment with 10, 20, and 40 individuals in each population for 600 and 2,000 epochs. Each experiment was repeated 20 times on a network with three hidden layers, each consisting of 20 nodes. The results are reported as the arithmetic average of all trials.

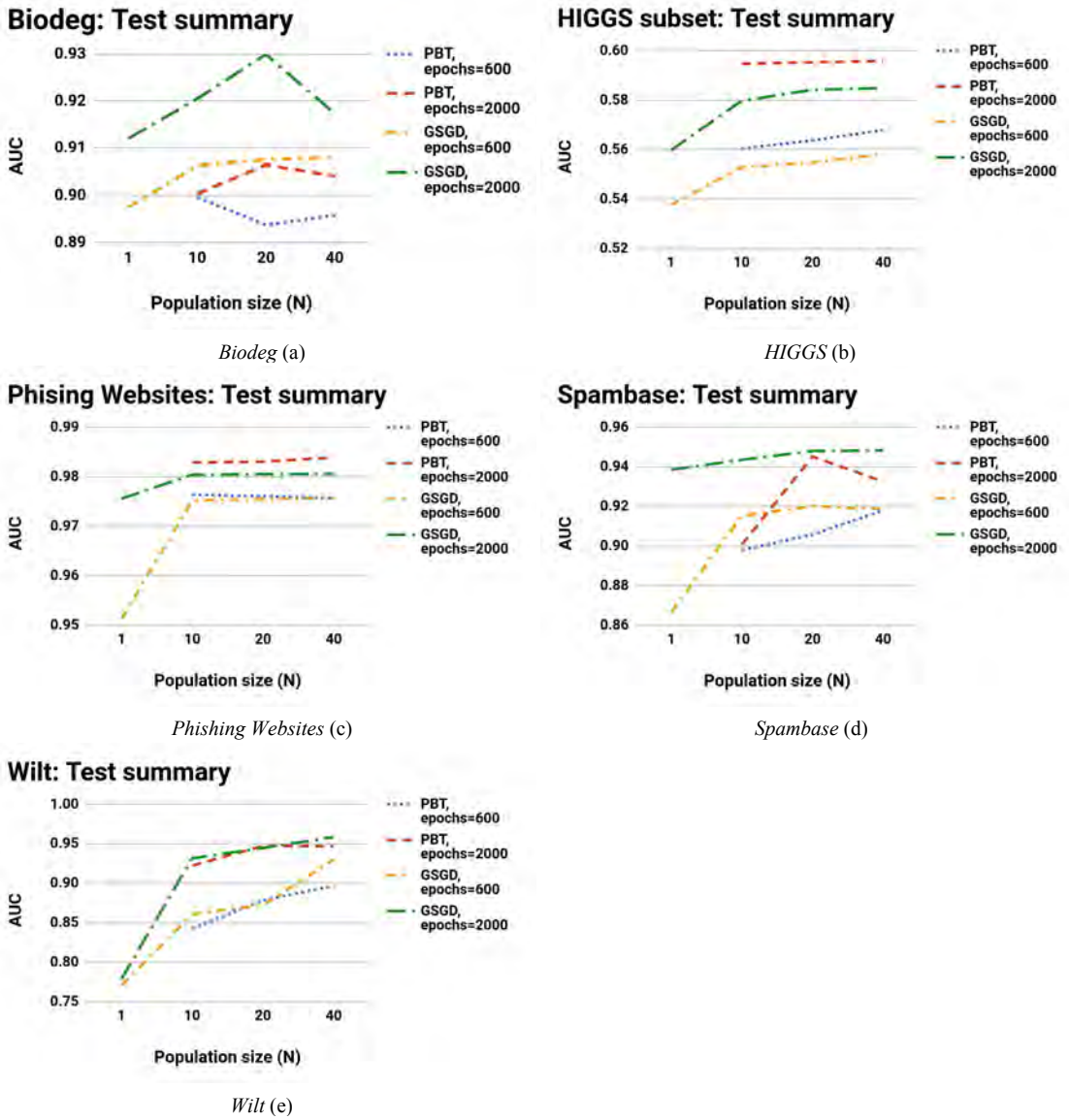
The only hyperparameter to be optimized for this first experiment was the learning rate (LR), η. For PBT, it was randomly initialized from a uniform distribution between 0.0001 and 0.1, and for GSGD the same range was uniformly divided to match the respective number of individuals in PBT. The results can be seen in Table 2 and Figure 1.

[1] http://archive.ics.uci.edu/ml/index.php

[2] 21 features are low-level, kinematic properties measured by the particle detectors in an accelerator, and seven features are functions of the first 21 features: high-level features derived by physicists to help discriminate between the two classes.

Epochs	600				2,000			
Population size (N)	**1**	**10**	**20**	**40**	**1**	**10**	**20**	**40**
Dataset								
Biodeg, PBT		0.900	0.894	0.896		0.900	**0.906**	0.904
Biodeg, GSGD	0.897	0.906	0.908	0.908	0.912	0.920	**0.930**	0.917
HIGGS subset, PBT		0.560	0.564	0.568		0.595	0.595	**0.596**
HIGGS subset, GSGD	0.537	0.553	0.555	0.558	0.560	0.580	0.584	**0.585**
Phishing websites, PBT		0.976	0.976	0.976		0.983	0.983	**0.984**
Phishing websites, GSGD	0.951	0.975	0.975	0.976	0.976	0.980	0.981	**0.981**
Spambase, PBT		0.898	0.906	0.918		0.901	**0.945**	0.933
Spambase, GSGD	0.867	0.915	0.920	0.919	0.939	0.944	**0.948**	0.948
Wilt, PBT		0.843	0.879	0.897		0.922	**0.947**	0.947
Wilt, GSGD	0.771	0.861	0.872	0.930	0.779	0.932	0.945	**0.959**

Table 2: AUC results of the initial PBT implementation and GSGD rounded to three significant digits for all datasets for four different population sizes and two different run-times in epochs.



Biodeg (a) *HIGGS* (b)

Phishing Websites (c) *Spambase* (d)

Wilt (e)

Figure 1: Graphical representation of AUC (y-axis) for the *Biodeg* (a), *HIGGS subset* (b), *Phishing Websites* (c), *Spambase* (d), and *Wilt* (e) datasets for PBT and GSGD. The x-axis shows the population size for PBT or the equivalent number of grid values for GSGD. Only LR has been optimized.

4.2.1 Results

For the *HIGGS* and *Phishing Websites* datasets, PBT scores better than GSGD. These two datasets are also the two largest with 100,000 and 11,000 instances, respectively. This indicates that PBT is more sensitive to the number of samples than GSGD. Jaderberg et al. [8] got their results using either very large datasets or generating as many samples as they needed.

For the other three datasets, PBT scores worse or comparable to GSGD. This result could be caused by overfitting because of the low number of training instances in combination with a relative high architecture complexity. However, it could also be caused by the AUC metric being too sensitive to the small number of examples in the dataset, as described by Hanczar et al. [22].

While PBT achieves better predictive performance on one dataset, and only slightly worse results on three of five datasets compared to the state-of-the-art results for these datasets, see Table 6, the results are still impressive given that the state-of-the-art results were achieved using methodology which was optimized for each dataset.

Even though the PBT and GSGD optimize the hyper-parameters with respect to the AUC, the predictive performance of the models could still suffer when the models are trained on imbalanced datasets. In our next experiment, we will address this by attempting to balance the dataset using oversampling.

4.3 PBT with oversampling and L_2 regularisation

Several strategies have been developed to deal with class imbalance, and Buda et al. [23] have discussed several of these methods and their use for training convoluted neural networks and concluded that imbalance is detrimental to classification performance. They also use AUC as an evaluation metric and justify this by arguing that overall accuracy metric is associated with notable difficulties for imbalanced data. Chawla et al. [24] note that oversampling can lead to overfitting for general machine learning techniques. To mitigate this and other problems, they developed Synthetic Minority Oversampling TEchnique (SMOTE). This technique works by adding synthetic data. However, they only compared against undersampling and did not consider the case of training a neural network. Therefore we will consider regular random, minority oversampling to deal with class imbalance for our next experiment.

In [14], Goodfellow et al. discuss several methods to handle overfitting and some of the computationally less expensive methods are drop out, L_1 and L_2 regularisation. We have used a variant of L_2 regularisation which modifies the normal performance function, the mean sum of squares of the network errors, by adding the term mean sum of the squares of the network weights and biases as described in Equation 2.

$$mse_w = \frac{1}{N}(1-\lambda)\sum_{i=1}^{N}(t_i-\alpha_i)^2 + \frac{1}{n}\lambda\sum_{j=1}^{n}w_j^2 \tag{2}$$

This is from the MATLAB 2018a implementation, which is similar, but not identical, to how others like Krogh and Hertz [25] and Goodfellow et al. [14] describe L_2 regularisation, but the general principle, and therefore the result should be similar.

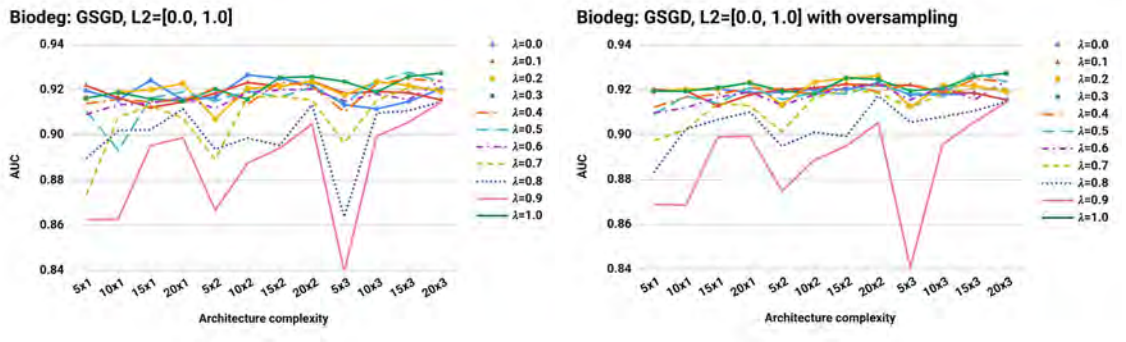
Based on the results of the previous experiment, we opted to focus only on the *Biodeg* dataset since this dataset achieved the worst performance in our N experiment, when compared to GSGD. In order to keep the complexities down, we looked at only two variants for this experiment, one with only oversampling and one with both oversampling and L_2 regularisation.

For this experiment we trained PBT for 600 epochs with a population of N=20 individuals for 20 independent trials, but to rule out any overfitting in the relatively large architecture in the first experiment, we tested multiple different architectures; one, two and three hidden layers with five, ten, fifteen, and twenty nodes. For GSGD, we used the same architectures as for PBT and the same single level grid search for learning rate as in the first experiment. Also, the hyperparameter distribution is the same as described in the first experiment for both PBT and GSGD. The results can be seen in Table 3 and Figure 3a.

Architecture	AUC				Accuracy in %			
	PBT, LR (SD)	PBT, LR, oversampling (SD)	GSGD, LR (SD)	GSGD, LR, oversampling, (SD)	PBT, LR (SD)	PBT, LR, oversampling (SD)	GSGD, LR (SD)	GSGD, LR, oversampling (SD)
5x1	0.895 (0.015)	0.905 (0.012)	0.900 (0.025)	0.909 (0.016)	80.3 (2.40)	82.1 (1.81)	83.1 (3.03)	82.9 (2.57)
10x1	0.897 (0.013)	**0.911 (0.012)**	0.912 (0.026)	0.918 (0.013)	80.9 (1.76)	83.0 (2.24)	84.4 (2.47)	83.9 (3.00)
15x1	0.895 (0.019)	0.908 (0.012)	0.917 (0.016)	**0.926 (0.020)**	80.9 (2.48)	82.9 (1.67)	84.9 (2.56)	85.5 (3.37)
20x1	0.899 (0.011)	0.909 (0.013)	0.915 (0.019)	0.924 (0.016)	81.2 (1.68)	82.7 (1.47)	84.5 (1.95)	84.1 (2.33)
5x2	0.900 (0.015)	0.910 (0.009)	0.917 (0.018)	0.923 (0.022)	81.9 (2.01)	**83.4 (1.12)**	85.2 (1.94)	**85.5 (2.97)**
10x2	0.904 (0.015)	0.910 (0.013)	0.919 (0.028)	0.915 (0.028)	**82.2 (1.87)**	83.0 (1.49)	84.9 (2.85)	84.7 (3.25)
15x2	**0.904 (0.011)**	0.911 (0.014)	0.910 (0.022)	0.921 (0.022)	81.6 (1.84)	83.0 (1.64)	84.2 (1.85)	84.1 (3.30)
20x2	0.902 (0.012)	0.908 (0.017)	**0.924 (0.024)**	0.925 (0.019)	81.9 (1.91)	82.5 (1.59)	85.2 (2.30)	84.6 (2.50)
5x3	0.900 (0.016)	0.903 (0.024)	0.916 (0.023)	0.915 (0.024)	82.2 (2.19)	82.9 (2.60)	**85.3 (2.63)**	84.5 (2.81)
10x3	0.900 (0.020)	0.910 (0.013)	0.915 (0.016)	0.918 (0.017)	81.8 (1.99)	82.9 (1.53)	84.6 (2.18)	84.3 (3.40)
15x3	0.898 (0.019)	0.907 (0.011)	0.917 (0.024)	0.923 (0.017)	81.4 (2.14)	82.5 (1.56)	85.1 (2.70)	84.0 (2.51)
20x3	0.899 (0.016)	0.911 (0.012)	0.922 (0.026)	0.921 (0.013)	81.5 (2.00)	83.1 (1.72)	84.4 (3.35)	84.4 (2.61)

Table 3: AUC and accuracy results with standard deviation in parenthesis for the *Biodeg* dataset. Only learning rate was optimised for PBT. For GSGD, single level grid search on learning rate was done. Results are with and without oversampling rounded to three significant digits for twelve different architectures.

In order to find out how L_2 regularisation works on this dataset, we compared GSGD with only L_2 regularisation and with both oversampling and L_2 regularisation with levels of λ=[0.0, 1.0] in steps of 0.1. We randomly initialized η from a uniform distribution between 0.0001 and 0.1. The results can be seen in Figure 2.



(a) (b)

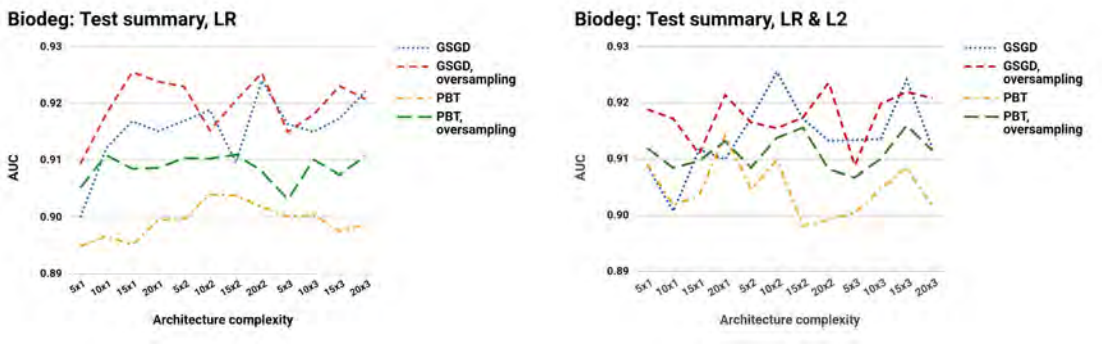
Figure 2: Graphical representation of AUC (y-axis) for GSGD for various architectures and levels of L_2 regularisation with (b) and without (a) oversampling for different architecture complexities (x-axis). λ=1.0 is not shown as it is equivalent to random classification with AUC ~= 0.5.

λ=1, λ=0.9, λ=0.8, and to a lesser extent also λ=0.7, particularly without oversampling, have a clear, negative impact on the performance for all architectures. Particularly noticeable is this for the less complex architectures with only five nodes in each layer. Otherwise, there is little difference between the results with or without

oversampling. For PBT's *exploit* we, therefore, decided to draw λ from a uniform distribution between 0.0 and 0.7, and for *explore,* we truncated λ to 0.7. GSGD was run as a grid search with λ=[0.0, 0.7] in steps of 0.23… and η=[0.0001, 0.1001] in steps of 0.025. The result can be seen in Table 4 and Figure 3b.

Architecture	PBT, LR & L_2 (SD)	PBT, LR & L_2, oversampling (SD)	GSGD, LR & L_2 (SD)	GSGD, LR & L_2, oversampling (SD)
5x1	0.909 (0.023)	0.912 (0.021)	0.909 (0.022)	0.919 (0.020)
10x1	0.902 (0.023)	0.908 (0.020)	0.901 (0.029)	0.917 (0.021)
15x1	0.904 (0.021)	0.910 (0.027)	0.912 (0.021)	0.911 (0.017)
20x1	**0.914 (0.020)**	0.913 (0.019)	0.910 (0.023)	0.921 (0.020)
5x2	0.905 (0.013)	0.908 (0.020)	0.917 (0.015)	0.916 (0.023)
10x2	0.910 (0.019)	0.914 (0.016)	**0.925 (0.013)**	0.915 (0.013)
15x2	0.898 (0.023)	**0.916 (0.017)**	0.917 (0.017)	0.917 (0.015)
20x2	0.899 (0.019)	0.908 (0.017)	0.913 (0.021)	**0.924 (0.019)**
5x3	0.900 (0.030)	0.907 (0.029)	0.913 (0.026)	0.909 (0.017)
10x3	0.905 (0.025)	0.910 (0.023)	0.914 (0.019)	0.920 (0.026)
15x3	0.908 (0.017)	0.916 (0.018)	0.924 (0.017)	0.922 (0.019)
20x3	0.902 (0.027)	0.911 (0.022)	0.912 (0.016)	0.921 (0.021)

Table 4: AUC and standard deviation results of PBT and GSGD for the *Biodeg* dataset. For PBT both learning rate and L_2 regularisation is tuned. For GSGD a 5 x 4 grid search was used. Results are with and without oversampling rounded to three significant digits for twelve different architectures.



(a) (b)

Figure 3: Graphical representation of AUC (y-axis) for PBT and GSGD with LR optimization with (b) and without (a) L_2 regularisation optimization, with and without oversampling for different architecture complexities (x-axis).

4.3.1 Results

From Figure 3a, we can see that oversampling alone gives a better overall performance for PBT while it does not show overall improvements for GSGD.

Comparing Figure 3a and 3b, L_2 regularisation appears to overall have a positive effect on PBT without oversampling, but not with oversampling nor for GSGD. The p-value for conducting an independent two-tailed Student's t-test comparing the best results of PBT and GSGD with and without oversampling from Table 3 and 4 are given in Table 5.

Based on this, we can therefore reject the alternative hypothesis that there is a statistically significant difference between the best results optimizing only the learning rate and optimizing both learning rate and L_2 regularisation term within a 5% significance level.

Results	LR (SD) (from Table 3)	LR & L_2 (SD) (from Table 4)	*p*-value
PBT	0.904 (0.011)	0.914 (0.020)	0.081
PBT, oversampling	0.911 (0.012)	0.916 (0.017)	0.307
GSGD	0.924 (0.024)	0.925 (0.013)	0.816
GSGD, oversampling	0.926 (0.020)	0.924 (0.019)	0.760

Table 5: Overview of the best AUC and standard deviation results from the *Biodeg* datasets for PBT and GSGD with only LR optimization and with both LR and L_2 regularisation optimization, with and without oversampling and the calculated *p*-value.

This indicates that oversampling and L_2 regularisation has little or no effect on the *Biodeg* dataset. This possibility is supported by Figure 2, where there is little difference in results between λ=[0.0, 0.7] with and without oversampling.

Similarly, the *p*-values comparing the best results of PBT with the best result of GSGD (bold in Table 4) are 0.042 and 0.169 without and with oversampling, respectively. We can, therefore, accept the alternative hypothesis that there is a statistically significant difference between PBT and GSGD without oversampling, but not with oversampling, within a 5% significance level.

Our overall best results (bold in Table 4) has as 95% CI at [0.909, 0.923] for PBT and [0.919, 0.931] for GSGD. Both PBT and GSGD are better than state-of-the-art results listed in Table 6.

5 Discussion and future work

In this paper, we have proposed a general PBT configuration to train neural networks on classification datasets. We then evaluated the PBT configuration on a selection of binary classification problems using small datasets in a scenario that is relevant for IoT applications. We have shown that our PBT configuration outperforms GSGD on the larger of these datasets. But on the smallest datasets, PBT performs comparable or only slightly worse than GSGD. Using the Biodeg dataset, we have also shown that it can optimize both the learning rate η and the penalty term λ of L_2 regularisation simultaneously, and we have established that oversampling, when relevant, has a positive effect on PBT.

When compared to the state-of-the-art results reported in the literature for these datasets, as summarised in Table 6, both PBT and GSGD give better results for *Biodeg*.

Dataset	Best-in-class	Experiment 1	Experiment 2	
		Best initial PBT (from Table 2)	Best PBT with LR (from Table 3) (SD)	Best PBT with LR & L_2 (from Table 4) (SD)
Biodeg	0.887 [26]	0.906	0.911 (0.012)	0.916 (0.017)
HIGGS	0.891 [27]	0.596 (subset)		
Phishing websites	0.993 [28]	0.984		
Spambase	0.980 [29]	0.945		
Wilt	0.990 [30]	0.947		

Table 6: Overview of the datasets used giving state-of-the-art results found in published papers to the best of our abilities together with the best results from our experiments.

We found that PBT achieves better or similar results on four of five datasets compared to the state-of-the-art results for these datasets, see Table 6. Given that the state-of-the-art results were achieved using a methodology which was optimized for each dataset, this is quite impressive and strengthen the idea that state-of-the-art or near state-of-the-art results can be achieved in constrained IoT devices in the edge. The inferior performance on the *HIGGS* dataset, is likely because we only used a subset of

the complete dataset, in addition to only considering a small class of network architectures.

Jaderberg et al. [8] claim that PBT does not have wall-clock run time that is greater than that of a single optimization process and is also able to use fewer computational resources than conventional search methods. In a separate paper by Jaderberg et al. [31] describing how they used PBT to train a neural network to playing the game of *Quake III Arena Capture the Flag* against human players using reinforcement learning on in real-time on the same commodity workstations used by the human players without adversely affecting the frame-rate of the game.

This presents intriguing possibilities for using PBT to train neural network architectures on low-powered IoT devices such as smart light bulbs, microwaves, and thermostats; devices where powerful processors are not available, and training common neural network architectures are currently not possible, thereby enabling new technologies that require these low-powered devices to learn, in vivo, how to respond to complex sensor information which will not be available until after deployment and when methods based on online learning is not suitable.

Automated training after deployment also highlights an important difference between our work and that of Jaderberg et al. [8]. Training after deployment will likely prevent the oversight of a human machine learning practitioner and therefore will require the training method to consistently achieve satisfactory predictive performance. In our experiments, we only used a single setup for all datasets, and still achieved results that either surpass or are similar to the state-of-the-art on those datasets. This is in contrast to the experiments by Jaderberg et al. [8] where the PBT algorithm only served as a framework which were extensively adapted to every problem.

This paper serves as a starting point for creating a general PBT configuration for training neural networks on classification datasets, and further work is likely to yield significant improvements. In the remainder of this section we will outline some possible further work.

In our experiments, we used the error rate as a measure of fitness to calculate the gradient during GD. Cortes and Mori [32] have pointed out that directly optimizing performance using AUC instead of indirectly by minimizing the error rate, can result in substantially better AUC scores.

In our experiments, we only tuned at most two hyper-parameters; the learning rate and the L_2-regularisation level. But PBT can, in principle, tune any number of hyper-parameters, and Jaderberg et al. [8] have tuned up to four hyper-parameters in their experiments. The complexity of optimizing hyper-parameters grows exponentially with the number of tunable hyper-parameters for a given algorithm using a grid search. PBT utilizes a property similar to random search [5], which should lead to much faster convergence, since the good regions of the hyper-parameters are explored more. More distinct values are used for each hyper-parameter to identify the good regions of hyper-parameters and ensures that they are explored more. Therefore, more complex algorithms with more hyper-parameters might produce better results with PBT.

The PBT algorithm itself also has hyper-parameters such as the size of quantile, time of evaluation, size of population and perturbation factors. Further work should explore ways of dispensing with these, or provide generally justified values. An option is to treat these parameters as tunable by PBT itself, together with the architecture as described by Zoph and Le [33]. This could make the optimization process more automated and robust. This is a type of automation that is an emerging strategy in the neural architecture search, a subfield of AutoML, according to Elsken et al. [34].

6 References

[1] D. Kwon, M. R. Hodkiewicz, J. Fan, T. Shibutani, and M. G. Pecht, "IoT-Based Prognostics and Systems Health Management for Industrial Applications," *IEEE Access*, vol. 4. pp. 3659–3670, 2016 [Online]. Available: http://dx.doi.org/10.1109/access.2016.2587754

[2] N. H. Motlagh, M. Bagaa, and T. Taleb, "UAV-Based IoT Platform: A Crowd Surveillance Use Case," *IEEE Communications Magazine*, vol. 55, no. 2. pp. 128–134, 2017 [Online]. Available: http://dx.doi.org/10.1109/mcom.2017.1600587cm

[3] K. Akkaya, I. Guvenc, R. Aygun, N. Pala, and A. Kadri, "IoT-based occupancy monitoring techniques for energy-efficient smart buildings," *2015 IEEE Wireless Communications and Networking Conference Workshops (WCNCW)*. 2015 [Online]. Available: http://dx.doi.org/10.1109/wcncw.2015.7122529

[4] S. Liu, J. Tang, Z. Zhang, and J.-L. Gaudiot, "Computer Architectures for Autonomous Driving," *Computer*, vol. 50, no. 8. pp. 18–25, 2017 [Online]. Available: http://dx.doi.org/10.1109/mc.2017.3001256

[5] J. Bergstra and Y. Bengio, "Random Search for Hyper-Parameter Optimization," *J. Mach. Learn. Res.*, vol. 13, no. Feb, pp. 281–305, 2012.

[6] E. Brochu, V. M. Cora, and N. de Freitas, "A Tutorial on Bayesian Optimization of Expensive Cost Functions, with Application to Active User Modeling and Hierarchical Reinforcement Learning," *arXiv [cs.LG]*, 12-Dec-2010 [Online]. Available: http://arxiv.org/abs/1012.2599

[7] J. H. Holland, "Genetic Algorithms," *Sci. Am.*, vol. 267, no. 1, pp. 66–73, 1992.

[8] M. Jaderberg *et al.*, "Population Based Training of Neural Networks," *arXiv preprint arXiv:1711.09846*, 27-Nov-2017 [Online]. Available: http://arxiv.org/abs/1711.09846. [Accessed: 23-Jan-2018]

[9] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge Computing: Vision and Challenges," *IEEE Internet of Things Journal*, vol. 3, no. 5. pp. 637–646, 2016 [Online]. Available: http://dx.doi.org/10.1109/jiot.2016.2579198

[10] T. Nishio and R. Yonetani, "Client Selection for Federated Learning with Heterogeneous Resources in Mobile Edge," *ICC 2019 - 2019 IEEE International Conference on Communications (ICC)*. 2019 [Online]. Available: http://dx.doi.org/10.1109/icc.2019.8761315

[11] K. Weiss, T. M. Khoshgoftaar, and D. Wang, "A survey of transfer learning," *Journal of Big Data*, vol. 3, no. 1. 2016 [Online]. Available: http://dx.doi.org/10.1186/s40537-016-0043-6

[12] T. Chen, Q. Ling, Y. Shen, and G. B. Giannakis, "Heterogeneous Online Learning for 'Thing-Adaptive' Fog Computing in IoT," *IEEE Internet of Things Journal*, vol. 5, no. 6. pp. 4328–4341, 2018 [Online]. Available: http://dx.doi.org/10.1109/jiot.2018.2860281

[13] M. Mukherjee, L. Shu, and D. Wang, "Survey of Fog Computing: Fundamental, Network Applications, and Research Challenges," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 3. pp. 1826–1857, 2018 [Online]. Available: http://dx.doi.org/10.1109/comst.2018.2814571

[14] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016.

[15] D. Whitley, V. S. Gordon, and K. Mathias, "Lamarckian evolution, the Baldwin effect and function optimization," in *Parallel Problem Solving from Nature — PPSN III*, 1994, pp. 5–15.

[16] N. V. Chawla, "Data Mining for Imbalanced Datasets: An Overview," in *Data Mining and Knowledge Discovery Handbook*, Second edition., O. Maimon and L. Rokach, Eds. Springer, 2009, pp. 875–886.

[17] K. Mansouri, T. Ringsted, D. Ballabio, R. Todeschini, and V. Consonni, "Quantitative structure-activity relationship models for ready biodegradability of chemicals," *J. Chem. Inf. Model.*, vol. 53, no. 4, pp. 867–878, Apr. 2013.
[18] P. Baldi, P. Sadowski, and D. Whiteson, "Searching for exotic particles in high-energy physics with deep learning," *Nat. Commun.*, vol. 5, p. 4308, Jul. 2014.
[19] R. M. Mohammad, F. Thabtah, and L. McCluskey, "An assessment of features related to phishing websites using an automated technique," in *2012 International Conference for Internet Technology and Secured Transactions*, 2012, pp. 492–497.
[20] M. Hopkins, E. Reeber, G. Forman, and J. Suermondt, "Spambase dataset," *UCI Machine Learning Repository*, 1999. [Online]. Available: https://archive.ics.uci.edu/ml/datasets/spambase
[21] B. A. Johnson, R. Tateishi, and N. T. Hoan, "A hybrid pansharpening approach and multiscale object-based image analysis for mapping diseased pine and oak trees," *Int. J. Remote Sens.*, vol. 34, no. 20, pp. 6969–6982, Oct. 2013.
[22] B. Hanczar, J. Hua, C. Sima, J. Weinstein, M. Bittner, and E. R. Dougherty, "Small-sample precision of ROC-related estimates," *Bioinformatics*, vol. 26, no. 6, pp. 822–830, Mar. 2010.
[23] M. Buda, A. Maki, and M. A. Mazurowski, "A systematic study of the class imbalance problem in convolutional neural networks," *Neural Netw.*, vol. 106, pp. 249–259, Jul. 2018.
[24] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic Minority Over-sampling Technique," *1*, vol. 16, pp. 321–357, Jun. 2002.
[25] A. Krogh and J. A. Hertz, "A Simple Weight Decay Can Improve Generalization," in *Advances in Neural Information Processing Systems 4*, J. E. Moody, S. J. Hanson, and R. P. Lippmann, Eds. Morgan-Kaufmann, 1992, pp. 950–957.
[26] Z. Luo and M. Hauskrecht, "Group-Based Active Learning of Classification Models," *Proc. Int. Fla. AI Res. Soc. Conf.*, vol. 2017, pp. 92–97, May 2017.
[27] P. Sadowski, J. Collado, D. Whiteson, and P. Baldi, "Deep Learning, Dark Knowledge, and Dark Matter," in *NIPS 2014 Workshop on High-energy Physics and Machine Learning*, 2015, pp. 81–87.
[28] T. Chin, K. Xiong, and C. Hu, "Phishlimiter: A Phishing Detection and Mitigation Approach Using Software-Defined Networking," *IEEE Access*, vol. 6, pp. 42516–42531, 2018.
[29] M. Khalid, I. Ray, and H. Chitsaz, "Scalable Nonlinear AUC Maximization Methods," *Proceedings of the European*, 2018 [Online]. Available: http://www.ecmlpkdd2018.org/wp-content/uploads/2018/09/82.pdf
[30] Y. Zhang, J. Wu, C. Zhou, and Z. Cai, "Instance cloned extreme learning machine," *Pattern Recognit.*, vol. 68, pp. 52–65, Aug. 2017.
[31] M. Jaderberg *et al.*, "Human-level performance in first-person multiplayer games with population-based deep reinforcement learning," 03-Jul-2018 [Online]. Available: http://dx.doi.org/10.1126/science.aau6249. [Accessed: 18-Aug-2019]
[32] C. Cortes and M. Mohri, "AUC Optimization vs. Error Rate Minimization," in *Advances in Neural Information Processing Systems 16*, S. Thrun, L. K. Saul, and B. Schölkopf, Eds. MIT Press, 2004, pp. 313–320.
[33] B. Zoph and Q. V. Le, "Neural Architecture Search with Reinforcement Learning," *arXiv [cs.LG]*, 05-Nov-2016 [Online]. Available: http://arxiv.org/abs/1611.01578
[34] T. Elsken, J. H. Metzen, and F. Hutter, "Neural Architecture Search: A Survey," *arXiv [stat.ML]*, 16-Aug-2018 [Online]. Available: http://arxiv.org/abs/1808.05377

Automated salamander recognition using deep neural networks and feature extraction

Jørgen Bakløkken*, Felix Schoeler*, Hugo Nørholm*,
Marius Pedersen*, Sony George*, Børre Dervo+
* Department of Computer Science, NTNU
+ Norwegian Institute for Nature Research

Abstract

This paper presents a study conducted to recognize salamanders by using their unique body markings based on images. The detection and matching of unique patterns in a salamander's body can be complex due variability in individual animals size, shape, orientation and also influence from the external enviornment. While traditional methods require time intensive manual image corrections of the salamanders to achieve accurate recognition, in this work we propose a fully automatic techinque for straigthening. We also propose a matching technique based on the corrected images. The convolutional neural network ResNet50 and dense scale-invariant feature transform (DSIFT) are used for belly pattern localization, and matching for salamander recognition.

1 Introduction

To understand the trend of animal life and migratory patterns, it is important to systematically quantify their population in an accurate way. Salamanders are amphibians which experience declination in their population. In Norway there are two different species of salamander, the great crested newt (Triturus crustatis) and the smooth newt (Lissotriton vulgaris). As the great crested newt (Triturus cristatus), has the status near threatened in Norway's national red list, it is crucial to know changes in their population. In order to evaluate the effectiveness of the actions taken in order to increase the number of salamanders, it is important track their population in an accurate way.

There are different ways to count the population of species. One of the most commonly used methods is pit tagging. A pit tag is a small transponder containing a material that is harmless to the animal to which this transponder is the placed. Later the count is performed by scanning its tag ID. However, to implement pit tagging it is necessary to catch the animal, inject a pit tag and document this in the database. This is very time consuming, need careful handling and also stressful for the animal. Image based pattern recognition methods becomes very common for performing population measure, monitoring migration, habitat usage etc. for several animal species because of it is less time consuming and less stressful for the animal. There are notable works in this topic [1].

This paper was presented at the NIK-2019 conference; see http://www.nik.no/.

Image based recognition methods solves the issues with pit tagging. However, there exists little efforts towards using automatic image-based pattern recognition for salamanders. Some of the commercial or open source softwares available requires user interventions and manual preprocessing [2, 3, 4, 5]. The great crested newt and smooth newt have body markings of small black patches that are individual-specific [1] (see Figure 1 and Figure 2). These individual-specific body markings can serve as a natural mark for capture–recapture models for estimating population demography. Accurate recognition requires good quality images of the salamanders. This includes proper illumination of the animal while the images are captured, correct orientation of the salamander so that the patterns are included in the images used for recognition. In some cases preprocessing is needed in order to correct the above mentioned factors.

One of the fields of research at the Norwegian Institute for Nature Research (NINA) is salamanders. NINA has interest to learn more about salamander's life cycle, living area and population. To gain more knowledge on this, NINA started a pit tagging project in 2010. In this research, NINA captures images of the salamander and is interested to develop an application that takes an input image of salamander, recognizes the pattern, and identifies whether the same animal is in the database. The objective of the current study is to propose a solution which identifies salamanders based on pattern recognition with a high accuracy, that is comparable to existing solutions. In addition, the solution should reduce the amount of manual work required to match salamanders by automating the workflow. In the current paper, we present results of a study conducted to develop an automated salamander recognition algorithm using deep neural networks and feature extraction.

The paper is organized as follows; first we present the dataset, then further the proposed method, after this we present our results, conclusion and future research.

Figure 1: Example image of a salamander.

Figure 2: Example image of a salamander.

2 Dataset

The dataset captured by NINA contains a total of 6700 images. These images were of the northern crested newt (Triturus crustatis) and the smooth newt (Lissotriton vulgaris).

The resolution of the images is 1240 x 1748 in JPEG format. Pit tagged information was also available and verify that the images were from the same salamander. Images in the dataset were of different quality levels: this includes images with with partially or fully occluded belly pattern, dark images with low contrast, curved pose, dirt and noisy, varying background, out of focus, overexposed etc. Images with no possibility to extract any information even after preprocessing were excluded from the dataset. Figure 1 and Figure 2 show example images from the dataset.

3 Proposed method

The workflow of the proposed method for salmander recognition is presented in Figure 3.

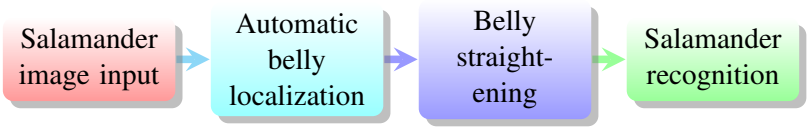


Figure 3: Workflow of the proposed method.

Images of the salamander have been recorded in different conditions and need to be preprocessed before they go to the recognition stage. Preprocessing steps include belly localisation and straightening. In most of the currently available recongition softwares, the preprocessing part is performed manually. One of the main contributions of the presented research is the automated processing of the complete workflow from input image to the recognition.

Belly pattern localisation

Salamander belly pattern

The salamander´s unique patterns is in the body area between the front and back legs. The animal pose is not the same for all the images and in several cases, this needs to be corrected in order to be able to compare patterns accurately. According to A.J. Ijspeert [6], the salamander can be modeled as having three joints in between the front legs and back legs, as demonstrated in Figure 4.

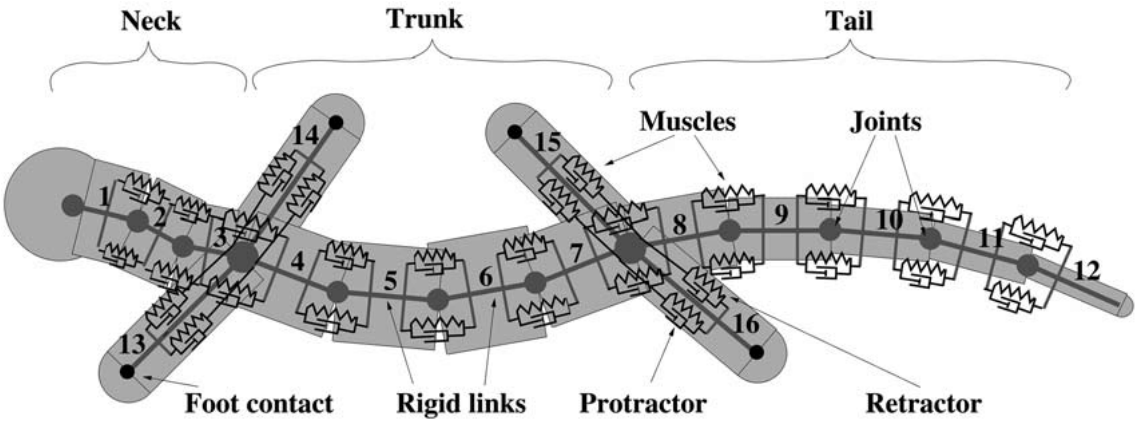


Figure 4: Mechanical model of a salamander, reproduced from [6].

This gets us to a total of five joints required to localize the trunk and belly pattern of the salamander. Based on this, we defined the belly pattern to be from the point in

between the front legs to the point in between the back legs. We tried to label these five points equidistantly along the spine. The spine will then be estimated using bicubic interpolation [7] of these five points. The five points were defined as seen in Table 1 and Figure 5.

Table 1: Labeled points on salamander

Point	Description	Color
Front legs	Point between the front legs	Orange
Belly point 1	First intermediary point along the spine	Yellow
Belly point 2	Second intermediary point along the spine	Green
Belly point 3	Third intermediary point along the spine	Turquoise
Back legs	Point between the back legs	Blue

The simplest method to locate the belly pattern is by manual labeling. In currently available systems, a human labeler goes through the images and selects a set of points along the spinal cord. With a large dataset, this can take a large amount of time. Accurate manual labeling by the authors took about 30 seconds, which coincides with the findings of Matthé et al. [1]. To do this for a dataset with 10000 images, it requires more than 83 hours of manual labeling. Additionally, the human labeler is subject to bias and error rates may also increase when working on labeling for a prolonged amount of time.

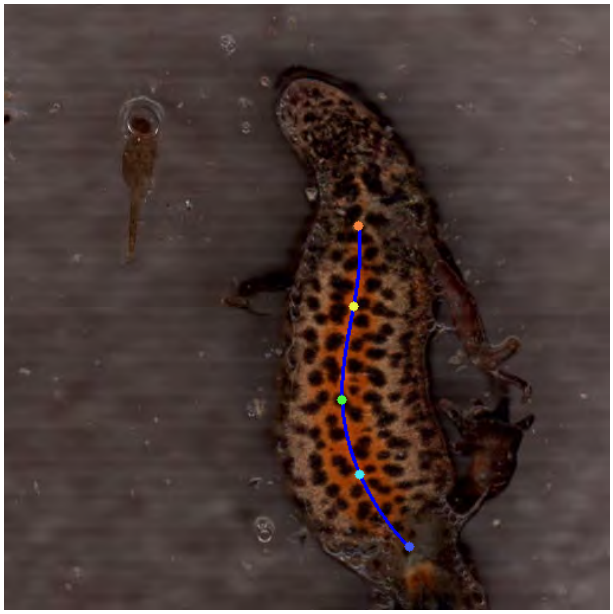


Figure 5: Example of a manually labeled salamander. The color points indicate the different joints of the salamander. The figure is best viewed in color.

Belly localization

Three different methods for belly localization have been tested and they are presented in the following paragraphs.

Segmentation and skeletonization: This method works by first segmenting out the salamander from the background. Further, it applies skeletonization, also referred to as a medial axis transform to estimate the salamander's spinal cord. This method is intended

to be used by having user input making the initial line near the object you want to create a contour, however this method proved unsuitable for two reasons: (1) requires longer time to process (2) mixing of background information into the area of interest. After the salamander is segmented out a medial axis transform is performed [8]. This constructs the skeleton image. After the skeleton image is constructed it is converted to a skeleton network [9]. Additionally, we grouped vertices that are close to each other in proximity together to form the skeleton seen in Figure 6. Getting the belly line then simply consists of extracting the longest edge that is not connected to any leaf vertex.

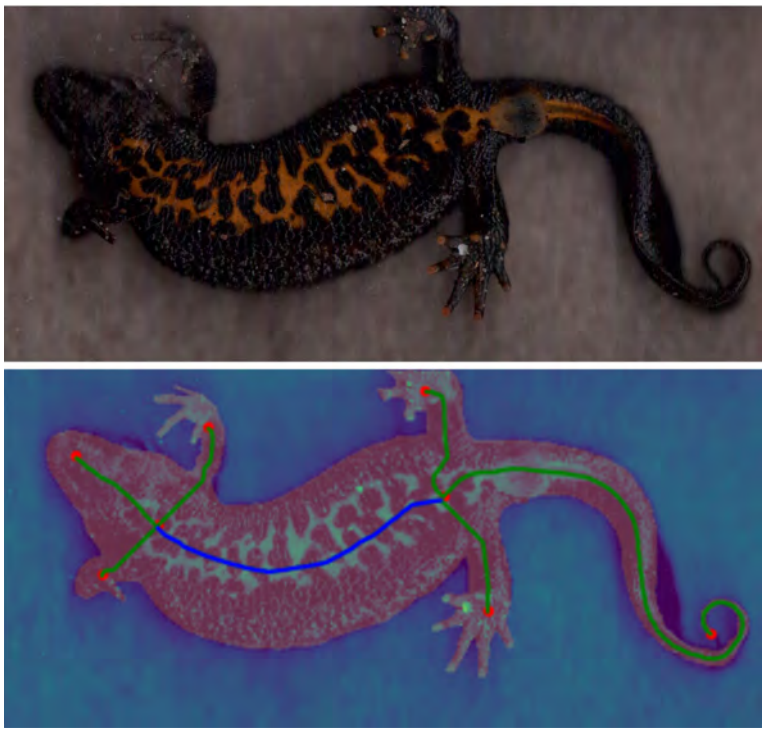


Figure 6: Salamander after manual segmentation and skeletonization.

The segmentation and skeletonization method has shown to quite error-prone due to a number of reasons. One problem with the method is that it requires the segmentation to be very accurate. If there are small errors in the segmentation, often caused by shadows or other artefacts in the image, this causes the skeletonization to create additional branches as seen in Figure 7. In this case a part of the tail, marked in blue, is actually erroneously identified as the belly pattern. If we would be able to attain perfect smooth segmentation as in the example of Figure 6, the same problem also occurs when the salamander does not have its feet perpendicular to body. This causes misalignment of the point between the feet which is used as the respective start- and end-point of the belly pattern. In Figure 6 we see that the salamander is bending and at the same time its's laying down further to one side. This causes the spinal cord and the belly pattern to not align with the medial axis of the segmented salamander. It also causes the pattern to differ depending on how the salamander poses on the image, which in turn negatively affects matching.

Pose estimation using LEAP: In order to more accurately extract the belly pattern, we tested a convolutional neural network (CNN) developed to estimate pose quickly called LEAP [10]. This is to be used in real-time applications. In order to make the images for network, images are converted to grayscale with a resolution of 196 x 196 or 256 x 256 as recommended [11]. Labeled images are split into a training set 90% and test

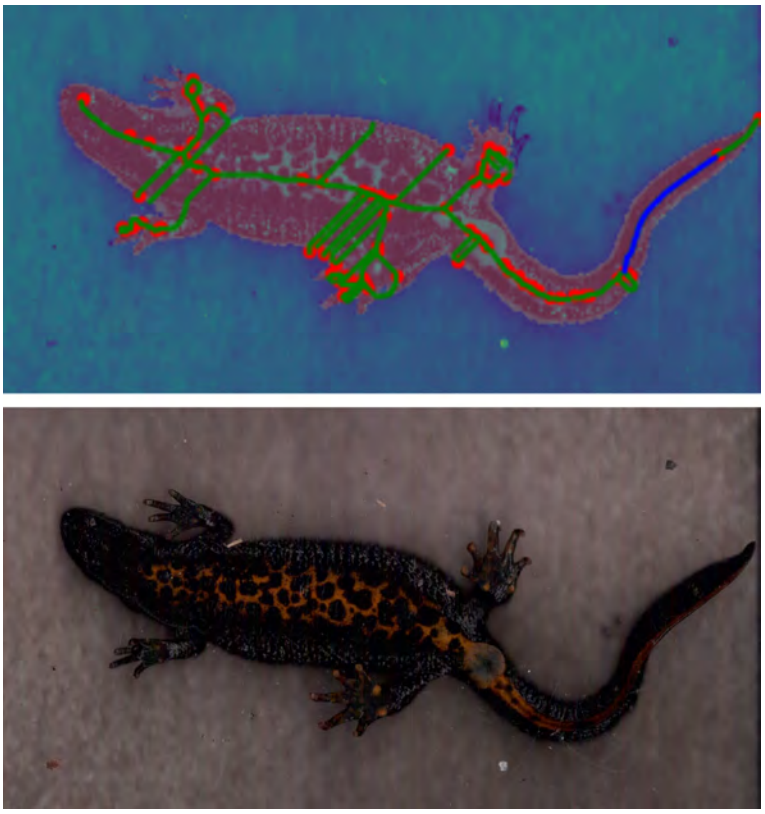


Figure 7: Consequences of minor segmentation errors. Original image on bottom, and segmented image on top. Best viewed in color.

set 10% with a batch size of 16, and rand 50 batches per epoch, for a total of 15 epochs. The dataset is augmented using random rotations (+-45) and mirroring. We have used 10% of the images in each batch for batch-level validation and used 10 of the 50 batches for epoch-level validation. The training finished in less than 5 minutes using hardware mentioned in Table 2.

Table 2: Hardware system used for testing in this work.

Component	Model
GPU	Geforce GTX 1060 6GB
CPU	Intel Core i5-2310@2.9 GHz
RAM	DDR3 8GB

The error is calculated by taking the Euclidean distance between the point predicted by LEAP and the human labeled points. The system was evaluated on 10 randomly selected images and is presented in Table 3.

It is observed that error rates are relatively high, with an average Euclidean error of 66px which is about the width of a typical salamander in our images. LEAP was able to generate useful results in only 1 out of our 10 testing images. It is able to predict the location of the belly pattern on some basic images, but fails quickly if there is rotation and/or bend in the images. Based on the results, we concluded that this neural network is not good enough for the present study.

Pose estimation using DeepLabCut: DeepLabCut [12, 13] is another tool for markerless pose estimation of user-defined body parts with deep learning. DeepLabCut

Table 3: LEAP testing results.

Metric	Error mean	Standard deviation
Front legs	106.9 px	103.8 px
Belly point 1	68.7 px	45.8 px
Belly point 2	37.2 px	26.3 px
Belly point 3	57.0 px	61.0 px
Back legs	65.4 px	96.4 px
All	**66.0 px**	**72.8 px**

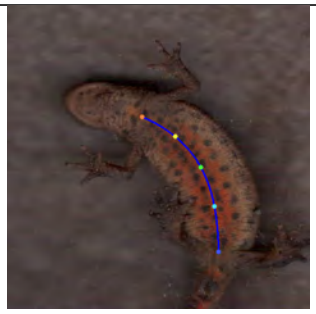
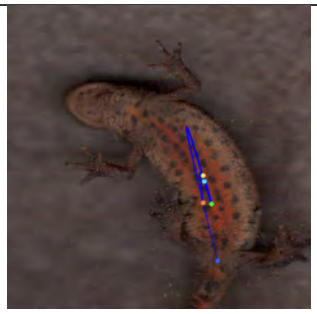
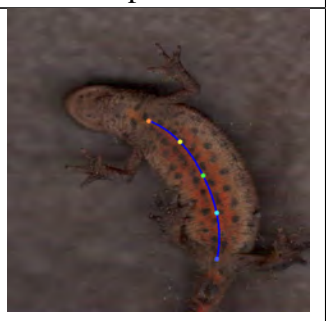
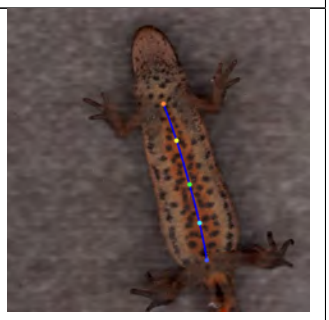
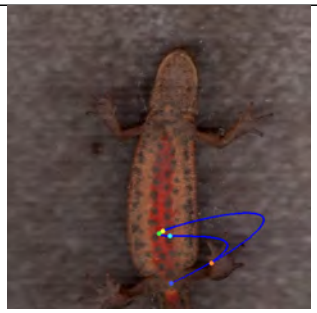
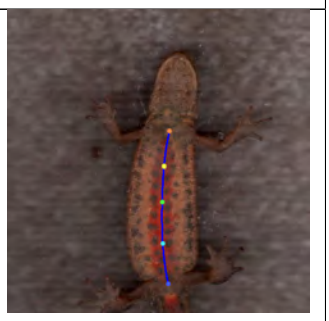
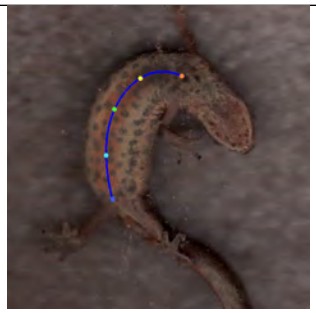
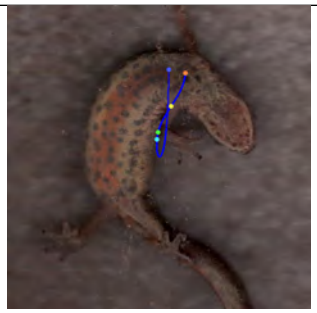
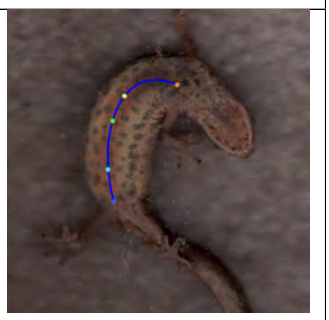
uses Huber loss as it's loss function and is less sensitive to outlier data than the mean-squared-error loss that is used in LEAP [10], as it is linear instead of quadratic for large errors. In order to test the accuracy of DeepLabCut, we provided it with the 200 labeled images. We provided the images in a 512 x 512 RGB format. The labeled images were split into 95% training and 5% testing sets. Data augmentation was done by mirroring the images, applying random rotations (+-30) and resizing (+-25%) the images and labels. Additionally, we used pretrained weights for the backbone, which is a deep residual network, ResNet [14], that has run 5000 iterations on the ImageNet [15] dataset, which according to it's authors makes it able to accurately predict pose with only 200 labeled images [12]. It is recommended to train the network for at least 200,000 iterations in order to arrive at a plateau of the loss function [13]. We trained the network for 350,000 iterations which took 12 hours using our hardware.

The error is calculated by taking the Euclidean distance between the point predicted by DeepLabCut and the human labeled points. The system was evaluated on 10 randomly selected images, and the results are shown in Table 4. DeepLabCut was able to predict images at a rate of 5.2 FPS.

Table 4: DeepLabCut testing results.

Metric	Error mean	Standard deviation
Front legs	6.4 px	2.9 px
Belly point 1	9.3 px	9.8 px
Belly point 2	10.5 px	5.8 px
Belly point 3	8.1 px	4.4 px
Back legs	3.3 px	1.9 px
All	**7.5 px**	**6.2 px**

Table 5: Comparison of belly localization methods.

#	Manually labeled	LEAP	DeepLabCut
1			
2			
3			
4			

Although DeepLabCut at 5.2 FPS is considerably slower than LEAP at 107.3 FPS, we also have a better accuracy with a significant margin for DeepLabCut. From the results in Table 4 there is a mean error of 7.5 px. We also observe that the endpoints have a considerably lower mean error than the three belly points. It is therefore reasonable to assume that the labeled points where not perfectly equidistantly aligned although they were relatively accurately placed on the spinal cord. This causes some fluctuations along the spinal cord. This does increase the mean error but does not affect the resulting bicubic interpolation as much. We could also validate this by simple visual inspection by comparing the bicubic interpolation of the manually selected points to the bicubic interpolation of the points selected by the neural network in Table 5.

As the input images were required to be a straight image, all the images are converted into a rectangular image, with the spinal cord of the salamander in the center. This

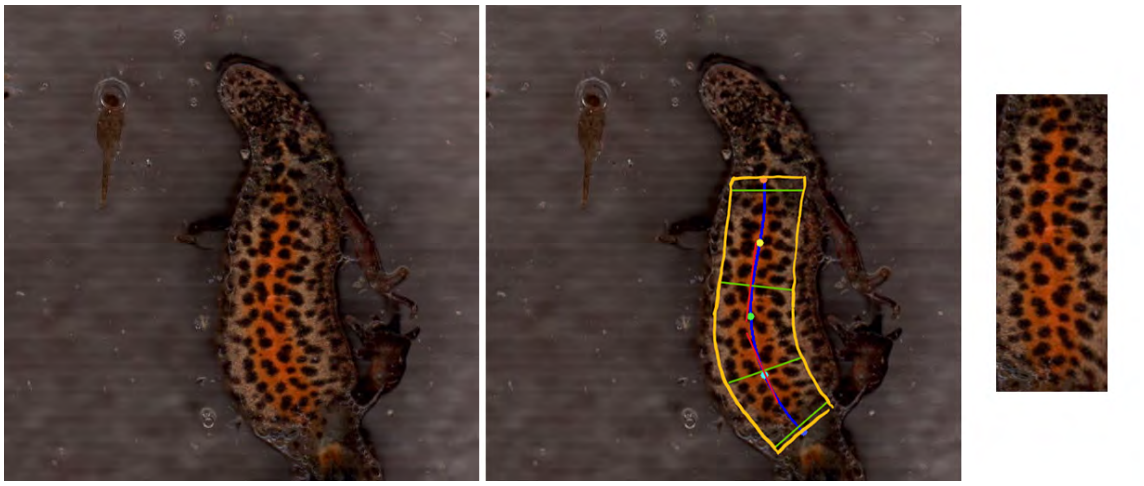


Figure 8: Straightening using DeepLabCut and the derivative of the interpolated spinal curve.

has been implemented using a nonlinear transformation based on the gradient of the interpolated curve. An example is shown in Figure 8.

Salamander recognition

Two different methods for recognition have been tested; a pixel based method and a method based on brute force matching.

Pixel based method

The method itself revolves around comparing the images pixel by pixel and looking at the difference to make an average pixel difference between the images to find matches. In order to make the images suitable for pixel-based comparison, it is necessary to perform few pre-processing steps: resizing and thresholding. Images are resized into 80 x 280 using bilinear interpolation. In addition to making the images the same size this also shrinks them by roughly 75% which makes the comparison algorithm faster.

To increase the speed of the algorithm the images are binarized. Doing this step does not cause any loss of information because only the difference between the images is the presence of a spot from the pattern or not. Compared to other thresholding methods, Otsu binarization [16] worked very well across the images we tested. Images in the present study are bimodal, which makes it easy for the algorithm to calculate a good threshold value. In order to reduce the noise after binarization, the images were blurred to smooth out the noise without compromising the pattern quality in the salamander body. Out of different blurring methods tested, median blur removed the most noise and resulted in least effect on the details of the pattern.

The average pixel difference adding up the differences of all the pixels and dividing by the dimensions of the image. One drawback with this method is that even a minor shift in the animal and thus resulting pattern, the images will not match and can drastically increase the difference between the images. To solve this careful image translation in each direction is performed and followed by comparison algorithm over and over again and picks the best score. In the present experiment, we have shifted images 20 pixels left and right with a step size of 2 and up and down 10 pixels and picking the best result.

This method showed a decent performance to find matches for ten images from a pool of another ten images consisting of of pairs of the original ten. While testing for

bigger pool trying to find the same 10 matches in a pool of 48 images the best match of each image was still a real match, however when we looked at all the match numbers there were plenty of images that gave potential matches meaning that if there had been no match in the system it would have given a false match as a result. These false positives varied in severity and we consider increasing the threshold in order to filter more of them out. However, some of the false positives reached as high as 92% and changing the threshold that high would result in most of the true matches as not good enough.

Brute force matching

This is a very simple but fast matching method. It takes two set of data, in this case two descriptors. The function will use the first descriptor as a "template" and then try to find the closest one in the second set and result in an image the closest to the template. In these two objects there's something called distance, and this will tell us the distance between the two matched points. If the distance parameter is within a certain number (in the present case 0.75, this number is commonly used for ratio tests) it's a match. In this work we set a minimum number of matches in order to make sure it's a match. This is because of the images are very similar so it's possible to get a match even though the two descriptors are from two different animals. We tested many different numbers and with some bad images we will get a low number of good matches. Since the lowest number in the present test set was 15 and is been chosen to use this as a minimum. This implies the two descriptors need at least 15 matches before we can be sure the pattern belongs to the same individual salamander.

We tested four popular feature based methods, ORB (Oriented FAST and rotated BRIEF) [17], SURF (Speeded up robust features) [18], SIFT (Scale-invariant feature transform) [19] and DSIFT (Dense SIFT). DSIFT showed the best performance, and the results for this is shown.

Evaluation

Table 6 shows the results obtained using a brute force matching with DSIFT and the pixel based method. As shown we have achieved a high accuracy with brute force matching and DSIFT.

Table 6: Match rates on DSIFT and pixel based method.

	Correct result	Match not found	False positive	Potential false positives
DSIFT	45	4	0	0
%	91.84%	8.16%	0.00%	
Pixel	21	9	19	511
%	42.86%	18.37%	38.78%	

We have analysed the results particularly the failed cases. In some cases, the belly pattern is occluded by the salamander's feet, and this led to the failure of pose estimation by DeepLabCut. Other reasons for false detection are the poor image quality mainly by the low lighting when the images were taken.

The main contribution of our system, contrary to existing software, is that the system provides a fully automated workflow as shown in Table 7.

Table 7: Comparison between existing software and our solution.

Software	Matching	Straightening	Licensing
Aphis	Pixel-based method and SURF[18]	No	Open Source
AmphIdent	Pixel-based method	Manual	Commercial
Wild-ID	SIFT [19]	No	Open Source
I3S Pattern	SURF [18]	Manual	Open Source
Proposed	**DSIFT**	**Automatic**	**Open Source**

4 Conclusion and future work

The goal of this work was to propose a solution which identifies salamanders based on pattern recognition with a high accuracy comparable to existing software's and to reduce the amount of manual work required to match salamanders by automating the workflow. We have reviewed the existing methods and software solutions available and outlined their differences, individual advantages and disadvantages. Based on the state-of-the-art methods and techniques used, we have identified an optimal solution with regard to the research objective. On the basis of this, a set of methods were selected for implementation and evaluated for this project. The methods selected were the neural network DeepLabCut for pose estimation and Dense SIFT for pattern recognition. The selected methods were later implemented as software solution. One main contribution of the work is a system that uses a matching method with a high accuracy on the current dataset. To the best of our knowledge, no efforts has produced any similar results which is been to the direction of automated solution for salamander recognition.

Recent advancements in deep neural networks allowed us to create this automated solution. With improved cameras, the images capture in future would have better image quality and expect increase in performance of the matching. NINA is already moving ahead in this direction. It is possible to extend the proposed methods for performing recognition tasks in other animals.

References

[1] Matthé, M., Sannolo, M., Winiarski, K., Sluijs, A. S. V. D., Goedbloed, D., Steinfartz, S., & Stachow, U. 2017. Comparison of photo-matching algorithms commonly used for photographic capture-recapture studies. *Ecology and Evolution*, 7(15), 5861–5872. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5552938/, doi:10.1002/ece3.3140.

[2] Management, R. URL: http://www.reijns.com/i3s/download/I3S_download.html.

[3] Bolger, D. T. URL: https://home.dartmouth.edu/faculty-directory/douglas-thomas-bolger.

[4] Amphident. URL: http://www.amphident.de/en/pages/download.html.

[5] d'Estudis Avançats, I. M. URL: http://imedea.uib-csic.es/bc/ecopob/index.php?option=com_content&view=article&id=73&Itemid=89&lang=en.

[6] Ijspeert, A. 06 2001. A connectionist central pattern generator for the aquatic and terrestrial gaits of a simulated salamander. *Biological cybernetics*, 84, 331–48. doi:10.1007/s004220000211.

[7] scipy.interpolate.interp1d.

[8] Scikit image documentation morphology.skeletonize. Available at https://scikit-image.org/docs/0.15.x/api/skimage.morphology.html#skimage.morphology.skeletonize, Accessed 08:58, 12 May 2019.

[9] Build network from skeleton image (2d-3d). Available at https://github.com/yxdragon/sknw, Accessed 09:05, 12 May 2019.

[10] Pereira, T., Aldarondo, D., Willmore, L., Kislin, M., Wang, S., Murthy, M., & Shaevitz, J. W. 2018. Fast animal pose estimation using deep neural networks. *bioRxiv*. URL: https://www.biorxiv.org/content/early/2018/05/25/331181, arXiv:https://www.biorxiv.org/content/early/2018/05/25/331181.full.pdf, doi:10.1101/331181.

[11] Opencv documentation - cv2.resize.

[12] Mathis, A., Mamidanna, P., Abe, T., Cury, K. M., Murthy, V. N., Mathis, M. W., & Bethge, M. 2018. Markerless tracking of user-defined features with deep learning. *arXiv preprint arXiv:1804.03142*.

[13] Nath, T., Mathis, A., Chen, A. C., Patel, A., Bethge, M., & Mathis, M. W. 2018. Using deeplabcut for 3d markerless pose estimation across species and behaviors. *bioRxiv*. URL: https://www.biorxiv.org/content/early/2018/11/24/476531, arXiv:https://www.biorxiv.org/content/early/2018/11/24/476531.full.pdf, doi:10.1101/476531.

[14] He, K., Zhang, X., Ren, S., & Sun, J. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 770–778.

[15] Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., & Fei-Fei, L. 2009. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, 248–255.

[16] Otsu, N. 1979. A threshold selection method from gray-level histograms. *IEEE transactions on systems, man, and cybernetics*, 9(1), 62–66.

[17] Rublee, E., Rabaud, V., Konolige, K., & Bradski, G. Nov 2011. Orb: An efficient alternative to sift or surf. In *2011 International Conference on Computer Vision*, 2564–2571. doi:10.1109/ICCV.2011.6126544.

[18] Bay, H., Tuytelaars, T., & Van Gool, L. 2006. Surf: Speeded up robust features. In *European conference on computer vision*, 404–417. Springer.

[19] Lowe, D. G. Sep. 1999. Object recognition from local scale-invariant features. In *Proceedings of the Seventh IEEE International Conference on Computer Vision*, volume 2, 1150–1157. doi:10.1109/ICCV.1999.790410.

Finite element method application of ERBS triangles

Tatiana Kravetc

R&D group Simulations
Department of Computer Science and Computational Engineering
Faculty of Science and Technology
UiT – The Arctic University of Norway

Abstract

In this paper we solve an eigenvalue problem on a circular membrane with fixed outer boundary by using a finite element method, where an element is represented as an expo-rational blending triangle.

ERBS triangles combine properties of B-spline finite elements and standard polynomial triangular elements. The overlapping of local triangles allows us to provide a flexible handling of the surface while preserving the smoothness of the initial domain, also over the nodes and edges. Blending splines accurately approximate the outer boundary, while keeping a coarse discretization of the domain.

We consider a mesh construction for such type of elements, evaluating of basis functions and their directional derivatives, local-to-global mapping, assembling of element matrices.

This paper was presented at the NIK-2019 conference; see http://www.nik.no/.

1 Introduction

Isogeometric analysis invokes a methodology that represents the solution space for dependent variables in terms of the same functions which represent the geometry [1].

Triangulation is a common approach to the domain discretization. Triangulation is more general than the tensor product surfaces, due to less geometrical constraints. When approximating solutions by the finite element method, we should choose the finite element mesh in such a way that it must not only get an accurate approximate solution but also the edges of the outer elements must approximate the boundary well. Triangular elements are particularly suited to the task of filling domains with smooth boundaries, thus minimizing the difference between the initial domain and the finite element domain [2].

One way to increase the accuracy of the solution is to provide mesh refinement, i.e. increase the number of elements, especially along the smooth boundary. Another way is to refine the elements. It implies higher-degree polynomials in the basis functions, and a mapping between local and global coordinate systems.

In this paper we introduce ERBS triangles as an alternative type of elements. These elements are based on expo-rational blending splines, which have been introduced in [3]. ERBS triangles are defined in [4]. The main distinction between B-splines and expo-rational blending surfaces is that the second ones can be relatively easy evaluated on a triangular mesh. The reason for this is that the expo-rational basis possesses an interpolatory property and it is strictly local under the support of the local geometry. In the presented research we show that the mesh constructed by ERBS triangles gives a very good approximation of the PDE solution with very coarse domain discretization, but geometrically exact at the boundary.

The following related works demonstrate an utilization of blending spline surfaces on triangulations. An implementation of the spline blending surface approximation over a triangulated irregular network is shown in [5]. The first instances of expo-rational finite elements on triangulations was presented in [6] and their application can be found in [7].

This paper is organized as follows: Section 2 focuses on basis functions, which combine ERBS basis and Bernstein polynomial basis of local triangles. In Section 3 we define a model problem and derive the finite element method applied to the considered problem, utilizing the ERBS triangles as elements. In Section 4 we show the results of the described method, implemented in MatLab, and compare them to the exact solution. The conclusions are given in Section 5.

2 ERBS triangles

In this section we collect previous results from [3], [4] and consider a construction of ERBS triangles, a combination of barycentric representation of Bernstein basis polynomials and expo-rational basis, and its directional derivatives.

An ERBS triangle is a surface that blends three Bézier triangles of the degree d via expo-rational basis functions. A comprehensive study of barycentric coordinates and Bézier triangles can be found in [8]. We define a simple version of the underlying basic expo-rational basis function over the formal parameter u.

Definition 2.1. The underlying basic expo-rational basis function in barycentric coordinates is defined by $B(u)$, $u \in (0, 1]$, as follows

$$B(u) = \begin{cases} \Gamma \int_0^u \phi(s)ds, \text{ if } 0 < u \leq 1, \\ 0, \text{ otherwise} \end{cases}$$

where

$$\phi(u) = \exp\left(-\frac{\left(u - \frac{1}{2}\right)^2}{u(1-u)} \right),$$

and the scaling factor

$$\Gamma = \frac{1}{\int_0^1 \phi(u)du}.$$

A set of such expo-rational functions forms a basis for the blending type surface construction.

Definition 2.2. For any point $\nu = (u_1, u_2, u_3)$, $u_1+u_2+u_3 = 1$, a set of expo-rational basis functions in barycentric coordinates is defined as follows

$$\beta_i(\nu) = \frac{B(u_i)}{B(u_1) + B(u_2) + B(u_3)} \quad \text{for } i = 1, 2, 3, \tag{1}$$

where $B(u_i)$ are as defined by Definition 2.1.

A set of expo-rational basis functions on a triangle is shown in Figure 1(a).

A construction of an ERBS triangle is based on a linear combination of three Bézier triangles of degree d and the set of expo-rational basis functions in barycentric coordinates.

Definition 2.3. For a set of local Bézier triangles $\ell_i(u_1, u_2, u_3)$, $i = 1, 2, 3$, and corresponding expo-rational basis functions $\beta_i(u_1, u_2, u_3)$, the general formula for the ERBS triangle is

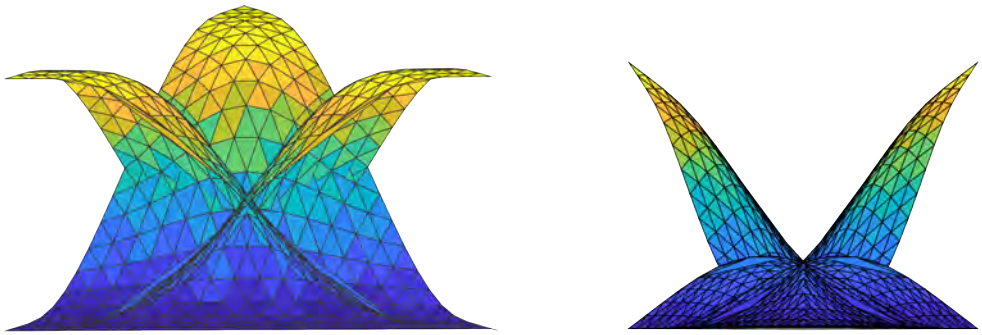
$$S(u_1, u_2, u_3) = \sum_{i=1}^{3} \ell_i(u_1, u_2, u_3)\, \beta_i(u_1, u_2, u_3), \tag{2}$$

where $u_1 + u_2 + u_3 = 1$ and $u_1, u_2, u_3 \geq 0$.

So far we have collected results from previous work on ERBS on triangular domains.

To use the ERBS triangles in a finite element context, we need to introduce a basis for ERBS triangle that combines both Bernstein basis and expo-rational basis. Locally, i.e. on the one ERBS triangle, the number of combined basis functions is equal to $q = 3\binom{d+2}{2}$.

The Bernstein polynomial basis, defined on the triangle K, can be written as a matrix W with ordered elements



(a) ERBS basis in barycentric coordinates

(b) Combined expo-rational basis in barycentric coordinates, constructed along parameter u_l

Figure 1: Basis functions in barycentric coordinates.

$$W_d^K = \begin{bmatrix} b_{0,0,d}^d & b_{0,1,d-1}^d & \dots & b_{d,0,0}^d \end{bmatrix},$$

where

$$b_{i,j,k}^d = \frac{d!}{i!j!k!} u^i v^j w^k, \quad u+v+w=1.$$

Then the combined basis $G^K = G^K(u_1, u_2, u_3)$ on the same triangle is formulated as

$$G^K = \begin{bmatrix} \beta_1 W_d^K & \beta_2 W_d^K & \beta_3 W_d^K \end{bmatrix}. \tag{3}$$

Thus, the formula (2) can be rewritten in a compact matrix form as

$$S(u_1, u_2, u_3) = (P^K)^{\mathrm{T}} G^K, \tag{4}$$

where P^K is a set of corresponding coefficients of three local triangles.

Figure 1(b) demonstrates a set of combined expo-rational basis functions G^K, evaluated along one of parameters, the degree of local triangles is equal to 1.

Figure 2 shows an example of ERBS triangle with local Bézier triangles of degree 1. This construction is very flexible and can be fitted to a geometry of relatively high degree of smoothness.

We now construct a set of partial derivatives of the combined expo-rational basis.

Definition 2.4. Let $D_{u_\iota} W_d^K$ be a set of directional derivatives in the direction u_ι of Bernstein basis functions of degree d. Then, from (3), it follows that the partial derivatives of the combined expo-rational basis in barycentric coordinates can be found as

$$D_{u_\iota} G^K = \begin{bmatrix} D_{u_\iota}\beta_1 \, W_d^K + \beta_1 D_{u_\iota} W_d^K \\ D_{u_\iota}\beta_2 \, W_d^K + \beta_2 \, D_{u_\iota} W_d^K \\ D_{u_\iota}\beta_3 \, W_d^K + \beta_3 \, D_{u_\iota} W_d^K \end{bmatrix} = \begin{bmatrix} D_{u_\iota}\beta_1 & \beta_1 \\ D_{u_\iota}\beta_2 & \beta_2 \\ D_{u_\iota}\beta_3 & \beta_3 \end{bmatrix} \begin{bmatrix} W_d^K \\ D_{u_\iota} W_d^K \end{bmatrix},$$

for each $\iota = 1, 2, 3$.

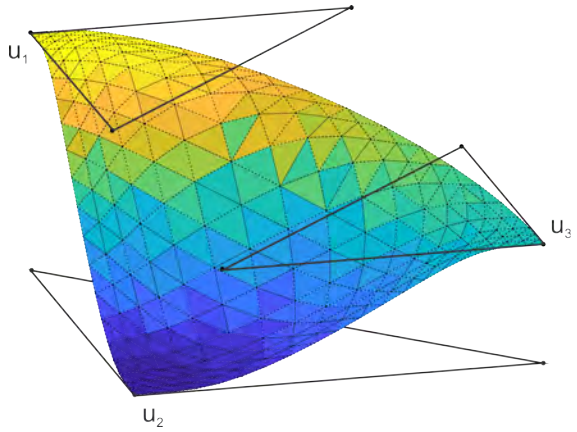


Figure 2: An example of the ERBS triangle with local Bézier triangles of the first degree.

3 Model problem

In this section we demonstrate an exact solution of the eigenvalue problem on a circular membrane and its approximation using the finite element method.

A detailed description of obtaining an analytical solution of the problem of membrane vibrations can be found in [9]. Consider a circular membrane Ω having a radius $r = a$ and a fixed outer boundary $\partial\Omega$. Introduce a given material constant c and a circular frequency ω.

Define the eigenvalue problem in the following form

$$\Delta\vartheta + \frac{\omega^2}{c^2}\vartheta = 0, \quad \text{in } \Omega, \tag{5}$$

$$\vartheta = 0, \quad \text{on } \partial\Omega. \tag{6}$$

An analytical solution of the problem (5) with boundary condition (6) is represented by two independent orthogonal eigenfunctions, referred to as the cosine and the sine modes, respectively

$$\vartheta_C^{(m.n)} = J_m(\omega_{(m,n)} r/c)\cos m\phi \quad \text{and} \quad \vartheta_S^{(m.n)} = J_m(\omega_{(m,n)} r/c)\sin m\phi, \tag{7}$$

where J_m is a Bessel function.

The circular eigenfrequencies $\omega_{(m,n)}$ of the (m, n) mode can be found from the formula

$$\gamma = \omega/c.$$

The eigenvalues $a\gamma_{(m,n)}$ denote from the boundary condition (6), which yields the characteristic equation

$$J_m(\gamma a) = 0.$$

Now, we consider a discrete solution for the eigenvalue problem. A detailed finite element approach is described in [10].

By replacing $\frac{\omega^2}{c^2} = \lambda$, we formulate the finite element method for the problem (5)-(6) as follows

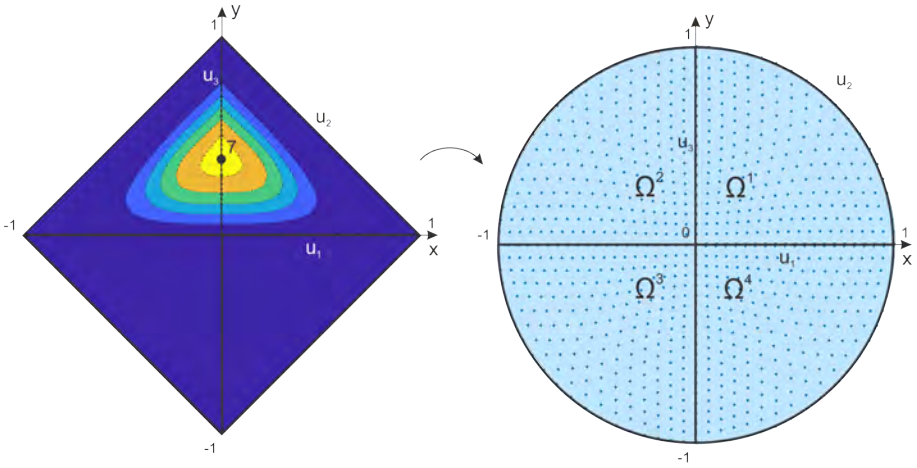


Figure 3: A mapping between triangular mesh and circular domain. A contour plot of seventh combined expo-rational basis function is shown on the left figure. Parameter lines of the circular domain are show on the right figure.

$$\left(\int_{\Omega} \nabla \mathbf{G}^{\mathrm{T}} \nabla \mathbf{G} \, d\Omega + \int_{\partial\Omega} \kappa \mathbf{G}^{\mathrm{T}} \mathbf{G} \, d(\partial\Omega) \right) \zeta = \lambda \int_{\Omega} \mathbf{G}^{\mathrm{T}} \mathbf{G} \, d\Omega \, \zeta, \tag{8}$$

where $\mathbf{G}$ is a set of all basis functions G^K, defined on the triangulated domain Ω.

In a compact matrix representation (8) is written as

$$(A + R)\zeta = \Lambda M \zeta, \tag{9}$$

where A is the stiffness matrix, M is the mass matrix and R is the boundary matrix.

The eigenvectors ζ and eigenvalues Λ come in pairs (ζ, Λ), and there are as many pairs $(\zeta_i, \Lambda_i)_{i=1}^n$ as there are coefficients $\mathbf{P}$. Then the solution ζ_i substitutes the z-coordinate of the coefficients $\mathbf{P}$ and thus the approximation of the eigenfunction ϑ_h can be obtained by the linear combination of coefficients and basis functions.

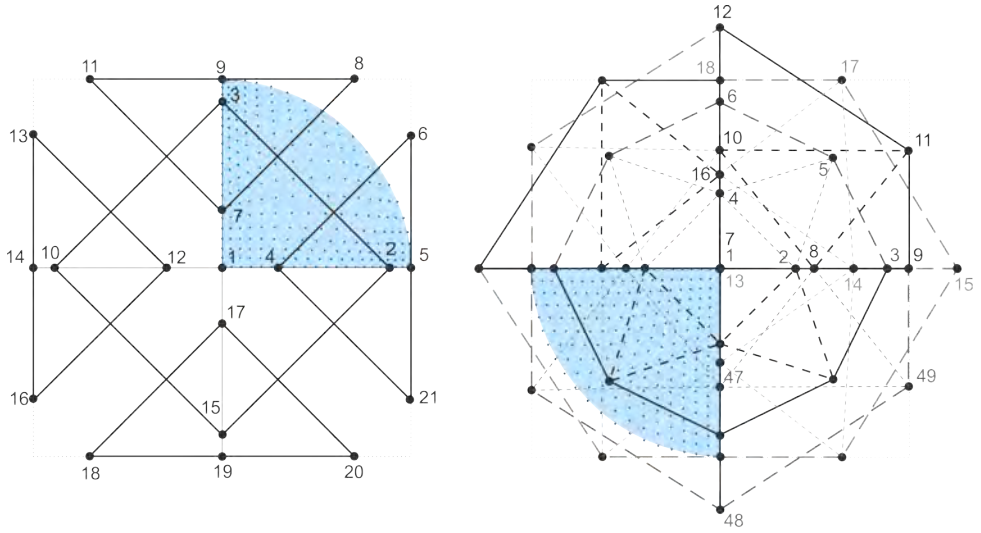
Domain construction

The main interest of the considered implementation is to explore how the very coarse mesh and low-degree local triangles handle a complex shape of the solution.

A set of combined expo-rational basis functions for one element is defined in Section 2. Each basis function G_i, $i = 1, ..., n$, is continuous, piecewise, and has its support on a set of corresponding elements.

Let $\mathbf{P} = \{P\}_{i=1}^n$, $P_i \in \mathbb{R}^2$ be a set of coefficients and a set of combined expo-rational basis functions be $\mathbf{G} = \{G_i\}_{i=1}^n$. We divide the entire domain Ω symmetrically into four elements Ω^e, $e = 1, ..., 4$. One element will be represented as a fixed triangle, with respect to which we will construct an ERBS triangulate surface to approximate a solution of the problem. A set of four elements represents a mesh. A mapping obtained by a linear combination of expo-rational basis functions and local triangles yields an approximation of the real domain. This mapping is shown in Figure 3. The set of coefficients of local triangles is called *a control net*.

A control net based on ERBS triangles has a layered structure in our example. An upper layer consists of four local triangles, which are connected by a central point. A bottom layer consists of triangles, connected by two, which construct the



(a) d = 1 (b) d = 2

Figure 4: Local triangles and their coefficients of a circular domain, constructed by four ERBS triangles. (a) First degree of local triangles. (b) Second degree of local triangles.

outer boundary. To avoid discontinuity in the domain, we merge together matching points. The number of coefficients n varies depending on the point configuration. A number of basis functions on the entire domain corresponds to the number of control points.

Figure 4 shows two examples of a control net for a mesh, which approximate a circular domain Ω with radius a. Figure 4(a) demonstrates local triangles of degree 1, Figure 4(b) shows local triangles of degree 2. The outer boundary for both cases is approximated by the L^2-projection and blending splines.

Coordinate transformation

Since the elements that we consider are curvilinear, we need to define coordinate transformations for them to compute the integrals constituting formula (8). Construction of two- and three-dimensional triangular and rectangular isoparametric elements, coordinate transformations and numerical integrations are detailed in [11], and an application of this to a problem of membrane vibration is exploited in [2].

To define correspondence between curvilinear and Cartesian coordinates we first focus on one triangular element Ω^e. The mapping between coordinates was introduced earlier, see formula (4), as a linear combination of control points $P = \{\begin{pmatrix} p_i^x & p_i^y \end{pmatrix}\}_{i=1}^q$ and basis functions $G = \{G_i(u_1, u_2, u_3)\}_{i=1}^q$, where $q = 3\binom{d+2}{2}$. This relation is valid for any local coordinate system. A slight complication with the barycentric coordinates is that they are not independent and the number of them is one more than in Cartesian coordinate system. To avoid this issue, we introduce new dependent formal variables

$$\begin{aligned} \xi &= u_1, \\ \eta &= u_2, \\ 1 - \xi - \eta &= u_3. \end{aligned} \tag{10}$$

For computation of the matrices A, M and R we need to provide two transformations. First, we express global derivatives through local derivatives. Secondly, a differential element of area has to be represented in local coordinates and the integration limits should be correspondingly changed.

We can write partial derivatives with respect to new variables of the basis functions as

$$\begin{aligned}\frac{\partial G_i}{\partial \xi} &= \frac{\partial G_i}{\partial u_1}\frac{\partial u_1}{\partial \xi} + \frac{\partial G_i}{\partial u_2}\frac{\partial u_2}{\partial \xi} + \frac{\partial G_i}{\partial u_3}\frac{\partial u_3}{\partial \xi},\\ \frac{\partial G_i}{\partial \eta} &= \frac{\partial G_i}{\partial u_1}\frac{\partial u_1}{\partial \eta} + \frac{\partial G_i}{\partial u_2}\frac{\partial u_2}{\partial \eta} + \frac{\partial G_i}{\partial u_3}\frac{\partial u_3}{\partial \eta}.\end{aligned} \tag{11}$$

Using (10) and (11), we get

$$\begin{aligned}\frac{\partial G_i}{\partial \xi} &= \frac{\partial G_i}{\partial u_1} - \frac{\partial G_i}{\partial u_3},\\ \frac{\partial G_i}{\partial \eta} &= \frac{\partial G_i}{\partial u_2} - \frac{\partial G_i}{\partial u_3}.\end{aligned} \tag{12}$$

Partial derivatives is evaluated by the formula defined in Definition 2.4. At once, the transformation between local coordinates ξ, η and the corresponding global coordinates x, y can be written in matrix form as

$$\begin{bmatrix}\frac{\partial G_i}{\partial \xi}\\ \frac{\partial G_i}{\partial \eta}\end{bmatrix} = \begin{bmatrix}\frac{\partial x}{\partial \xi} & \frac{\partial y}{\partial \xi}\\ \frac{\partial x}{\partial \eta} & \frac{\partial y}{\partial \eta}\end{bmatrix}\begin{bmatrix}\frac{\partial G_i}{\partial x}\\ \frac{\partial G_i}{\partial y}\end{bmatrix} = J\begin{bmatrix}\frac{\partial G_i}{\partial x}\\ \frac{\partial G_i}{\partial y}\end{bmatrix},$$

where J is the Jacobi matrix, which depends on the local coordinates. The differential element of area, according to [2], is

$$dxdy = |J|\, d\xi d\eta, \tag{13}$$

in which $|J|$ is the Jacobi determinant, or Jacobian of transformation.

By using (12) we find the global derivatives as

$$\begin{bmatrix}\frac{\partial G_i}{\partial x}\\ \frac{\partial G_i}{\partial y}\end{bmatrix} = J^{-1}\begin{bmatrix}\frac{\partial G_i}{\partial u_1} - \frac{\partial G_i}{\partial u_3}\\ \frac{\partial G_i}{\partial u_2} - \frac{\partial G_i}{\partial u_3}\end{bmatrix}. \tag{14}$$

Deriving J from the basis functions G_i, $i = 1, ..., q$, which define the coordinate mapping (4), we obtain

$$J = \begin{bmatrix}\sum \frac{\partial G_i}{\partial \xi}p_i^x & \sum \frac{\partial G_i}{\partial \xi}p_i^y\\ \sum \frac{\partial G_i}{\partial \eta}p_i^x & \sum \frac{\partial G_i}{\partial \eta}p_i^y\end{bmatrix} = \begin{bmatrix}\frac{\partial G_1}{\partial u_1} - \frac{\partial G_1}{\partial u_3} & \frac{\partial G_2}{\partial u_1} - \frac{\partial G_2}{\partial u_3} & ...\\ \frac{\partial G_1}{\partial u_2} - \frac{\partial G_1}{\partial u_3} & \frac{\partial G_2}{\partial u_2} - \frac{\partial G_2}{\partial u_3} & ...\end{bmatrix} P. \tag{15}$$

Global matrices A, M and R are filled in by the local-to-global mapping. A concept of element matrices is fully described in [12]. The global matrix breaks up into sums of elemental contributions A^e, $e = 1, ..., m$, where m is the number of elements.

Integration limits are changed to limits corresponding to a triangle. Finally, using formulas (13)-(15), the element stiffness and mass matrices are computed as follows

$$A^e = \int_0^1 \int_0^{1-\eta} \begin{bmatrix} \dfrac{\partial G^e}{\partial x} & \dfrac{\partial G^e}{\partial y} \end{bmatrix} \begin{bmatrix} \dfrac{\partial G^e}{\partial x} \\ \dfrac{\partial G^e}{\partial y} \end{bmatrix} |J| \, d\xi d\eta, \quad M^e = \int_0^1 \int_0^{1-\eta} (G^e)^{\mathrm{T}} G^e \, |J| \, d\xi d\eta. \quad (16)$$

A transformed integral in the boundary matrix R should be computed along a curve, which represents the boundary $\partial\Omega^e$ of the triangular element Ω^e. According to the formula for computing curvilinear integrals, we write the element boundary matrix as

$$R^e = \int_0^1 \kappa \, (G^e)^{\mathrm{T}} G^e \, ||(\partial\Omega^e)'_\sigma|| \, d\sigma. \quad (17)$$

4 Results

We solve the model problem (5) with Dirichlet boundary conditions (6). A few mode shapes are found on the circular domain Ω, which is constructed by using ERBS triangles, as shown in Figure 4. Approximations are found by formulas (9), (16)-(17) and compared with the exact solution (7).

The errors are presented in Figure 5 and computed as functions of the approximation for each of the considered mode. The error is approximated in L^2-norm

$$||\vartheta - \vartheta_h|| = \sqrt{\int_\Omega |\vartheta - \vartheta_h|^2 \, d\Omega}.$$

Figure 4 shows the construction of the circular domain formed by ERBS triangles and two types of local Bézier triangles: of degree 1 and 2, respectively. One of the resulting elements Ω^e is shown for both cases as a circle sector. Points on the element represent local parameters. One can see that on Figure 4(b) parameters are distributed more uniformly compared to Figure 4(a). By flexibility of ERBS triangle construction, we can construct specific parameter distribution, taking into account derivatives of the target surface.

Figure 6 demonstrates the results of the algorithm implementation. Simple shapes of the solution can be handled by the first degree of local triangles at the same accuracy level as for the second degree local triangles, for example for modes $(0, 1)$, $(0, 2)$, $(2, 1)$. On the other hand, the appearance of the nodal circles together with nodal diameters immediately implies incrementation of the degree of local triangles. For example, the mode $(1, 2)$ has some irregularities on the local degree $d = 1$, comparing with degree 2. It can be explained such as the shape of this mode is too complex for such low degree of local triangles.

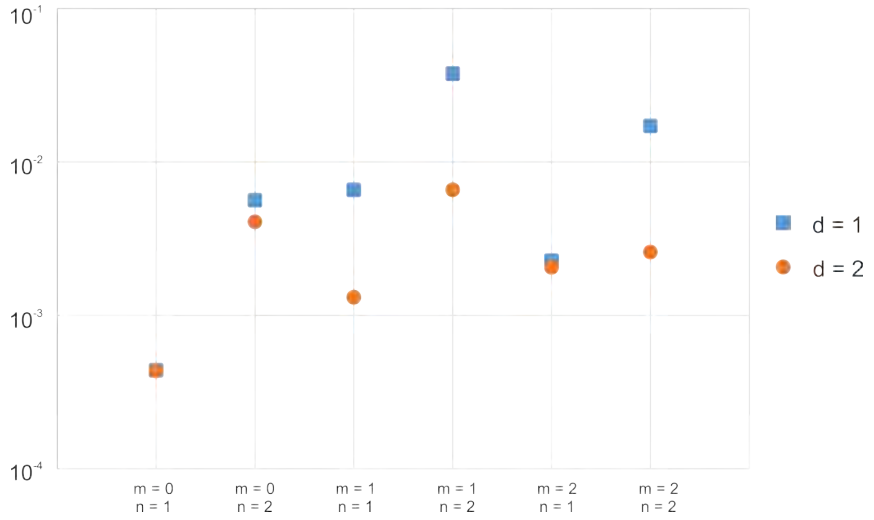


Figure 5: Comparison of L^2-error for different mode-shapes and their FEM approximation by using two types of local triangles: of the first ($d = 1$) and second ($d = 2$) degree.

5 Conclusion

We have successfully implemented the finite element method that solve the eigenvalue problem on a circular membrane with fixed outer boundary, where elements are represented as ERBS triangles.

Increasing the degree of local triangles, one can provide many different approximations of the initial surface, which satisfy the required intrinsic properties of its geometry. Blending splines allow for accurately approximation of the boundary while keeping a coarse discretization of the domain. Even complex smooth domains can be constructed on a base of a few triangular elements. The overlapping of local triangles allows us to provide a flexible handling of the surface while preserving the smoothness of the initial domain, also over the nodes and edges.

The use of finite elements based on ERBS-triangles improves the computational efficiency. The structure of local triangles and a symmetric basis allow for the preliminary computation of integrals constituting the stiffness and mass matrices. Based on this, one can parallelize the computational process. Combining the above with the fact that the computations are performed on a very coarse mesh, we conclude that the proposed algorithm can be effectively optimized.

As future work we consider the development of automatic mesh generation, optimization of computations and a more thorough mathematical analysis of the presented basis functions.

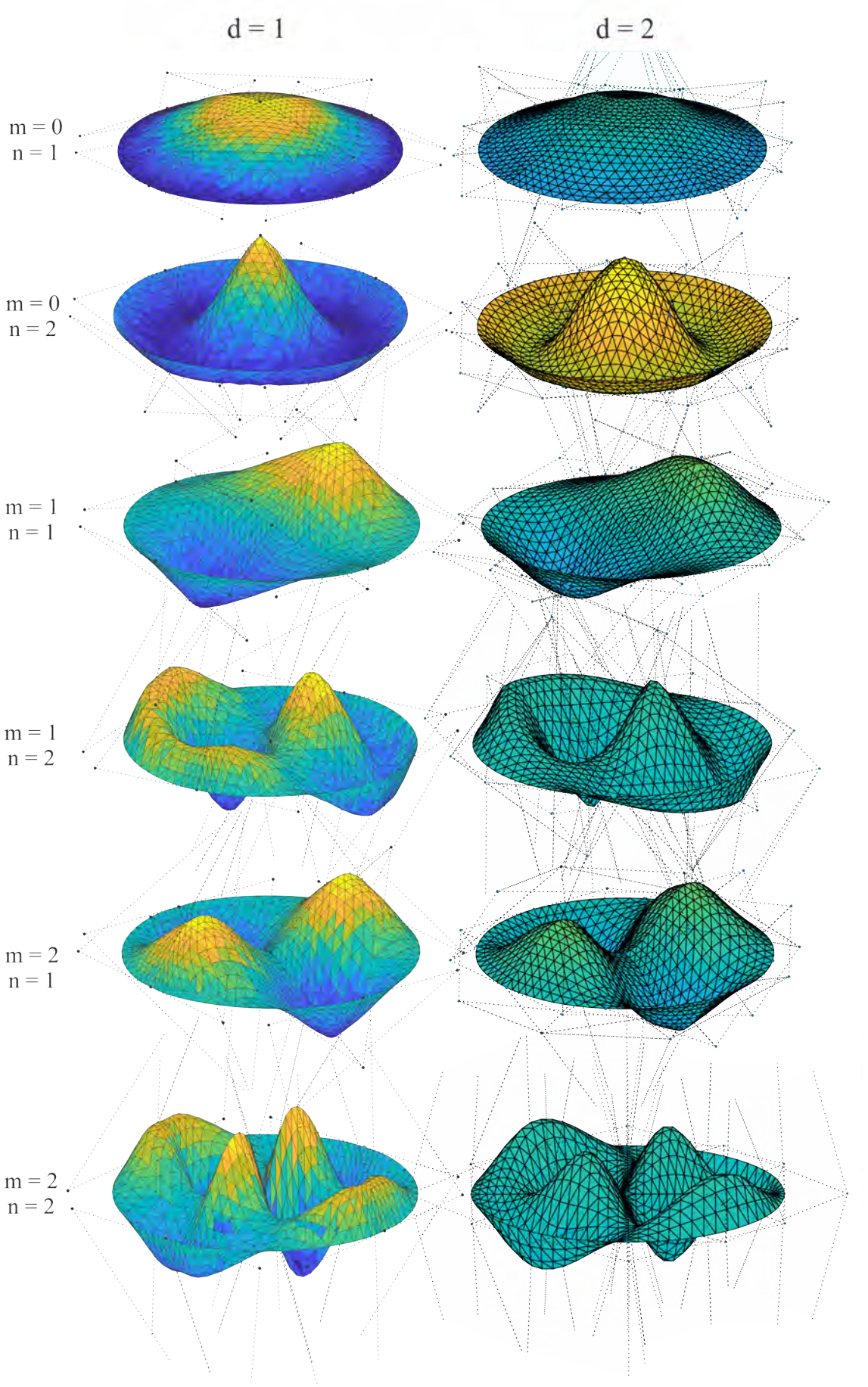


Figure 6: First few mode shapes, obtained by FEM utilizing ERBS triangles as elements. Two types of local triangles are presented: Bézier triangles of the first degree (left hand side), and of the second degree (right hand side). Local triangles are shown by points and dotted lines.

References

[1] T.J.R. Hughes, J.A. Cottrell, and Y. Bazilevs. Isogeometric analysis: CAD, finite elements, NURBS, exact geometry and mesh refinement. *Computer Methods in Applied Mechanics and Engineering*, 194:4135–4195, 2005.

[2] L. Meirovitch. *Principles and Techniques of Vibrations*. Prentice-Hall, Inc., Upper Saddle River, New Jersey, 1997.

[3] L.T. Dechevsky, A. Lakså, and B. Bang. Expo-Rational B-Splines. *International Journal of Pure and Applied Mathematics*, 27(3):319–367, 2006.

[4] A. Lakså. *Blending technics for curve and surface constructions*. Narvik University College, Narvik, Norway, 2006.

[5] R. Dalmo, J. Bratlie, B. Bang, and A. Lakså. Smooth spline blending surface approximation over a triangulated irregular network. *International Journal of Applied Mathematics*, 27(1):109–119, 2014.

[6] L.T. Dechevsky, P. Zanaty, A. Lakså, and B. Bang. First instances of generalized expo-rational finite elements on triangulations. *In: Applications of Mathematics and Engineering and Economics 2011*, 1410:49–61, 2011.

[7] P. Zanaty. Finite element methods based on a generalized expo-rational B-splines with harmonic polynomial coefficients. *International Journal of Applied Mathematics*, 26(3):379–390, 2013.

[8] M.-J. Lai and L.L. Schumaker. *Spline Functions on Triangulations*. Cambrige University Press, Cambridge, UK, 2007.

[9] P. Hagedorn and A. DasGupta. *Vibrations and Waves in Continuous Mechanical Systems*. Wiley, 1st edition, 2007.

[10] M.G. Larson and F. Bengzon. *The Finite Element Method: Theory, Implementation, and Practice*. Springer, 2010.

[11] O.C. Zienkiewicz, R.L. Taylor, and J.Z. Zhu. *The Finite Element Method: Its Basis and Fundamentals*. Elsevier Butterworth-Heinemann Linacre House, Jordan Hill, Oxford, 6th edition, 2005.

[12] T.J.R. Hughes. The Finite Element Method: Linear Static and Dynamic Finite Element Analysis. *Computer-aided civil and infrastructure engineering*, 4(3):245–246, 1989.

Unveiling the Data Shadow: A Scalable Software Architecture for Public Health and Electronically Assessed Data (PHEAD)

Tor-Morten Grønli[1], Andreas Biørn-Hansen[1], Siri Fagernes[1], Ragnhild Eg[2], Kjeld Hansen[1], and Per Morten Fredriksen[2]

[1]Mobile Technology Lab, Department of Technology,
Kristiania University College, Norway
[2]Kristiania University College, School of Health Sciences, Norway

Abstract

Health expenditures in Norway amounted to 10,2 percent of GDP in 2018, and the budget was dedicated predominantly to treating disorders and maintaining support functions. Only 3 percent of the budget went to preventive health, despite the fact that preventive measures hold the greatest promise. At the same time, computer-based technology enables measuring and gathering large amounts of health-related data, like a person's activity level, heart rate, sleep patterns, eating habits, exercise habits and even emotional state. The popularity of personal electronic recording equipment, along with the extensive use of social media, result in large collections of data that are continuously expanding. Currently this wealth of information is not accessible to healthcare professions, instead it is contained within a data shadow. Even with access, the majority of health personnel would be unable to break the data down to sensible and usable information.

This research project is founded in the perspective of preventive health, aiming to synthesize available personal health information by utilizing commodity mobile and wearable hardware. It sets out to harvest the potential inherent in the constellation of shadow data, returning the insight to the individual as personalised health advice. Through a persuasive technologies experiment, the project will explore efficient means to motivate healthy lifestyle choices. Furthermore, the project presents a prototype architecture for the collection and processing of unfiltered data that will serve as a foundation for health workers and medical doctors to base diagnoses, treatments and council upon. Our major contribution is a proof of concept implementation and leveraging state of the art cloud based function as a service approach to build a scalable software architecture for a ubiquitous and heterogeneous environment harvesting the data shadow through activity tracking devices.

1 Introduction

Countries worldwide are experiencing an increase in life-style related diseases of epic proportions. The global burden of disease is mainly caused by overweight, obesity and

This paper was presented at the NIK-2019 conference; see http://www.nik.no/.

inactivity. World-wide numbers count more than 650 billion overweight or obese adults and close to 400 million children and youths [1]. Combined with an ageing western population, these numbers bring with them disorders such as diabetes, muscle skeletal conditions and cardiovascular diseases [2]. One consequence is a financial burden that will increase dramatically. Health expenditure in Norway amounted to 360 billion in 2018, corresponding to 10,2 percent of the national GDP, a 0,6 increase per inhabitant since the previous year [3]. 97 percent of this budget is spent on treating disorders and on support functions. A meager 3 percent are allocated preventive health measures, despite the amassing research body on their beneficial effects [3]. Considering that the challenge with many of the most common illnesses today is their relation to lifestyle and individual choices, it seems the economically most feasible and effective solution is the prevention of illness. The catch is that prevention requires information. To provide the appropriate preventive care and advice, health workers, medical doctors and other professionals would need all relevant information related to the current condition of a patient. However, significant amounts of information related to a person's lifestyle are largely concealed from these professionals. Consequently, it is difficult (if not impossible) to make the correct diagnosis, prescribe the correct treatment, supply meaningful advice or give life counseling. Up until now, information regarding a person's health has only been available if: measured by a healthcare professional, provided through medical history, and knowledgeably shared by the patient.

Meanwhile, the world drifts towards digitisation and digitalisation in close to every field and area, so also in terms of health related information. There are vast amounts of information available and health related information is now used to improve a number of aspects of personal life. In a matter of short time, changes in health diagnosis, treatment and research are forthcoming. Large international players such as Apple, Google and Amazon are investing heavily and general opinion awareness is growing together with governmental and national enquiries and campaigns. Internet of Things, mobile devices and wearable computing have led to a growing ecosystem for data rich services and a diverse universe for the end users. Systems and application development have typically been conducted targeting specific user needs. We now see a shift towards more heterogeneous environments, which are *task oriented* instead, hence recognizing the users' flow of action over multiple devices to complete a task [4].

Rapid technological innovations and continued digitalisation ensure that almost any and every routine and activity can be recorded and analysed, and many choose to do so. Over time, conscious choices can become habitual; furthermore, some systems come with with interfaces that are designed to incentivate continued activity. Something as simple as a milestone reached or an acknowledging remark can serve as rewards, and rewards increase motivation [5]. With continued use, a system can analyse input data and adapt its rewards to the user's behaviour. In social media and various entertainment applications, the adaptation can go beyond simple reward mechanisms, for instance by prioritising content to pique interest further [6]. The same type of adaptation could serve as a motivational mechanisms in health-promoting applications, using adaptive feedback to establish and reinforce a reward cycle for healthy lifestyle choices.

2 Related Work

In this section we will elaborate on related work to our research in accordance with the three major pillars in the project, personal health information, software architecture and persuasive technologies.

Personal Health Information

Diseases move along a continuum from healthy to sick, yet current treatment strategies prioritise patients at the sicker end of the spectrum [7]. This implies that the vast majority of health measures is directed at a minority of the population, while those at lower risk are left to themselves. Assuming an eventual progress along the disease continuum, the majority of the population can be considered in a stage of sick, albeit without clinical diagnoses or symptoms. This creates a paradox; although the number of sick or high risk patients are reduced, the number who will eventually require medical help increases.

The challenge remains to identify potential risks for future disease, to enable early action for the larger portion of the population. One fruitful solution is to create an analytical model that can assess grade of disease along the continuum and to identify early stages of action [7]. This redefines the current disease-reducing strategy to a risk-reducing strategy, encouraging a wider focus on population health and a larger budget allocation to preventive health work. Population-based research will be crucial for future public health.

Information is key to preventive health care; not only knowledge about the relation between lifestyle and illness, but insight about the individual and their behaviour. At the moment, information on patients' health and lifestyle is confined both in access and applicability:

- **Information measured by healthcare professionals.** Administrative data recorded during routine health care include attendance, procedures and diagnoses. These are entered digitally into administrative systems, and are then pooled at local, regional and national levels [8].

- **Information provided in patient journals.** Patient data include medical history, journals and previous treatments, hospital visits, physiological tests and laboratory tests.

- **Information shared by patients.** Health assessment is no longer restricted to healthcare professionals, anyone can register and monitor their own symptoms, physiological functions and various activities using any number of devices and applications. For example, activity levels, heartrate, sleep rhythms, eating habits, exercise habits, emotional states, and sentiments shared in social media, to name a few [8].

Daily interactions with technology have extended to many different aspects of life, from mundane routines and work chores to entertainment and physical activity. Moreover, social media applications, such as Facebook, Instagram, Snapchat and Twitter, gather indirect data on interests, relations and emotional states through consumed content and communication patterns. Most of these behaviours are voluntary, yet the expanse of data left behind online is difficult for the casual user to comprehend. The more we do online, the larger the shadow of our online actions grows. The data shadow encompasses all unintentional, or less intentional, data that an individual generates online, including clinical data. Together, activities on different technologies and arenas make up arrays of data that share the potential to yield unique insights about an individual's life-style related behaviour.

The inherent potential in these massive data collections is left largely untapped, for several reasons. Firstly, although a lot of this information is publicly available online, little or nothing is openly accessible to healthcare professionals. Secondly, health personnel

underestimate the value of self-recorded and self-generated information. And thirdly, there is a general lack of knowledge and skill in using this type of data [8]. Nonetheless, advances in data analytics demonstrate the potential in applying statistical algorithms on patient data, for instance in calculating the risk of outbreaks of infectious diseases [9]. With the power of algorithmic analyses, a system could monitor all measures and activities and offer customised health advice, which in turn could be adjusted in accordance with the resulting effect. In other words, everyday actions can deliver the insight needed to provide the still-healthy population with customised life-extending advice.

The puzzle pieces are all there, the only thing needed to complete the puzzle is to assemble the pieces. Public Health and Electronically Assessed Data (PHEAD) aims to do just that, by developing a system that collects and analyses data from mobile and wearable devices. With the potential that follows large-scale data analysis, PHEAD will enable early detection of disease and other consequences related to life-style. Fine-meshed data limits will ensure that the system can divide the continuum from completely healthy to sick in the same way that many diseases are now divided into stages. This will socially, economically and not least individually be very important for future treatment and prevention of illness and disease.

Architectural Paradigms

Through years of evolution different software architectures and paradigms have developed. From the early days of client-server to modern day serverless architectures and with many stages in between. We will further highlight some of these paradigms with particular importance.

Service Oriented Architecture (SOA) was coined as a term in 1996 and it has since become the state-of-the-art in many software architecture. All large software vendors today offer various frameworks and implementations of SOA [10] and Microsoft, Amazon, Heroku to name a few host easy configurable cloud based environments for such deployments. SOA is a framework for designing flexible and loosely integrated services, in distributed environments with the main goal to address the challenge of supporting architectural innovation and flexibility in an ever changing software development environment. SOA as an architectural principle should aim to support architectural flexibility with the ability to include both new and legacy systems into the software platform. Further, its distributed nature should support efficiency, reuse, security, adaptability and maintainability [11].

Following the work on Service Oriented Architectures other perspectives evolved as well with Microservice Software Architectures as the most predominantly one [12, 13]. Miroservices is debated and not uniformly defined. However, usually most descriptions will recognise it by design principles. The design principles [14] often include the possibility to rapidly change and adapt, that the individual units (services) are small, coherent functions of work solving one specific task and by living in a cloud based environment they out of the box fully support a highly scalable approach.

In recent years, the serverless architectural paradigm has enjoyed the attention of practitioners and researchers alike – and interest is growing [15]. From the perspective of a software developer, the serverless paradigm's main purpose is to provide an abstraction over traditional operations related to hosting and orchestration of servers, and hardware-based scaling (e.g., vertical and horizontal system scaling) [16]. In Figure 1, five common abstractions in the X as a Service umbrella are displayed, ordered from closest to the

metal (left) to the furthest away (right). The serverless paradigm aligns closer to the right, typically seen as an umbrella term covering Back-end as a Service (BaaS) while incorporating the Functions as a Service (Faas) paradigm as a core enabling component for executing business logic. The choice of runtime environment for FaaS is directly tied to cost and warm and cold startup performance of the FaaS functions, challenges investigated in depth by Jackson and Clynch [17]. The BaaS paradigm is composed of a set of Software as a Service (SaaS) components, typically user management, databases, file storage and similar. For the PHEAD project, the FaaS paradigm is in focus – however, we also make use of other BaaS and SaaS components including CosmosDB for persistent data storage, and Azure Event Grid enabling our event-driven architecture. Also (federated) identity management is likely a component we will integrate with in future work.

Figure 1: Layers of abstraction for X as a Service

Persuasive Technologies

Many behavioural responses are triggered by external stimuli, be it the reply to a question or the orientation towards a sound. Traditional learning theories attribute behavioural learning to the association that follows repeated exposure to a new stimulus and a subsequent response, mediated by an established stimulus [18]. This approach to learning has been adopted by a large number of technology providers, who incorporate basic stimulus-response mechanisms in their product designs. Names may vary, but playful, motivational and persuasive designs all aim to motivate the continued use of a product [5]. For instance, gamification mechanisms such as reward schemes are implemented in many social media applications [19]. These mechanisms are typically presented as rewarding responses that provide value to a voluntary behaviour, thereby reinforcing the relation between the behaviour and the response [6]. The response can be as simple as a high-score, a badge, a running streak, or a number of views, likes or comments to a piece of content shared by a user. However, the mechanisms are not as simple as they appear. A user's behaviour and responses provide data that can be used to accommodate the user further, such as filtering out relevant content or adjusting the frequency of notifications [6]. This type of nudging can certainly get someone hooked on an application [20], but it can also be used to encourage lifestyle changes.

Fogg [21] took a conceptual view on persuasive technologies and proposed the idea that there are five primary types of social cues for people to recognize computers as social actors: Physical, psychological, language, social dynamics and social roles. Fogg [22] developed a behavioral model for persuasive design named the Fogg Behavioral Model (FBM). The model consists of three elements: Motivation, ability, and triggers.

Persuasive design of health applications incorporates a number of features to encourage users both to use the application, and to change their behaviour. The features can be grouped into four categories, social support, system credibility support, dialogue support and primary task support. These categories contain features with different mechanisms, some appeal to social needs, some use call-for-actions to trigger behaviour, and some are aimed at the individual's preferences [23]. However, none of these features

consider the context or the timing of the nudge. Previous work has suggested that increased focus on the user's current context of use is of key importance for improving the user experience of the wearable. In particular, lack of context-awareness in fitness trackers has been listed as a significant drawback, for instance when the user is not able to act on a nudge motivating to move, when the user is sitting on an airplane [24].

3 Scenario

Imagine the following scenario for our user, *Laura*: After getting out of bed this morning the user opens the smartphone to read the latest news, check Facebook, e-mail, and complete the morning ritual in the online world. Whilst doing this the activity tracker on Laura's arm are synchronizing the sleep pattern to the vendor cloud via the smart phone - she gets a thumbs up for having completed a good nights sleep. On the way out the apartment, the system picks up on this and calculates the likeness for that she now will choose the bus, rather than walk, to work. A gentle nudge with a motivating suggestion are shown encouraging Laura to walk today since the exercise pattern from the previous two days are below her expected standards. Well at work she has a busy day with meetings and office time combined. The system is aware of her schedule and gives updates on the progress towards the daily activity goal in between meetings and work session to be non-intrusive. After work hours, when she has finished her round of errands, the systems has a suggestion ready for an evening activity in line with the system profile for a user of her category - she completes her power-yoga session successfully. When she is at bed for the night, again the smartphone synchronizes today's activities and information to the backend cloud. An analytics-engine is crunching the data from the day from all the registered users, detecting anomalies, identifying patterns and organizing the data storage. Where consent has been given, data is shared with health care services for individual health profile updates, sharing the newly gained knowledge. In preparation for the next day, machine learning algorithms are gaining new insight from the last 24 hours of data, and accordingly updates the individual users profile. Whilst all this has happened, Laura has almost completed a good nights sleep. She is now one hour away from waking up to a new day with personalised, individually tailored health profile activities.

4 Prototype Architecture

Instead of manufacturing our own wearables, PHEAD is using existing off-the-shelf hardware widely available to the general public. The challenge is to find a common ground to make these wearables accessible for use. Establishing a joint software platform and an Application Programming Interface (API) to collect data from wearables of different manufactures is therefore necessary. All manufactures have their own mechanisms of collecting data based on their own hardware and algorithms, hence; we will access and pull this information from the dedicated manufacturers upon given user consent.

When examining the architecture components in Figure 2, which shows the software architectural implementation with emphasis on the major components, we can identify three major components: End user component with smartphone and wearable; wearable device vendor cloud; Azure Cloud Environment. All three components are vital parts and we will further elaborate on them.

The wearable device vendor clouds represent the remote service for the individual producers such as Garmin, Polar, FitBit etc. This is the standard cloud based services

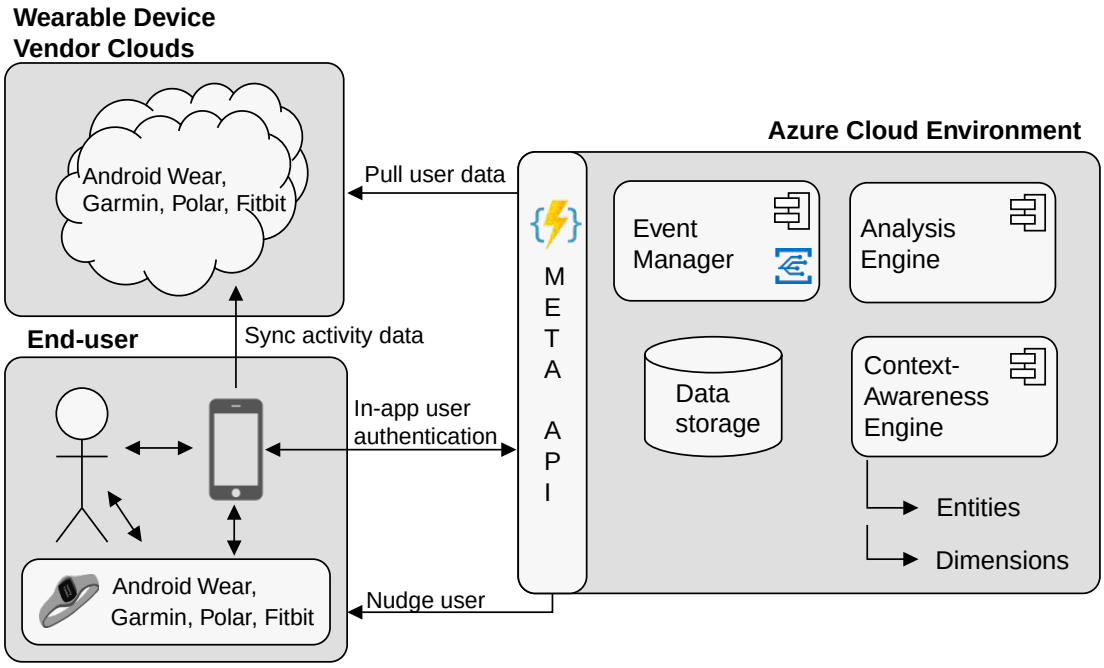


Figure 2: Conceptual PHEAD architecture

for each vendor where users typically register an account upon purchase of a new device. Vendors enable access to user data through proprietary APIs, given that a user token is acquired.

On the end-user side, PHEAD is build to expand across and to use several different platforms i.e. the central cloud based software platform, a variety of wearable devices and native applications on Android and iOS platforms. Taking a persuasive technology approach, the design of this applications incorporates a number of features to encourage users both to use the application, improve their health and to change their behaviour. The user perspective contains two major components, where the first one is the wearable device i.e. typically a watch (Garmin, Polar, Apple etc) or an activity tracker (FitBit, JawBone etc). Secondly, the users smartphone device is important as it has two main features: Firstly to act as the synchronization hub towards the vendor cloud and secondly providing the local application which is used for token generation for the backend system so that data can be fetched from the vendor cloud on the users behalf.

5 Result and Discussion

To elaborate on our findings, we will describe and discuss them in comparison with related work to our research. By doing so we will revisit it from the perspective of the three pillars of: personal health information, software architecture and persuasive technologies.

Personal Health Information challenges

Health personnel typically faces three challenges when consulting a patient; inadequate update of healthcare workers in the medical field due to limited time resources, lack of knowledge of the behaviour of the patients and an inability to change patients habits for improving health. Firstly, a GP typically sees 30 patients a day, and there is little room for self-study. As there are close to 2 million medical papers published each year, keeping updated in the field of medicine is an impossible task for a GP. Secondly, patients,

intentionally or unintentionally, does not reveal all information the GP needs to perform qualified medical treatment. The information perhaps does not occur to be important to the GP nor the patient, the patients does not remember and sometimes the patients are unwilling to share important information. Thirdly, making people change to a more healthy lifestyle in order to prevent diseases has proven very difficult, and many GP's feel helpless as their advice are not embraced by the patients.

All of these three challenges may be improved by using information gathered by wearables, social media and algorithms detecting important health-related behaviour, as well as connecting the personalised information with the wast amount of medical information available on internet and conveying it to health personnel as the GP. Perhaps more importantly, health personnel may encourage health enhancing behaviour and dissuade behaviour related to poor future health. The latter by using premeditated and situation related nudging.

The challenge is to easily obtain these data, decipher the output and make appropriate use of the information to help patients. Unfortunately, the amount of data is so large it is impossible to process this in a normal manner. This, together with an increasing specialization of the medical field, the amount of information will cause problems for healthcare professionals. That is, physicians are not expected to possess the knowledge available about the individual diseases and the best treatment. The physicians and other healthcare professionals must therefore make use of the knowledge available on the internet. This information is very complex and fragmented. Using algorithms to search the internet for the latest information directly related to the patients symptoms and diseases will allow health personnel to give the latest treatment and advises to the patients with significantly more precision.

Architectural Challenges

The main architectural flow is showcased through one vendor in Figure 3 by exemplifying with FitBit. Initial dataflow is started by having the local user activity synced to the vendor cloud through a smartphone device. Given that new data is uploaded, the FitBit vendor cloud will POST to the webhook endpoint that new data is available, accordingly triggering a FaaS function to publish this event on the Azure Event Grid. Following this, another FaaS function collects data from the vendor cloud and transforms it to a local entity format for further processing. We build and run a data validation and reliability tests for the different wearables to ensure correct flow of data from user, through vendor systems, to our software system in the Azure cloud. Wearables that are not accessible or which fails to produce valid and reliable data are omitted. The approach with FaaS architecture, support continued development and addition of new functions for data processing as needed following established principles from previous work [14, 16]. The last step of data processing is the computing of the activity context for a given user. By doing this we achieve through a context-aware taxonomy to decide whether the user should be nudged using a persuasive approach to reach h/s goal, or to terminate the current operation.

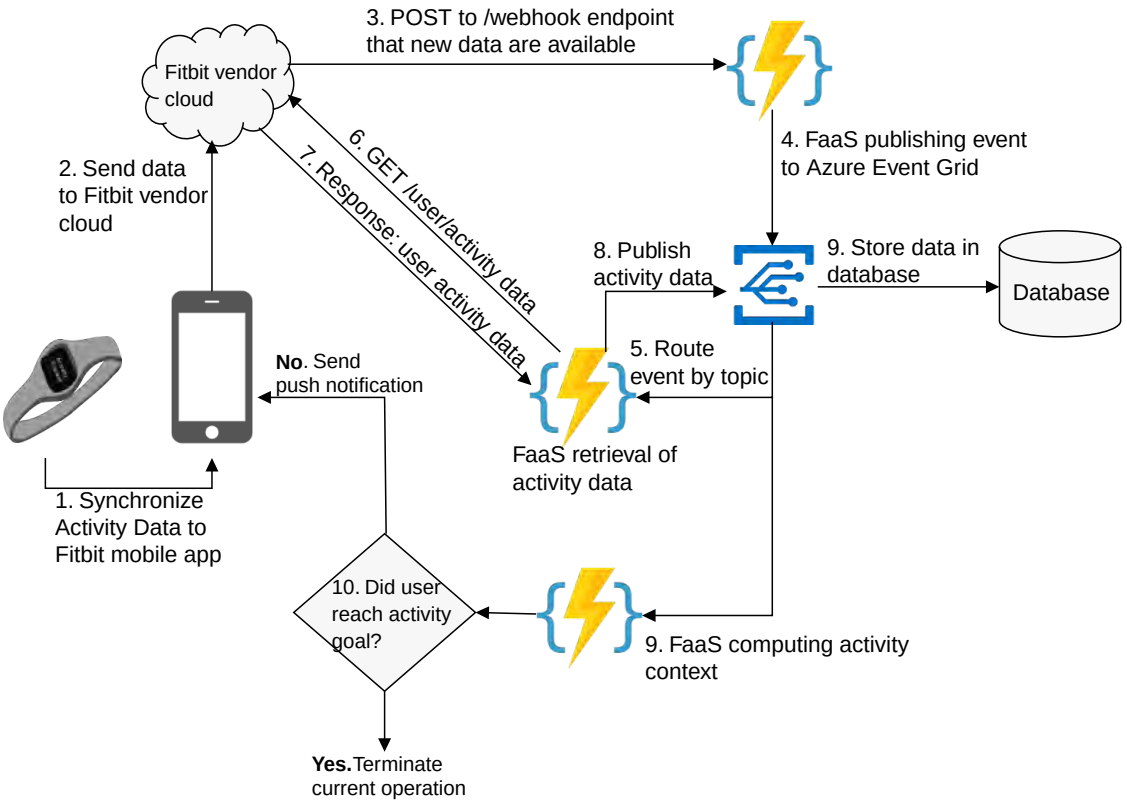


Figure 3: Fitbit PHEAD architecture

Persuasive Technology Challenges

This application function as a hub, collecting pre-planned data from the chosen applications by applying playful, motivational and persuasive designs all aim to motivate the continued use [5]. These mechanisms are typically presented as rewarding responses that provide value to a voluntary behaviour, thereby reinforcing the relation between the behaviour and the response [6]. As the main goal is to help people improve their life-style and prolong peoples life, an applicable feedback system must be developed. Telling people to change life-style on a general and not specific level has limited effect. However, human computer interaction studies have shown that so called "nudging" do have effect [23]. To encourage people to follow guidelines given based on the personalised data collection, nudging needs to be individualised and appropriate for the situation the individual is in. Small comments on a smartwatch telling you to move every hour is not very helpful, especially if you are on a plane for five hours, nor has it shown any large effects. The nudging have to be based on all the information given by the PHEAD-setup and programmed to nudge people in a setting where the nudging has an effect and people are able to follow the instructions.

Technical Challenges

The process of developing the FaaS-based PHEAD architecture is illustrated in Figure 4. One particular aspect of the process, which complicates local development and debugging of this architecture, is the absence of tooling. At the time of writing, Microsoft Azure does not provide an official emulator or local environment for their Event Grid product, a central component of the architecture, enabling routing of- and temporary storage for events. As the loosely coupled Azure Functions depend on being triggered by events

published by the Event Grid, the lack of a local emulator renders development and debugging cumbersome. The issue of debugging is also raised in a whitepaper by Fox et al. [25] reporting on the status of FaaS and serverless computing both in industry and research. Based on our experience developing this system using the Microsoft Azure platform, we note that years after Fox et al. published their report, the state of debugging and local development is still work-in-progress.

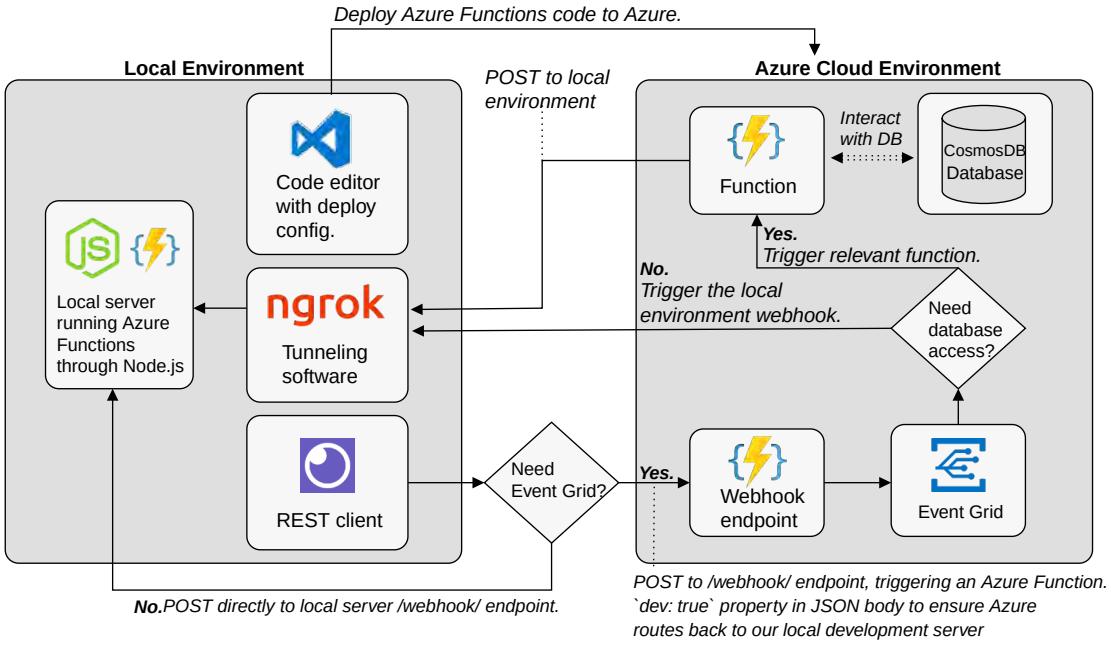


Figure 4: The process of developing PHEAD FaaS semi-locally

To route events from the Azure Event Grid back to the local development environment, the *ngrok* tunneling software is typically used to expose externally a server running on localhost, allowing the localhost server to be reached from outside the local network [26]. A RESTful API endpoint from the localhost server is then provided to the Azure Event Grid, formatted as follows after being exposed by the ngrok tunnel: `https://<unique_app_id>.ngrok.io/api/endpoint`. While this method of semi-local debugging is recommended by Microsoft [26], expanding the development team from one developer to two (or more) will require the Azure Event Grid to also handle event routing not only back to the local environment, but more specifically to the correct developer's environment. Setting up and configuring an online development environment, such as the upcoming Microsoft cloud-hosted development environments [27], to which multiple developers could connect to could be one solution to avoid riddling the Azure Event Grid with configuration and patterns for handling routing to multiple and distributed local development environments.

Limitations

We acknowledge that since this initial result and evaluation does not involve actual users with personal information, we do only initial verification of the suitability of the data and related security mechanisms. For further studies a robust security platform and data protection regime must be incorporated and tested.

6 Conclusion

This research project anchored in the cross-disciplinary fields of preventive health and software architecture showcased how to synthesizing personal health information from wearable devices to gain insight into the health data shadow. Through a developed prototype architecture and data flow test with the FitBit device, we show how individuals can gain a fact-based awareness of own health and make informed choices. Our proof of concept implementation further leverage the state of the art cloud based function as a service approach to build a scalable software architecture for a ubiquitous and heterogeneous environment harvesting the data shadow through activity tracking devices. Further to this, we highlight the benefit for such a solution for health workers and medical doctors that can be provided with a comprehensive, unfiltered data foundation to base diagnoses, treatment and council upon. This work on establishing a solution to unveil the data shadow show the huge potential for demystifying this huge amount of data, utilising state of the art serverless architectures and further highlight openings for data analysis, information tailoring and context-aware health information as highly relevant areas worthy of further pursue.

7 Acknowledgement

This work was supported by interdisciplinary funds from Kristiania University College. Regional Ethic Committee reference number: 2019/648/REK south-east C.

References

[1] World Health Organization. Obesity and overweight. `https://www.who.int/news-room/fact-sheets/detail/obesity-and-overweight`. (Accessed on 05/09/2019).

[2] World Health Organization. Double burden of malnutrition. `http://www.who.int/nutrition/double-burden-malnutrition/`, 2019. (Accessed on 03/17/2019).

[3] Statistisk Sentralbyra. 68 000 per innbygger til helse. `https://www.ssb.no/nasjonalregnskap-og-konjunkturer/artikler-og-publikasjoner/68-000-per-innbygger-til-helse`, 2019. (Accessed on 04/09/2019).

[4] Tor-Morten Grønli, Andreas Biørn-Hansen, and Tim A Majchrzak. Software development for mobile computing, the internet of things and wearable devices: Inspecting the past to understand the future. In *Proceedings of the 52nd Hawaii International Conference on System Sciences*, 2019.

[5] Sebastian Deterding. Eudaimonic design, or: Six invitations to rethink gamification. 2014.

[6] Ryan Tan Rui Yang and Vivian Hsueh Hua Chen. Gamification: Influencing value-perception of target behaviors. In *Proceedings of GamiFIN Conference*, 2017.

[7] Geoffrey Rose, Kay-Tee Khaw, and Michael Marmot. *Rose's strategy of preventive medicine*. Oxford University Press, 2009.

[8] Sarah R Deeny and Adam Steventon. Making sense of the shadows: priorities for creating a learning healthcare system based on routinely collected data. *BMJ Qual Saf*, 24(8):505–515, 2015.

[9] Jeremy Ginsberg, Matthew H Mohebbi, Rajan S Patel, Lynnette Brammer, Mark S Smolinski, and Larry Brilliant. Detecting influenza epidemics using search engine query data. *Nature*, 457(7232):1012, 2009.

[10] Michael Rosen, Boris Lublinsky, Kevin T Smith, and Marc J Balcer. *Applied SOA: service-oriented architecture and design strategies*. John Wiley & Sons, 2012.

[11] Haluk Demirkan, Robert J Kauffman, Jamshid A Vayghan, Hans-Georg Fill, Dimitris Karagiannis, and Paul P Maglio. Service-oriented technology and management: Perspectives on research and practice for the coming decade. *Electronic commerce research and applications*, 7(4):356–376, 2008.

[12] Cesare Pautasso, Olaf Zimmermann, Mike Amundsen, James Lewis, and Nicolai Josuttis. Microservices in practice, part 1: Reality check and service design. *IEEE Software*, 34(1):91–98, 2017.

[13] Cesare Pautasso, Olaf Zimmermann, Mike Amundsen, James Lewis, and Nicolai Josuttis. Microservices in practice, part 2: Service integration and sustainability. *IEEE Software*, (2):97–104, 2017.

[14] Martin Fowler and James Lewis. Microservices a definition of this new architectural term. *URL: http://martinfowler. com/articles/microservices. html*, page 22, 2014.

[15] Cloudability. State of the cloud 2018. Technical report, 2018.

[16] Neil Savage. Going serverless. *Communications of the ACM*, 61(2):15–16, January 2018.

[17] D Jackson and G Clynch. An investigation of the impact of language runtime on the performance and cost of serverless functions. In *2018 IEEE/ACM International Conference on Utility and Cloud Computing Companion (UCC Companion)*, pages 154–160. IEEE, December 2018.

[18] Burrhus F Skinner. Operant behavior. *American psychologist*, 18(8):503, 1963.

[19] Natasha Singer. Can't put down your device? that's by design. *The New York Times*, 2015.

[20] Nir Eyal. *Hooked: How to build habit-forming products*. Penguin UK, 2014.

[21] B Fogg. Computers as persuasive social actors, chapter 5, 2003.

[22] Brian J Fogg. A behavior model for persuasive design. In *Proceedings of the 4th international Conference on Persuasive Technology*, page 40. ACM, 2009.

[23] John Matthews, Khin Than Win, Harri Oinas-Kukkonen, and Mark Freeman. Persuasive technology in mobile applications promoting physical activity: a systematic review. *Journal of medical systems*, 40(3):72, 2016.

[24] Vivian Genaro Motti and Kelly Caine. Smart wearables or dumb wearables?: Understanding how context impacts the ux in wrist worn interaction. In *Proceedings of the 34th ACM International Conference on the Design of Communication*, page 10. ACM, 2016.

[25] Geoffrey C Fox, Vatche Ishakian, Vinod Muthusamy, and Aleksander Slominski. Status of serverless computing and Function-as-a-Service(FaaS) in industry and research. Technical report, IEEE, August 2017.

[26] Craig Shoemaker, Theano Petersen, and Nick Schonning. Azure functions event grid local debugging. `https://docs.microsoft.com/en-us/azure/azure-functions/functions-debug-event-grid-trigger-local`, October 2018. Accessed: 2019-9-6.

[27] Jonathan Carter. Intelligent productivity and collaboration, from anywhere. `https://devblogs.microsoft.com/visualstudio/intelligent-productivity-and-collaboration-from-anywhere/`, May 2019. Accessed: 2019-9-6.

Translating 2D art into Virtual Reality and comparing the user experiences

Daniel Patel[1,2] Runar Tistel[1] Atle Geitung[1] Harald Soleim[1]
[1] Bergen University College (HVL), Norway [2] NORCE Technology, Norway
dapa@norceresearch.no

Abstract

Recent advancements in virtual reality on the hardware and software front have made high-quality virtual reality experiences both cheaper, and easier to obtain. This paper explores how virtual reality changes the way a user experiences art and if virtual reality is suited as a medium for expressing art. Based on two existing artworks, we have created VR versions using a game engine, and conducted a user study to get a comparison of how the experience of the traditional artworks differ from the VR versions. The artworks have been created in 3D using algorithmic modelling techniques.

Introduction

Displaying a computer-generated 3D scene to a user by showing stereoscopic images through a head mounted display (HMD) is a type of Virtual Reality (VR). The 3D scene can be updated based on the user's view direction, giving the user the feeling of being inside and looking around in the scene. Recent advancements in VR hardware and software have made high-quality VR both more affordable, and easier to obtain. VR has established itself in the domain of games, training and design, but has not been used much for communicating art. VR, compared to e.g. paintings, offers several additional dimensions such as stereoscopy, the ability to display arbitrarily large works of art, free and effortless movement in the virtual space, dynamic/animated 3D structures, sound, interaction between the viewer and the art, and interaction between multiple viewers. This makes VR an attractive medium for expressing art. Compared to classical museums, new digital technology such as VR is to a larger degree embraced by youngsters. Digitized content has also the advantage that it can be distributed directly to homes. Instead of bringing the youngsters to the museums, one can imagine bringing museums to the youngsters in a new and exciting format. This way of thinking has been brought forward by the Norwegian artist Bjarne Melgaard with his VR artwork "My Trip" [1]. VR can enable art to reach more people and new age segments.

As we are not artists, we decided to expand already existing artworks into VR. The artworks we chose are the recently deceased (24 April, 2018) Norwegian artist Pushwagner's paintings Selvportrett (Self-portrait), and Manhattan. We chose these two works because they are highly regular and therefore easy to reproduce with a computer program. In addition, the works are interesting as they are spatial and have lots of structure. Using existing art makes it possible for us to compare the evaluated VR version with the original version which allows us to find out how virtual reality changes the way a user experiences conventional 2D art. In this paper we also want to find out to which degree VR can be considered as an art medium and not only a digital display technology.

According to the Norwegian Copyright Act (Lov om opphavsrett til åndsverk m.v.,), artworks do not enter public domain before 70 years after the authors death (§ 11). Therefore, we do not depict the artworks but refer the user to look up the works which are easily accessible on the web. Regarding our VR versions, the copyright act (§ 6) states that translated, adapted or transferred pieces of work are under the copyright of the

This paper was presented at the NIK 2019 conference. For more information see http://www.nik.no/

original work, whereas new and independent works created by making use of existing work are not. It possible that our VR versions can be interpreted to fit under the first case, therefore we do not make our VR software available and we only show screenshots of our derived works that we consider to be sufficiently different from the original works.

Related Work

VR Hardware. A heightened sense of immersion has been sought after in the film industry for years. 3D cinema using stereoscopic image projection and higher quality surround sound systems are just some examples. An early attempt at VR was the Sensorama [2] by Morton in 1962. It attempted to give viewers a more immersive movie experience by stimulating as many senses as possible. Electric fans simulated wind, and the release of aromas and vibrating motion was produced to give the user a heightened sense of presence. Sutherland's work in 1968 [3] is considered to be the first HMD system, incorporating many of the same techniques seen in modern HMD systems such as head-tracking and stereoscopic display. However, it was so heavy that the HMD unit required to be suspended in the air through a ceiling mounted mechanical arm. The CAVE project [4] (1992) is a later attempt at VR that projects the virtual environment onto the surrounding walls of a square room using projectors, to create the illusion of a 3D environment. The CAVE project has been applied extensively in engineering environments to e.g. enhance product design and development. In 2016, several high quality and affordable HMD units together with supporting software and development kits were released to the market from companies such as Oculus, HTC, Valve and Sony. This brought VR experiences for the first time to the general public. Tracking of position and orientation is supported both on the HMDs and on hand-held input devices. Factors that define the quality of the VR experience are the pixel density and the field of view of the HMD display, the latency and polygon count of the rendering hardware as well as the need for cables required for display and tracking. In this paper, the Oculus Rift Consumer Version 1 with 1080×1200 resolution per eye and 90 Hz refresh rate was used.

Art and VR. Long before VR hardware was made, artists had been working on related ideas. In 1792, Barker [5,6] painted the world's first panorama which depicted a full 360 view of London and was installed in a purpose-built circular building. A platform in the middle of the building allowed the viewer to look around at an uninterrupted 360 view of the painted surroundings of London. In 1849, the classical music composer Wagner described his vision in the essay "The Artwork of the future" [7] of integrating art forms such as music, dance and poetry in a single total artwork. His vision may now be possible to achieve with today's VR technology. In 1995 Char Davies who was originally a painter created one of the first VR art installation [8]. The installation consisted of an HMD displaying computer graphics that was controlled by the user's breath and balance. Breathing out caused the user to sink and breathing in caused them to float upwards. The user could move laterally by leaning in the wished direction. In 1997, Murray in her landmark book [9] predicted the rise of digital storytelling. She stated that VR enables the participant to enact stories rather than merely witnessing them since the participant is the center of the virtual world. VR's ability to give a strong sense of presence in a work, and its ability to fuse multiple art forms as was the vision of Wagner, shows that it has a large potential for use in arts. Around 2016, museums started utilizing VR to expand the availability of their art collection. The VR Museum of Fine Art [10] is a VR application that presents digital replicas of works of art from the Museum of Fine Arts Boston, in a fully explorable 3D environment. In addition, translating existing art into the VR medium has been done for the works of famous painters such as Van Gogh (2016) [11], Dali(2016)

[12], Bosch (2016) [13] and Magritte (2018) [14]. These VR adaptations have also been extended with animations and sound.

In our work we also translate existing art into VR. Our main purpose is however not the end-product itself, but to perform a systematic survey to learn more about VR as an art medium and how it differs from 2D paintings. In 2017, the exhibition "The Unframed World" [28] explored Virtual Reality as an artistic medium. The exhibition conveys the aesthetic potential of Virtual Reality and examines its role as a critical medium for reflection on states of being in the world today. Like we do, the Master's thesis [15] also explores the question about VR as an artistic medium, however only qualitatively through studying existing VR artworks and by discussing them. He concludes that "the capacity of virtual reality is not currently used to its full extent when it comes to artistic manifestation." The VR art applications reviewed so far have been created by programmers and 3D model designers, possibly guided by artists. Artists without a technical background can still create VR art directly in VR using familiar paintbrush metaphors. Tools like Tilt Brush [16], and Oculus Medium [17] give users the ability to paint and sculpt models in three dimensions using HMD's and tracked hand controllers which act as digital 3D brushes.

Programmatic and algorithmic generation of art. VR experiences found in HMD's are generated from software running on a connected PC or running in the HMD itself. Established software for creating VR applications are game engines such as Unreal [18] and Unity [19] which offer a programmatic approach and a visual user interface to the graphical assets. The solution presented in this paper was made in Unreal.

VR and 3D applications consist of a computer program that does the rendering and digital assets such as 3D models and sound clips. The 3D models can be designed in modelling software such as Blender [20] or Maya [21]. As opposed to manually creating models and content for a digital art project, there is a field of digital art called generative art where the artist creates a computer program that then automatically or semi automatically generates art, possibly seeded by a random number generator or some user defined input. The first example of such art was made by Noll in 1964 at Bell Labs [22]. He created a program that automatically generated a digital version of Mondrian's "Composition With Lines" painting.

Translating 2D art to VR

The works Selvportrett and Manhattan by Pushwagner were chosen because of their spatial structure and visual appeal. The Selvportrett scene is made with black pen/marker on white paper and depicts a large crowd of people looking out from inside of a cylindrical structure consisting of many mezzanines, reminiscent of a tall stadium. The top is domed and covered with images of faces. The bottom of the scene seems to stretch downwards endlessly. In the center of the cylindrical room there is a spiraling line of people stretching from bottom to the top. Manhattan is a colored painting of several tall and thin skyscrapers in bright colors along a passage. The buildings bend and sway in an unnatural way and stretch into infinity downwards, upwards and into the perspective. The sides of the skyscrapers have bright colors and windows with people looking out from. Both paintings have a strong sense of space and scale due to the relation between repeating structure, empty space and perspective. They portray large expansive spaces which can be represented well in VR. The artworks also feature regularity and repeated elements which makes them easier to translate to 3D than other works of less regular structure. The Manhattan painting also indicates a waving movement of the skyscrapers which we represent as animation in VR. Pushwagner typically does not want to talk about the meaning of his artworks, but willingly talks about perspective, depth and the feeling of

3D. He wants the observer to feel he/she becomes transported into the artworks [23]. This is exactly what VR can achieve. The regularity in our two selected paintings, also found in several other works by Pushwagner, makes it possible for us to define the scenes programmatically/algorithmically instead of modelling each element in the scene manually in 3D modelling software. Programmatic definition of 3D models, also known as parametric modelling, is the act of describing a model based on input parameters and using loop counters and mathematical expressions to define it. For instance, the geometry of a chessboard can be defined by having its width and number of squares in x and y direction as input parameters, then inside a for loop, the geometry of each of the rows is created. This for loop has another for loop inside that creates each individual cuboid shaped square using two alternating color. Doing parametric modelling of our two paintings has several advantages, it is faster for the designer to create the scenes procedurally compared to manual modelling, it becomes easier to dynamically change the scenes at runtime by changing the parameters, and the scenes take up less space in memory during rendering since they can be generated on the fly.

We deconstructed the geometry in Selvportrett and Manhattan into building blocks and reconstructed them by positioning the building blocks according to e.g. number of levels in the buildings. The human geometry shown in Selvportrett was made externally in the software MakeHuman [24] which consists of an editor where a human model can be generated based on different parameters such as age, height, gender and muscularity. The human model consists of 20K triangles, and we had 5400 humans in the scene, resulting in 109M triangles not counting in the triangles for the building. To achieve interactivity, we reduced the triangle count by representing the human mesh at three decreasing levels of detail (LOD), se right figure. Humans that are far away and have smaller footprint on the screen due to perspective projection, are rendered at lower level of detail than humans that are close to the viewer. Lower levels of detail are only used when the footprint is so small, that the reduced detail level is not discernable.

In addition to producing the geometry of the scene, we need to alter the drawing style of the geometry to fit the drawing medium and technique the artist has used. To replicate this style in 3D, we changed the rendering style from the default style of drawing colored surfaces to instead only draw the edges between surfaces. This was done by rendering depth values (Figure 1a) and surface normals (Figure 1b) in hidden buffers and setting pixels to black that have sufficiently different depth or normal values than its neighbor pixels, otherwise pixels are set to white (Figure 1c).

Often when an artist draws by hand, small imperfections in the lines in the form of wiggles are created from the drawing tool, the canvas, or the artists hands being unsteady. Therefore, to introduce an extra layer of realism to the artistic style, and to further replicate the style of Pushwagner, some noise is added to the lines (Figure 1d). This noise offsets the pixels in the lines slightly both vertically and horizontally. The noise is sampled from a 2D buffer of same resolution as the output image. The 2D buffer can be interpreted as unevenness in the drawing canvas which produces the wiggles in the drawn lines. Therefore, the noise values are not recalculated but stay the same from

frame to frame. Getting the right rendering style for the Manhattan painting required less work, we could basically use standard rendering with colored surfaces. In addition, we used 2D textures to render the windows with the people looking out. More detailed information on the rendering process is described in the master thesis [25].

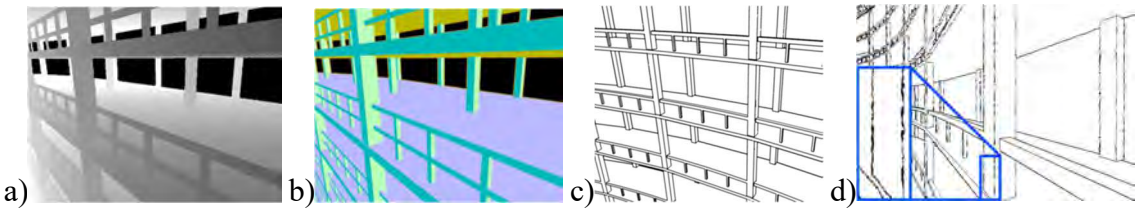


Figure 1 a) Distance from viewer to surface mapped to grayscale (depthmap). b) Surface normals mapped to RGB color. c) Resulting line drawing by using data from a) and b). d) Adding wiggles to lines by offsetting pixels in c) with random values. Blue large box shows a zoom-in on the blue small box. (Images in a)-d) are shown from slightly different viewpoints).

Evaluation of VR art

In this paper, we wanted to find out how people experience VR art compared to conventional 2D art, and if VR is suitable for expressing art (Table 1). For this we translated two existing 2D art paintings into VR, exposed both the 2D and the VR versions to respondents, and asked them to do a comparison (research question 1). By letting the respondents compare two versions of the "same" piece of art, we could formulate concise questions that would be easy to answer and analyze, as opposed to answering more open non-grounded questions. After having started a thought process in the responders, we followed up with open questions to answer our second research question.

Table 1. Research questions

1. How does VR change the way a user experiences art?
2. Is VR suited as a medium for expressing art?

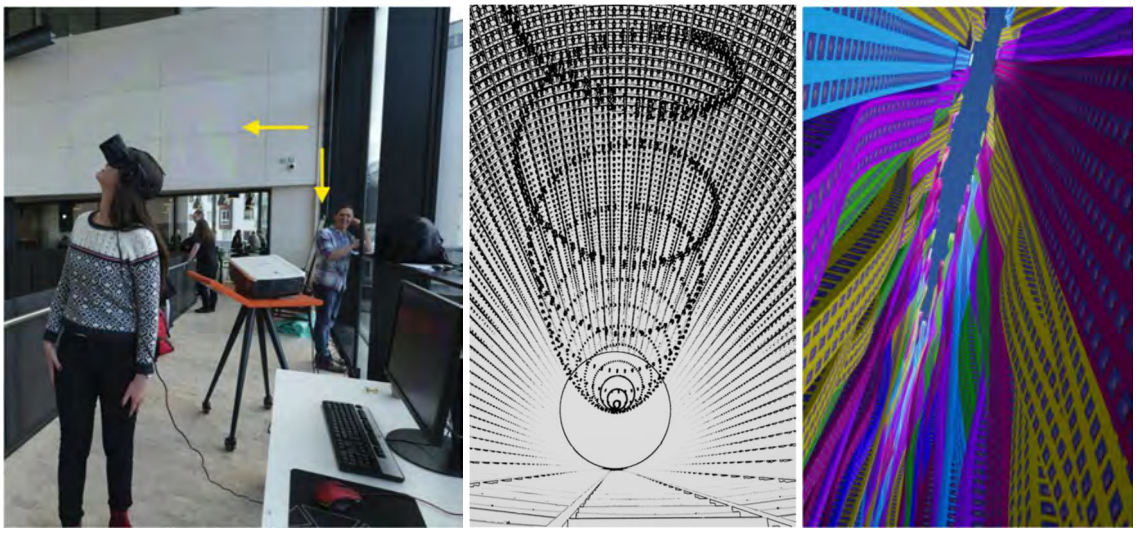


Figure 2 Left: Public demo. Middle: Screenshot of Selvportrett. Right: Screenshot of Manhattan.

We conducted a public demo session at the Faculty of Fine Art, Music and Design, University of Bergen (Figure 2). This location was chosen to attract qualified respondents that more likely had knowledge on aesthetic and arts and the ability to formulate insightful answers. Each respondent was first shown an original print of Selvportrett (positioned below the yellow vertical arrow in Figure 2), followed by a picture of Manhattan, on a

computer monitor. We were not able to get hold of an original print of Manhattan, and posters do not exist, therefore the answers to question 3 and 4 shown below may have been influenced by this situation. However, several respondents were already familiar with this painting. The respondent was then shown our VR version of Selvportrett, and of Manhattan. The rendering in the VR headset was also displayed on a wall, vaguely visible at the yellow horizontal arrow in Figure 2 left. Afterwards, the user was given the seven questions shown in Table 2. A total of 18 people tried the VR application, and 15 completed questionnaires were returned to us.

Table 2 Questions given after having experienced the VR artworks.

1. Are you a student or do you have higher academic rank?
2. What are your initial impressions of the scenes?
3. How do you think the scenes work, compared to the original art?
4. Do you think the scenes capture the spatial experience of the original scenes?
5. Do you think virtual reality changes the possibilities in artistic expression?
6. Do you use, or do you think you will use VR tools to create art? If so, why?
7. What do you think my role is in this production, am I an artist?

The first question was made to get the educational background of the responder so we could evaluate if the answers were influenced by this. We found no clear correlations between the answers and the background and have therefore not further analyzed the responses on this question. Questions 2, 3 and 4 were designed to get shed light on research question 1 by getting the responders' feedback on the impression of our VR scenes compared to the original 2D work of art. Question 2 was an open question, while the following were more specific. The open question enabled us to capture additional information which would not have been captured from the answers on the specific questions. Finally, questions 5, 6 and 7 were meant to get feedback on research question 2 regarding VR as an art medium in itself as opposed to only a digital presentation tool. All individual answers can be found in the Master's thesis [25]. In addition to the written feedback on the questions, we noted down exclamations and comments given during the demonstration. Examples of noticeable oral feedback was

- "Yes, now I really am inside it!"
- "Woa, I don't dare look down!"
- "With the movement (in the scene) I could look at this for a long time."
- "Freaky! So you've made this in 3D? That must have taken a lot of time!"

Research question 1: How does VR change the way a user experiences art?

To answer this question, we specifically asked responders to compare an original work with a derived VR version of it that we had made. This was done for two works. Basing our work on existing art, gives us a baseline to do the comparison against and enabled us to create VR art without being artists. We will start reviewing the answers on questions 3 and 4 while taking material from question 2 for detailing the answers.

For question 3, "How do you think the scenes work, compared to the original art?", we have listed the most prominent parts from individual answers: "something entirely different"; "feeling of height, space"; "adds a dimension"; "you have truly entered the artworks"; "we are integrated in the work"; "from reading the art to being apart of it"; "added a new layer (but kinda blurry)"; "much more vivid"; "better depth"; "makes you feel small and not important"; "much better 3D feeling … different

perspective"; "pulls in the observer as part of the work"[1] ; "compliments each other"; "you're part of it .. different perspective.". No respondent was negative or stating that the original art was better. Twelve out of 15 specified some sort of added value in the VR versions. The three that did not explicitly state an added value were the respondents writing: "something entirely different", "added a new layer (but kinda blurry)" and "compliments each other". The added value stated from the twelve others can be categorized in these categories:

Improved spatial feeling (4 respondents): "feeling of height, space"; "adds a dimension"; "better depth"; "much better 3D feeling … different perspective".

Being inside the artwork (5 respondents): "you have truly entered the artworks"; "we are integrated in the work"; "from reading the art to being part of it"; "pulls in the observer as part of the work"; "you're part of it .. different perspective".

Feeling of insignificance in relation to something larger (4 respondents): "it makes you feel small and not important. You are just a small piece of a big pie"; "Massive, kinda scary. It feels like I'm being part of a cell"; "I felt small"; "I feel the scenes focus on that I am only a small part of an infinitely large universe"[2]. Other sorts of additional contribution not categorized were "good added value"[3]; "much more vivid"

For question 4, "Do you think the scenes capture the spatial experience of the original scenes?", 14 out of 15 respondents answered positively, where four of these stated that it was even better. The only respondent answering negatively stated: "Not quite. In the original it seems like the viewer has a lot more in a fisheye perspective". The perspective is indeed different and more exaggerated in the original Selvportrett work, however we decided not to use it as the unnatural fish-eye lens perspective distortion combined with interactive change of viewpoint reduced realism and had a nauseatic effect. The answers show that the spatial experience is at least as good or better than in the original work and this further supports the observation of improved spatial feeling from question 3. In retrospect, we see that this question might have been too narrow as the spatial dimension is only one of the many aspects when observing art.

The answers from Question 2, "What are your initial impressions of the scenes?" can be categorized into that of feeling small and unimportant as already discussed, feeling dizzy or vertigo and the following generic positive feedback: "Interesting and fascinating"[4]; "very realistic despite the simplicity"; "huge, large space"; "Well thought out and executed"; "Very impressive"; "Properly executed"[5]; "very exiting", "Impressive and describing"[6]; "interesting".

It is evident from both immediate oral response, and written response, that a viewer of our VR scenes is transported to an active viewing position inside the artwork. This active viewing position brings a different perspective on the art, and can help an artist invoke emotions, such as that of feeling insignificant and small. Feelings of space, scale, and movement are transmitted well in VR. The responders were also in general very positive to our VR works. The respondents stated that the VR version closely resembled the original art when seen from the original viewpoint. We do however not to show comparison images in the article due to copyrights.

Research question 2. Is VR suited as a medium for expressing art?

Questions 5-7 were used to shed light on research question 2. For question 5, "Do you think virtual reality changes the possibilities in artistic expression?", 12 out of 15

Original quotes*: [1]«dra inn betrakter som ein del av verket. [2]«eg føler at scenene fokuserer på at eg berre er ein liten del i eit uendelig stort univers» [3]«god merverdi». [4]"Interessant og fascinerende" [5]"Fullført skikkelig" [6]"Imponerende og beskrivende"*

answered yes, 2 answered maybe, where one stated "maybe. It will give a new dimension and experience so that you can find new ways.", and one answered blank. Responders that also gave additional information stated: "I think the future will be exciting"[7]; "opens up new worlds of opportunities never ever imagined before"; "new mode of expression"; "It will come to people directly"; "It will give a new dimension and experience so that you can find new ways."; "Definitely, but also makes you depend on more tools like cables, glasses, pc etc."; "VR does exactly this by moving the observer from a static position to an almost active role in the work. This gives the audience a completely different role, so yes! I think VR will change the expression dramatically."[8]; "Yes you can do things that otherwise would be impossible. (Would have been even cooler with sound)"[9]; "Yep! It will not replace anything but creates a different kind of artistic expression". The respondents answered very positively and pointed out that VR expands the possibility of artistic expression.

Question 6, "Do you use, or do you think you will use VR tools to create art? If so, why?" received five yes answers, three maybe answers and five no answers. One answered "Not me, but my students" and one answered blank. One responding no, stated he did not like digital art and that he got dizzy, another pointed out the need for cables and setup: "Personally, no, I neither have the tools or the knowledge or the motivation to". This was also pointed out by another in question 4: "makes you depend on more tools like cables, glasses, pc etc.". Equally many answered yes as no. Two answering no pointed out the need for equipment and one didn't like digital art. It is probably natural to be critical to using a technique for creating art that one is not trained in. As demonstrated in this paper, it is not trivial to create custom VR experiences. Knowledge of programming, computer graphics and mathematics is required. When simpler content-creation tools and equipment will be available, it is possible that the adoption in arts will be higher. This view is also expressed by one of the responders: «This is still relatively new and for a special interest group. In my opinion the possibilities and use of VR will explode and become a natural way to communicate and interact with each other»[10]

Question 7, "What do you think my role is in this production, am I an artist?" received two yes answers, four answers that could be categorized as maybe ("yes and no"; "kinda half way"; "not sure"; "somehow"[11]), eight no answers, and one blank answer. Responders answering no, argued that the VR work is not original but derived from existing art. This may implicitly indicate that the respondents would have answered yes if the VR work was original and that they therefore do consider VR as a medium for expressing art. Some responders stated this explicitly. One of the two respondents answering yes, argued that although the work is a copy, the result is sufficiently different to consider the creator an artist. The other respondent answering yes stated" Well yeah, isn't all art about imitation? To give form to a shape where there was none? To create is to be a creator and that is to be an artist, coding or not.". As we know, there are many definitions of what art is. To create the VR artwork, we had to do a certain degree of creative work as opposed to only performing a mechanic and methodical translation. A 2D figurative drawing is a projection of an underlying 3D scene. The VR creator had to reproject the 2D drawing into 3D. Also, he had to decide on what should exist behind the viewer in case the viewer rotates 180 degrees. For the second artwork he had to decide

Original quotes*: [7] "Tror vi går en spennende tid i møte" [8] "VR gjer nettopp dette med å forflytte betrakter frå ei statisk stilling til ei nesten aktiv rolle i verket. Dette gjer at ein får ei heilt anna rolle som publikum, så ja! Eg trur VR endrar uttrykka ganske så voldsomt" [9] "Ja du kan gjøre ting som vanligvis ville vært umulig. (Hadde blitt enda kulere med lyd)" [10] «Ennå er dette relativt nytt og for en spesielt interessert gruppe. Etter min mening kommer mulighetene og bruken at VR til å eksplodere og bli en naturlig del i hvordan vi kommer til å kommunisere og samhandle med hverandre» [11] "på sett og vis" [12] "Et sterkt utrykk som gir en god merverdi til originalverket. Denne form for kunstformidling har et stort potensiale i å formidle kunst til et nytt publikum"*

how animation of the buildings should behave. These can be considered the VR creator's interpretation of the art and may be the reason why respondents stated that the VR version was something more than the original artwork. ("A strong expression that gives added value to the original work. This form of art mediation has a substantial potential in communicating art to a new audience"[12])

For the second research question, we registered enthusiasm and 12 out of 15 respondents stated that VR expands the possibilities in artistic expression. When asked if they use or will use VR, there was an even distribution of opinions with five yes, three maybe, and five no answers. VR is under rapid development, and the responses point towards that when the technology has advanced further, it will become viable for users that are used to more traditional artistic tools. The question "Am I an artist?" received mostly negative response since very little original content was added in the VR version. However, from the answers one can deduce that if the VR work had been more independent from the original work, the producer would be considered as an artist. In conclusion, the answers indicate that the responders do consider VR as an art medium, and as an exciting and promising one.

Discussion and Conclusions

Our user survey showed that VR is able to get a user more involved in a piece of art than when looking at a painting. We also showed that VR is good at representing 3D space and movement. In an interview [26], the VR artist Zakarian summarizes the experience by stating "With VR art people can visit the world in my head instead of looking at it through a window". These findings may not be surprising as they are qualities inherent in the technology itself. More interesting is the open and positive attitudes of the responders regarding art in VR. They find VR to be a fascinating and promising new medium but are reluctant to start using it before it has matured more. Chung, the CEO of a virtual reality film production studio states about VR: "You have to basically define the medium as you go. It's almost like you're trying to create the paintbrush while trying to create a painting." [27]. The VR hardware and software technology are evolving fast, and it is therefore hard to learn something that is still changing. In addition, the art medium has not established its own terminology, visual expression and set of techniques yet and it is complex to use. Zakarian states "I had to teach myself about 12 different programs and extensions and to get intimate with the hardware. It seems a bit crazy, since I could as well hire a team to produce VR art for me like other artists do, but I think a lot of the original vision gets lost when the artist doesn't get their hands dirty." [26]. It is interesting to notice that in VR, the art consumer gets a direct access to an artist's vision, while producing VR art is hard as the artist does not have a very direct access to the medium the art is created in. It is easier to express oneself in a medium one is so trained in that it becomes an extension of oneself, and here VR still has a way to go.

Also, worth pointing out is the additional creative process needed for translating a 2D piece of art into VR. This required more individual decisions than we had anticipated. This process required extending the art and therefore captures an individual interpretation of the art, which may deviate from the artists original vision.

Although not the main topic, an example of the deconstruction of (figurative) art into its geometry and drawing style and the parameterization of each component has been demonstrated in this paper. The parameterization can be considered a capture of the essence of an artwork and opens up for changing the parameters to create variations of the artwork. This aspect was to a larger degree explored in the Master's thesis this work builds on [25]. For most types of figurative art, it can be hard to find parameterizations that facilitate the creation of a 3D model. This is possibly true for art depicting irregular

and organic shapes such as the ones found in nature. However, in the Master's thesis we demonstrated the recreation of the organic effect of Salvador Dali's melting watches by using soft-body physics simulation in real-time through NVIDIA's FleX simulator. This enabled us also to interact and touch the watches in VR while they behaved as thick cloth-like objects.

Acknowledgements

We would like to thank Torkel Bernsen and Fredrik Salhus from the Faculty of Fine Art, Music and Design, University of Bergen for taking part in this project and giving valuable advice. We also thank the anonymous reviewers for insightful feedback.

References

[1] NRK, "Sender unge inn i en «syretripp» for å lokke dem til kunsten," 2019. [Online]. Available: https://www.nrk.no/kultur/halvparten-av-alle-unge-gar-aldri-pa-museum-1.14540177 .

[2] M. L. Heilig, "Sensorama simulator". US Patent US3050870A, 28 August 1962.

[3] I. E. Sutherland, "A head-mounted three dimensional display," *AFIPS '68 (Fall, Part I): Proceedings of the December 9-11, 1968, Fall Joint Computer Conference,* pp. 757-764, 1968.

[4] C. Cruz-Neira, D. J. Sandin, T. A. DeFanti, R. V. Kenyon and J. C. Hart, "The CAVE: audio visual experience automatic virtual environment," *Communications of the ACM,* vol. 35, no. 6, pp. 64-72, June 1992.

[5] R. Barker, Artist, *London from the Roof of the Albion Mills.* [Art]. 1792.

[6] Wikisource, "A Compendium of Irish Biography/Barker, Francis," 2011. [Online]. https://en.wikisource.org/wiki/A_Compendium_of_Irish_Biography/Barker,_Robert

[7] R. Wagner, The artwork of the future and other works., Lincoln: University of Nebraska Press, 1993.

[8] K. Fieldgate, "Osmose by Char Davies," 2017. [Online]. Available: http://www.immersence.com/ .

[9] J. H. Murray, Hamlet on the holodeck: The future of narrative in cyberspace., New York: The Free Press, 1997.

[10] "The VR Museum of Fine Art," 20 August 2016. https://store.steampowered.com/app/515020/The_VR_Museum_of_Fine_Art/ .

[11] Samsung VR, "The Starry Night VR," 2016. [Online]. Available: https://samsungvr.com/view/4cS0iSadKdm .

[12] Salvador Dali Museum, "Dreams of Dali," Goodby Silverstein & Partners, 2016. [Online]. Available: https://thedali.org/exhibit/dreams-vr/ .

[13] BDH, "Bosch VR," [Online]. Available: https://www.bdh.net/immersive/bosch-vr

[14] BDH, "Magritte VR," [Online]. Available: https://www.bdh.net/immersive/magritte-vr .

[15] B. Kim, "Virtual reality as an artistic medium: A study on creative projects using contemporary head-mounted displays," Aalto University, Helsinki, 2016.

[16] Oculus, "Tilt Brush," 2017. [Online]. Available: https://www.oculus.com/experiences/rift/1111640318951750/ .

[17] Oculus, "Oculus Medium," 2016. [Online]. Available: https://www.oculus.com/experiences/rift/1336762299669605/ .

[18] Unreal Engine, "Make Something Unreal," [Online]. Available: https://www.unrealengine.com/en-US/.

[19] Unity, "Unity for all," [Online]. Available: https://unity.com/.

[20] Blender, "Blender, made by you," https://www.blender.org/.

[21] Autodesk, "Maya," [Online]. Available: https://www.autodesk.com/products/maya/overview.

[22] A. M. Noll, "Human or Machine: A Subjective Comparison of Piet Mondrian's "Composition with Lines" (1917) and a Computer-Generated Picture," *The Psychological Record,* vol. 16, no. 1, pp. 1-10, 1966.

[23] O. Bhar, "Push meg forsiktig" in *Pushwagner*, Oslo, Galleri Pushwagner. ISBN 978-82-998762-0-9, 2011, p. 31.

[24] MakeHuman Community, "MakeHuman," [Online]. Available: https://bitbucket.org/MakeHuman/makehuman/src/default/.

[25] R. Tistel, "Projecting Art into Virtual Reality. Creating artistic scenes through parametrization utilizing a modern game-engine," The University of Bergen, 2018.

[26] Denis Semionov VR Art, "Mariam Zakarian: "With VR art people can visit the world in my head instead of looking at it through a window"," 2018. [Online]. Available: http://semionovdenis.com/en/2018/02/17/mariam-zakarian-with-vr-art-people-can-visit-the-world-in-my-head-instead-of-looking-at-it-through-a-window/ .

[27] Artsy, "Virtual Reality Is the Most Powerful Medium of Our Time," 2016. [Online]. Available: https://www.artsy.net/article/artsy-editorial-virtual-reality-is-the-most-powerful-artistic-medium-of-our-time .

[28] The Unframed World. Virtual Reality as Artistic Medium for the 21st Century. HeK (House of Electronic Arts Basel), Switzerland. 2017. [Online]. Available: http://www.peertospace.eu/the-unframed-world/ .

RadixInsert, a much faster stable algorithm for sorting floating-point numbers

Arne Maus (em)
Dept. of Informatics, University of Oslo
arnem@ifi.uio.no

Abstract

The problem addressed in this paper is that we want to sort an array *a*[] of *n* floating point numbers conforming to the IEEE 754 standard, both in the 64bit double precision and the 32bit single precision formats on a multi core computer with *p* real cores and shared memory (an ordinary PC). This we do by introducing a new stable, sorting algorithm, RadixInsert, both in a sequential version and with two parallel implementations. RadixInsert is tested on two different machines, a 2 core laptop and a 4 core desktop, outperforming the not stable Quicksort based algorithms from the Java library – both the sequential Arrays.sort() and a merge-based parallel version Arrays.parallelsort() for $500<n<250$mill by a factor from 3 to 10.

The RadixInsert algorithm resembles in many ways the Shell sort algorithm [1]. First, the array is pre-sorted to some degree – and in the case of Shell, Insertion sort is first used with long jumps and later shorter jumps along the array to ensure that small numbers end up near the start of the array and the larger ones towards the end. Finally, we perform a full insertion sort on the whole array to ensure correct sorting. RadixInsert first uses the ordinary right-to-left LSD Radix for sorting some left part of the floating-point numbers, then considered as integers. Finally, as with Shell sort, we perform a full Insertion sort on the whole array. This resembles in some ways a proposal by Sedgewick [10] for integer sorting and will be commented on later. The IEE754 standard was deliberately made such that positive floating-point numbers can be sorted as integers (both in the 32 and 64 bit format). The special case of a mix of positive and negative numbers is also handled in RadixInsert. One other main reason why Radix-sort is so well suited for this task is that the IEEE 754 standard normalizes numbers to the left side of the representation in a 64bit double or a 32bit float. The Radix algorithm will then in the same sorting on the leftmost bits in *n* floating-point numbers, sort both large and small numbers simultaneously. Finally, Radix is cache-friendly as it reads all its arrays left-to right with a small number of cache misses as a result, but writes them back in a different location in *b*[] in order to do the sorting. And thirdly, Radix-sort is a fast O(*n*) algorithm – faster than quicksort O(*nlogn*) or Shell sort $O(n^{1.5})$. RadixInsert is in practice O(*n*), but as with Quicksort it might be possible to construct numbers where RadixInsert degenerates to an $O(n^2)$ algorithm. However, this worst case for RadixInsert was not found when sorting seven quite different distributions reported in this paper. Finally, the extra memory used by RadixInsert both in its sequential and parallel versions, is *n* + some minor arrays whereas the sequential Quicksort in the Java library needs basically no extra memory. However, the merge based Arrays.parallelsort() in the Java library needs the same amount of *n* extra memory as RadixInsert.

Keywords: Radix sort, Insertion sort, IEEE 754, parallel algorithms, multicore, Java, Shell sort.

1. Introduction

The chip manufacturers have since 2004 not delivered what we really want, which simply is ever faster processors. The heat generated with an increase of the clock frequency will make the chips malfunction and eventually melt above 4 GHz with today's technology. Instead, they now sell us multi core processors with 2-32 processor cores. Some special products with 50 to72 cores are also available [2, 21], and the race for many processing cores on a chip is also found in the Intel Xeon Phi processor with its fast, unconventional memory access and 62 to72 cores [2]. However, many cores on a chip does not guarantee much faster execution – more often than not the bottleneck for such processing chips is memory and the memory channels. Many of these processors, but not all, are hyperthreaded, where some of the circuitry is duplicated such that each core can switch between two threads within a few instruction cycles if the active thread is waiting for some event like access to main memory.

To the operating system one such hyperthreaded core then acts as a separate core. The conclusion to all this parallelism is that if we faster programs, we must make either new faster algorithms and parallel versions of current or such new algorithms. This paper presents a new sorting algorithm for floating point numbers with two different parallel implementations.

The rest of this paper is structured as follows. First, the sequential RadixInsert algorithm is explained with some comments on how it is implemented in Java. Then RadixInsert is compared with other algorithms in the literature – where the most relevant turned out to be Shell sort from 1959[1] and the ideas in [10]. The two parallel implementations of RadixInsert, one uses an improved merge algorithm [11], and the second uses a full parallel Radix-sort that is explained in more detail. Then graphs for sorting different distributions of numbers on two different machines are presented comparing RadixInsert with the Quicksort based Arrays.sort and Arrays.parallelsort in the Java library. Observations on the differences between the performance on 32bit and 64bit numbers are discussed. Finally, this paper concludes.

2. The sequential RadixInsert algorithm

The problem addressed is that we want to sort a array *a*[] of length *n* (with 64 bit or 32 bit floating point numbers in the IEEE 754 [2,3,13] standard) on a shared memory machine with *p* cores. This we do by first sorting on *m* bits the left part of all the numbers by using the ordinary right-to-left Radix sort (LSD) reading the IEEE 754 numbers as 64 bit (or 32 bit) integers and sort them as integers. The IEEE 754 standard defines many floating point representations; in this paper we focus on the 64bit and 32 bit formats. An IEEE754 number consists of three parts – the leftmost bit is a signbit, then a modified exponent part of length 11 bits for the 64 double representation and 8 bit for the 32 bit format. The thirdrightmost part is the mantissa – the significant bits (with this exponent and sign bit).
We assume that the LSD Radix-sort is well known[4,10]. If we have sorted the left part on *m* bits from bit 63-*m* to bit 63, the sign bit (bits are numbered 63..0), we have then created 2^m sub regions or buckets where all elements in one bucket are larger than all elements in any bucket to the left, and smaller than all elements in any bucket to the right – this because they all have different values in the *m* leftmost binary digits and are sorted on these *m* bits. How we treat negative numbers is described later.

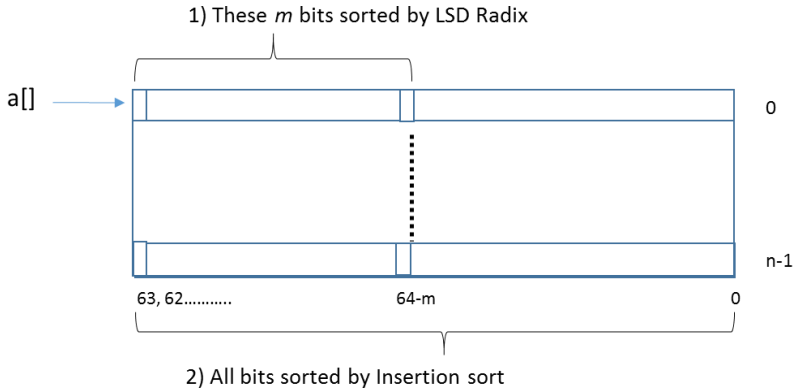


Figure 1. *The array a[] of 64 bit IEEE754 numbers, first partially sorted by LSD Radix and finally by Insertion sort.*

Within each such bucket the elements are not sorted, they are in the order they appear on input. That is another way of saying that this partial LSD sorting of *a*[] is stable.

It is also the case that these small buckets are not of the same size - that is data dependant. Finally, in RadixInsert the whole array *a*[] is sorted using the ordinary, stable Insertsort. This ensures that we have done a stable full sorting of all elements in *a*[] – regardless of their initial distribution. Because LSD sorting makes such a good job of localizing almost equal sized numbers, this last Insertsort phase can be made very quick, but in a theoretical worst case can be $O(n^2)$.

Some more details of the rationale behind this new algorithm. It is based on three observations:

1) The IEEE 754 is such that positive numbers can be sorted as integers.
2) The IEEE 754 standard left-justifies all numbers such that large and small numbers alike have their most significant bits starting at bit 63 (31). When sorting on the same left part of all numbers, they are all sorted to the same degree. If the left part of *a[]* is sorted on *m* bits, then *a*[] is dvided int 2^m buckets, and:
   all elements in bucket $_i$ > elements in bucket $_{i-1}$ (but each bucket is not in any way sorted internally – they are in the same order as on input).
3) When sorting a mix of positive and negative floating-point numbers with LSD Radix, this way, because negative numbers have their sign bit set, negative numbers will be sorted last in reverse order posing as the largest numbers and with the smallest negative number rightmost in *a[n-1]*. Negative numbers we solve by testing the last element after the LSD sorting but before Insertionsort. If that element is negative, by binary search we find the first, leftmost negative number. Then we use the extra array *b*[] of the same length as *a*[], copy first the negative part of *a*[] to the front of *b*[] while swaping it. Last in *b*[] we copy the positive section. As the last but one step we copy *b*[] to *a*[].To keep the stable property also for negative numbers, we then have to walk through the negative section once more because equal elements was first stably sorted to the last section in *a*[]. When we swaped it to the front of *b*[], and later back to *a*[], we reverse that order. As a last step we walk through the negative section element by element and when finding a subsection with equal elements, we swap them a second time, and hence ensure stable sorting.

3. Comparison with other algorithms

The use of Insertion sort for finalizing sorting of a partial sorted array using LSD Radix was nowhere to be found for sorting of floating point numbers, but for integer sorting, Sedgwick proposes first sorting half of the bits with LSD Radix and then finalizing the sorting by applying insertion sort on the whole array. A webpage on sorting IEEE 754 numbers by only using Radix sorting on all 32bits on the short format was found [9]. Almost all papers on sorting concern the sorting of integers, and might miss the interesting observation 2) above which might make floating point numbers more easy to sort. Not many specific algorithms specifically suited for IEEE 754 are presented. Almost all remarks on the net [9] describe using Radix sorting of floating-point numbers with first converting IEEE754 number to some other format, byte or decimal, before sorting. They seem unaware of observation 1) above. These authors seem to rely on the correct observation in Donald Knuth [5] that algorithms for integer sorting are also well suited for sorting floating point numbers in the sense that they concentrate on integer sorting algorithms. A mixed sorting of LSD Radix finished with a sweeping insertion sort on the last bits is not found for floating point numbers. One good reason might be that a pure Radix sorting of all significant bits on the (right justified) integers is usually faster. However, when

investigating distribution based integer sorting algorithms such as Quicksort, the use of Insertion sort for finalizing the sort, are abundant.
RadixInsert for IEEE754 numbers resembles in many ways Shell sort[1] from 1959. In Shell sort Insertion sort is first used along long jumps for shifting small elements to the left and larger to the right of *a*[], later progressively shorter jumps are used until Shell sort finish by sorting all elements using Insertion sort. Shell sort can be described as an Insert-Insert sort algorithm.

4. The run time efficiency of sequential RadixInsert.

a) The general efficiency on two computers,
We first give performance figures for the two computer used for development of this algorithm; a 2core (4 hyperthreaded) laptop, with an Inteli7-4600U @ 2.1 GHz-2.2.7GHz CPU, and a Desktop with 4 core(8 hyperthreaded) Intel i7- 6700 CPU @ 3.40 GHz.

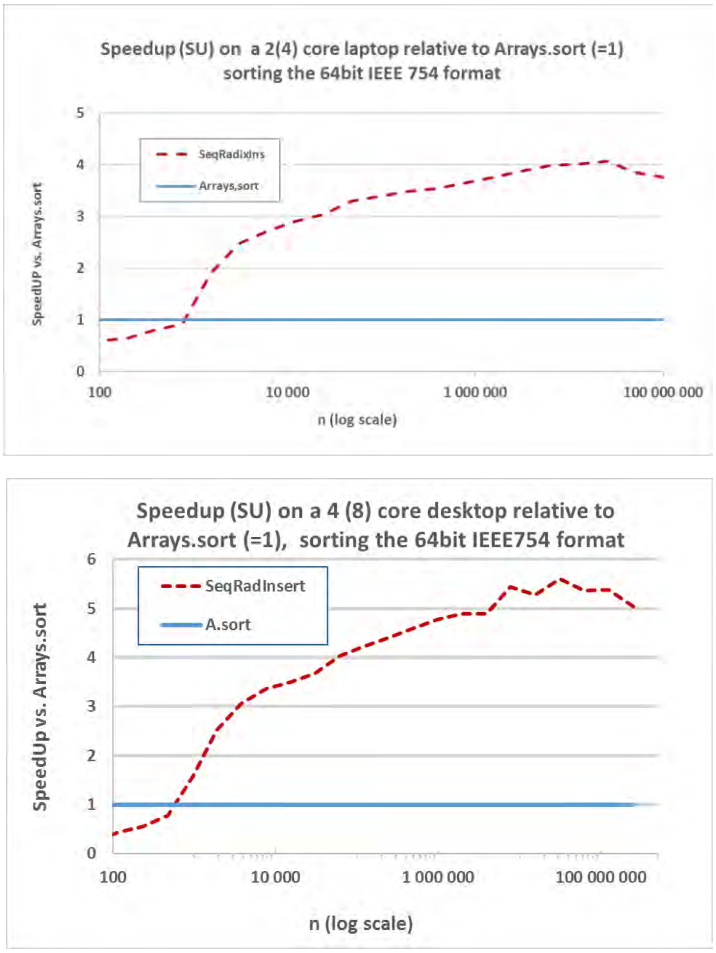


Figure 2. *The Speedup of a Desktop (n= 100.. 250 mill.) and a Laptop (n=100..100mill.) for sorting n numbers with the sequential RadixInsert algorithm compared with the sequential QuickSort based Arrays.sort() algorithm (=1) in the Java library.*

We see that sequental RadixInsert is at best 4 to 5 times faster than Arrays.sort when sorting 64bit IEEE754 floating point numbers. This speedup is basicly only sensetive to *m* , the number of bits we sort on in the Radix phase as explained in the next section.

b) The number of bits used by LSD Radix.
The most inportant issue for the speed of this algorithm is how we determine *m*, the number of bits we sort on in Radix phase . The following formula is used when sorting *n* numbers in the 64 bit IEEE754 format:

$$m = log_2 n + 13$$

The reason for the $log_2 n$ term is simple, to make the length of the average bucket equal to 1. The reason for the addition of 13 bits is motivated by the sign bit plus the 12 bit exponent part of IEEE754 for 64 bit numbers. For most distributions of numbers to be sorted, this part varies little, but must be sorted on. See fig.3 below.

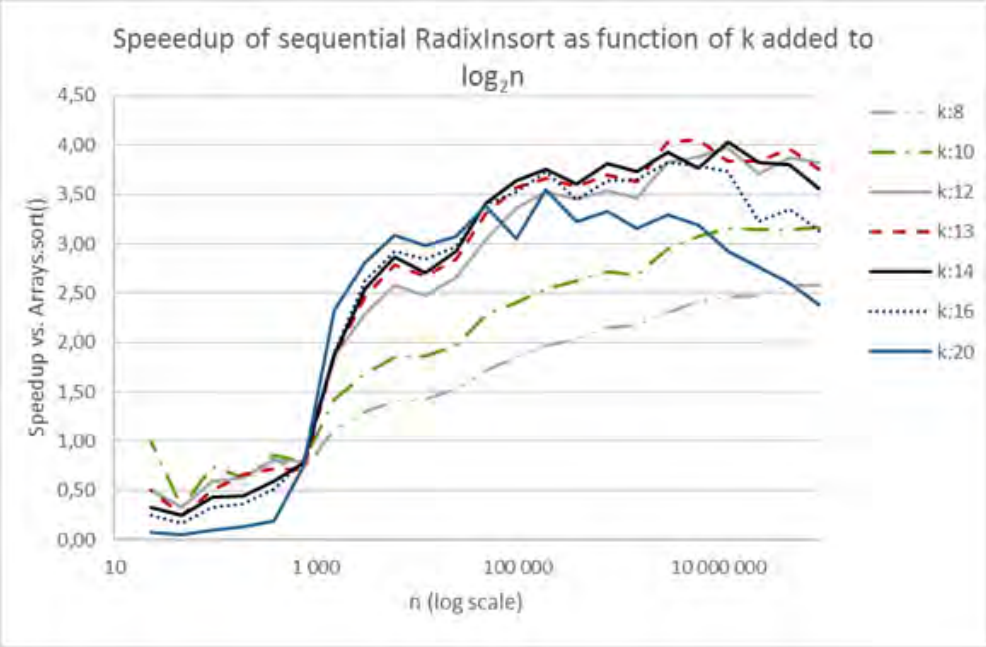
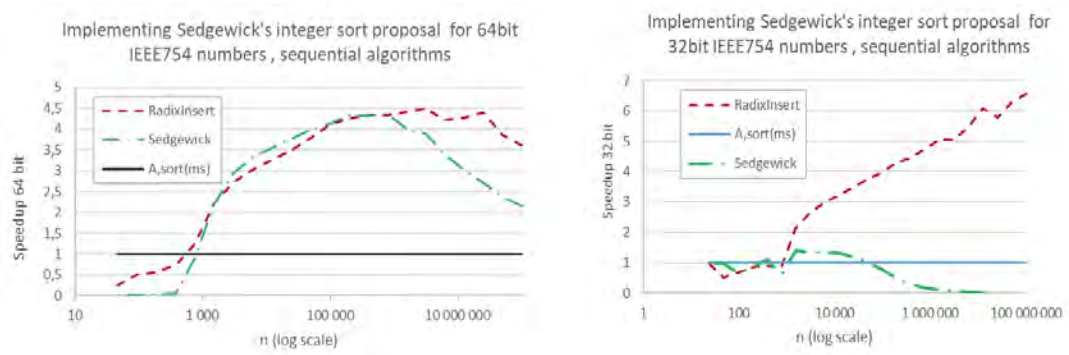


Figure 3. *The Speedup of the sequential RadixInsert algorithm as a function of the number of additional bits to log2(n) we sort on versus the sequential Java library Arrays.sort algorithm on a 4(4) core Desktop for the 64bit IEEE 754 format, n = 10.. 100mill.*

Finaly, it must be mentioned that the extra amount of memory used by RadixInsert is an array *b*[] of size *n* + some minor arrays, whereas the sequential Quicksort in the Java library needs basically no extra memory. However the merge based Arrays.parallelsort() in the same library needs the same amount of extra memory as RadixInsert does in both its sequential and parallel versions.

c) Comparison with Sedgewick's proposal.
Sedgewick has proposed[10], for integer sorting that we should sort on half the bits with Radix and then use Insertsort. If we implement this idea for IEEE754 sorting in 64 bit and 32 bit floating point rutines stating that *m*= 64/2 and *m*= 32/2, we get Fig 4a and Fig 4b below with RadixInsert compared with Sedwick's proposal: 4a for 64bit and 4b for 32bit.



Figures 4a and 4b. *We see that Sedgewick's integer proposal (fig a) of 64/2 bit radix sort is good for 64 bit IEEE754 numbers but inferior to the formula in this paper. However, it is a total disaster (fig b) with 32/2bit radix sort for 32 bit numbers.*

We conclude that Sedwicks propososal does not work well, especially for the 32bit version.

d) The Java optimizer.

Java code is optimized during runtime. The tests calling the sort algorithm in Arrays.sort() and the three RadixInsert classes, are iterated many times, for most graphs 5 times for the largest value of *n* tested and progressively more times for smaller values of *n*. This we do because Java does runtime optimization of programs as they are used. First time the byte-code from the class-file is executed, it is compiled into machine code. With more runs of the same code this machine code is optimized two or three times. The final result is a speedup of more than 100 000 for some operations like the new operation on a class or method calls, while a user written Insert sort method will be optimized with a speedup of 20-30 [13].

The figures presented are always the median of these many runs. It must also be made clear that the two methods from the Arrays class undergo the same optimizations as they are executed and called the same number of times.

e) The distribution of numbers sorted.

The performance of the three RadixInsert algorithms reported in this paper, the sequential and the two parallel versions of that, varies little with the distribution of numbers they sort. We tested seven different distributions; some of which were constructed to make RadixInsert slow, but the results vary little. The seven distributions used when initialising n elements *a*[i], were: " nextDouble()", (which generates a random 64bit double between 0 and 1), "nextDouble()*nextInt(n) (nextInt(n) generates a random integer between 0 and *n* – here converted to a double)", " nextDouble – 0,3"(for generating a mix of positive and negative numbers) ,"nextDouble()+nextInt(n)", "1.0+nextInt(n)", " (n-i) *nextInt(n)-i/10.0", and " (n-i) * r.nextDouble()".

The highest speedup is found with distribution: nextDouble()*nextInt(n) with a best speedup of 4.04, while the worst performing is: nextInt(n) – i/10.0 with speedup of 3.83. Hence most figures use the nextDouble() distribution.

The balance between the 1) Radix and 2) Insert-sort phases in sequential RadixInsert

1) When counting the values of the initial count-arrays (one for each digit), they are all declared and counted at the top method. Since most of this sorting use a 4 digit LSD radix sort, this saves 4 – 1 = 3 reads and a conversion double to long for each element, but the elements also have to be read for each digit it sorts on. Thus, a total of. (1+4)*n = 5n reads

and 4n writes is done in the Radix phase.
Insertsort consist of a double loop. In the outer loop it tests if a[i] ≤ a[i+1], and if that fails, it enters the inner loop to leftshift element a[i+1] in place. Each such shift adds one read and one write. We have then counted this number of such shifts as a function of the %-age of n, the numbers sorted. This %-age is almost constant for all n (1000 <n< 250 mill), but varied significantly with the k, the number of elements added to $\log_2 n$. This is shown in figure 4 for n=10 mill. If k =5, it results in 26n reads and 25n writes in Insert-sort, but if k=14, the figures are 1,025n reads (including 1 read for the outer loop) and 0,025n writes and hence the time used by insert phase is neglect able. We conclude that the time taken by Insert sort decreases rapidly with the number of bits sorted on by the Radix phase, but for values < 22 (last value tested) it 'never' goes to exact 0, so the Insertsort phase is always needed.

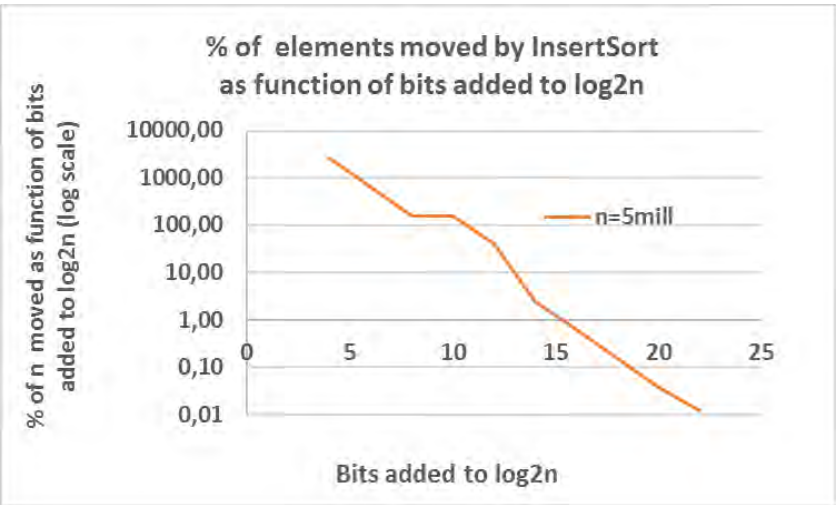


Figure 5. *The %-age of n elements moved by Insertsort as function of the number of bits added to $\log_2 n$ in RadixInsert. This figure shows the work done for 64bit format by InsertSort as a function of k, the number of bits added to log2(n). We see that when k = 5, InsertSort shift 10 times as any elements than n, the amount of elements we sort. But with k = 15, we only move 1% of these n elements – obviously a negligible amount of work.*

5. Two parallel implementations of RadixInsert

The two parallel algorithms described here parallelize both the Radix part of the RadixInsert and the Insertsort part of the algorithm.

a) Merge Para

In [14], with *k* cores, it first divides data into *k* parts and then, in a top down fashion, starts two threads at each level until it has started k threads at the last level. Each thread then sorts its part with sequential RadixInsert which include Insertsort. On backtrack each node merge two segments from both ends, small elements from the left and largest elements from the right end. This is an all parallel merge algorithm, meaning that the top level it's two-parallel, at the next level its four-parallel, ... In the paper it is demonstrated that this merging is faster than ordinary merging when $k > 2$ and $n > 1000\,000$. Ordinary merging, which only merge from the left part of its segments, has a sequential merge stage at the top.

b) Full Para RadixInsert

This is a full parallel algorithm meaning that it starts *p* threads, one for each core, and apart from two synchronizations between the threads for each digit sorted on, all threads can work at full speed until a full LSD Radix sort is done. Since it has not yet been published, a short description is also given here. Right to left Radix sorting of *a*[] on *m* bits first determine a number of digits to sort on, RadixInsert basically use 4 digits (for n <1000 it uses 2 digits), dividing the *m* bits into 4 more or less equal parts (each 6 to 10 bits long). LSD Radix, starting with the least significant (rightmost) digit, will then, for each digit move data back and forth between arrays *a*[] and *b*[] based on the values on that digit after all elements with smaller values . Doing this four times, then final sorted result ends in *a*[]. The following describes sorting on one such digit. The full sorting is just an iteration of doing this four times sorting on the next set of bits to the left.

Stages in sorting on one digit with $2^{digbits}$ different values 0: $2^{digbits}$-1, with the threads numbered: 0, … (p-1). With *p* threads we divide the array *a*[] into *p* equal parts. $thread_0$ owns the leftmost part of *a*[], $tread_1$ the next part,…Each $thread_i$ then owns $a[from_i \,..to_i]$ and does all its sorting on that part to $b[from_i \,..to_i]$.

1. Each thread has an integer array *count*[0..2^{dbits}-1] and counts all different values of the digit in its part of *a*[] .
2. In the shared data area there is declared a two-dimensional array *allCount*[0:*p*-1][]. Each thread sets its *count*[] array into *allCount*. In Java that is a single statement: *allCount*[i] = *count*;
3. All threads synchronize on the same ReentrantCyclicBarrier.
4. Each $thread_i$ creates a second array *count2*[0..$2^{digbits}$-1] and initializes its content following this rule: count2[s] = $\sum_{t=0}^{k-1}[\sum_{j=0}^{s-1} allCount2[\,t][j] + \sum_{r=0}^{i-1} allCount2\,[s][r]\_$, or verbally: *count2[s]* is initialized to the sum of all elements in *allCount [][]* with smaller value than *s* + the sum of all values in allCount [][] with the same value *s* and a smaller thread-index than *i*. An example is that if $thread_i$ finds a 3 in a[] it must be placed after all 0, 1 and 2s from all threads including itself and also after all 3's found by threads with smaller index than *i*. This last part of the rule also secures that RadixInsert is stable.
5. Now all threads can in parallel go through its part of *a*[] and sort its different values *w* to *b*[] and after each such placement in place *b*[count2[*w*]] and increase count2[*w*] by one. Then all threads will write to the correct placement and hence into different elements in *b*[].
6. All threads synchronize on the same ReentrantCyclicBarrier.

This gives a full parallel stable sort in the Radix phase.

6. Making the Insert phase full parallel

The above section describes how the radix phase can be made full parallel. As decribed earlier, we have now partitioned *a*[] into buckets where all elements in one bucket are larger than all elements in all buckets to the right and smaller than all elemets in buckets to the right, but internally no bucket is sorted – they are in the input order. On the average they are of

length 1, but that is data dependant. They might be of any length ≥ 0. Here is how we make the insertsort full parallel:

1. After the Radix phase, all threads sort their part of *a*[] with Insertsort in parallell.
2. All threads synchronize on the same ReentrantCyclicBarrier.
3. All threads but the last has to fix a possible problem with the next thread in case a bucket with two or more elements is split between this thread and the next thread (see fig. 6). Even though each part on this division line is sorted, it might not be sorted across this division.

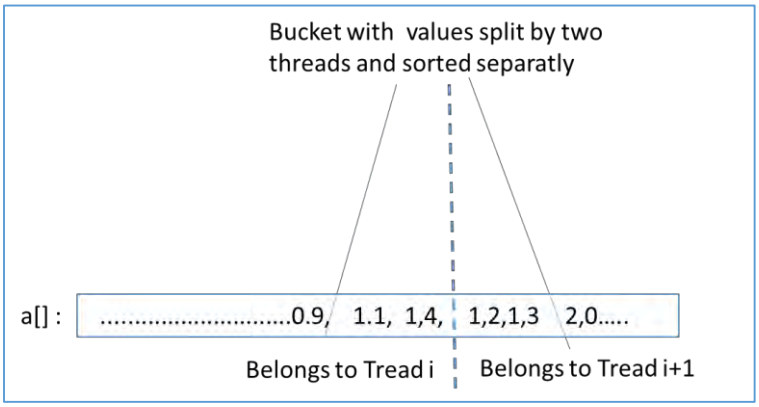


Figure 6. *The problem that might arise if a bucket with more than one element is divided by two threads. This problem is solved by starting insertion sort, beginning with the leftmost element of thread i+1 and shifting smaller elements leftwards into the area for thread$_i$ until this split bucket is fully sorted.*

This problem occurs empirically less than one in 100 million numbers sorted by RadixInsert, but has none the less to be solved. This sorting will be very short stopping with the next buckets left and right.

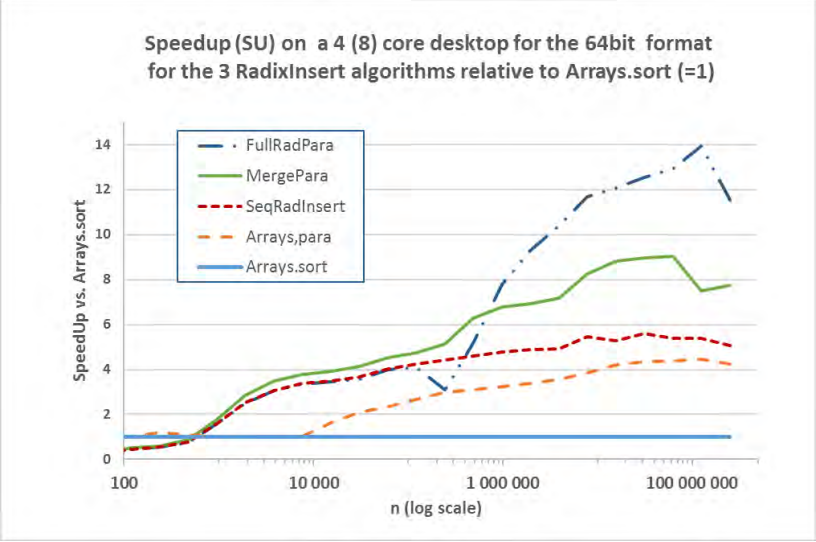


Figure 7. *The Speedup of the sequential, the two parallel RadixInsert algorithms and the Arrays.para algorithm on a 4(8) desktop, the 64bit format*

7. Sorting the 32bit IEEE754 format with RadixInsert

In figure 7 we gave measurements when sorting 64bit floating point numbers. What if we sort 32 bit IEEE754 numbers? The short answer is that the speedup is then even better. The only change except for the obvious recompiling double arrays to float, is that the formula now for selecting m, the number of bits for the LSD Radix to sort is now:

$$m = Math.\min(log_2\, n + 9,\ \ 32)$$

The reason for $\log_2 n$ is the same as for the 64bit double. The addition of 9 here is because the exponent part + sign bit is only 9bit.

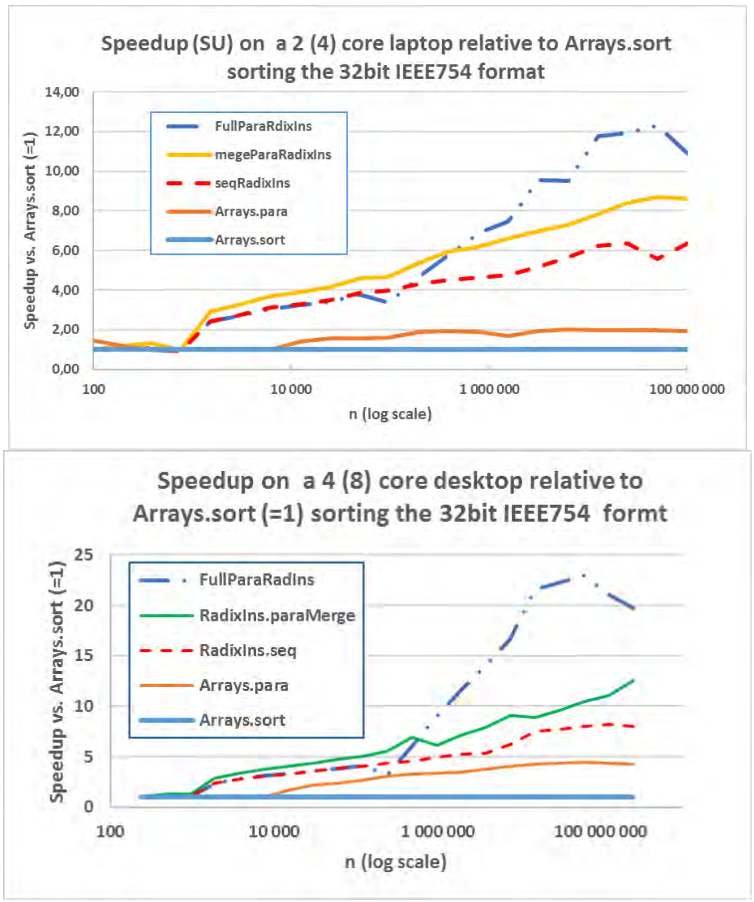


Figure 8. *The speedup of the three RadixInsert algorithms sorting 32bit IEEE 754 numbers on a 2(4) core laptop and a 4(8) core desktop with the nextFloat() distribution. In general RadixInsert sorting the 32 bit format is almost twice as fast as sorting the 64 bit format.*

We also note that the 4(8) core Desktop is approximately twice as fast as the laptop because it has twice as many cores. In addition to the full parallel RadixInsert is up to more than 20 times as fast than Arrays.sort and almost 5 times as fast as Arrays.parallelsort on the Desktop for the 32bit format.

8. Notes on the implementations

The Java program that implements sequential RadixInsert and its two parallel implementations and the tests producing the graphs are implemented as four classes:

a. TestParaRadixInsDouble – does all statistics and collecting run times for all tested algorithms including Arrays.sort() and Arrays.parallelsort(),
b. SeqRadixIns – containing all tuning parameters and the sequential version of RadixInsert. Its user interface is static method: sort (double a[])
c. RadixMergePara – containing the merge parallel version of RadixInsert. This class calls the sorting method in SeqRadixIns. Its user interface is method: *sort* (double a[])
d. RadixFullPara – containing the full parallel version of RadixInsert. This class calls the sorting method in SeqRadixIns. Its user interface is method: sort (double a[])

In a sorting libray, only class SeqRadixIns and class RadixFullPara are needed, This code will be available on the authors home page [15] before the conference. The three last classes also is available in 32bit *float* versions.

- To ensure that all RadixInsert algorithms are correct, the result from the call to Arrays.sort() is kept for each run, and all arrays sorted by the three RadixInsert method are afterwards compared element by element with the Arrays.sort() sorted array.

- For reading a double as a long, the Java library method *Double. doubleToRawLongBits(double value)* is used. If one implementst this algorithm in C, a union beween a *long long* and a *double* can be used to the same effect.

- In some algorithms it is a marked effect to overbook the number of cores/threads we tell the program to use compared with the actual cores present. The effect of this is small here, but a slight 4% increase of speedup of the full parallel radix algorithm with a doubling the number of cores reported by the operating system can be used. In effect then the 2(4) core laptop then runs with 8 threads.

- Like most sorting methods that employs threads, it only starts parallelism when n > some limit (here: numCores * 15 000). If not, the parallel method only uses the sequential algorithm because the time it takes to start *p* threads is larger than sorting such a 'short' array. If $n < 50$, it only uses insertsort.

9. Discussion

We have presented RadixInsert, a new sorting algorithm for sorting 32bit and 64bit floating point numbers following the IEEE 754 standard. Sequential RadixInsert is up to some 4 times faster for the 64bit format and more than 6 times faster for the 32bit format than the standard sequential Java sort method. The best parallel RadixInsert is also at least some 3 times faster than the standard Arrays.parallelsort. What is new in these algorithms is that we have dynamic number of bits we sort on and first and foremost that we sort floating point numbers.

Although there are some disputes between Intel and NIVIDA on arithmetic on this standard [2,3], and that the Java library have two ways of reading such a floating point number, the other with normalized NaN values (Not a Number), it is our claim that as long as all such IEEE754 numbers comes from the same computing device with the same encoding RadixInsert is a valid and much faster sorting method. Two additional facts that strengthen this claim is that we do not do any arithmetic or change any bit – we only read bits in their representation. Also, our sort is always checked as being equal to what quicksort achieves element by element by value. The IEEE 754 standard is supported by Intel, AMD and the Arm (which dominates the mobile phone market) and probably all other CPU and GPU producers. As opposed to Quicksort, RadixInsert is a stable sorting algorithm, which makes serial sorting on more than one data field possible.

The reason that 32bit RadixInsert sorting is almost twice as fast as 64bit sorting, while the quicksort based routines in the java library have little speedup 32bit versus 64 bit, we explain by the execution times reflects more the number of bytes read & written to and from memory. The 32bit RadixInsert reads less than half the number of bytes than 64bit sorting and thus fit better into the caching system. Quicksort on the other hand has far more reads and writes (with n= 1mill, it is in the order of 20) and thus stressing the cache system.

10.Conclusion

We have presented RadixInsert, a new sorting algorithm for sorting 32bit and 64bit floating point numbers following the IEEE 754 standard. Sequential RadixInsert is up to some 4 times faster for the 64bit format and more than 6 times faster for the 32bit format than the standard sequential Java sort method. The full parallel RadixInsert is also 3 to 5 times faster than the standard Arrays.parallelsort.

11. References.

[1] *Shell, D. L. (1959). "A High-Speed Sorting Procedure" . Communications of the ACM. **2** (7): 30–32.*

[2] *ANSI/IEEE 754-1985. American National Standard - IEEE Standard for Binary Floating-Point Arithmetic. American National Standards Institute, Inc., New York, 1985.*

[3] *IEEE 754-2008. IEEE 754–2008 Standard for Floating-Point Arithmetic. August 2008.*

[4] *The Art of Computer Programming*, Volume 3: *Sorting and Searching*, Third Edition. Addison-Wesley, 1997. ISBN 0-201-89685-0. Section 5.2.5: Sorting by Distribution, pp. 168–179.

[5] https://docs.nvidia.com/cuda/floating-point/index.html inspected 19.05.2019

[6] https://software.intel.com/en-us/forums/watercooler-catchall/topic/681450 inspected 18.05.2019

[7] http://stereopsis.com/radix.html, inspected 19.05.2019

[8] Knuth, Donald E. (1997). "Shell's method". The Art of Computer Programming. Volume 3: Sorting and Searching (2nd ed.). Reading, Massachusetts: Addison-Wesley. pp. 83–95. ISBN 978-0-201-89685-5.

[9] V. J. Duvanenko, "Faster LSD Radix Sort", https://duvanenko.tech.blog/2019/02/27/lsd-radix-sort-performance-improvements/ February 2019 inspected 19.05.2019

[10] R. Sedgewick, "Algorithms in C++", third edition, 1998, p. 424-427

[11]M. J. Atallah (ed.): «Algorithms and theory of Computation Handbook», ch. 3, ISBN 0-8493-2649-4, CRC Press 1999.

[12] K.Hegna, A.Maus: «Javaprogrammering – kort og godt», s. 236, Universitetsforlaget 2017.

[13] https://en.wikipedia.org/wiki/IEEE_754 (inspected 03.spt. 2019)

[14] A. Maus: "A faster, all parallel Merge sort algorithm for multicore processors" NIK'2018, Norwegian Informatics Conf. Svalbard, 2018. Tapir, www.nik. no

[15] Arne Maus homepage: http://arnem.at.ifi.uio.no/sorting/

Survey of interactions in popular VR experiences

Kjetil Raaen and Hanne Sørum
Kristiania University College

Abstract

As the first step in a project to examine the quality of interactions in the relatively young field of Virtual Reality (VR), this study showcases the creative variation among modes of interaction. The present paper reports on a multiple case study reviewing VR applications focusing primarily on interactions. By surveying a set of popular applications, we explore the variety as well as the developing conventions within user interactions in VR. Because this research is work-in-progress we provide some preliminary insight that we can build on and discuss in upcoming studies. Our results show a wide array of different ways of interacting with such applications. Generally, these can be categorised in one of a few groups; menus, locomotion and interaction with the virtual environment. We also argue that theory from the field of Human Computer Interaction (HCI) can be applied to VR in regards to design of user interfaces.

1 Introduction

Lately, we have seen a steadily increasing interest in immersive, binocular displays [21]. When these exclude all external visual stimuli, they are referred to as Virtual Reality (hereafter VR). This is a medium with tremendous potential. The ability to be transported to other places, to be fully immersed in experiences, and to feel like you're really there - present - opens up previously un-imagined ways to interact and communicate [16]. Another definition of VR is "various technologies that, through digitally created sense impressions, give the user a sense of being somewhere else." [22]. For example, in a zoo or on a boat. There are also several application areas where one can use VR, and among other things we see that it is used in brain surgery, architecture visualization, exercises related to fire and training of employees [22]. With this as a starting point, it is crucial to create good interactions and user-friendly interfaces. In this study, we define *interaction* as any opportunity for the user to control or influence the VR environment. Because VR applications exist in many different subject areas, they have a varied target group. Good design and good functionality are important elements to make the applications work appropriately for the users. It should be easy for users to understand what to do and when.

Current implementations of VR technology track users' movements in six degrees of freedom. This means that the user can rotate their head along any axis, as well as move around within an area while the headset tracks the movements and updates displayed images appropriately. This is called room-scale VR. Previous paradigms include sitting and standing VR, where the user has limited or no ability to move around. Along with realistic sound and motion tracking hand-held controllers, this technology allows a

This paper was presented at the NIK-2019 conference; see http://www.nik.no/.

much higher level of immersion in visual media than any previous technology. Potential applications range from simple fun to treating serious psychiatric illnesses. While the precise tracking of head movements allows such interaction to feel extremely natural in VR, many other interactions necessitate more abstract approaches. One such case is moving longer distances. The natural movements are limited by the area available for the VR setup, at most a few meters in each direction. To allow movement beyond this in the virtual world, developers have to introduce alternatives.

Another tricky problem is direct interaction with the world. VR systems still give very limited tactile feedback, therefore, opening doors and operating machinery needs some abstraction. Further, some interactions needed to control the application itself, such as menus and text input might sometimes be necessary. In this work, we qualitatively survey a selection of popular VR applications looking at interaction design choices.

With this work, we aim to provide insight into common interaction modes in current products. For now we will not attempt to judge the quality or effects of such design choices, leaving this for future work. Our plan is to utilize this information when deciding which applications to use for future user studies into interaction in VR.

2 Background

Research into Human Computer Interaction (hereafter HCI) in VR has focused on constructive, or design, research, that is research focusing on new designs proposed by the researcher. "Virtual reality has captured the world's imagination. Over the last few years, developers and enthusiasts in the thousands have devoted countless hours to coding, designing, and speculating about the possibilities of this exciting new medium" [16]. One literature review [3] found that full 55% of recent publications were in this category. Thus, there is a lack of research into how current commercial applications work. Among these the authors found that "A significant observation is that a number of publications presented constructive research without any form of evaluation or contribution to the understanding of relevant phenomena" [3]. Such an approach may create new designs; however, it has limited applicability outside the narrow problem addressed. Boletsis et al. further argue that we need more empirical work as well as conceptual research to explain phenomena and find connections. Such work has higher problem-solving capability. In use of technologies (such as VR), the users typically interact with an interface and external devices, like for instance a hand-held controller and headphones.

These patterns of interactions may map to the four types of activities that users are likely to be engaged in when interacting with any system [17]: (1) *Instructing* where users type in commands and give instructions to a system or interacts through a menu system, for instance by typing in commands and using menu systems for choosing actions to take in a given environment. (2) *Conversing* where users have a dialogue with the system in use and talk via an interface, then the system replies. The system can reply by using speech or text. The voice command system Siri is one example. (3) *Manipulating* where users interact with objects and manipulate by for instance opening, holding and selecting digital objects in a virtual space. One example can be that the user can select and move a picture from one place on the PC to another. (4) *Exploring* where users move around in virtual worlds and explore the environments. An example can be that the user explore the environments, such as the interior and functions of a car and get a realistic feeling of how it works. It seems clear that these types are relevant when analysing VR applications.

Developing and designing interactive systems and interfaces, involve various considerations, among them design issues, technological choices, identifying the users, and

lastly activities performed by the users in the context of use [2]. High quality interactions are a vital contributor to satisfaction among the users and great experiences. Although satisfaction can be assessed in many and different ways, usability plays an important role in interaction design and HCI. Usability can be defined as following "The effectiveness, efficiency and satisfaction with which specified users achieve specified goals in particular environments" [9]. In order to measure usability, methods such as heuristic evaluation, eye tracking, user testing and observation can be applied. All these methods may also be applicable when designing and testing VR applications. There also seems to be some benefit of applying Don Norman's design principles [14] for usability and user experiences. These six principles are related to:

- *Consistency*: This principle recommends designing user interfaces to have similar operations and use similar elements for achieving the same tasks. For example, consistency in design is that you always right-clicking to mark an image.
- *Visibility*: Being able to see the state of the system and which actions that are possible to perform. The more visible functions are, the greater there is for the users to know what to do.
- *Affordance*: Affordance means to give the user a hint for how to use a given solution or product, and provide clues for how to operate things. For example, the buttons on the elevator give the user a hint about how the doors are opened and closed.
- *Mapping*: Users should understand the connection between a function and the effect it has. For example, the stove's positioning of buttons and wheels in relation to how to put on the plate and adjust the temperature.
- *Feedback*: Feedback is about the system sending a response to the user regarding actions that have been performed and what has been accomplished.
- *Constraints*: Limiting the possibilities a user has for interactions to prevent the user from performing tasks that lead to errors. It shall be visible to the user what is possible and not possible, in a given user interface.

Jerald [10] provides some design guidelines that can help designers and developers in creation of VR experiences. This groups interactions in *patterns.*

- *Selection* patterns allows the user to choose a specific object in the environment for future actions.
- *Manipulation* patterns are ways for the user to modify or influence objects.
- *Viewpoint control* patterns allow the user to manipulate their point of view in the world. The purpose is usually to move around.
- *Indirect control* patterns uses some intermediary to control the environment.
- *Compound* patterns combine the previous patterns in various ways.

According to [6] some rules should be considered regarding movement in VR: moving forward is better than moving backwards, up/down is better than strafe left/right and, finally, better than a gentle camera rotation is a fast camera cut. Moreover, if the movements is wrong the users can get confused and they might get sick. These factors is of particular

importance in VR compared to web- and mobile apps. Other useful principles in designing user interfaces and interactions are Jakob Nielsen's 10 usability heuristics [13] and Ben Shneiderman's 8 yellow rules of usability design [20]. Most likely, in many cases, guidelines typically used in design of physical products, desktop and mobile applications, are applicable in VR environments.

Empirical work looking into interaction in VR include Santos [18] which argues that designers of VR experiences are in the need of guidelines to ensure efficiency and usability. They conducted a study of menus in VR and focused on efficiency and satisfaction among users, by shedding light on geometry and position (four types of menu configuration). They did however not find a clear preference for any type of menu. Regarding efficiency measured by selection time, the results show that in both non-diegetic and spatial menus there were statistically significant differences among linear and radial menus, with radial menus giving shorter selection time. The time for completing tasks is smaller on radial menus compared to linear menus. Concerning usability and error rate, there were no statistically significant differences between such menus. Consequently, the user experience is not a large extent influenced by this.

Based on the literature review conducted for the present paper, interactions in VR seems to be an under-investigated area with a great potential. Although usability in interfaces is a well studied subject, it is not clear to what extent these results are applicable to VR. For VR itself, only a few very specific types of interaction are evaluated.

3 Method

For inclusion in this study we wanted a selection of applications that are commonly in use by a broad audience. Based on this criterion, we used the store (Viveport) associated with the headset, the HTC Vive, and filtered for free applications for room-scale VR. This resulted in a selection of 304 applications. We reasoned that free applications will be among the first tested by new users. Further, these applications are necessarily not business critical to the developers, thus allowing them more freedom to experiment. From this list we selected and reviewed the 10 most downloaded applications. This resulted in a wide variety of different genres, ranging from entertainment and lifestyle, to shooter and 360° videos.

For each application one of the authors tested the application, while the other took notes and asked questions. The purpose was to gain as much information as possible during this phase. Thus, we were able to record data simultaneously with testing the VR application. When examining an application, we explored the application until we were confident we had seen most of the content, as well as the full variety of interactions. We observed and noted how a user could interact with the application. This evaluation involved only observation, and we did not do any subjective assessment of the qualities of what we found. Some applications were clearly limited in scope and brought us to a definite *ending* within a few minutes, while others were complex systems where we spent upwards of an hour before we felt confident we had seen the relevant parts.

For analysing the data, we employed an inductive approach to identify themes in the data [15], focusing on the HCI aspects of the experience. When we were satisfied we had identified the most relevant themes, identified which application represented which theme. This allowed us to see how prevalent each theme was, as well as getting a more clear idea about what the dataset as a whole implied about each theme. Most, 4, experiences were games, while others where educational experiences.

There are some limitations to this study that it is worth mentioning. The selection

Application (alias)	Release date	Genre	No. of players	Rating (no. reviews)
Vive Video [8] (A)	Jan 25, 2017	360° video, causal, entertainment, education, lifestyle, creativity	Single-player	3.6 (216)
Ready Player One: OASIS beta [5] (B)	Mar 23, 2018	Action, sci-fi, travel, travel & exploration, shooter, adventure	Multi-player	4.0 (276)
The Last Day Defense [1] (C)	Nov 15, 2018	Strategy	Single-player	4.8 (32)
Sniper Rust VR – Trial Version [25] (D)	Dec 11, 2018	Action, adventure, causal, shooter	Single-player	3.2 (10)
theBlu: Whale Encounter [24] (E)	Mar 23, 2017	Adventure, causal, entertainment, travel & exploration	Single-player	4.4 (126)
VRChat [23] (F)	Sep 14, 2017	Adventure, live event, creativity, education, entertainment, social	Multi-player	4.1 (52)
Buzz Aldrin: Cycling Pathways to Mars [11](G)	Mar 17, 2017	News, entertainment, education, social, travel & exploration	Single-player	4.4 (60)
Star Wars: Trials on Tatooine [12] (H)	May 3, 2017	Sci-fi, action, causal, entertainment	Single-player	3.8 (74)
Sharecare VR [19] (I)	Aug 3, 2017	Education, health & fitness	Single-player	4.7 (32)
Undersea Park [4] (J)	Feb 22, 2019	Travel & exploration, education, documentary, art & design, entertainment	Single-player	5 (2)

Table 1: Overview of the applications included in the study.

consists of the ten most popular VR applications downloaded from Viveport. Since the sample also consisted of only ten applications, it may be advantageous to include more in future studies. It should also be mentioned that all the applications subject to evaluation were free. As we have mentioned earlier, this study does not investigate the quality and usability of the various applications, but primarily focuses on interactions and actions needed from the user to control the application.

4 The applications

Each of the applications, presented in Table 1, is given an alias (A-J), the letter in parentheses, by which we refer to in the rest of the paper. The data reported here is taken from www.viveport.com[1]. The table shows an overview of the diversity of applications that we have reviewed, both in terms of release date, latest update and types of applications. The categorization is done by Viveport and shows that most of the applications belong to several categories/genre. 8 out of 10 are single-player applications, while 2 out of 10 are multi-player. The numbers in the right column refer to the ratings given by users (max score is 5.0, minimum 1.0), followed by the number of users that have provided a review.

Although the applications are on the Top 10 list provided by Viveport, we see a significant variation in the number of users giving feedback and score given by the users. The scores vary from 3.2 (as the lowest) to 4.8 (highest with more than 10 reviews). The number of ratings varies from 2-276. There may also be a relation between number of reviews and for how long the application has been available for download in Viveport. As the content and purpose of these applications are different, we can also expect that the user groups are shared (with some overlap), depending on the context and scope of use. Because we are more interested in studying interactions than actual content of the application, we consider this variety of applications a strength. The review of all the applications also showed a huge variation in scope.

5 Interaction modes

After reviewing the applications, we summarised the identified themes. We soon found that each theme we had identified represented one way in which the user was able to interact with the application. To describe these different themes we use the phrase *interaction modes*. These modes are comparable to Jerald's [10] *interaction patterns*, however the scope of each mode is somewhat broader, and includes all the individual elements used to achieve an objective. Thus, *selection* and *manipulations* can be combined to one mode. We also identified three main categories among these interaction modes. Menus, locomotion and interaction with the virtual environment. Each has various implementations and purposes. In the following subsections we will describe each of the interaction modes grouped by category, listing applications using the described concept in parenthesis. Figure 1 shows the various interaction modes categorised by purpose with a list of applications which use them.

Menus

Menus in VR are in concept very similar to how menus are used in traditional user interfaces. A list of options is presented to the user as text or icons, and the user selects one or more items by "clicking" on them. In VR, there is some variation in how this is implemented. This is a practical example of what Jerald [10] terms *indirect control patterns*. Menus may be anchored in the world as shown in Figure 2a, and the user needs to navigate in front of the menu to select an item (A, B, C, F, H). Alternatively, the menu can be attached to one controller (B, I), giving an illusion of a menu connected to the user's wrist as shown i figure 2b. This necessitates another controller to select an item. How the user selects an item is not consistent. In some experiences it is enough to touch a controller to the button, while others require using a button on the controller. The last option can be combined with a beam from the controller to allow selection at a distance. Lastly,

[1] accessed 14. March 2019.

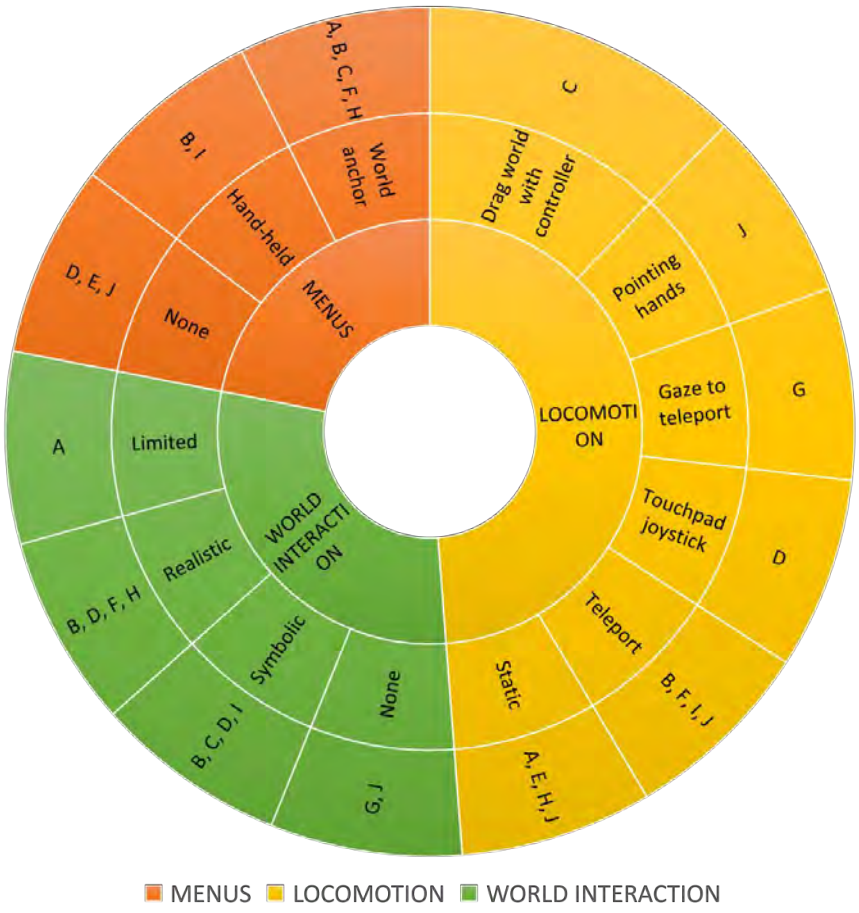


Figure 1: All interaction modes categorised by purpose showing which application use each mode. For example, locomotion is divided into three different modes: *limited*, *realistic* and *symbolic*, in addition to *none*.

some applications avoid menus completely, opting instead for symbolic interactions in the world. Menus are an old idea, used from the first graphical user interfaces. It seems clear that the concept stays with us in VR for now.

Locomotion

Another important aspect of interaction in VR is locomotion. Jared [10] terms this *viewport control patterns*. Because we limited ourselves to applications supporting room scale VR, all applications allow limited movement by walking around within the limitations of the VR setup. Some applications do not allow users to move outside the bounds of their VR setup (A, E). Other applications use cinematic techniques such as fading out and fading in at another location (H).

A common solution is letting the user indicate a new position in the world, and then immediately move there. This approach is usually called *teleportation* and illustrated in Figure 3. Indicating a destination can be achieved with pointing a controller (B, F, I) or simply holding a steady gaze at a specific point in space (limited use in G). This approach emerges as the preferred mode of locomotion in VR, despite Jared mostly mentioning it as an afterthought.

Another approach to locomotion is using input directly on the controllers, which Jared calls *steering pattern*. The Vive controller has a touch sensitive area for the thumb.

(a) A menu anchored in the world (Application C).

(b) A menu connected to the users left hand (Application I).

Figure 2: The two main ways of displaying menus in VR.

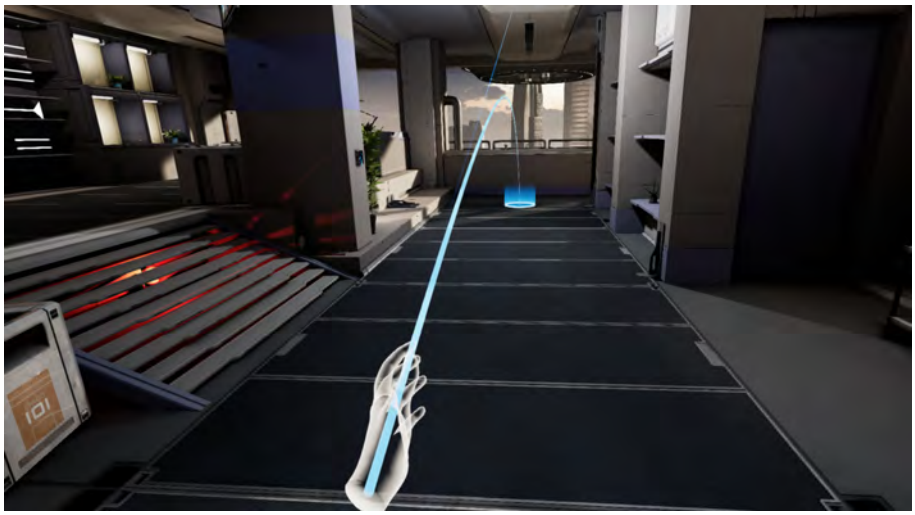


Figure 3: When the user pushes and holds a button on the controller, the application shows an arc and a circle on the ground. The circle indicates where the user will end up. To move, you release the button (Application B) [5].

One application (D) use these to control locomotion. One thumb controls direction of movement, while the other controls turning. Another application (C) use the movement of the controller to move around in the world. Pressing a button on the controller locks the world to the controller. Thus, moving the controller moves the world around the player. This can be repeated for longer distance travel.

Finally, one application (J) allowed the user to select one of three modes of which two are previously introduced: Teleportation and static. The third option used here is movement by pointing the controller in the desired movement direction. Extending the arm would increase the speed, while moving the controller towards the chest would slow down and stop movement.

World Interaction

Many of the applications allow some form of interaction directly with the virtual world. By moving the controller to an object and clicking a button, the user may pick up or activate an object. This concept include both *selection* and *manipulation* patterns according to Jared [10]. The result of such an interaction can be classified in one of two distinct groups. Realistic interactions cause the object to react close to how you would expect in the real world (A, F, H), as shown in Figure 4. The alternative to realistic interactions is what we

Figure 4: The user has picked up a pen, and may draw in the air (Application F).

Figure 5: A heads-up display showing critical information about the character's status in game.

term *symbolic interactions*. In such interactions (C, I), when the user interacts with an item it is interpreted as an abstract input to the application triggering some action not expected in the real world. You may touch markers on a board to trigger construction of buildings (C). Some applications (B, D) combine these two modes of world interaction. Interaction with the virtual world is core experiencing virtual worlds. Except for the non-interactive experiences, all implement some form of this. Symbolic interactions may also replace menus in many situations.

Other considerations

In addition, one application (B) used a Head's Up Display (HUD), that is, information overlaid on the world moving with the user's head (as shown in Figure 5). This allows critical information to be visible continuously to the user while minimising the disruption of immersion. This could be categorised as another interaction mode, however as implemented it only allows one way communication. That is, the application shows information to the user while the user interacts in other ways. In this case, using realistic world interactions.

The scope of the applications varied significantly. Some are short experiences with a clearly limited amount of content, while others are open-ended. One application (F) even allow users to create content. This is is however mostly implemented outside VR. Some

applications were more difficult to understand and explore than others. Most applications clearly showed up front what was possible and others had good tutorials. Conversely, some were confusing and difficult to navigate. In sum, the 10 applications show two fundamentally different types of menus, three different implementation of locomotion and 2 ways of interaction with the virtual world.

6 Discussion

This paper provide an overview of existing interaction modes, while not attempting to judge the quality or effects of such design choices. Facing the fact that there is limited research on interactions in VR, the purpose of this work is to identify the most important modes of interaction in VR. Because this research is work-in-progress, we aim to further develop and test this in future studies. One approach will be to apply a HCI perspective and include evaluation of usability [9] and satisfaction among users. Additionally, we may evaluate application of design principles [17] and usability heuristics [13]. However, grounded in the preliminary findings we clearly see that interactions in VR can be divided into different modes and dimensions, represented by menus, locomotion and world interaction. Each of these three modes has a variety of implementation.

Moving beyond the limitation of the VR setup has proven tricky to implement well. Current approaches suffer a trade-off between two contradictory goals: Presence and minimized discomfort [7]. Removing this option completely keeps presence high and discomfort low but limits the experience to small areas. Teleportation has been shown to interrupt presence but induces minimal discomfort. Indirect control of movement, such as by pointing hands or using buttons create some discomfort, but does not reduce presence. It is clear that developers are currently experimenting with a wide array of approaches to solving it.

Additionally, it is also clear from our findings that interaction modes in VR may benefit from being evaluated according to Don Norman's design principles [14]. All six principles are as meaningful and relevant in VR as they are in traditional interfaces. For instance, locomotion interactions usually suffer from limited *visibility*, forcing the user to click all buttons to figure out how it works. Conversely, VR interactions often have clear *mapping* between what the user does and its effects.

Among the interaction types [17] we see that in VR *instructing* is usually implemented with menus or symbolic world interactions, while realistic world interactions map well to *manipulating*. Lastly *exploring* is implemented in VR using one of the many options for locomotion. Compared to Jared [10] we see a very limited selection of interaction patterns. While we have tested relatively few applications, our findings still indicate a winnowing of the field as designers tend towards agreement about what works.

We plan to study some of these interaction modes further in the future. Such studies may then refer to this framework for classification of interaction in VR applications. The value of such knowledge is twofold, (1) this seems to be an under-investigated area with a great potential as VR receive interest for various activities and business purposes. (2) To study the design of VR applications including interactions, there is a need for frameworks and guidelines describing modes of interactions. We argue that there is a lack on how to approach evaluation of interaction in VR applications. Consequently, we aim to introduce HCI literature [17] in to this field and subject of interest.

7 Conclusion

In this work we have identified three possible criteria to categorise VR applications. Although one may argue that VR to some extent differ from traditional digital interfaces (e.g. web, mobile), we still find the relevance of classical HCI theories and guidelines, design principles and methods for designing interactions. Among the goals for every application is producing effective user interfaces, great interactions and satisfaction among the users. By shedding light on interactions in VR we will hopefully contribute to a better understanding and a further discussion on how to understand interaction modes. In the next round, we will evaluate usability of some of these interaction modes in an empirical study using participants.

References

[1] ARVI lab: *The Last Day Defense*. Game [PC/Vive] (November 2018), last played March 2019.

[2] Benyon, D.: Designing Interactive Systems: A Comprehensive Guide to HCI, UX and Interaction Design. Pearson (2013), https://books.google.no/books?id=u6cpnwEACAAJ

[3] Boletsis, C., Cedergren, J.E., Kongsvik, S.: HCI research in virtual reality: A discussion of problem-solving. In: Proceedings of the International Conference on Interfaces and Human Computer Interaction 2017 (2017)

[4] Bruno Leiseur: *Undersea Park*. Game [PC/Vive] (February 2019), last played February 2019.

[5] Directive Games: *Ready Player One: OASIS beta*. Game [PC/Vive] (March 2018), last played March 2019.

[6] Fictum, C.: VR UX: Learn VR UX, Storytelling & Design. CreateSpace Independent Publishing Platform (2016), https://www.amazon.com/VR-UX-Learn-Storytelling-Design/dp/1533273022

[7] Griffin, N.N., Liu, J., Folmer, E.: Evaluation of Handsbusy vs Handsfree Virtual Locomotion. In: Proceedings of the 2018 Annual Symposium on Computer-Human Interaction in Play. pp. 211–219. CHI PLAY '18, ACM, New York, NY, USA (2018). https://doi.org/10.1145/3242671.3242707, http://doi.acm.org/10.1145/3242671.3242707

[8] HTC: *Vive Video*. App [PC/Vive/Oculus] (March 2017), played March 2019.

[9] ISO: ISO 9241-11:1998 Ergonomic requirements for office work with visual display terminals (VDTs) – Part 11: Guidance on usability. Tech. rep., International Organization for Standardization (1998), http://www.userfocus.co.uk/resources/iso9241/part11.html

[10] Jerald, J.: The VR Book: Human-Centered Design for Virtual Reality. Association for Computing Machinery and Morgan & Claypool, New York, NY, USA (2016)

[11] LIFE VR: *Buzz Aldrin:Cycling Pathways to Mars*. Game [PC/Vive] (March 2017), last played March 2019.

[12] Lucasfilm: *Star Wars: Trials on Tatooine*. Game [PC/Vive] (May 2017), last played March 2019.

[13] Nielsen, J.: Usability inspection methods. chap. Heuristic Evaluation, pp. 25–62. John Wiley & Sons, Inc., New York, NY, USA (1994), http://dl.acm.org/citation.cfm?id=189200.189209

[14] Norman, D.A.: The Design of Everyday Things. Basic Books, Inc., New York, NY, USA (2002)

[15] Oates, B.J.: Researching Information Systems and Computing. SAGE Publications (2005), https://books.google.no/books?id=5TxdBAAAQBAJ

[16] Parisi, T.: Learning Virtual Reality: Developing Immersive Experiences and Applications for Desktop, Web, and Mobile. O'Reilly (2015), https://books.google.no/books?id=T5XtrQEACAAJ

[17] Preece, J., Rogers, Y., Sharp, H.: Interaction Design: Beyond Human-Computer Interaction. Wiley, Hoboken, NJ, 4 edn. (2015)

[18] Santos, A., Zarraonandia, T., Díaz, P., Aedo, I.: A comparative study of menus in virtual reality environments. In: Proceedings of the 2017 ACM International Conference on Interactive Surfaces and Spaces. pp. 294–299. ISS '17, ACM, New York, NY, USA (2017). https://doi.org/10.1145/3132272.3132277, http://doi.acm.org/10.1145/3132272.3132277

[19] Sharecare: *Sharecare VR*. Game [PC/Vive] (August 2017), last played May 2018.

[20] Shneiderman, B., Plaisant, C.: Designing the User Interface: Strategies for Effective Human-computer Interaction. Addison-Wesley (2010), https://books.google.de/books?id=pddxRQAACAAJ

[21] Statista: Global augmented/virtual reality market size 2016-2022 (2019), https://www.statista.com/statistics/591181/global-augmented-virtual-reality-market-size/

[22] Urke, E.H.: VR og AR - en norsk introduksjon til virtual og augmented reality. Cappelen Damm Akademisk (2018)

[23] VRChat: *VRChat*. Game [PC/Vive] (September 2017), last played March 2019.

[24] Wevr: *theBlu:Whale Encounter*. Game [PC/Vive] (March 2017), last played March 2019.

[25] Zatun Game Studio: *Sniper Rust VR - Trial Version*. Game [PC/Vive] (December 2018), last played March 2019.

