university of groningen / faculty of mathematics and natural sciences / computing science

SC@RUG 2015 proceedings

# 12th SC@RUG 2014-2015

Rein Smedinga, Michael Biehl and Femke Kramer (editors)

www.rug.nl/research/jbi

# SC@RUG 2015 proceedings

Rein Smedinga
Michael Biehl
Femke Kramer
editors

2015
Groningen

ISBN (e-pub pdf): 978-90-367-7941-8
ISBN (book): 978-90-367-7942-5
Publisher: Bibliotheek der R.U.
Title: 12th SC@RUG proceedings 2014-2015
Computing Science, University of Groningen
NUR-code: 980

## About SC@RUG 2015

### Introduction

SC@RUG (or student colloquium in full) is a course that master students in computing science follow in the first year of their master study at the University of Groningen.

SC@RUG was organized as a conference for the twelfth time in the academic year 2014-2015. Students wrote a paper, participated in the review process, gave a presentation and chaired a session during the conference.

The organizers Rein Smedinga, Michael Biehl and Femke Kramer would like to thank all colleagues who cooperated in this SC@RUG by collecting sets of papers to be used by the students and by being an expert reviewer during the review process. They also would like to thank Agnes Engbersen for her very inspiring workshops on presentation techniques and speech skills and Michael Wilkinson for giving lectures on how to write scientific papers.

### Organizational matters

SC@RUG 2015 was organized as follows. Students were expected to work in teams of two. The student teams could choose between different sets of papers, that were made available through the digital learning environment of the university, *Nestor*. Each set of papers consisted of about three papers about the same subject (within Computing Science). Some sets of papers contained conflicting opinions. Students were instructed to write a survey paper about this subject including the different approaches in the given papers. The paper should compare the theory in each of the papers in the set and include their own conclusions about the subject. Of course, own research was encouraged.
Two teams proposed their own subject.

After submission of the papers, each student was assigned one paper to review using a standard review form. The staff member who had provided the set of papers was also asked to fill in such a form. Thus, each paper was reviewed three times (twice by peer reviewers and once by the expert reviewer). Each review form was made available to the authors of the paper through *Nestor*.

All papers could be rewritten and resubmitted, independent of the conclusions from the review. After resubmission each reviewer was asked to re-review the same paper and to conclude whether the paper had improved. Rereviewers could accept or reject a paper. All accepted papers can be found in these proceedings.

In her lectures about communication in science, Femke Kramer explained how researchers communicate their findings during conferences by delivering a compelling storyline supported with cleverly designed images. Lectures on how to write a scientific paper were given by Michael Wilkinson and a workshop on reviewing was offered by Femke Kramer.

Agnes Engbersen gave workshops on presentation techniques and speech skills that were very well appreciated by the participants. She used the 2 minute madness presentation as a starting point for improvements.

Rein Smedinga was the overall coordinator, took care of the administration and served as the main manager of *Nestor*.

Students were asked to give a short presentation halfway through the period. The aim of this so-called two-minute madness was to advertise the full presentation and at the same time offer the speakers the opportunity to practice speaking in front of an audience.

The conference itself was organized by the students themselves. In fact half of the group was asked to fully organize the day (i.e., prepare the time tables, invite people, look for sponsoring and a keynote speaker, etc.). The other half acted as a chair and discussion leader during one of the presentations. The audience graded both the presentation and the chairing and leading the discussion.

The gradings of the draft and final paper were weighted marks of the review of the corresponding staff member (50%) and the two students reviews (each 25%).

Students were graded on the writing process, the review process and on the presentation. Writing and rewriting counted for 35% (here we used the grades given by the reviewers), the review process itself for 15% and the presentation for 50% (including 10% for being a chair or discussion leader during the conference and another 10% for the 2 minute madness presentation). For the grading of the presentations we used the assessments from the audience and calculated the average of these.

In this edition of SC@RUG students were videotaped during their 2 minute madness presentation and during the conference itself using the new video recording facilities of the University and with thanks to the CIT crew (special thanks to Adri Mathlener for providing and operating a mobile recording kit during the conference). The recordings were published on *Nestor* for self reflection.

On 8 April 2015, the actual conference took place. Each paper was presented by both authors. We had a total of 12 student presentations this day.

**Sponsoring**

The student organizers invited two keynote speakers and both the corresponding companies sponsored the event as well by providing lunch and drinks afterwards and payed for the additional costs like programme leaflets and such. We are very grateful to

- Procam Talent Wings
- Capgemini Consulting Nederland
- Quintor Groningen

for sponsoring this event.

**Thanks**

We could not have achieved the ambitious goasl of this course without the invaluable help of the following expert reviewers:

- Fatimah Alsaif
- Michael Biehl
- Frank Blaauw
- Ilche Georgievski
- Andrea Pagini
- Alex Telea
- Michael Wilkinson

and all other staff members who provided topics and provided sets of papers.
Also, the organizers would like to thank the *Graduate school of Science* for making it possible to publish these proceedings and sponsoring the awards for best presentations and best paper for this conference.
Rein Smedinga
Michael Biehl
Femke Kramer

Since the tenth SC@RUG in 2013 we added a new element: the awards for best presentation, best paper and best 2 minute madness. Therefore, from that edition on, we will have a Hall of Fame:

**Best presentation awards**

**2015**
Diederik Greveling and Michael LeKander:
*Comparing adaptive gradient descent learning rate methods*
Johannes Kruiger and Maarten Terpstra:
*Hooking up forces to produce aesthetically pleasing graph layouts*
**2014**
Diederik Lemkes and Laurence de Jong:
*Pyschopathology network analysis*
**2013**
Jelle Nauta and Sander Feringa,
*Image inpainting*

**Best 2 minute madness presentation awards**

**2015**
Diederik Greveling and Michael LeKander:
*Comparing adaptive gradient descent learning rate methods*
**2014**
Arjen Zijlstra and Marc Holterman:
*Tracking communities in dynamic social networks*
**2013**
Robert Witte and Christiaan Arnoldus:
*Heterogeneous CPU-GPU task scheduling*

**Best paper awards**

**2015**
Jasper de Boer and Mathieu Kalksma:
*Choosing between optical flow algorithms for UAV position change measurement*
**2014**
Lukas de Boer and Jan Veldthuis:
*A review of seamless image cloning techniques*
**2013**
Harm de Vries and Herbert Kruitbosch:
*Verification of SAX assumption: time series values are distributed normally*

## Contents

# A Comparison of Cloud Computing Services in Smart Learning Systems

Guntur Dharma Putra

**Abstract**—Smart learning system is a form of e-learning service that is enhanced to be smarter and more efficient using context-aware technologies, which are based on the users behavior. Cloud computing services offer some advantages in its implementation to smart learning systems, for example, increased cost savings and also improved efficiency and convenience of educational services. The rationale behind this paper is to compare and discuss the existence or lack of existing approaches regarding the implementation of cloud computing services in smart learning systems. This is done by surveying the state of the art in the area, and illustrating the requirements of context-aware smart learning systems with regard to some important factors: context-awareness, security, ontology, multi-device support, and flexibility. This paper discusses four different approaches in smart learning systems and cloud computing services: smart cloud computing, elastic model, web 3.0, and content-oriented approach. The result shows that smart cloud computing is the approach that covers all important factors mentioned before. This paper is also eager to help investigating the work that have been done before for cloud computing services in smart learning systems and to show the possible requirements for the future smart learning systems.

**Index Terms**—e-learning, smart learning services, cloud computing, context-aware, Internet enabled learning.

## 1 Introduction

The rapid enhancement of digital technology is creating not only new possibilities but also new challenges. The current society is being redefined by the advancement of technologies in almost every field of human being. Nowadays, e-learning and cloud computing are emerging as the complex paradigm of modern education with reduced investment for teachers and educators. E-learning is electronic and Internet based learning, using Internet technology to design, implement, select, manage, support and extend learning. E-learning will not replace traditional educational methods, but will greatly improve the efficiency of higher education [1].

An increasing number of universities and educational institutions in the USA and UK are adopting cloud computing not only for incrementing cost savings but also for improving the efficiency and convenience of educational services [2]. The cloud computing systems have been implemented for e-learning services. However, most of the current cloud-based education systems are focusing on delivering learning materials rather than supporting and establishing an integrated cloud-based educational service environment.

Smart learning (s-learning) is a new paradigm of learning. The concept of s-learning acts as an important role in the creation of an efficient learning environment that offers personalized contents. It also supplies students with a nice communication environment and thousands of resources. However, the existing learning infrastructure is still not complete. For instance, it does not allocate necessary computing resources for s-learning systems dynamically [3]. Today, the majority of s-learning systems have some problems in interfacing and sharing data with other systems. This might lead to duplication of data and low utilization of resources. To overcome this problem, it is advised to use cloud computing to support resource management. The cloud computing environment has the needed foundation for the integration of platform and technology. It combines teaching resources distributed over various locations by utilizing existing conditions as much as possible to meet the demands of the teaching activities.

In this paper, some approaches of cloud computing in s-learning systems are discussed and evaluated. The evaluation of cloud computing in a s-learning system approach is based on several studies that attempted to define factors that drive successful online education [4, 5, 6]. Those works stated that there are some factors that push an e-learning system to be successful. The result showed that the factors are learner's computer anxiety, instructor attitude towards e-Learning, e-Learning course flexibility, e-Learning course quality, perceived usefulness, perceived ease of use, student collaboration, and diversity in assessments are the critical factors affecting learner's perceived satisfaction. However, our study only tries to investigate the technical implementation of cloud computing without the involvement of students, instructors, or courses. Thus, from this reference, the only factor that is possible to be used is only flexibility of the system. Furthermore, in order to evaluate the approaches deeper, we add some more factors that are relevant with s-learning system, such as context-awareness, security, multi-device support, and ontology utilization.

The rest of this review paper is organized as follows. Section 2 starts with general introduction into the difference between conventional e-learning and s-learning. Section 3 elaborates the relation of cloud computing in educational systems. Section 3 also includes the necessity of implementing cloud computing services and cloud-based applications in educational systems. Section 4 describes the current approaches that has been carried out according to the context of the study. Section 5 provides a discussion about the approaches that are mentioned in section 4. Finally, concluding remarks and future works are drawn in section 6.

## 2 E-Learning and S-Learning

There are several terminologies that refer to a learning environment with electronic devices, computers, or Internet. A study tried to investigate these terminologies when applied to some particular scenarios [7]. With around 40 respondents involved in the survey, the result alleged that definitions found in various articles mirror the conflicting responses provided by the respondents in this study. The findings showed great differences in the meaning of foundational terms that are used in the field, but also provide implications internationally for the referencing, sharing, and the collaboration of results detailed in varying research studies [7].

Conventionally e-learning provides teaching and learning by computers connected using wire connections and in a lecture-style classroom setup. Although learners are able to browse and download resources anytime and anywhere through the existing e-learning platform, they were limited to traditional lecture-class setup. Afterwards, e-learning was developed with the advancements of Internet. Thus, there are a number of cloud-based applications available in the e-learning field [8]. However, E-learning will not in any way replace traditional educational methods. Nevertheless, this will significantly improve the efficiency of the education [1]. A research has shown e-learning impact on individual performance. Moreover, the study has

• *Guntur Dharma Putra is a Master Student Computing Science at the RuG, e-mail: g.d.putra@student.rug.nl.*

offered various suggestions to different communities of practitioners to improve their performance with regards to the adoption and continued use of e-learning [9].

S-learning has become an important method of learning during the recent time [10]. It has been made possible by the new advancements of Internet and Information Technology. The s-learning has a big role in creating a nice and personalized learning situation, and also being well adapted to the current education model wherever possible [3]. Usually, the teaching and learning that e-learning offers is only inside of a lecture-style classroom with desktop computers. Although students are able to download resources and browse through the existing e-learning platform regardless of time and place, they were still confined to the limits of the classical classroom-setups.

Yet, there is no exact definition of s-learning. Related scholars who are involved with education business are discussing that the concept of s-learning should not be limited to just utilizing smart gadgets. Thus, the government, academics, and the educational industry have been working on defining s-learning. At the s-learning Korea forum 2010 [11], a concept of s-learning was proposed as follows: first, it is focused on humans and content more than on devices; second, it is effective, intelligent tailored-learning based on advanced Information Technology (IT) infrastructure [10].

The Korean Ministry of Education, Science and Technology (MEST) defined s-learning as Self-directed, Motivated, Adaptive, Resource-enriched, and Technology-embedded [12]. More information on S.M.A.R.T Learning promoted by MEST is as follows:

- S: Self-Directed, which means that the education system is progressing toward a self-learning system more than ever. Students' roles transition from knowledge adopters to knowledge creators. Also, teachers become facilitators of learning.
- M: Motivated means education becomes experience centered and involves learning by doing; creative problem solving and individualized assessment are pursued.
- A: Adaptive means strengthening of the education system's flexibility and tailoring learning for individual preference and future careers.
- R: Resource-enriched means that s-learning utilizes rich content based on open market, cloud education services from both public and private sectors. In other words, it expands the scope of learning resources to include collective intelligence, Social Learning.
- T: Technology-embedded means that in the s-learning education environment, students can learn anywhere, any time through advance technologies.

## 3 Cloud Computing and Education

Electronic devices, especially computers, have been playing an important role in modern education since the emergence of e-learning. Education also has a close relation with the Internet as many e-learning platforms or systems are based on on-line applications. An example of e-learning platform that recently has been addressed a new form of on-line learning is Massive Open on-line Course or MOOC for short[13]. Some examples of these MOOCs are edX [1], Coursera[2], and Udacity[3]. Several universities, such as MIT[4] and UC Berkeley[5], also put their teaching materials, ranging from undergraduate to graduate-level on-line, so that they are openly available and easy to access. A study asserted that openness and reputation are important for MOOC providers especially for course offering [14]. Openness and reputation are ways that MOOC providers can both differentiate themselves from competitors and enhance an individual's intention for continued MOOCs enrollment.

This section describes the close relation between cloud computing and educational field, especially e-learning and s-learning. Cloud computing that introduces efficient scale mechanism can let construction of e-learning systems be entrusted to suppliers and provide a new mode for e-learning [15].

### 3.1 Benefits of Cloud Computing in Educational System

A study by Bouyer et al. [16] alleged that cloud computing is reducing the difference between on campus education and distance education. Still there are some limitations of e-learning for lab based education due to computation power. Fortunately cloud computing is the technology that is able to offer distinguished services in three layers. Cloud computing enables students to access the knowledge by distributed e-learning resources in a public, private, or hybrid cloud types. Because of using cloud computing systems for deploying a modern education environment, universities and other educational organizations have to take into account various things, such as cost and accelerate delivery of learning services, quick learning, and privacy issues.

Cloud computing also owns several important benefits for education [16]. Those important advantages are quick delivery of various services, cost minimization, risk reduction, security enhancements, reshaping teaching, and collaboration expansion.

### 3.2 Cloud-Based Application in Education Systems

A definition of cloud computing declares that it is a technology that provides users with information resources by using the Internet as a medium. Users can make use of information resources such as application software or storage space from the cloud without needing to download them beforehand. Users only have to pay per usage charges for resources they used. The concept of cloud computing is a combination of distributed computing, grid computing, utility computing, and so on [8]. When a particular user requests a service from cloud server, the server immediately provides the requested services to the user based on the request details. This implies that the server has the ability to complete the user's request personally. These features allow the users to use the service only the amount they need at their desired time and pay according to the usage proportionally.

Several researchers have presented their approach to implement cloud computing in educational systems. For example, Casquero et al. [17] presented a framework based on iGoogle and using the Google Apps platform for the development of a network of cooperative personal learning environments. They discussed the integration of institutional and external services in order to provide customized support to faculty members in their daily activities. They take the advantage of Google's framework as a testbed for the research, implementation and testing of their educational purpose services as well.

Even though much work has been carried out with regard to adopting cloud computing for educational systems, further studies need to be conducted to develop more diverse forms of cloud-based education systems, in more innovative and efficient ways [18]. Meanwhile, most of the existing cloud-based education systems are concentrating on delivering and sharing of learning materials and teaching activities, rather than constructing and supporting an integrated, total cloud-based educational environment.

## 4 Cloud Computing in Smart Learning System Approaches

There are several approaches for implementing cloud computing in a smart learning environment. This study has evaluated several recent approaches [10, 8, 18, 2, 19]. Those approaches are categorized into certain categories: smart cloud computing, elastic model, cloud and Web 3.0, and content oriented approach.

### 4.1 Smart Cloud Computing

Smart Cloud Computing (SCC) has the capability to provide a s-learning environment by using elastic computing for 4S model. Elastic 4S is carried out through an intelligent learning engine that consists of four service rules - Smart Pull, Smart Prospect, Smart Content and Smart Push. SCC offers system standardization and describes how to manage it properly. A conventional e-learning system is only capable to display a single content on a single device or multiple contents

[1]https://www.edx.org/
[2]https://www.coursera.org/
[3]https://www.udacity.com/https://www.coursera.org/
[4]http://ocw.mit.edu/
[5]http://webcast.berkeley.edu/

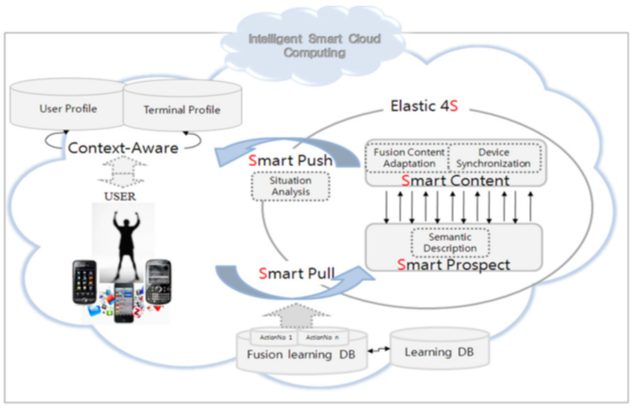


Fig. 1. SCC Architecture [8].

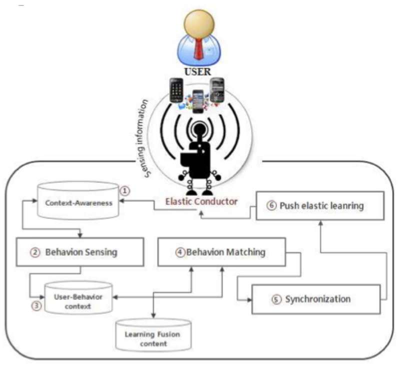


Fig. 2. Elastic conductor architecture [10].

on one device. The SCC can deliver s-learning to the users so they can use multiple devices to render multi learning contents. The SCC uses context-aware sensing, a sensing process that will extract user's preference, to provide personalized contents. Sensing is carried out through the location and Internet Protocol (IP) address of each device. Furthermore, the architecture of the model is shown in Figure 1 [8].

Figure 1 shows how the SCC provides s-learning to the user. The proposed system utilizes Elastic 4S based on information obtained from the user. The information from users contain information about the user themselves and device they are using, received by context-aware sensors. Context-aware monitoring monitors user requests and the kind of devices that the user is currently using. SCC provides user-aware services based on elastic 4S by utilizing the information collected by the sensors. Elastic 4S is carried out through an intelligent learning engine that consists of four service rules - Smart Pull, Smart Prospect, Smart Content and Smart Push. The definition of Elastic 4S (E4S) is described as follows:

$$\{E4S_i\} = \{(Spull_i, Spros_i, Scon_i, Spush_i)\}, 1 \leq i \leq N \qquad (1)$$

where:

$Spull_i$: Smart pull analyze the extractable content from the sensing information.

$Spros_i$: Smart prospect description of the content for target devices and delivery time.

$Scon_i$: Smart content connection establishment between server and target devices.

$Spush_i$: Smart push synchronized delivery of contents to target devices.

As shown in the definition, E4S pulls the sensing data and analyzes the contents that are possible to be extracted. The context-aware module will only extract the intended information based on sensing data. The system then will prospect what contents are appropriate based on the sensing data and finally push the content to specified users.

The system also offers context-aware services, since Context-aware is important [20]. Context-aware is also a key point in s-learning. Another research also contributed to implement context-aware services in educational system [21]. However, cloud computing was not used, as this study only focuses on context-aware in classroom setups only.

## 4.2 Elastic Model

A study conducted by Kim et al. [10] proposed the elastic conductor that performs provisioning and scheduling for the decision of smart activities. The provisioning and scheduling are performed through an inference engine that uses the rules based on three attributes: an object id for user context, a predicate relationship for user behavior, and a value for thinking. The elastic conductor is utilized in Platform as a Service (PaaS) cloud type as a smart activity.

The elastic conductor is able to generate user interface configurations as well, for instance, personalized views and content rendering. These processing tasks consist of four smarts concept: behavior sensing, behavior matching, synchronization and push for displays multi-contents on the multi-device. This system does the sensing of context-awareness through the location and IP address of each device. The architecture of the model is show in Figure 2.

Figure 2 shows how the conductor delivers s-learning to the user. Information of the user includes the information about the user itself and the device, which is received by context-awareness sensors. Context-awareness monitors user requests and the kind of devices that the user is currently using. By using the information collected by the sensors, the conductor pulls the sensing information and analyses the extractable contents. The behavior sensing concept acts as an information filter that extracts only the intended information from sensing data and stores it in the user-behavior context Data Base (DB). There can be multiple contexts in sensing data, which depends on the services available in the learning management system.

To provide the s-learning service to each unique user, the behavior sensing concept has to automatically deduce the real situation of the user. The behavior sensing is the process of extracting user's behavior information through a variety of sensors to filter information from the sensing information. The filtered information is analyzed to figure out user's behavior patterns. This patterns consist of set of user preference, GPS and value of terminal MAC-ID.

The filtered process in behavior sensing is defined with the rating function as:

$$R : Ua \times Ui \times T \times L \times D \rightarrow Rating \qquad (2)$$

where $Ua$ is user action, $Ui$ is user interest, $L$ is location, $T$ is time, $D$ is device, $Rating$ is the information of rating. The $Ua$ dimension is defined as $User \subseteq Uaction \subseteq Urequest \subseteq Learning\ object \mid title$ and consist of a set of user situation. Similarly, the $Ui$ dimensions are defined as $User \subseteq Uinterests \subseteq Uneeds \subseteq Uexpertise \mid experience$. The $L$ is defined as $Location \subseteq Homes \subseteq Street \subseteq Company$. The $T$ is defined as $Time \subseteq Month \subseteq Day \subseteq Morning \subseteq Lunch \subseteq Afternoon \subseteq Evening$. Finally, the $D$ dimension can be defined as $Device \subseteq Terminal\ MAC\ ID \subseteq Application\ type$. Visually, ratings R on the filtered process is can be stored in a multidimensional cube.

The double cube is stored rating $R(Ua, Ui, T, L, D)$ for the filtered proves $Ua \times Ui \times T \times L \times D$, where the five tables define the sets of user action, interest, location, time and device associated with Action, Interest, Time x Location and Device dimensions respectively.

For example, the rating $R(303, 1302, 2, ASP1) = 7$ means that for the action with action ID 303, the user's interest learning object is interest ID 1302 and using this item mainly Time ID 5 in the Location Company, rating was specified during in the device ID ASP1. In other words, the user uses the application (ID 1302) every afternoon by using a smart phone at the street. So that filtered data is the basis for

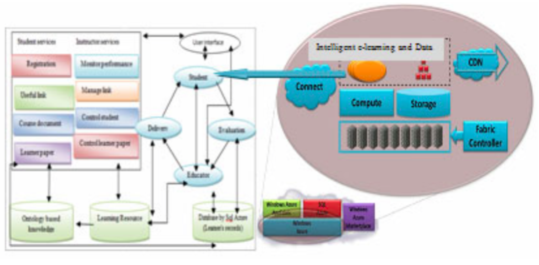


Fig. 3. Windows Azure provides compute and storage services for intelligent e-learning in the cloud [19].

creating user behavior database (DB) for providing s-learning service. According to following classification the situation is determined and then the user behavior database will be created.

### 4.3 Cloud and Web 3.0

Nasr et al. [19] proposed an approach of e-learning system that is supported by Platform as a Service (Paas), Infrastructure as a Service (Iaas), and Web 3.0. This work is basically using a cloud computing platform provided by Microsoft, which is known as Windows Azure[6]. As described in this paper, Windows Azure has several important parts such as a computing part, a storage part, a fabric controller, and a Content Delivery Network (CDN). The parts are depicted in Figure 3.

An intelligent e-learning system based on an integration between cloud computing and web 3.0 is developed in this approach in order to enhance the efficiency of a learning environment. This proposed system also provide an up-to-date, self-regulated, stability, QoS(Quality of Service) guaranteed system. However, this system is highly dependent with the Windows Azure platform.

### 4.4 Content Oriented

A research carried out by Jeong et al. [2, 18] proposed a content-oriented smart education system based on cloud computing that integrates a number of features required for implementing a cloud-based educational media service environment. The objective is to develop an integrated education content service system based on cloud computing to deliver and share a variety of enhanced forms of educational content. This proposed approach developed six main features as its foundation. Firstly, by establishing a private cloud platform to install and operate a cloud-based educational media service environment. Secondly, developing a common file format enabling manipulation of various forms of media content on multiple platforms. Thirdly, implementing an authoring tool, allowing teachers to create various types of smart media content, including text, images, sound, and video. Fourthly, developing a content viewer to display media content on diverse types of devices through a multi-platform based design. Fifth, implementing an inference engine to provide students with customized individual learning content by analyzing their learning and content usage patterns. Sixth, including a security system to encrypt data and to control user access for dependable smart media content services.

Figure 4 presents the proposed cloud-based education system for smart media content services. The proposed system enables delivery and sharing of a variety of enhanced educational content by integrating a number of features required for the deployment of a cloud-based educational media service environment. Figure 2 shows the proposed system with its six main features required for deploying cloud-based educational content service.

In detail, this proposed system has six main features, those are cloud platform, common file format, authoring tool, content viewer, inference engine, and security system. Private cloud platform provides an infrastructure for the implementation of a cloud-based educational media service environment by applying several cloud computing technologies, such as data synchronization, virtualization, service

[6]http://azure.microsoft.com/

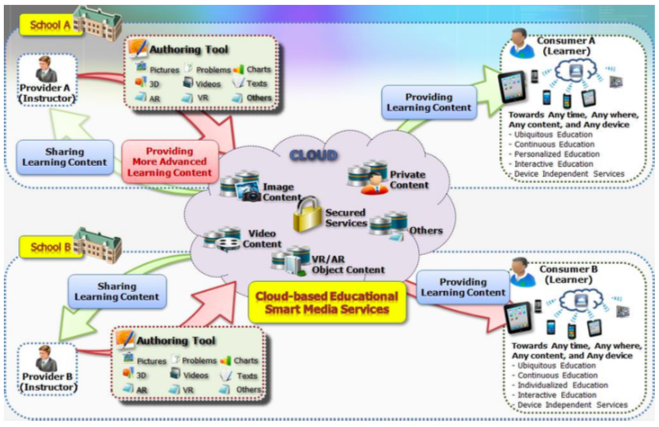


Fig. 4. Architecture of the proposed cloud-based education system for smart content services [18].

provisioning, and multi-sharing services. Common file format is also developed in order to be able to manipulate various types of media content on multiple device platforms based on an XML document format with HTML5, eXtensible 3-Dimensional (X3D), and JavaScript. Authoring tool allows teachers to create many types of smart media content ranging from text, images, sound, and video. Then, the content viewer is developed to display media on multiple platforms and inference engines will provide students with personalized learning content by analyzing their preferences, learning styles, and content usage patterns. Finally, a security system is included to encrypt data and control privileged user access.

## 5 Comparison of Approaches

Some benefits of cloud computing in education are drawn in a study by Gonzalez et al. [22]. Those advantages are a wealth of online applications to support education, flexible creation of learning environments, support for mobile learning, computing-intensive support for teaching, learning, and evaluation, scalability of learning systems and applications, costs saving in hardware, cost saving in software.

This study tries to evaluate the above mentioned approach of cloud computing in s-learning systems based on several studies that attempted to define factors that drive successful online education [4, 5, 6]. That works alleged that there are some factors that push an e-learning system to be successful. The result showed that the factors are learner's computer anxiety, instructor attitude toward e-Learning, e-Learning course flexibility, e-Learning course quality, perceived usefulness, perceived ease of use, student collaboration, and diversity in assessments are the critical factors affecting learner's perceived satis-

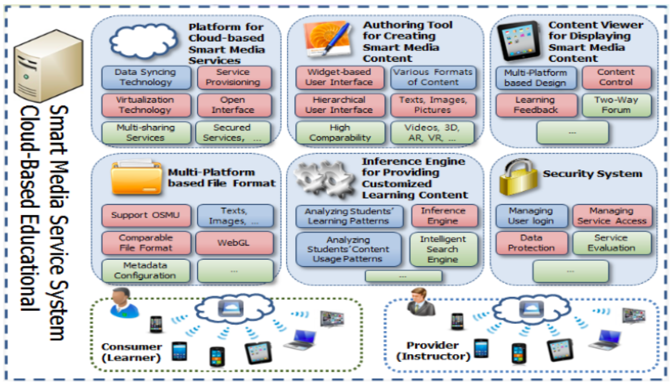

Smart Media Service System
Cloud-Based Educational
Platform for Cloud-based Smart Media Services
Data Syncing Technology
Service Provisioning
Virtualization Technology
Open Interface
Multi-sharing Services
Secured Services, ...
Authoring Tool for Creating Smart Media Content
Widget-based User Interface
Various Formats of Content
Hierarchical User Interface
Texts, Images, Pictures
High Comparability
Videos, 3D, AR, VR, ...
Content Viewer for Displaying Smart Media Content
Multi-Platform based Design
Content Control
Learning Feedback
Two-Way Forum
...
Multi-Platform based File Format
Support OSMU
Texts, Images, ...
Comparable File Format
WebGL
Metadata Configuration
...
Inference Engine for Providing Customized Learning Content
Analyzing Students' Learning Patterns
Inference Engine
Analyzing Students' Content Usage Patterns
Intelligent Search Engine
...
Security System
Managing User login
Managing Service Access
Data Protection
Service Evaluation
...
Consumer (Learner)
Provider (Instructor)

Fig. 5. Infrastructure of the proposed system with its six main features [18].

faction. However, since our study only tries to investigate the technical implementation of cloud computing without the involvement of students, instructors, or courses, the only factor that is possible to be used is flexibility. Furthermore, in order to evaluate the approaches deeper, we add some more factors that are relevant with s-learning systems, such as context-awareness, security, multi-media support, and ontology utilization.

Kim et al. presented a s-learning service that is based on SCC. As mentioned above, this approach offers s-learning services through 4S model, which are smart pull, smart prospect, smart content, and smart push. Security is also covered in this approach, although it is just securing user's personal information through some security setting without data encryption. SCC also manages to implement context-awareness by utilizing context-aware sensors. The context-aware module considers the characteristics of each user individually, such as learners' knowledge interests, needs, expertise, and experiences. Thus it can provide highly customized and relevant learning services to each user. Each cloud type (Iaas, Paas, Saas) is covered using smart cloud approach. Furthermore, no specific platforms are used in this approach, this will lead to a good flexibility. Semantic description based on UVA (Universal Video Adaptation) are used to provide accurate and meaningful information for the fusion content (content-database). The paper also managed to show the implementation with four fusion media, which includes video, audio, Microsoft Power Point presentation, and text.

Elastic Model (EM) [10] focuses to meet users' need intelligently. The approach is slightly similar with the SCC because EM also use the four smart concept to cloud service. This proposed system also support multiple devices just as SCC approach. Furthermore, this system is independent from any platform and this makes this approach flexibile to be implemented. Security factor is also slightly described to secure each user's personal information using some security setting e.g. user's schedule and location. This approach also utilizes context-aware by implementing behavior sensing to provide s-learning service to each individual user. However, there are no ontological approaches explained in the paper.

The approach from Nasr et al. [19] integrates cloud computing as a platform with the help of Web 3.0 to build an intelligent learning system. This is done by utilizing Windows Azure as the platform for the system. The system also proposed ontology based model. For implementing the knowledge the leaning resources have to be described by means of meta-data. This is also a resource for contextual learning. Security system in this approach is fully covered in Windows Azure platform that obviously has the enterprise level security system. However, this approach does not mention any description in multi device support and since this system depends on Windows Azure platform to be developed, this system is more likely to have less flexibility.

Content-oriented approach [18] makes use of cloud computing in smart education systems and integrates a number of features that will enable a school to deliver and share a variety of enhanced forms of educational content including text, images, videos, and even 3D or even virtual scenes. Thus, this approach has a rich amount of contents and does support multimedia content. Moreover, this content-oriented approach also supports multiple-devices with multiple screen sizes and features. This concept also offers a security system. The author mentioned that security is needed not only to encrypt data and control privileged user access but also to protect and solve network problems in the cloud. An inference engine is utilized to provide students with personalized learning content by analyzing their preferences, learning styles, and content usage patterns. This proposed system does not use an ontological approach to model the educational database or user preference. However, this system offers flexibility as it utilizes XML for data and document exchange and it also develops Common File Format to be able to manipulate various types of media content on multiple device platforms. Although this approach seems to have a comprehensive concept of cloud computing in s-learning systems, this proposed concept has not been fully implemented yet.

To sum up the above discussion, Table 1 depicts the four approaches in summary. As seen in Table 1, we can conclude that SCC has all factors that the author wants to assess. Some other approaches lack in one or two factor and there are no approach that lacks three or more factors.

Table 1. Comparison between the existing cloud computing services implementation in s-learning system.

| | SCC | EM | Web 3.0 | Content-oriented |
|---|---|---|---|---|
| Security | ✓ | ✓ | ✓ | ✓ |
| Context-awareness | ✓ | ✓ | ✓ | ✓ |
| Multi-device support | ✓ | ✓ | – | ✓ |
| Ontology | ✓ | – | ✓ | – |
| Flexibility | ✓ | ✓ | – | ✓ |

Moreover, in order to deliver an effective and successful s-learning system, intuitive user interface and an out of the box user experience that will make the system easier to be used might be factors that has to be keep in mind since it is mentioned in [4] that ease of use is one factor that drives a successful learning system. Moreover, the author suggests that adaptation of cloud computing in s-learning systems can be considered as successful if it is able to prevent duplication of data and can manage computing resources efficiently.

## 6 Concluding Remarks

Cloud computing services offer several advantages in its implementation to e-learning system, such as increased cost savings and also improved efficiency and convenience of educational services. Furthermore, e-learning services can be also enhanced to be smarter and more efficient using context-aware technologies as context-aware services that are based on the users behavior. This paper has compared and reviewed several researches and articles about cloud computing services in smart learning environments in terms of context-awareness, security, ontology, multi-device support, and flexibility. These factors are drawn out from a research that tried to determine factors that drive the successfulness of e-learning. The result showed that SCC is the only approach, which is able to cover all factors used in assessment.

Further studies with regard to cloud computing services in smart learning systems should consider security as an important issue as users' information are stored in the system. Encryption can be taken into account for improving the security of the system and to protect personal data from any unwanted access. The comprehensive surveys are needed to be undertaken in order to assess the capability and the usefulness of the system. Moreover, a benchmark about how system runs will show the benefits and how efficient cloud computing has made the smart learning system. To sum up, further studies, which take more variables into account, will need to be undertaken.

## Acknowledgements

The author would like to thank the expert reviewer, Fatimah Alsaif, for the valuable help and suggestions on reviewing this paper. Also, the other peer reviewers, Mathieu Kalksma and Stephan Boomker, who have provided a lot of useful feedbacks that really helped during the rewriting process.

## References

[1] Monisha Singh Sudhir Kumar Sharma, Nidhi Goyal. Distance Education Technologies: Using E-learning System and Cloud Computing. *International Journal of Computer Science and Information Technologies*, 5(2):1451–1454, 2014.

[2] Ji-Seong Jeong, Mihye Kim, and Kwan-Hee Yoo. A Cloud based Smart Education System for e-Learning Content Services. *Advanced Science and Technolohy Letters (ASTL)*, 25:131, 2013.

[3] L Uden, I T Wangsa, and E Damiani. The future of E-learning: E-learning ecosystem, 2007.

[4] Pei-Chen Sun, Ray J. Tsai, Glenn Finger, Yueh-Yang Chen, and Dowming Yeh. What drives a successful e-Learning? An empir-

ical investigation of the critical factors influencing learner satisfaction. *Computers & Education*, 50(4):1183–1202, May 2008.

[5] B Fetaji and M Fetaji. E-Learning Indicators Methodology Approach in Designing Successful e-Learning, 2007.

[6] N Laily, A Kurniawati, and I A Puspita. Critical success factor for e-learning implementation in Institut Teknologi Telkom Bandung using Structural Equation Modeling, 2013.

[7] Joi L. Moore, Camille Dickson-Deane, and Krista Galyen. e-Learning, online learning, and distance learning environments: Are they the same? *The Internet and Higher Education*, 14(2):129–135, March 2011.

[8] Svetlana Kim, Su-Mi Song, and Yong-Ik Yoon. Smart Learning Services Based on Smart Cloud Computing. *Sensors*, 11(8):7835–7850, 2011.

[9] Soheila Mohammadyari and Harminder Singh. Understanding the effect of e-learning on individual performance: The role of digital literacy. *Computers & Education*, 82:11–25, November 2014.

[10] S Kim and Yongik Yoon. Elastic Service Model for Smart Learning Based on Cloud Environment, 2013.

[11] Taisiya Kim, JiYeon Cho, and BongGyou Lee. Evolution to Smart Learning in Public Education: A Case Study of Korean Public Education. In Tobias Ley, Mikko Ruohonen, Mart Laanpere, and Arthur Tatnall, editors, *Open and Social Technologies for Networked Learning SE - 18*, volume 395 of *IFIP Advances in Information and Communication Technology*, pages 170–178. Springer Berlin Heidelberg, 2013.

[12] MEST: Ministry of Education, Science and Technology of the Republic of Korea, Smart education promotion strategy, Presidents Council on National ICT Strategies, 2011.

[13] Anoush Margaryan, Manuela Bianco, and Allison Littlejohn. Instructional Quality of Massive Open Online Courses (MOOCs). *Computers & Education*, 80:77–83, August 2014.

[14] Khaled M. Alraimi, Hangjung Zo, and Andrew P. Ciganek. Understanding the MOOCs continuance: The role of openness and reputation. *Computers & Education*, 80:28–38, August 2014.

[15] Xiao Laisheng and Wang Zhengxia. Cloud Computing: A New Business Paradigm for E-learning, 2011.

[16] Asgarali Bouyer and Bahman Arasteh. The Necessity of Using Cloud Computing in Educational System. *Procedia - Social and Behavioral Sciences*, 143:581–585, August 2014.

[17] Oskar Casquero, Javier Portillo, Ramón Ovelar, Jesús Romo, and Manuel Benito. igoogle and gadgets as a platform for integrating institutional and external services. *Mash-Up Personal Learning Environments (MUPPLE08)*, page 37, 2008.

[18] Ji-Seong Jeong, Mihye Kim, and Kwan-Hee Yoo. A Content Oriented Smart Education System based on Cloud Computing. *International Journal of Multimedia & Ubiquitous Engineering*, 8(6), 2013.

[19] Mona Nasr and Shimaa Ouf. A proposed smart E-Learning system using cloud computing services: PaaS, IaaS and Web 3.0. *International Journal of Emerging Technologies in Learning (iJET)*, 7(3):19–24, 2012.

[20] A R Pratama, Widyawan, and G D Putra. An infrastructure-less occupant context-recognition in energy efficient building, 2014.

[21] K Scott and R Benlamri. Context-Aware Services for Smart Learning Spaces, 2010.

[22] José A. González-Martínez, Miguel L. Bote-Lorenzo, Eduardo Gómez-Sánchez, and Rafael Cano-Parra. Cloud computing and education: A state-of-the-art survey. *Computers & Education*, 80:132–151, September 2014.

# Natural interaction from a graph theory perspective

Julien van der Land, Jorrit Idsardi

**Abstract**— This paper surveys the usage of graph theory in a natural world context. We highlight several studies that have used graph theory and provide an in depth view on how these studies have used graph theory within their application domain. Specifically the studies that were examined cover the following topics; termite nests, landscape connectivity, trophic relationships, and protein structures. We found that the application of graph theory within the natural domain was varied and that graph theory is often used in these studies to gain insight into the complex systems that are found in this domain. During the survey we identified two cases where in graph theory is used; gaining insight into the underlying data that a graph represents and, using graphs for visualization purposes. These two cases can of course be used together as is the case with the insect nests studies and, landscape connectivity.

## 1 Introduction

In this paper we describe complex systems in the natural world from a graph theory perspective. Graph theory has been used in the field of biology to create insight into many different aspects of the natural world. The following topics were examined in this survey;

- The dynamics and equilibria of ecological predator-prey networks[4].
- Mapping the topological structure of termite nests[5].
- Creating insight into food-web structures[2]
- The application of graph theory for for protein structures [10]

Animal inter- and intraspecies interactions are often analyzed using food-webs, when these interactions concern the predation relations between different species. Layout characteristics that insect nests exhibit, can be interesting to study due to the properties found within these nests. Many animals/insects display certain patterns in terms of migration routes or other environmental features. Insight into these routes/features allow preservation authorities to map the interactions between different habitats for the preservation of species. Lastly protein structures are extremely complex and in order to study these structures, graph-theory is used to predict protein folding in order to gain insights into the stability of proteins and, more importantly, function.

The application of graph theory allows us to model interactions/networks within a mathematical construct. Graphs are a combination of vertexes and edges, where in vertexes are connected by edges. Depending on the context vertexes can be used to model chambers within a insect nests or animal habitats, edges can be used to model pathways between the chambers in insect nests or migration routes that link habitats. Extending this analogy to trophic relationships between species, it can also be animals eating other animals. The vertexes would then be animals and the edges be the predator-prey relationship. Finally, graph theory can also be used to represent the structure of proteins, using several meanings for the vertexes and edges.

This paper is organized in the following manner; In section 2 we examine the application of graph theory in studies related to insect nests. In section 3 we examine how graph theory is used to study the connectivity of fragmented animal habitats. Section 4 details the application of graph theory in studies that examine the trophic relationships between species. Finally 5 details how graph theory is used when studying protein folding and protein interactions. In section 6 we discuss our findings during our examinations of the subjects.

- *Julien van der Land with University of Groningnen, E-mail:J.van.der.land.2@rug.nl.*
- *Jorrit Idsardi with University of Groningnen, E-mail: J.Idsardi@student.rug.nl*

## 2 Insect Nests

The structure of insect nests can be expressed quite naturally as a graph. In particular termite nests have been studied quite closely by [5] and [9]. In [9] Valverde et al. attempts to prove that insect nest networks are optimal. They do this by proving that termite nests are close to the percolation threshold. This is the threshold where large connected components either exist or not. This means that termites build efficient connections spanning the entire nest at low cost.

To analyze the termite nests, 3D scans of 6 different nests from different locations were used. Because the hallways between chambers are quite a bit smaller than the chambers the process of converting the nests to a 3D graph representation could be automated. This was done by scanning the nests with a medical scanner and then automatically creating a 3D graph model in which chambers are represented as vertexes and hallways as edges. The result of this process can be seen in figure 1.

After analysis of the 6 nests, Valverde et al. created a model that generates graphs similar to the nests. The model does the following: it starts with a fully connected 3D lattice, consisting of $S^2H$ nodes, in which all neighboring nodes are 1 distance away from each other. $S$ stands for the breadth and width in nodes on a layer. $H$ is the amount of vertical layers of the model. Some links are removed until only a certain amount of links are left. The removal is link is done at random using the following algorithm:

1. Select a random node
2. Select a random link connected to the node select at step 1
3. Remove remove the link according to the probability p(q)
4. Iterate over steps 1-3 until the following criteria; $2(S-1)SHp$ horizontal links left, and $S^2(H-1)q$ vertical links left. $p$ is the survival probability of horizontal links. $q$ is the survival probability of vertical links. When $p > q$, the structure is layered

Nodes are then displaced a small distance in a random direction and finally, nodes that are too close are merged in to one node. After running an evolutionary algorithm on the original nests, metrics for acceptable forms, links and size were determined. the numbers were then put into the model. Valverde et al. found that their model produces graphs similar to real termite nests. They then measured statistics regarding the efficiency and found that termite nests allow for maximizing network functionality while minimizing its connectivity.

Perna et al.[5] investigated the sparseness of the termite network as there is no additional cost in adding more connections between rooms. Also of note is fact that nodes that have a lot of connections are far from the external walls of the nest ( see figure 2). [5] Perna et al. found that chambers adjacent to the external wall usually only had one edge to the internal chambers. They propose that the reason, that outer chambers are connected by only one edge, is a defensive evolution, as the hallways are about the size of a soldier termite. This allows the termites to close off the external chamber and thereby stopping the entire invasion in their nest.

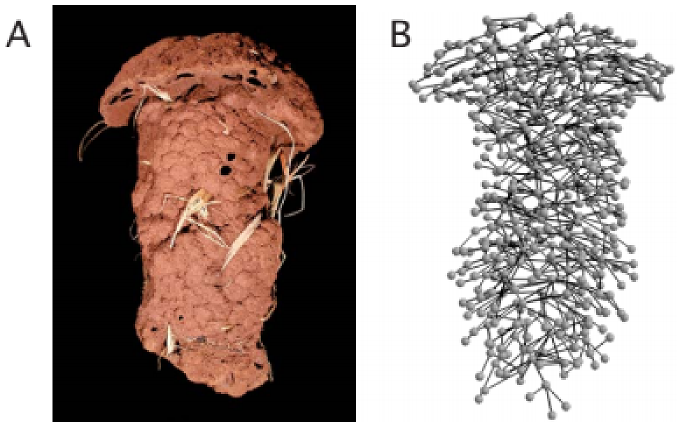


Fig. 1. (A) a termite nest and (b) the spatial 3D graph of that nest. Courtesy of [9].

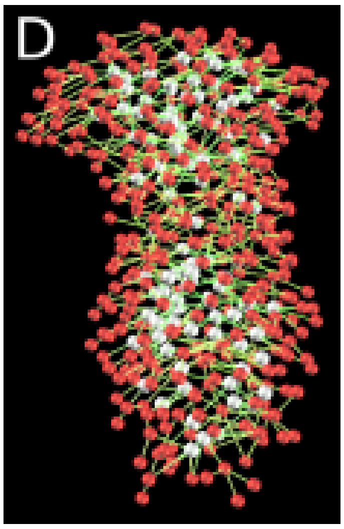


Fig. 2. Same nest as in figure 1. Red nodes are nodes at an exterior wall, white nodes are interior. Courtesy of [5].

## 3 Landscape Connectivity

In the field of biology, graph theory is used to measure landscape connectivity between high quality habitat patches on a given landscape. This is interesting for biologists because the measuring of the connectivity of these patches can aid in the preservation of animal species. Fortin et al. [3] writes that obstructions between high quality habitats reduce gene flow, which in turn increases inbreeding. Gene flow is the propagation of genes between different animal habitats for a certain species.

Cushman et al. [1] details that different features in a landscape have a certain resistance to movement for animals. Cushman et al. [1] specifically looked at the gene flow of blacks bears within a $3000km^2$ area in northern Idaho. Cushman et al. [1] found that landscape coverage impacts gene flow. Specifically Cushman et al [1] found that forest cover had a positive influence on gene flow. Areas with low forest cover and, roads were found to reduce gene flow. Cushman et al.[1] also found that areas with middle elevation levels are most optimal for gene flow. Gene flow resistance was found to increase at high and low elevation levels. In addition to that, Fortin et al. [3] also notes that these obstructions and unconnected habitat areas increase species invasion. From this we can conclude the connectivity of habitats is impacted by the features of the landscape between different habitats.

Within biology there is a lot of interest in maintaining landscape connectivity so that species can move between different habitats in such a way that gene flow positively impacted. These habits are often spaced out in a mosaic of patches[3]. This connectivity is important to keep the population of a species varied enough. In order to determine the connectivity between these habitats Fortin et al.[3] uses spatial graph algorithms by considering the structural configuration of high quality habitat patches (nodes) and the Euclidean distance between them(links) [3].

The links between these nodes may not represents the actual path that the species take while migrating between habitats due to the landscape which may impede a species movement [3]. As such [3] uses Delaunay tessellation to form a minimum planar graph.

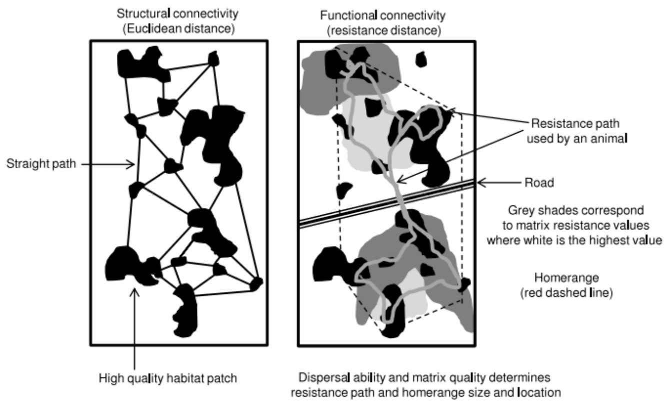

Structural connectivity (Euclidean distance)
Functional connectivity (resistance distance)
Straight path
Resistance path used by an animal
Road
Grey shades correspond to matrix resistance values where white is the highest value
Homerange (red dashed line)
High quality habitat patch
Dispersal ability and matrix quality determines resistance path and homerange size and location

Fig. 3. Landscape connectivity from [3].

Figure 3 shows the connectivity between different habitats, on the left images shows the connectivity between habitats using euclidean distance. The right images hows connectivity based on species, taking into account dispersal ability, which results in least-cost paths.

Another paper that uses graph theory to describe habitat connectivity is Rayfield et al.[6]. In this paper graph theory is used to provide an efficient measure of potential connectivity pathways in heterogeneous landscapes. In Rayfield et al[6] spatially explicit least-cost habitat graphs are used to evaluate how matrix and spatial configurations influence the assessment of habitat connectivity [6]. Rayfield et al.[6] does this by generating artificial landscapes with qualities ranging from low to high. The area and degree of the fragmentation of the landscapes is controlled in a factorial experiment. Rayfield et al. [6] found that the highest sensitivity, to relative costs of landcover, was in landscapes with fragmentation of 20% - 50%. Rayfield et al.[6] suggests several methods for identifying low-cost routes between pairs of habitat patches.

## 4 Food webs

Trophic levels are an hierarchical ordering of organisms classified into 5 different categories as shown in figure 4. This classification allows us to determine the place of animals along the food-chain. What follows is a short description of these trophic levels

1. Producer: Plant and Algae, the lowest level of the trophic pyramid.
2. Primary consumer: Animals(Herbivores) that consume producers.
3. Secondary consumer: Carnivores that only feed on primary consumers.
4. Tertiary consumer: Carnivores that feed and secondary and, possibly primary. consumers
5. Quaternary consumers: Animals that feed on tertiary and, lower trophic level animals. Quaternary consumers have no natural enemies.

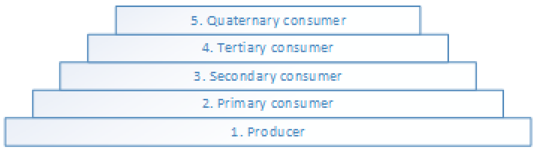


Fig. 4. Trophic levels

A food web is a web of animals connected to each other. The connections are directed and mean that the two connected animals have a trophic relationship. A trophic relationship is the relationship between animals where one animal eats the other. In figure 5 we seen an example of a food web represented as a directed graph. In this figure trophic relationships between animals(nodes) are represented by directed edges and should be read as *m* is consumed by *n* and *n* is consumed by *o*.

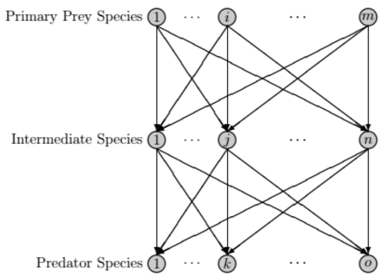


Fig. 5. An example of a food web from [4].

In biology graph theory is used to analyze different characteristics of food-webs such as whether a food-web is in equilibrium or to analyze the connectivity between predator-prey species. The application of food-webs is varied and can be used to, for example, manage the population of fish species within fishing communities. For the most part, however, food-webs are used to create insights into questions relating directly or indirectly to population dynamics.

Within the food-web context, there are some different categories of food-webs:

- Connection web - All the food resources that consumers consume.
- Energy flow web - Quantizes the energy between resources and consumers.
- Functional web - Shows interactions between species, specifically to determine population levels and, growth or decline of organisms within a ecological system.

Connection webs are normally represented by directed graphs. The direction of the links between nodes indicate which species is the consumer and which species is a resource(consumers eat resources). The energy web shows relations between the trophic levels of different species, an energy web can, for example, show that a certain species consume more energy because it's lower in trophic pyramid.

Depending on the interactions between species, the vulnerabilities and traits of a species may be taken into account when linking consumers and resources as indicated by [7]. Naturally food-webs are dynamic webs that can change with time as different species interact with each other. Population-dynamical food-web models[7] show this trophic interaction over time. Population-dynamical food-web are parameterized by using sub-models such as [7]:

- functional and numerical response of consumers to varying resource abundance.
- non-trophic losses (due to death or metabolic losses)
- populations dynamics of basal species.

Given the complexity of food-webs, a distinction has to be made between patterns in empirical data and models that can explain and reproduce sets of patterns [7] found in food-webs. Because natural food webs are so complex, a trade-off has to made between accuracy of the food-web and its completeness. As such trophic links between species are often inferred, rather than measured from the natural world.

## 5 Protein structure

Graph theory is used in the study of proteins[10] to understand many different topics of the protein. Among these topics are:

- Fold and pattern identification
- Testing of folding rules
- Functionality and structurally important motif recognition
- Identification of clusters important for function, structure and folding
- Identification of proteins with similar folds
- Protein dynamics

Because the topics are very different from one another they use different definitions for the nodes and edges within a graph. In this section we will discuss the application of graph theory for each of these topics.

**Protein structure comparison: pattern identification** There are a lot of different proteins in nature, but there is a limited number of 3D structures that they occur in. Learning what certain structures, that perform a certain function, look like, allows researchers to understand what a protein does by examining its structure, which is modeled as a graph. The comparison of graphs is generally an expensive computation although there are some methods of approximation that are relatively fast and reliable.

**Clusters in protein folding and function** The interaction of residues with other residues and the environment, determines the functional structure of the protein. These clusters can be found using the *graph spectral technique*. Particular types of clusters are known to be important for specific functions of the protein. The graph spectral technique can differentiate between these clusters by setting criterions corresponding with the type of cluster you are tying to find. Some specific residues have been identified that correlate closely to the folding of the protein. Places where the protein interacts with other proteins can be determined using a combination of graph spectral techniques, and some the more traditional methods of determining protein interaction sites, with good accuracy.

**Protein dynamics** A protein should not only fold, but it should be able to move as well. How proteins move is currently not fully understood. Simulating the physics in full of a protein in motion, down to the molecular level however is possible. This simulation is computationally expensive, in addition to that for large structures this simulation is prohibitive. There are ways to simulate the movement of protein as it can be extracted from the molecular topology. The Gaussian network model is one such a method. Using amino-acids residues as nodes and, edges that represent the spring-like interactions between the residues. The graph spectra is then used to extract the interesting properties from the principle of statistical thermodynamics, which can be used to determine the movement. Constrained graph theory can also be used, as it can determine the flexible and rigid substructures of proteins, which then can be used to determine how a protein moves.

## 6 Discussion

As section 2 shows, the usage of graph theory in biology is somewhat new, but it enables researches to find many interesting facts. In case of the insect nests it gave researchers insight into why it was structured as it was. It gave researches insight into connections between nature areas. In the case of food-webs questions of population are considered.In the case of protein structures, graph theory is used to determine various structures, folding methods and even functionality of proteins.

Within biology graph theory is often use in two different cases;

- One wants to gain insight into the underlying data that the graph represents
- One wants to have a method that allows for visualization purposes of the underlying data

Evidence of this is found in all the studies that were surveyed. For example in section 2, graph theory is used to represented a insect nest as a 3 dimensional graph, this 3d graph is then used to gain insight into the topology of the insect nest. Further evidence is found in of this is presented in section 4 where graph theory is used to gain insight into the trophic relationships of species. During our survey of landscape connectivity, and protein structures we found that graph theory is used to gain insight into underlying data by representing the underlying data as a graph.

### 6.1 Food-webs

In the case of food-webs, there is currently no standard model available. A generic model could be built and made extendable to the various usages of food-webs by building it with graph theory in mind. This generic model can then be used by future researchers, to more easily build food-webs and for other researchers to better understand those standardized food-webs.

### 6.2 Insect nests

The models of the termite nests, were built using different methods. However Valverde et al.[9] could have used Perna et al.[5] model for generating random termite nests. Currently there exists no standard or preferred method to generate spatial 3D graph from insects nests. Defining such a standard method would aid researchers when analyzing insects nests because both Valverde et al[9] and Perna et al[5] spend significant effort on generating the graph models that represent a insect nest.

### 6.3 Protein structure

Protein structure has many models, which use graphs, but in those graphs the nodes and edges have a different meaning. Sometimes the edges of the protein structure graphs are based on similar things but with different thresholds. It would be nice, if images of those graphs are always specific about what the nodes and edges are, since the applications of those graphs do differ depending on those definitions.

### 6.4 Landscape connectivity

For the landscape connectivity it is difficult to define a single graph model that can always be applied in every situation. Even though one could assign fixed resistance values to terrain features such as roads and forest coverage the impact that these terrain features differs for different species. This would of course invalidate such an approach. Current studies into landscape connectivity have so far only examined connectivity for specific animal species. No general model has been developed that allows for the optimization for landscape connectivity for different animal species. In addition it is also unknown which properties(small world, Scale-free, etc) connected habitats exhibit.

### 6.5 Relevance

This paper is written as part of a computing science colloquium, but the relevance may seem far away with all these nature related graphs. However, there is always something to learn from observing nature even for computing scientists, as Unger et al[8] prove in their paper about building a search engine for P2P-networks, inspired by the pheromone trails used by ants. The natural world has had millions of years of evolution to solve its problems, it is time we use that evolution to our benefit.

## Acknowledgements

We would like to acknowledge and thank G.A. Pagani., G.D Putra and, E. Karountzos for reviewing our paper. In addition to that we would also like to thank Prof. M. Biehl and, Prof. F. Kramer for supplying us with valuable lessons while writing this paper.

## References

[1] S. A. Cushman, K. S. McKelvey, J. Hayden, and M. K. Schwartz. Gene flow in complex landscapes: testing multiple hypotheses with causal modeling. *The American Naturalist*, 168(4):486–499, 2006.

[2] J. A. Dunne, R. J. Williams, and N. D. Martinez. Food-web structure and network theory: the role of connectance and size. *Proceedings of the National Academy of Sciences*, 99(20):12917–12922, 2002.

[3] M.-J. Fortin, P. M. James, A. MacKenzie, S. J. Melles, and B. Rayfield. Spatial statistics, spatial regression, and graph theory in ecology. *Spatial Statistics*, 1:100–109, 2012.

[4] A. Nagurney and L. S. Nagurney. Dynamics and equilibria of ecological predator–prey networks as natures supply chains. *Transportation Research Part E: Logistics and Transportation Review*, 48(1):89–99, 2012.

[5] A. Perna, C. Jost, S. Valverde, J. Gautrais, G. Theraulaz, and P. Kuntz. The topological fortress of termites. In *Bio-Inspired computing and communication*, pages 165–173. Springer, 2008.

[6] B. Rayfield, M.-J. Fortin, and A. Fall. The sensitivity of least-cost habitat graphs to relative cost surface values. *Landscape Ecology*, 25(4):519–532, 2010.

[7] A. G. Rossberg. *Encyclopedia of theoretical ecology - Food webs*. Number 4. University of California Press, Berkeley, 2012.

[8] H. Unger and M. Wulff. Towards a decentralized search engine for p2p-network communities. In *Parallel, Distributed and Network-Based Processing, 2003. Proceedings. Eleventh Euromicro Conference on*, pages 492–499. IEEE, 2003.

[9] S. Valverde, B. Corominas-Murtra, A. Perna, P. Kuntz, G. Theraulaz, and R. V. Solé. Percolation in insect nest networks: Evidence for optimal wiring. *Physical Review E*, 79(6):066106, 2009.

[10] S. Vishveshwara, K. V. Brinda, and N. Kannan. Protein structure: Insights from graph theory. *Journal of Theoretical and Computational Chemistry*, 01(01):187–211, 2002.

# A Comparison of Decision-making Approaches in Smart Spaces

Joris Schaefers, Toon Albers

**Abstract**—Smart spaces can make autonomous decisions in order to minimize the interactions the user has to make with their environment, by adapting the environment to their needs and preferences. By controlling certain parts of the environment that people live in, smart spaces try to optimize the comfort and safety of the inhabitants, while at the same time trying to minimize costs like power consumption and user interaction. This way, the productivity of the users will be increased, because they can focus on tasks that are more important to them.
In our research we compare different approaches behind smart spaces, these are rule-based approaches, planning algorithms and learning algorithms. We first describe these methods and in order to clearly understand how each approach works, some notable implementations of the approaches will also be discussed. Finally we compare them based on the expectations users have of a smart space and the ease of implementation of the methods. The comparison will provide insight into the applicability of each automation approach and shows that an approach should be chosen based on the needs of a system on a case-by-case basis. We found that rule based approaches are best if you require an easy to produce and easy to adjust smart space. Approaches using a planning algorithms are also very adjustable and perform well on providing safety to the smart space. The learning algorithms also provide safety and are the easiest approach to integrate.

**Index Terms**—Smart spaces, decision-making, automated control.

## 1 Introduction

Smart spaces are environments that are monitored and controlled by computers. They can make autonomous decisions in order to minimize the interactions the user has to make with their environment by adapting it to their needs and preferences. Their goal is often to make the life of the user as efficient as possible, by for example performing certain actions on behalf of the user, or by managing climate controls to keep the energy bills low. Providing safety is often also seen as a major feature. By controlling certain parts of the environment that people live in, smart spaces try to optimize the comfort and safety of the inhabitants, while at the same time trying to minimize costs like power consumption and user interaction.

Consider the following example of an inhabitant of a smart home. When the alarm clock goes off, the room is still dark so the user turns on the lights. If it is light outside the user opens the curtains and in order to save power should turn off the room lights again. Some automation can be performed by allowing the user to open the curtains from their bed or by using preprogrammed timers. However, smart spaces can simplify this process by being configured or learning to do the right thing at the right time. For example, they can automatically choose to open the curtains when the alarm clock goes off and if there is sufficient outdoor light. This way the user has to do even less work.

Pragnell, Spence and Moore [21] state that the demand for smart homes and smart spaces is steadily growing. Especially so as the technology matures. However, increased maturity of smart spaces also results in increased complexity of the underlying systems. This makes research on this topic very interesting.

Smart spaces aim to perform actions automatically and dynamically based on what the user actually wants. Timed actions are not always sufficient because timers have to be set and reset to perform actions at the right time and they cannot sense nor react on situations happening in the home. There are a few approaches that enable smart space behaviour, namely rule-based, planning, and learning approaches. However, there is not a lot of published research that compares these methods and thus it can be difficult to choose an appropriate method when designing a smart space. In this paper we compare the aforementioned approaches in order to provide insight into their applicability.

- *Joris Schaefers is a MSc. Computing Science student at University of Groningen, E-mail: j.schaefers@student.rug.nl.*
- *Toon Albers is a MSc. Computing Science student at University of Groningen, E-mail: t.albers.2@student.rug.nl.*

We start by describing our methodology and looking at the expectations users have of smart spaces, which can be used to compare different smart-space approaches. This is followed by the description of the three decision-making approaches. Afterwards some well-known smart space implementations will be discussed, as well as a project focused on energy management through automation. The described methods are compared and conclusions are presented.

## 2 Method

In this section we look at the way to compare the different methods of automation of the smart space discussed in section 3.

A survey was done in 2013 among people 25 years or older with a moderate income [12]. The results of this survey showed that the most important reasons for users to use a smart space system are (in order of importance):

1. Security and safety
2. (Remote) control
3. Convenience
4. Energy management

A survey by Pragnell et al. [21] obtained similar results. The main reason for using a smart space, people said, was to have more security and safety. The surveys show that people mostly think about fire detection and gas leak detection. Controlling and remotely controlling the space are also often wanted as features. The next item is convenience. When thinking about convenience, the survey shows that the main feature people seek is climate control and lighting control. The last item is energy management. Users want to save power and money by, for example, only turning on lights in rooms that are occupied.

Apart from the expectations users have of the system it is also necessary to look at factors regarding the implementation of the different automation methods. The comparison is therefore done based on the following properties:

- Safety: Enhancing the safety of the inhabitants by reducing the risks in case of danger.
- Control: The ability of controlling the home from a single device in or out of the house.
- Convenience: How well the home reduces the amount of actions the inhabitants have to perform.
- Energy management: How applicable the approach is for saving energy.
- Production: How little effort is required to develop a system.
- Integration: How little effort is required to set up the system in a new home.

- Adjustability: How much the users have the ability to interfere with the system and adjust the actions the smart space takes.

## 3 Methods of automation

There are several ways to automate smart spaces, which are described by Huber [11]. They distinguish rule-based approaches, planning algorithms and learning algorithms. These approaches will be discussed in this section.

All methods require telemetry from a number of sensors of the smart space, and perform actions by instructing actuators, like light bulbs or automatic curtains. The general architecture of a smart space [8] is shown in Fig. 1.

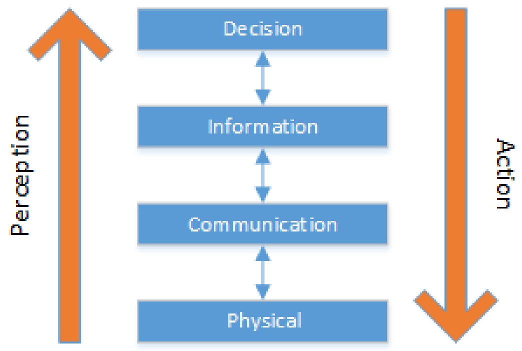


Fig. 1. The general architecture of a smart home

The physical layer contains all the physical elements (that is, the hardware) of the home. The communication layer takes care of all the necessary communication. This includes giving information to the users when needed, communicating with external resources and possibly communicating with other agents as is discussed later in section 4.1. The information layer holds all the information that might be useful in the decision-making process. It gathers, generates and stores this information. The decision layer decides what actions to be taken by the agent. These actions get selected based on the information that is obtained from the other layers. The information flow is bottom-up. This means sensors in the physical layer communicate their status to the information layer (through the communication layer). The decision layer then decides what action to take based on that information. The handling of the actions the decision layer chooses to do flows top-down. The information layer is used in deciding what action to take and then gets updated to include the information about the decision that has been taken. The chosen actions to execute are then passed to the communication layer, which will execute the actions on the appropriate devices.

The methods discussed in this section are general methods of automation used by smart spaces. Section 4 discusses several well-known implementation of smart spaces in which these general methods are used.

### 3.1 Rule based decision making

In a rule-based approach, the status of the devices are directly determined using rules. Rules have the form of 'if the user enters the house, then turn on the lights'. Of course, most rules will be much more specific and complex. For example, lights may not have to be turned on when there is enough natural light from outside. This approach is described schematically in Fig. 2. The user defines rules that the system should follow using the user interface. The solver knows the rules and verifies them against the current context of the home to see if actions needs to be taken. The devices in the smart space, like automatic curtains and lighting, are called actuators. The solver uses these actuators to satisfy the rules and take the necessary actions.

There are a multitude of ways to implement a rule-based approach. The rules are mostly defined by a programmer, using a rule-based programming language like Jess, Prolog or SWRL [11, 18]. Degeler and Lazovik [7] provide a way to define rules and execute them efficiently using a Constraint Satisfaction Problem (CSP). In a CSP, constraints are defined over variables and the solver tries to find a solution to these constraint by trying to change these values using the available actuators. When the value of a sensor or the overall availability of a device changes, the rules need to be re-evaluated. However, it would be inefficient to evaluate all rules if there are a large number of them. Degeler and Lazovik use a dependency graph to efficiently find the rules that are affected by changes. Lu, Li, Jin, Xing and Hao [18] propose separating the rules of the system from the controlling and decision-making components into what they call a 'knowware' containing only the logic rules. This is done to increase the manageability and reusability of the rules, since the rules are separated from the underlying system they are executed on. However, users will need experience in the rule-programming languages before they can modify these rules themselves. A solution to this is found with visual programming languages such as Blockly, where users can drag and drop predefined elements to define rules [9].

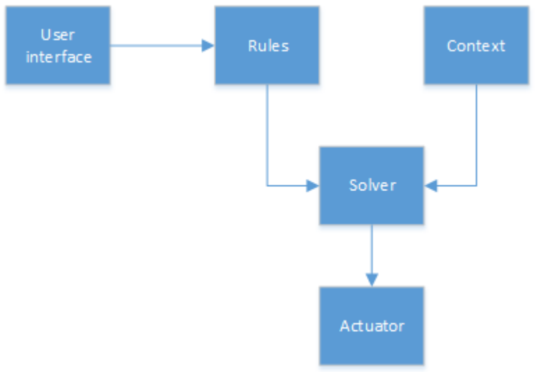


Fig. 2. Schema of a rule-based system. Rules are retrieved from a database and executed according to available sensor data.

### 3.2 Planning algorithms

Unlike the previous approach, planning algorithms [20, 14] do not use a simple 'if this then that' system, but instead react with a composition of multiple steps. A schema of this approach is given in Fig. 3. The planning is done to achieve a goal, which is either triggered by the user or through means of a different decision algorithm, such as a rule-based system. A model of the home and a descriptions of available devices is used to determine what actions are needed to achieve the goal. Like the previous example, when the user enters the house the goal of lighting the rooms will be triggered. When there is enough natural light outside, the planner will choose to open the curtains, since this is the cheapest option. However, when it is dark outside the only option will be to turn on the lights. The environment model is needed in order for the smart space to know what lights are located where and which rooms are connected to each other. So in order to light up the room, the lights in the room itself could be turned on, but perhaps also lights in adjacent rooms. The planning algorithm also checks if the created plan, the sequence of actions resulting in fulfilling the goal, is being accomplished. While executing a plan, actions are being taken by the actuators, which in turn change the context of the home. The planning system will get feedback from the context and notices if something goes wrong. It can then create a new plan to fulfill the goal through other means, given the new context created by the failed previous plan.

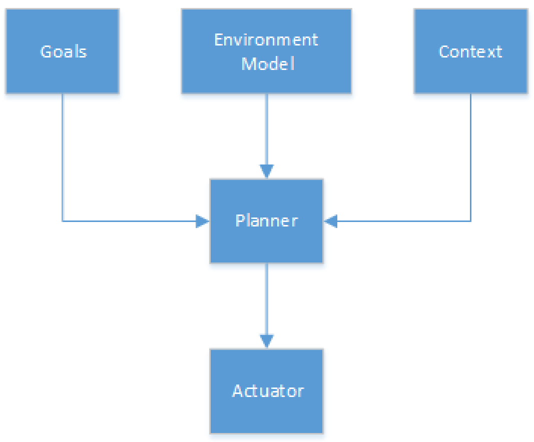


Fig. 3. The planner receives goals and tries to fulfill these goals by creating/planning a sequence of actions based on the model of the environment and the current context of the home

### 3.3 Learning algorithms

Unlike rule-based and planning approaches, learning algorithms [3, 11, 5] do not follow predefined rules created by a programmer or user. This approach is described schematically in Fig. 4. By monitoring the actions of a user, the system can detect patterns and automate certain actions. Following the above example, if the user turns on the light when they enter the home, the system can detect this and turn on the lights for them after a while. Even better, if after turning on the lights the user opens the curtains when it's light outside and then turns off the lights, the system would be able to optimize and open the curtains without having to turn on the lights at all. The learning algorithm must however be carefully designed to prevent unwanted behaviour. For example, consider a presentation, where during the presentation the user turns on the room lights to draw something on the board, and then switches the lights off again. The system may now infer that after a certain period of time during a presentation the lights should be turned on. And this is certainly not the intention of the user.

Learning algorithms can also provide safety to its inhabitants. For example, Jakkula [13] proposes a method using machine learning to detect anomalies. By using this anomaly detection method, the health of the inhabitants can be verified and whenever the system detects an anomaly and thinks something is wrong with a user, the system can interfere. This method provides safety for the inhabitants, but in an entirely different way than the rule-based and planning approaches. This method does not open windows in case of a gas leak like a rule or planning approach might do, but instead it focuses solely on situations and actions that are not normally taken. For example, it could detect a user has forgotten to turn off the stove and raise an alarm.

## 4 Well-known implementations

In order to see how these general methods operate, several well-known implementations of smart spaces will be discussed in which these methods are used. This provides insight into the current state of this field. The smart home implementations that will be discussed are:

- MavHome, using a learning-based approach
- Smart Homes for All (SM4ALL), using a planner linked with a rule-based approach

Each of these implementations make use of one or more of the methods of automation discussed in the previous chapter. This chapter looks at how these well-known implementations work and how each implementation tries to fulfil the needs of the users.

### 4.1 MavHome's approach

MavHome [5, 4, 6, 10] is a project by the University of Texas at Arlington. Its focus lies on maximizing comfort and productivity of the users while minimizing the operation costs. This project is discussed because it contains a lot of aspects that are common in learning based smart homes such as use of multiple agents and the use of several prediction algorithms.

MavHome is, as they call it, an agent-based smart home [5], consisting of a hierarchy of agents. Each agent has a specific role and contains all the layers of the general architecture of a smart home shown in Fig 1. The agents communicate information with other agents through their communication layer. The agent system is hierarchical, which means that communication layer might communicate with the decision layer of another agent. If the physical layer represents another agent, the actions that have to be executed are communicated with the decision layer of this other agent. This other agent can then again make its own decisions on how to handle this particular action. This way, a complex system can be divided into multiple less complex agents, thereby decreasing the complexity of the system and increasing maintainability and scalability.

While collecting and processing data, MavHome continually tries to thoroughly find out what is going on in the environment. Based on the actions and information that is recorded, it tries to identify which users are currently in the home. MavHome will then try to assist the user by performing the actions that it predicts the user will perform themselves. This way MavHome tries to automate the routine of the users. The decision layer decides what actions to take and tries to predict these actions by searching for patterns in the information layer. To make predictions, MavHome uses several different algorithms and a filter [5, 4, 6, 10]. These are:

- Smart Home Inhabitant Prediction (SHIP)
- Active LeZi (ALZ)
- Markov Model
- Episode discovery (ED)

#### 4.1.1 Smart Home Inhabitant Prediction (SHIP)

When a user performs an action (uses a device), this specific user-action is recorded and stored. Say the user performs action $A$. The home will then looks at the sequence of actions that were done by the user before they performed this specific action $A$. The history of actions of the specific user is analysed to find if part of this sequence of actions happened before, in the same state, and resulted in the same action $A$ being taken. When a previous sequence is found, it will increase the match count of this sequence which registers how many times this specific sequence resulted in this action. After recording the matches, the prediction algorithm processes all the matches found. It ranks the actions taken in a certain state based on the match-frequency and the length of the previous sequence of actions. Based on these action rankings it can take the action that has the highest ranking, if a specific sequence of actions occurs again in a specific state. The main disadvantage of this prediction algorithm is that the entire action his-

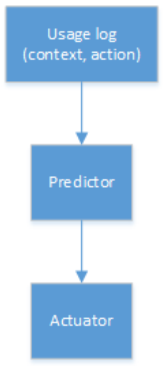


Fig. 4. Learning based algorithms predict what actions the user wants to perform using a history of previous usage and then executes them automatically.

tory must be stored. If this history becomes too big over time, looking for matches and processing them will take too long.

### 4.1.2 Active LeZi (ALZ)

The Active LeZi algorithm is based on data compression (LZ78 data compression). The general idea behind this algorithm is that good compression methods seem to act as good predictors as well. The algorithm works by first parsing the sequence of actions to a string, where each character represents an action. By using LZ78 to compress this string, the method calculates the probabilities of the character that would be next in the given string. And because the next character represents an action, compression methods can be used to predict next user actions.

### 4.1.3 Markov Model

A lot of context variables of the home are randomly changing, such as the weather. These randomly changing variables affect what actions are taken, either by the user or the smart-home. For example, on a warm sunny day the heating of the home may be reduced and the inhabitants may be more likely to be outside of the house. The actions that are taken here are only related to the current state of the environment, not the previous sequence of states. If it was a cold rainy day the day before, it won't have much influence on the actions taken on the sunny day. This means that actions in the smart home can be modelled according to the state the home is in at the moment the actions are taken. One representation that can be used for this kind of modelling is called a Markov Model. A Markov Model is a model of a system that has the 'Markov property', which means that the future state of the system only depends on the current state, not the past sequence of states.

MavHome collects all the actions performed in the home and in what context they were performed. To refine this Markov model, MavHome tries to identify tasks based on the actions performed. Actions might correspond to different tasks if the time and place of these actions differ significantly. Actions are first partitioned into groups that are most likely part of a single task. Then a k-means clustering algorithm is used to transform these different tasks into abstract classes of tasks. K-means clustering groups $N$-dimensional data (in this case the properties of each task's actions such as device and time of day) into clusters based on the similarity of their properties [19]. These abstract tasks can then be used to calculate probabilities of transactions between specific states.

### 4.1.4 Episode discovery (ED)

Episode discovery (ED) is focused on identifying 'significant episodes'. This algorithm looks at the action history of a certain user and tries to find related events. The key difference with the other algorithms is that these events do not have to be ordered. The related events can also be partially ordered or even unordered. Episode discovery does not make predictions on its own, but it is used to filter the history of events and actions. The prediction algorithms discussed earlier can use this filtered set of significant episodes to try to predict the actions, rather than using the direct history of the actions itself. The MavHome project found that using ED as a filter significantly increases the prediction accuracy.

## 4.2 Smart Homes for All's approach

The Smart Homes for All (SM4ALL) project [1] is a large smart home project in Europe. Many different people and universities from all across Europe contribute to this project. This project focuses on the comfort and safety of the inhabitants of the smart home. This project uses a planning approach together with a rule engine. Kaldeli et al. [14] discuss the architecture of this smart home. The goals for the architecture is to make it highly interactive and adaptive so it can be used in multiple houses and for multiple types of users. The architecture consists of three layers, as shown in Fig. 5.

The user layer consists of the devices and ways for users to interact with the system and to instruct the home. For example a smart phone, tablet or computer. The composition layer contains the components

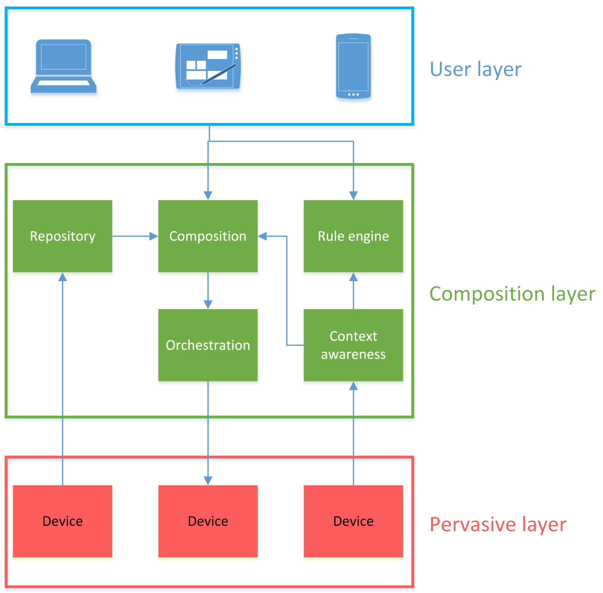


Fig. 5. Layers of the SM4ALL architecture

that will make the smart decisions in the home. This layer provides the intelligence of the home and takes all the necessary actions through means of the other layers. The pervasive layer is used to connect with devices in the home. The universal plug and play (UPnP) protocol [2] is used to provide a dynamic way to connect to the devices. The functionality the connected device has to offer is exposed using the OSGi platform, which is a Java platform that allows the creation of complex systems through composition of swappable components. The pervasive layer provides a way for the composition layer to utilize the functions the connected devices have to offer.

The service composition layer is the layer that actually does all the 'smart' thinking to control the elements present in the other layers. It is based on CSP solving [14]. The layer contains several different components, including a rule engine, a repository, a composition module, context awareness and and orchestration. The rule engine contains rules that will generate goals (instead of actions) based on the context, as is previously discussed in chapter 3.1. The goals that are generated by this rule engine are passed to the composition module, which acts as the planner. The planner receives high-level goals either from the user or the rule engine and makes a plan on how to fulfil these goals. To do this, the planner uses the repository, which contains the information of all the active devices that can be used. The planner generates a sequence of actions to be executed and passes this to the orchestrator, which executes the necessary low-level service-actions needed.

The planner is a smart component and searches for the different ways to fulfil the same goal. When something goes wrong while executing a goal, the planner will notice by either a timeout during the realization of the goal or through a notification from the context awareness. The planner will then create a different composition of actions that will realize the same goal. Upon replanning for the goal, the planner will take into account the new context that is created by the execution of the failed attempt to reach the goal. Also, whenever something is wrong and the planner does not receive all the data it needs, such as specific sensor information, it can still plan to fulfil the goal.

The entire home is modelled as a CSP, so when a specific context attribute changes, a constraint is added to the constraint satisfaction problem that reflects this new state. This way, whenever a goal is received by the planner, it does not have to load the entire context of the home.

## 5 Smart Grid

An important aspect of smart homes is the possible integration with smart power grids. The SmartHouse/SmartGrid project [15, 17, 16] focuses on the energy management and efficiency of the smart homes. It adds an energy utility to the smart-home environment that can be used to minimize the energy consumption.

This addition of the energy management utility to the smart home tries to help regulate the energy consumption. It measures the energy consumption of the devices in the home and gives information about energy costs and usage. The home tries to make their users aware of their energy usage and tries to let them help reduce energy consumption by advising them. To effectively add these energy features to the home, it is necessary that the system can automatically identify home appliances. A standard model is used to automatically register new devices and services.

The energy the home uses does not only come from an energy service company. The energy can also come from a distributed energy resource (DER). This distributed energy resource consists of relatively small energy generating devices, such as windmills or solar panels, which might provide energy to the smart home. Using these resources can lower the cost of energy. The smart home tries to predict how much energy the different resources within the DER will generate. The home also knows the market prices from power companies. Using this information about the cost of energy, a smart home can perform the high energy consuming tasks during a time where the cost (and load) is the lowest.

Smart homes can communicate and interact with each other in order to manage the energy consumption as a group. They tell each other how much energy they consume and how much they can produce. This way a smart grid is created. It might also happen, for example, that the energy network does not have the capacity to provide a sudden high demand of energy an area. The homes/devices can make a plan together to spread out their energy consumption to lower the energy load. The architecture of the smart home is shown in Fig. 6.

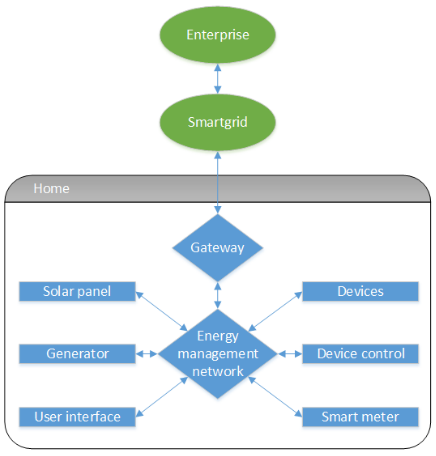


Fig. 6. The SH/SG home architecture

The home contains devices that are used by the inhabitants and consume energy. The device controller influences these devices and can try to save energy by, for example, switching a device off. The home may also be able to provide energy. In the figure, the home has a solar panel and a backup generator. This information is all communicated to the smart grid by using the gateway in the home. The grid is connected to several different energy producers and consumers, of which the smart home is one, and tries to balance their energy. The solar panel of the home might produce enough energy to provide to other smart homes in the smart grid. And in case of a blackout, the backup generator's energy might be sold to other people.

## 6 Results

The major benefit to rule-based approaches is that the system can be perfectly tuned to the needs of the user. The downside of this is that these rules have to be carefully designed, which requires a lot of work from a programmer or user.

Planning algorithms do not respond in a predefined way, but compose actions to fulfil a goal. However, this requires a precise model of the environment and a description of the effects of each action. It also still requires a programmer to create rules to trigger these goals. One advantage over rule-based systems is that it can be more fault tolerant, as it can choose to perform similar actions when some devices malfunction.

Learning algorithms are by far the most flexible, as little initial setup is required during installation, since the system learns what to do over time. The downside is that the system requires a training phase. That is, it is not actually a smart space until some time after deployment. This can be mitigated by loading custom learning data to define some behaviour in advance. However, this naturally requires more work from a programmer. Another downside is that it may pick up unwanted behaviour, where the user must act as a moderator to disable those actions.

The discussed methods can also be related to the user expectations as given in chapter 2. Rule-based and planning algorithms are both suitable for security and safety purposes, since they can be programmed to respond to threats in a certain way. Since the dynamic response of planning algorithms can be more fault tolerant, it would be the best choice for security and safety purposes. However, the learning approach can most easily be extended to detect anomalous behaviour from users. This could be used to automatically detect failures in the hardware of the system, but also to detect medical emergencies [13]. In this way it provides a different kind of safety.

Rule-based and planning algorithms are best suited for (remote) control, since they can more easily be given a simple user interface. Learning-based systems may be too abstract to directly add a user interface, since the underlying rules are created dynamically.

Convenience-wise all approaches can reduce the number of actions the user has to perform by the same amount. The only difference is how these actions are triggered (programmed versus learned). The same goes for energy management. In all approaches the energy consumption of the house can be reduced by making smart decisions such as opening window curtains instead of turning on lights.

Rule-based systems are simplest to produce, since they are relatively uncomplicated and readily available. While the sensors and actuators add complexity to the system, these are also common factors for the other methods. The planning method requires more components such as the environment model. Learning algorithms are most complex, since they require large scale analysis of data.

On the other hand, learning algorithms are most easily integrated because once in place, the system will automatically develop its behaviour, although this may take some time. Rule-based and planning approaches work immediately after installation, however they require a lot of effort in order to create rules or goals for the system. At the same time, these approaches can more easily be adjusted by adding, disabling or modifying the rule database, where learning based systems would require some sort of complex interface in order to adjust the underlying rule model.

These results have been summarized in table 1. Per property, points were divided among the discussed methods, following the above descriptions. The higher the number of dots, the more suited the method is for that property. This table is only meant as a convenience for the user and the rating is therefore only meant as an indication and not as a guide when evaluating automation methods.
Looking at the several different implementations, all focus on convenience while having a flexible way of connecting and discovering new devices. The way these implementations organize their architecture in order to fulfil these goals is very different. MavHome has no user-interaction and tries to fulfil the user-needs by predicting what the user wants based on their previous actions. This is very different from the SM4ALL approach which tries to best fulfil the user-needs by execut-

Table 1. Rating of the properties per automation method

| | Rule based | Planning | Learning |
|---|---|---|---|
| Safety | ●● | ●●● | ●●● |
| (Remote) Control | ●●● | ●● | ● |
| Convenience | ●● | ●● | ●● |
| Energy Management | ●● | ●● | ●● |
| Production | ●●● | ●● | ● |
| Integration | ●● | ●● | ●●● |
| Adjustability | ●●● | ●●● | ● |

ing the high-level goals that are issued. This SM4ALL approach is based purely on user-interaction with the planning system or with the rule engine. The way to connect and interact with the devices differs a lot too. MavHome tries to do this as flexible as possible by using a hierarchy of agents that can reason about the context and execute actions on the services. In the SM4ALL project this is done very differently. The connected devices send their state to a single repository which contains all device information and fires events based on status changes.

The SH/SG project is purely focused on energy management and tries to communicate with other smart homes in order to share and save energy. SM4ALL and MavHome show that rule-based, planning and learning based systems can successfully be used in the creation of a smart space.

## 7 Discussion

The results do not show a clear winner between the rule-based, planning and learning approaches as far as the discussed properties are concerned. Each approach has certain drawbacks and benefits. Where the rigidity of rule-based and planning approaches allow for precisely tuned usage, this is exactly the reason why integration requires a lot more effort. Therefore when building a smart home or smart space the developers must choose the most appropriate method on a case-by-case basis.

Looking at safety, the use of a smart space might greatly increase safety but might also introduce new risks. Many smart spaces are connected to the internet so the users can interact with the smart space from outside the space. This means, however, that this system can also be hacked, in which case hackers have control over the building and might have the ability to even start a fire. What can also happen is that the users make a mistake while adding a new rule which could do harm or cause a disaster.

## 8 Conclusion

In this paper we have discussed a number of expectations users have regarding smart spaces and smart homes. We described several general techniques, as well as some notable smart space implementations. We have looked at their architecture and described their components. We then compared the different techniques according to the user expectations described earlier.

Each of the well-known implementations of smart spaces focuses on a different type of user-need. None of the implementations effectively focus on all four most important user needs.

Our approach only looks at theoretical benefits of the discussed methods. In future work, it would be beneficial to see these methods applied to real environments in a basic form. This is different from the well-known implementations discussed earlier which are much more complicated and so do not provide accurate information with respect to the properties compared in this paper.

It would also be great to see an architecture which integrates a combination of the methods. This could be very difficult to create, because of the fundamentally different way these methods decide what action to take. For example, if a learning algorithm were to be combined with a constraint satisfaction problem (CSP) system (either for a rule-based or planning approach), then the learning system tries to learn from the events in the given context, while the CSP system constantly influences this context by solving this CSP. The learning algorithm will then be very limited in it's decision-making because when it predicts some user actions, the action might not be executed because of the constraints given to the context. Meaning that these two algorithms can work against each other. It might be possible to include a very simple rule based approach with a learning algorithm method like MavHome to enhance the security a little. For example by adding a very simple rule that if there is a gas leak detected, the windows should be opened.

We are also very interested in a cloud based approach. So that if you visit another smart space that is not yours, the space can automatically adjust to your preferences. For example if you go on a holiday and visit your holiday home, the holiday home automatically adjusts to your preferences and adjusts the light and heating to suit you. Of course the same idea could be applied to office spaces.

## References

[1] Smart homes for all. http://www.sm4all-project.eu/. Accessed: 2015-02-07.

[2] Universal plug and play. http://www.upnp.org/. Accessed: 2015-02-07.

[3] C. M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[4] D. J. Cook. Identifying tasks and predicting action in smart homes using unlabeled data. 2003.

[5] D. J. Cook, M. Youngblood, E. O. Heierman, K. Gopalratnam, S. Rao, A. Litvin, and F. Khawaja. Mavhome: An agent-based smart home. 2003.

[6] S. K. Das, D. J. Cook, A. Bhattacharya, E. O. Heierman, and T.-Y. Lin. The role of prediction algorithms in the mavhome smart home architecture. 2002.

[7] V. Degeler and A. Lazovik. Dynamic constraint reasoning in smart environments. In *IEEE International Conference on Tools with Artificial Intelligence*, 2013.

[8] A. Dixit and A. Naik. Use of prediction algorithms in smart homes. 2014.

[9] Google Developers. Blockly faq. 2015. https://developers.google.com/blockly/about/faq.

[10] E. O. Heierman and D. J. Cook. Improving home automation by discovering regularly occuring device usage patterns. 2003.

[11] M. Huber. Automated decision making. In *Smart Environments: Technologies, Protocols, and Applications*, chapter 10. 2005.

[12] icontrol. State of the smart home. 2014. http://www.icontrol.com/insights/2014-state-smart-home/.

[13] V. Jakkula, D. J. Cook, et al. *Anomaly detection using temporal data mining in a smart home environment*, volume 47, pages 70–75. Stuttgart [etc.] FK Schattauer [etc.], 2008.

[14] E. Kaldeli, E. U. Warriach, A. Lazovik, and M. Aiello. Coordinating the web of services for a smart home. *TWEB*, 7(2):10, 2013.

[15] K. Kok, C. Warmer, D. Nestle, G. Heusel, J. Ringelstein, P. Selzam, H. Waldschmidt, A. Weidlich, S. Karnouskos, A. Dimeas, and S. Drenkard. Smarthouse/smartgrid in-house architecture and interface description. 2009.

[16] K. Kok, C. Warmer, D. Nestle, P. Selzam, A. Weidlich, S. Karnouskos, A. Dimeas, and S. Drenkard. Smarthouse/smartgrid coordination algorithm and architecture. 2009.

[17] K. Kok, C. Warmer, G. Venekamp, A. Weidlich, S. Karnouskos, P. da Silva, D. Ilic, A. Dimeas, J. Ringelstein, S. Drenkard, and V. Liolou. Final architecture for smarthouse/smartgrid. 2011.

[18] Y. Lu, G. Li, Z. Jin, X. Xing, and Y. Hao. A knowware based infrastructure for rule based control systems in smart spaces. 2013.

[19] J. MacQueen. Some methods for classification and analysis of multivariate observations, 1967.

[20] D. Nau, M. Ghallab, and P. Traverso. *Automated Planning: Theory & Practice*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2004.

[21] M. Pragnell, L. Spence, and R. Moore. *The market potential for Smart Homes*. YPS for the Joseph Rowntree Foundation York, 2000. http://cisyr4backupsofeverything.googlecode.com/svn/trunk/CM30082-DissertationReadings/2.)Journals/Pragnell,Spence,Moore-TheMarketPotentialforSmartHomes.pdf.

# Comparing Adaptive Gradient Descent Learning Rate Methods

Diederik Greveling and Michael LeKander

**Abstract**—In machine learning, learning algorithms are often trained by minimizing a given cost function using function optimization algorithms. Gradient descent algorithms are a popular family of function optimization algorithms which utilize calculus-based methods. There are two leading classes of gradient descent algorithms: batch gradient descent (BGD), which considers every example at each time step, and stocastic gradient descent (SGD), which only looks at one example at a time.

Most optimization algorithms, including BGD and SGD, require multiple *hyperparameters* that are not immediately obvious from the data. In practice, this requires researchers to manually tune these hyperparameters depending on the dataset and desired results. Recent proposals have suggested modifications to the BGD and SGD algorithms aiming to improve performance and reduce their dependance upon hyperparameters.

In this paper, we investigate the waypoint averaging and vSGD algorithms in addition to standard BGD and standard SGD. We evaluate each of these algorithms with respect to their respective convergence properties, number of hyperparameters, and resulting final cost values. Empirical results on a large dataset of 25010 10-dimensional samples demonstrate that waypoint averaging and vSGD both reach better local optima faster than the standard algorithms. In addition, we find that these variations are less influenced by their hyperparameters, thus requiring less manual tuning.

**Index Terms**—Machine learning, optimization, gradient descent, learning rate, hyperparameters.

## 1 Introduction

One of the primary goals of machine learning is to notice patterns in example data, and then recognize these patterns in novel data. For example, we can classify different types of fruit based on their colour and size. Given a set of fruits, the learning algorithm can learn that bananas are typically yellow and that their size is usually larger than an apple. The algorithm is now able to "recognize" whether an unknown fruit is a banana or an apple.

Learning algorithms typically undergo a training process which attempts to find a set of internal parameters, $w$, which minimize an objective cost function, $E(w)$. This cost function gives a numeric value of how "good" a set of parameters performs over a given dataset, with lower costs preferred over high costs.

The cost function and internal parameters vary between different learning algorithms. For example, Generalized Learning Vector Quantization's (GLVQ) internal parameters represent the coordinates of various *prototypes* that estimate the centers of data clusters. GLVQ's cost function measures how close these prototypes are to the "correct" data points. GLVQ is discussed in more detail in Section 2.1.

Often, the cost function given by learning algorithms are differentiable, which allows the use of calculus-based methods to approximate a minimum $E(w)$ value. One popular technique is known as gradient descent optimization, which is one of the most-used techniques for large-scale learning problems[2].

At every time step $t$, gradient descent optimization calculates the gradient of the cost function, $\nabla E(w_t)$, which gives a vector pointing in the direction of the steepest cost function change at the current value of $w_t$. By constantly following this gradient vector, the procedure will theoretically always locate a local minimum value, although perhaps not the global minimum. When gradient descent reaches a minimum value, or equivalently when it reaches a value with a zero gradient, it is said to have converged. However, implementation details may prevent convergence from occurring, as explained in Section 4.

There are multiple variants of gradient descent optimization. However, most gradient descent variations can be grouped into one of two families: batch gradient descent, or stochastic gradient descent. Batch gradient descent (BGD) computes the gradient by considering every element in the dataset and then performing an update step. In contrast, stochastic gradient descent (SGD) approximates the gradient by only considering a single element at a time, immediately performing an update step after each element. Details about BGD and SGD can be found in Sections 4.1.1 and 4.2.1, respectively.

Most gradient descent methods require *hyperparameters* that influence their convergence properties, but that cannot be easily determined from the dataset. Hyperparameters are optimization parameters that are not automatically tuned by the learning algorithm, and must be set manually during initialization. In practice, these parameters are tuned by manually searching over various hyperparameter configurations until a good result is found. This manual tuning is costly, since often multiple full runs are needed in order to find optimal hyperparameter values. In addition, since these parameters are data-dependent, they must be recalibrated for every dataset.

One common gradient descent hyperparameter is the learning rate, $\alpha$, which determines how far to step in the direction of the gradient. For both BGD and SGD, the learning rate heavily influences how well the algorithm is able to minimize the cost function. If the learning rate is too high, then the algorithm will not converge. If it is set too low, then the algorithm will take a very long time to converge, and may easily get stuck in highly suboptimal local minima.

There are multiple methods proposed for adaptive learning rate control in order to improve finding an optimal solution. Some methods propose a learning rate schedule where the learning rate decreases with respect to time[1]. Other methods used higher order derivatives to approximate an "optimal" learning rate using recently encountered data[12].

In this paper we research methods proposed by various authors which each attempt to improve the standard SGD and BGD algorithms by adapting the learning rate during the course of the algorithm. The following gradient descent methods will be compared:

- standard BGD
- standard SGD
- variance-based SGD (vSGD) [12]
- BGD with waypoint averaging (WA-BGD) [6]
- SGD with waypoint averaging (WA-SGD, a method we propose).

- *Diederik Greveling is a MSc student, University of Groningen E-mail: diederikgreveling@gmail.com*
- *Michael LeKander is a MSc student, University of Groningen E-mail: m.l.lekander@gmail.com*

We carry out this analysis over the poker hand dataset, supplied by the UCI machine learning repository [4]. The poker dataset contains 25010 examples of possible poker hands, mapping 10 dimensional input (the suit and rank of each of the five cards) to ten classes (the type of poker hand; straight, three of a kind, full house, etc.).

The remainder of this paper is organized as follows: Section 2 describes the learning algorithm used. In Section 3 we describe the different criteria needed in order to compare the performance of the different optimization algorithms. Section 4 describes the various gradient descent methods we explore in this paper. We present our results in Section 5, and discuss these results in Section 6. Finally, Section 7 concludes the paper.

## 2 Learning Methods

One important task in machine learning is supervised learning, whereby an algorithm classifies data into one of several classes. This is done by generalizing an unkown rule from a set of training examples with pre-determined classifications. Various learning algorithms are used to accomplish this task. Some popular supervised learning algorithms include multilayer neural networks and learning vector quantization.

Often these learning algorithms require a training phase to “learn” the data by setting internal parameters. This training phase can be formalized as minimizing an objective cost function that predicts how well a set of parameters will perform against novel data. General optimization methods are often used during this phase to find learning algorithm parameters that minimize this cost function.

### 2.1 Generalized Learning Vector Quantization

In this paper, we use the generalized learning vector quantization (GLVQ) classification algorithm proposed by Sato et al. [11]. For a $n$-dimensional dataset, the GLVQ algorithm maintains $P$ different $n$-dimensional *prototypes*, each of which represent a cluster of data points from the same class. Each prototype is associated with a specific class. Once the prototypes have been trained, the algorithm classifies unknown $n$-dimensional inputs by finding the closest prototype to the input, and then returning the class of that closest prototype.

The goal of GLVQ training is to position each of these $P$ prototypes in the $n$-dimensional space such that each prototype is as close as possible to examples from its own class, while remaining as far as possible from examples from different classes.

The GLVQ function was chosen for this paper because it has a relatively simple and differentiable cost function,

$$E(w) = \sum e_i(w) = \sum \frac{d(i,p_{ij}) - d(i,p_{ik})}{d(i,p_{ij}) + d(i,p_{ik})}, \tag{1}$$

where $e_i(w)$ is the per-sample cost function, $d$ is the Euclidean distance function, $p_{ij}$ is the prototype closest to $i$ with the same class as $i$, and $p_{ik}$ is the prototype closest to $i$ of a different class than $i$.

The GLVQ cost function can produce negative values. This simply indicates that $d(i,p_{ij})$, the distance from a sample to its closest correct prototype, is smaller on average than $d(i,p_{ik})$, the distance to its closest incorrect prototype.

Since the GLVQ cost function is affected by multiple prototypes for each sample, prototypes cannot be trained separately. Thus, the optimization algorithm used to train GLVQ can be viewed as optimizing over a $P \times n$-dimensional space.

## 3 Optimization Algorithm Criteria

The following list of criteria enables us to make objective distinctions between the different algorithms. These criteria are based on how fast the algorithms converge and how many times hyperparmeters need to be set in order to find the optimal values for these parameters.

- *Influence of hyperparameters* - As mentioned before, the tuning of the different gradient descent methods is computationally costly since often multiple runs are needed to locate and verify optimal settings. Hyperparameters influence the final cost function value to different degrees for different algorithms. Since finding optimal hyperparameter settings is computationally expensive, algorithms with more lenient hyperparameters (even when not optimally tuned) are preferred.
- *Number of epochs before converging* - The goal of every optimization algorithm is to find a local optima in the least amount of epochs. We measure this by the number of epochs (full iterations through the data) the algorithm requires before it converges. The number of epochs plays a key role in how efficient the training process will be.
- *Final cost function value* - The cost function tells us how well the learning algorithm is expected to perform when presented with novel data. This is particularly useful since we can then make a direct comparison between the different algorithms in terms of how efficient the algorithms are, particularly when using identical initializations.

## 4 Gradient Descent

Since many learning algorithms produce cost functions that are differentiable, calculus-based methods can be used during the learning algorithm training phase. Gradient descent begins by starting from a given initial location. Taking the gradient (derivative with respect to every dimension) of the cost function at a location produces a vector in the direction of steepest change. Gradient descent then takes a step in this direction, moving in the direction following the gradient. However, how far to follow this gradient depends on the method used.

The choice of initial location plays an important role in the convergence of gradient descent methods. Different initial locations may converge to different local minima, and a “bad” initial location may cause an algorithm to never converge. The effect of different initializations is illustrated in Fig. 1.

In order for the gradient descent methods to converge, they run through each sample in the dataset multiple times. If the algorithm has gone through the whole dataset, i.e. every sample is used at least once, then the algorithm is said to have completed one *epoch*. The number of epochs required until convergence can be used as a measure for the run time of the algorithm.

It is often the case that an algorithm will come to a relatively good solution, even if it never fully converges. A technique known as *early stopping* terminates the algorithm $t_{\max}$ epochs even if the optimization parameters are not fully stable. Although early stopping is known to have some beneficial effects with avoiding over-fitting, it is often used to simply to reduce the required computation, as it effectively puts an upper bound on the number of epochs required by the algorithm[7].

### 4.1 Batch Gradient Descent Methods

In this section we describe the different batch gradient descent methods we used for our research. We first describe standard BGD, fol-

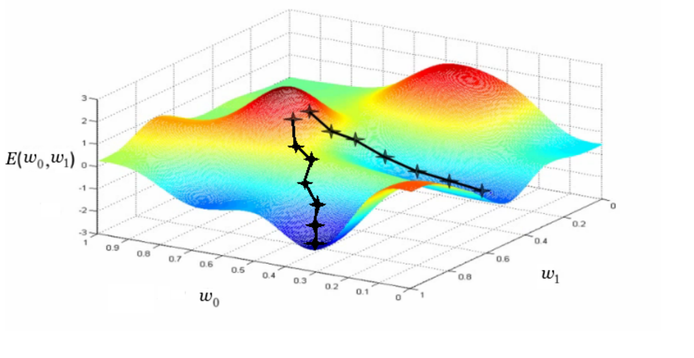


Fig. 1: Gradients descent steps starting at two different initializations (adapted from Andrew Y. Ng Stanford CS229 Machine Learning slides[5]).

lowed by a description of BGD with waypoint averaging.

#### 4.1.1 Standard BGD

In batch gradient descent, the gradient is accumulated over an entire epoch before performing an update step. This is in contrast with SGD, where the weight changes are applied directly (see Sec.4.2.1). BGD is thus able to calculate the "true" gradient direction for the dataset. However, it does not know how far it can safely go in the direction of the gradient before going back "uphill" again. This especially poses a problem when the dataset is large because the weight changes will in turn become large which could result in overshooting local optima. The learning rate can be reduced to solve this problem however this means that more computation is required for a weight change of the same magnitude.

The BGD update step is described by

$$w_{t+1} := w_t - \alpha \sum_{i=1}^{n} \nabla e_i(w_t), \tag{2}$$

where $t$ is the current epoch, $w_t$ is the vector of optimization parameter values at epoch $t$, and $i$ is the index of a dataset sample.

Some authors often imply that BGD is theoratically superior to SGD due to the fact the BGD follows the gradient more closely[8]. Some authors also state that BGD is just as fast or faster than SGD [3]. According to Wilson et al., SGD is often faster than BGD and is usually a better approximation of true gradient descent optimization[14].

#### 4.1.2 BGD with Waypoint Averaging

Papari et al. introduced a variation of BGD known as BGD with Waypoint Averaging (WA-BGD) that addresses the problem of continual overshooting[6]. They note that if the algorithm continually overshoots the optimal parameters due to the learning rate being too large, then it might be better to take the average over a history of the most recent parameter values (the *waypoint average*) and continue from there. If we continue from this waypoint average, then the learning rate should be reduced to avoid further continual overshooting.

The WA-BGD method is an attempt to extend BGD in a computationally efficient way while being easy to implement. Another benefit is that WA-BGD does not require careful tuning of a large set of algorithm parameters, compared to some other methods. It also does not require any learning rate schedules or higher derivatives to adapt the learning rate.

After a set of initialization epochs, both the waypoint average and gradient descent step is calculated at each subsequent epoch. The configuration that produces a lower cost function value is then chosen as the next set of optimization parameters, $w_{t+1}$. A normalized gradient is used to overcome plateau states and flat regions of the cost function[6].

WA-BGD introduces two additional hyperparameters:

- $k$, the number of epochs over which the waypoint average is calculated and
- $r < 1$, which determines how much the learning rate is reduced if the waypoint average is better.

For the first $k$ epochs, WA-BGD performs similarly to WA-SGD, since the history is not large enough to perform a waypoint average calculation, i.e.

$$w_{t+1} := w_t - \alpha_0 \frac{\nabla E_t}{\|\nabla E_t\|} \text{ for } t = 0,1,2,3,\ldots,k-1. \tag{3}$$

Note, however, that a normalized gradient is used.

After the first $k$ epochs, the following steps are performed every epoch:

1. A normalized gradient descent update is performed:

$$\widetilde{w}_{t+1} := w_t - \alpha_t \frac{\nabla E(w_t)}{\|\nabla E(w_t)\|}. \tag{4}$$

2. The waypoint average over the previous $k$ optimization parameters is calculated:

$$\widehat{w}_{t+1} := \frac{1}{k} \sum_{i=0}^{k-1} w_{t-i}. \tag{5}$$

3. The cost function is calculated at $\widetilde{w}_{t+1}$ and $\widehat{w}_{t+1}$.

4. Lastly, the new position and step size is determined based on the cost function value:

$$w_{t+1} := \begin{cases} \widetilde{w}_{t+1} & \text{if } E(\widetilde{w}_{t+1}) \le E(\widehat{w}_{t+1}) \\ \widehat{w}_{t+1} & \text{otherwise,} \end{cases} \tag{6}$$

$$\alpha_{t+1} := \begin{cases} \alpha_t & \text{if } E(\widetilde{w}_{t+1}) \le E(\widehat{w}_{t+1}) \\ r \times \alpha_t & \text{otherwise.} \end{cases} \tag{7}$$

### 4.2 Stochastic Gradient Descent Methods

In this section we discuss the different methods based on stochastic gradient descent. We first describe standard SGD, followed by a description of vSGD method. Finally, we introduce the waypoint averaging for SGD method.

#### 4.2.1 Standard SGD

Standard SGD (also known as the Robbins Monro procedure[10] in other contexts) approximates the actual gradient by only looking at the gradient for a single sample at a time. This sample-specific approximated gradient is called the *local gradient*. SGD updates the optimization parameters after each sample from the dataset.

While local gradients can sometimes contradict each other, overall SGD does approximately follow the true gradient direction. The noise generated by contradictory local gradients actually has a beneficial effect, as it may allow escaping some local minima[6]. Another benefit of SGD is that it can handle large training datasets, since local gradients can guide the optimization parameters within an epoch.

The SGD update step is described by

$$w_{t+1} := w_t - \alpha \nabla e_i(w_t), \tag{8}$$

where $t$ is the current time step, $\alpha$ is the learning rate, $i$ is the index of a random dataset sample, and $\nabla e_i$ is the local gradient for sample $i$.

SGD is said to have completed one epoch once it has performed $n$ update steps, where $n$ is the number of samples in the dataset.

#### 4.2.2 vSGD

Variance-based SGD estimates an optimal learning rate by keeping a history of the previous gradient, gradient variance, and higher-order derivative, eliminating the need to manually tune a learning rate [12].

Theoretically, an optimal adaptive learning rate could be determined by calculating the expected gradient and the gradient variance (under certain assumptions, details may be found in [12]). Unfortunately, in practice it is not computationally feasible to compute these values for each sample of a given dataset.

However, it is possible to estimate these values by taking the average gradient, average gradient variance, and average diagonal elements of the cost function Hesse matrix (the second derivative of the cost function) elements. This results in a learning rate that is specific per optimization parameter.

Let $d$ represent the number of optimization parameters. Let $\overline{g}$, $\overline{v}$, and $\overline{h}$ be $d$-vectors representing the current estimated average gradient, squared gradient, and Hesse matrix diagonal, respectively. A $d$-dimensional vector $\tau$ represents the length of the "memory" of each optimization parameter. Let $h_i(w_t)$ refer to the diagonal Hesse elements of the per-sample cost function at $w_t$ for sample $i$.

The following steps are performed every epoch:

1. Similar to standard SGD, a random sample is picked from the dataset at every time step $t$. However, before $w_{t+1}$ is calculated, the $\overline{g}$, $\overline{v}$, $\overline{h}$, and $\tau$ vectors must first be updated as follows:

$$\overline{g}_{t+1} := (1-\tau_t^{-1})\overline{g}_t + \tau_t^{-1}\nabla E(w_t) \tag{9}$$

$$\overline{v}_{t+1} := (1-\tau_t^{-1})\overline{v}_t + \tau_t^{-1}(\nabla E(w_t))^2 \tag{10}$$

$$\overline{h}_{t+1} := (1-\tau_t^{-1})\overline{h}_t + \tau_t^{-1}h_i(w_t) \tag{11}$$

$$\tau_{t+1} := (1-\frac{\overline{g}_{t+1}^2}{\overline{v}_{t+1}})\tau_t + 1. \tag{12}$$

2. The optimal learning rate can then be estimated by

$$\alpha_{t+1} := \frac{\overline{g}_{t+1}^2}{\overline{h}_{t+1}\overline{v}_{t+1}^T}. \tag{13}$$

   This method does not require any initial value of $\alpha$, thus eliminating the learning rate as a hyperparameter.

3. Finally, each update step is performed identically to SGD but using the estimated optimal learning rate:

$$w_{t+1} := w_t - \alpha_{t+1}\nabla e_i(w_t). \tag{14}$$

#### 4.2.3 SGD with Waypoint Averaging

SGD with waypoint averaging (WA-SGD) is essentially the same as the WA-BGD counter part, except it utilizes SGD update steps instead of BGD ones. In WA-SGD, the waypoint averages are calculated after each epoch, that is, after $n$ SGD update steps.

WA-SGD gain the benefit of using noise to overcome some local optima while still using waypoint averaging for automatic step size control. It is possible that WA-SGD will converge faster then WA-BGD due to this added benefit. However we are not able to calculate the normalized gradient over an entire epoch, due to the nature of SGD.

This method retains the same hyperparameters as WA-BGD, namely $k$ and $r$.

If we convert WA-BGD to WA-SGD we have to adjust only 2 steps, namely the initialization step and the first step of every epoch. In the initialization step we changed the BGD function to a SGD one. This can be described by

$$\widetilde{w}_{t+1} := w_t - \alpha_t\nabla E(w_t) \text{ for } t = 0,1,2,3,\ldots,k-1. \tag{15}$$

The first step of every epoch is changed by adjusting it for SGD use:

$$\widetilde{w}_{t+1} := w_t - \alpha_t\nabla E(w_t) \text{ and } E(\widetilde{w}_{t+1}). \tag{16}$$

## 5 Results

In this section we discuss the different results produced by the algorithms. We have implemented the discussed gradient descent algorithms specifically in order to compare their results in this paper.

Determining optimal initial optimization parameter values is beyond the scope of this paper. Thus, we use identical initialization for each method when obtaining our results, for fair comparisons between methods.

In addition, the random number generated was seeded with the same initial value at the start of each run. This ensures that the sample presentation order is the same across the stochastic methods.

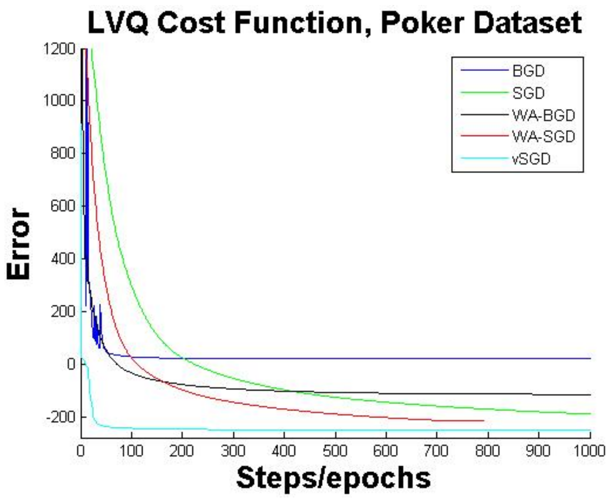


Fig. 2: LVQ Poker cost function, epochs 0-1000

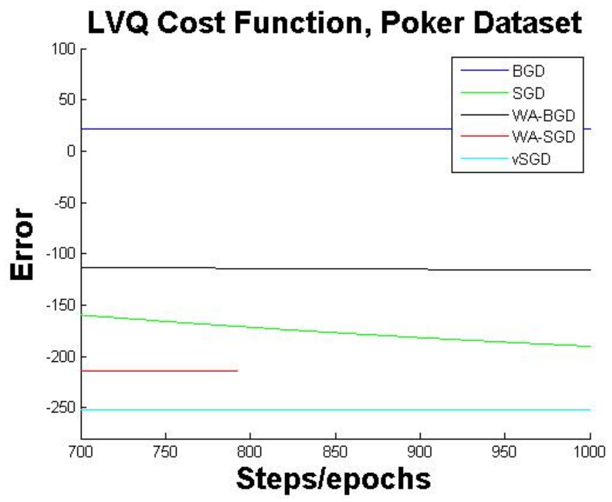


Fig. 3: LVQ Poker cost function, epochs 700-1000

### 5.1 Poker Dataset

The poker hand dataset, supplied by the UCI machine learning repository[4], holds 25010 records of possible poker hands. Each record (feature) consists out of five playing cards from a deck of 52 cards. Each card has two attributes respectively, suit and rank, so one record holds 10 attributes, resulting in a 10-dimensional dataset. Every hand holds a label for a total of 10 different type of hands in the game of poker; for example, a “three of a kind” or a “full house”.

We initialized 10 prototypes, 1 prototype for each class. Every algorithm ran for $t_{\max} = 1000$ epochs. Figure 2 displays the cost function value of each method from epoch 0 to 1000. The same learning rate was chosen for BGD and WA-BGD, as well as SGD and WA-SGD.

Figure 3 displays the final 300 epochs, more clearly distinguishing between the various final results. WA-SGD converged after epoch 793 while the other algorithms ran the full 1000 epochs.

### 5.2 Poker Dataset subset

To explore the influence of hyperparameters on optimization performance, we tested the methods under various hyperparameter settings. Due to the large computational costs to accomplish this, we used a subset of the poker hand dataset containing 2200 samples such that the number of samples from each class is proportional to the full training set.

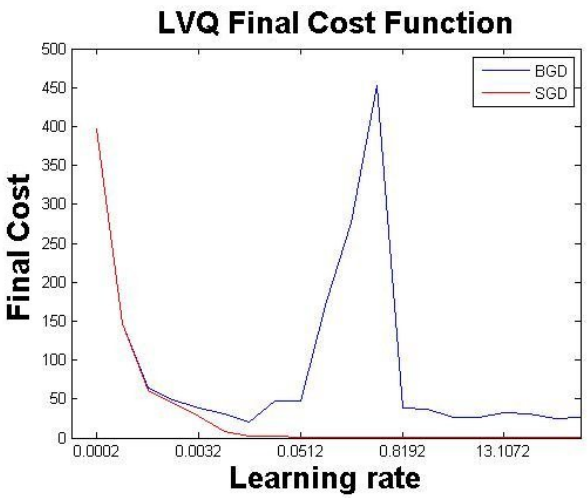
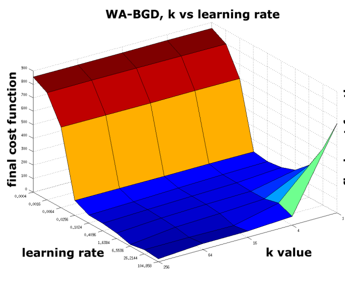
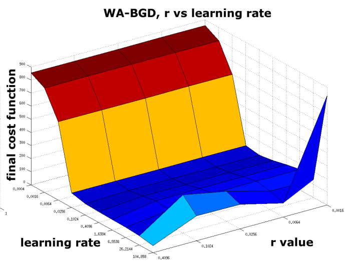


Fig. 4: Final LVQ Poker cost function value for all learning rates

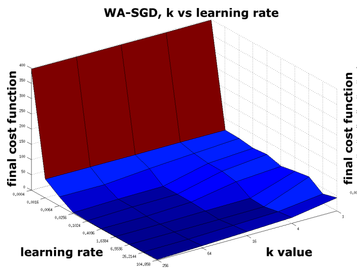


(a) Comparing various $k$ vs $\alpha$

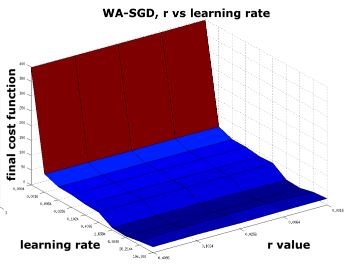


(b) Comparing various $r$ vs $\alpha$

Fig. 5: Hyperparameter influence, WA-BGD



(a) Comparing various $k$ vs $\alpha$



(b) Comparing various $r$ vs $\alpha$

Fig. 6: Hyperparameter influence, WA-SGD

We exhaustively searched over various hyperparameter combinations for each optimization method, again using identical initial prototypes for each algorithm. As before, we used 10 prototypes, and each algorithm ran for $t_{\max} = 1000$ epochs.

For SGD and BGD, we searched over learning rate values equal to $0.0001 * 2^p$, where $p$ ranged from 1 to 20. The plot of the final cost function values for these various learning rates can be found in figure 4.

For WA-BGD and WA-SGD, we looked at the combined influence of various values for $k$ and $\alpha$ while holding $r$ constant, as well as various values of $r$ and $\alpha$ while holding $k$ constant. The results for WA-BGD and WA-SGD can be found in figures 5 and 6.

Since vSGD has no hyperparameters, it was not tested on this dataset.

## 6 Discussion

In this section we discuss the results presented in 5 in the context of the criteria discussed in section 3.

### 6.1 Poker Dataset

No algorithm converged before 750 epochs on the full poker dataset, and only WA-SGD converged before $t_{\max} = 1000$ epochs.

Both BGD algorithms (WA-BGD and standard BGD) initially descended faster than either of WA-SGD or SGD (Fig. 2). However after 210 epochs standard SGD and WA-SGD both produce lower final LVQ cost function values than either standard BGD and WA-BGD. vSGD descended the fastest of all the algorithms.

BGD becomes stable after 90 epochs, with the highest final value. WA-SGD is relatively stable after 500 epochs. vSGD is the clear winner it becomes relatively stable after 100 epochs while WA-SGD converges and SGD does not become stable but keeps descending.

As seen in figure 3, WA-SGD converged after 793 epochs with the second-best final cost function value. Interestingly the standard SGD method is still descending and is still not stable after 1000 epochs. Because the learning rate is the same for SGD and WA-SGD and the samples are chosen in the same order we can state that WA-SGD greatly improves SGD. The same can be said for vSGD which outperformes all the other methods in terms of lowest final cost value.

WA-BGD and BGD have the same learning rate at the start, however WA-BGD performs better than BGD. WA-BGD does not perform better when compared to the SGD methods. It is surprising to see that WA-BGD initally decreased the faster than SGD and WA-SGD. Due to the nature of WA-BGD, we let the step size be very high, allowing it to make large jumps in initial epochs. Once suboptimal gradient steps were taken, the step size drastically reduced, producing the near-constant slope.

Overall vSGD is the clear winner, with a minimum cost function value of $-252.50$ and fast stabilization at epoch 300. In comparison with the other algorithms, vSGD also has the highest descent in the first 50 epochs.

### 6.2 Hyperparameter influence

The search for optimal hyperparameters was even more computationally expensive than originally anticipated. Even when using the reduced dataset, it took over 48 hours to produce results. WA-SGD took the least amount of time to compute, as it was the method that converged before $t_{\max}$ epochs the most often. Both SGD and BGD always ran for the full $t_{\max}$ epochs over this dataset, regardless of the learning rate.

The results for standard SGD (fig. 4) is somewhat confusing, as the result does not become noticeably worse for the extremely large learning rates. This is perhaps due to the largest tested learning rate not being large enough to observe divergent behavior. Unfortunately the computational requirements prohibit easily testing for a larger range of learning rates.

The standard BGD results (fig. 4) are equally confusing; the cost function values to get worse until $\alpha = 0.4096$, as is expected. However, after this the final cost function drastically drops for larger learning rate values. We are not sure how to explain this observed “peak” in the final cost.

The results from WA-BGD (fig. 5) indicate that, while too-small learning rates cause the algorithm to not converge after 1000 epochs, larger $\alpha$ values still give good results. Presumably, the adaptive learning rate control of waypoint averaging is able to compensate in this case, which is why this result is better than that of standard BGD. The $k$ value (fig. 5a) produced slightly better results as it grew larger, but all values of $k$ greater than one still performed well. The $r$ value (fig. 5a) had a larger impact on the final cost function value, although $r = 0.4096$ seems to be a good heuristic.

For WA-SGD (fig. 6), choosing an optimal learning rate did not largely impact the final result, so long as it was relatively large. Large $k$ values (fig. 6a) again performed well, producing nearly identical results for $k > 16$ and $\alpha > 0.0256$. The $r$ value (fig. 6b) had no noticeable impact on the final cost function value at all, as seen by the nearly flat surface along the "r value" axis.

Overall, the SGD-based algorithms performed better than the respective BGD-based ones. Standard SGD and WA-SGD produced lower final cost function values than standard BGD and WA-BGD respectively, while both being less impacted by sub-optimal hyperparameter settings.

## 7 Conclusion

In this paper, we outline our results from implementing various gradient descent algorithms by comparing them against an ideal simple cluster dataset, as well as a real-world Poker dataset. We conducted a search over various hyperparameter values, and present the results of the training runs.

The waypoint averaging methods produced better initial descent properties than their non-waypoint counterparts for the high-dimensional poker dataset. There was a somewhat surprising result that our ad-hoc WA-SGD method outperforms standard SGD.

The learning rate plays a large role on the convergence properties of the standard BGD and SGD algorithms, as predicted. For WA-SGD and WA-BGD, the inital learning rate still played a role, although it was less impactful for larger values. Although WA-BGD and WA-SGD do introduce additional hyperparameters, these additional hyperparameters seem to be more tolerant to a wider range of learning rate values in comparison with the normal BGD and SGD methods.

vSGD has the best performance on the poker dataset. vSGD is able to find the lowest cost function value and has the steepest descent in the first 50 epochs. vSGD does not require any hyperparameters, which eliminates the need for manual tuning.

There is much room for further related research. This paper only considers the cost function given by the GLVQ learning algorithm. However, there are many other learning algorithms that have well-defined and differentiable cost functions, such as the multilayer neural network[3] or the relevance matrix LVQ (GMLVQ)[13]. These various learning algorithms may provide cost fuctions that produce different convergence properties, potentially impacting various gradient descent methods differently.

Similarly, this paper only considers the poker data set. The dataset used to train a learning algorithm can also drastically impact the convergence properties of the resulting cost function. Future research should apply the analysis carried out in this paper to additional data sets, to see if the results observed in this paper can be replicated or if they are specific to the poker data set.

Although we compare five different gradient descent algorithms in this paper, there are many different gradient descent algorithms not considered here. Two popular methods not considered in this paper are gradnient descent with momentum[9] and gradient descent with learning rate schedules[1]. Future research should apply this analysis on these algorithms, for direct comparison with the methods presented here. For accurate comparisons, care should be taken that each of these algorithms have identical initializations as the results presented here.

Finally, the WA-SGD algorithm proposed in this paper produced promising results in our results. Future work needs to be done to investigate the theoretical aspects of this algorithm. In addition, this algorithm needs to be emperically verified over different learning algorithms and data sets.

## Acknowledgements

The authors wish to thank Michael Biehl for the support he has given us. The authors also wish to thank Jasper de Boer and Joris Schaefers for their valuable feedback.

## References

[1] C. Darken, J. Chang, and J. Moody. Learning rate schedules for faster stochastic gradient search. In *Neural Networks for Signal Processing [1992] II., Proceedings of the 1992 IEEE-SP Workshop*, pages 3–12, Aug 1992.

[2] A. D. Flaxman, A. T. Kalai, and H. B. McMahan. Online convex optimization in the bandit setting: Gradient descent without a gradient. In *Proceedings of the Sixteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '05, pages 385–394, Philadelphia, PA, USA, 2005. Society for Industrial and Applied Mathematics.

[3] M. Hassoun. *Fundamentals of Artificial Neural Networks*. A Bradford book. MIT Press, 1995.

[4] M. Lichman. UCI machine learning repository, 2013.

[5] A. Y. Ng. Stanford cs229 machine learning autumn 2014, 2014. File: *gradient-descent.png*.

[6] G. Papari, K. Bunte, and M. Biehl. Waypoint averaging and step size control in learning by gradient descent. In *MIWOCI 2011, Mittweida Workshop on Computational Intelligence*, volume MLR-2011-06, pages 16–26.

[7] L. Prechelt. Automatic early stopping using cross validation: quantifying the criteria. *Neural Networks*, 11(4):761 – 767, 1998.

[8] J. C. Príncipe, N. Euliano, and W. Lefebvre. *Neural and adaptive systems: fundamentals through simulations*. Wiley, 2000.

[9] N. Qian. On the momentum term in gradient descent learning algorithms. *Neural networks*, 12(1):145–151, 1999.

[10] H. Robbins and S. Monro. A stochastic approximation method. *The annals of mathematical statistics*, pages 400–407, 1951.

[11] A. Sato and K. Yamada. Generalized learning vector quantization. *Advances in neural information processing systems*, pages 423–429, 1996.

[12] T. Schaul, S. Zhang, and Y. LeCun. No more pesky learning rates. In *Proceedings of The 30th International Conference on Machine Learning*, pages 343–351, 2013.

[13] P. Schneider, M. Biehl, and B. Hammer. Adaptive relevance matrices in learning vector quantization. *Neural Computation*, 21(12):3532–3561, 2009.

[14] D. Wilson and T. R. Martinez. The general inefficiency of batch training for gradient descent learning. *Neural Networks*, 16(10):1429–1451, 2003.

# A Comparison of Combination Schemes for Multiclass Classification Using Binary Support Vector Machines

R. van Veen and L.E.N. Baakman

**Abstract**—Several real world problems involve the classification of data in more than two categories. Although a lot of research has been done on classifying patterns, it is mostly focussed on binary classification problems. One popular way to handle multiclass classification problems is by combining multiple binary classifiers into one multiclass classifier. We discuss and compare four of these combination schemes; One-Against-One, One-Against-All, Decision Directed Acyclic Graph and, Half-Against-Half by their classification accuracy, the time complexity of training them, the space complexity of the final model and the time complexity of using the models to classify a pattern.
In conclusion the four methods have a comparable classification accuracy, however Decision Directed Acyclic Graph and Half-Against-Half need less space to store the trained model and is faster in classifying a pattern.

**Index Terms**—Support vector machine (SVM), multiclass classification, machine learning, supervised learning, pattern recognition, one-against-one (OAO), one-against-all (OAA), half-against-half (HAH), decision directed acyclic graph (DDAG).

## 1 Introduction

The ease with which we humans recognize each others faces, understand spoken words, categorize objects around us by touch, sight, and other senses, fails to give a true impression of the incredibly complex process called pattern recognition. Pattern recognition - the act of taking in raw data and taking an action based on the 'category' or 'class' of the the pattern - is a widely researched field of interest within the world of computing science [7]. Comparing the performance of the state of the art in this field with that of toddlers in pattern recognition tasks truly illustrates the difficulty of pattern recognition.

This paper focusses on supervised multiclass pattern recognition. Multiclass classification is concerned with mapping raw data e.g. measurements of real world objects, to one of the more than two possible output classes. These classes represent the category to which the input 'belongs'. An example in the field of medical diagnosis could be the distinction among different types of tumours. The raw input data in this example would be measurements like shape, volume, and density of a not yet classified tumour. These measurements are called features and a collection of such measurements, as related to one specific tumour, is called a feature vector. A multiclass classifier answers questions such as "Given these input features, what type of tumour are we dealing with?" The classifiers in this paper are constructed using supervised learning, which is a branch of machine learning where the classifier algorithms are provided with a labelled training set.

Most research on pattern recognition is done on binary classifiers, i.e. systems that distinguish between two classes. Within the medical diagnosis, a binary classifier would only be able to differentiate between two types of tumours, i.e. classifiers that answer questions such as: "Given these input features, are we dealing with a type A or type B tumour?" An ongoing research issue is how to extend these binary classifiers to a system that discriminates between more than two classes.

Currently there are two types of approaches for multiclass classification. One method directly considers all data in one optimization formulation, the other decomposes the problem into multiple binary classification problems [9]. The first approach is non-trivial and often leads to unexpected complexity or worse performance [18]. In this paper we only consider algorithms of the second type.

Several different schemes exist to combine two-class classifiers to one multiclass classifier. We compare some of these schemes on accuracy of classification, and computational and spatial complexity. We consider the following combination schemes. 'One-Against-One' (OAO) [12] is a combination of binary classifiers that are trained to distinguish between one of the classes and every other class. 'One-Against-All' (OAA) [3] uses binary classifiers that are trained to distinguish one class of patterns from all other patterns. The 'Decision Directed Acyclic Graph' (DDAG) [20] is a method that combines the OAO binary classifiers in a decision tree structure. 'Half-Against-Half' (HAH) [13] is the only method that considers to group the classes and train the binary classifiers to distinguish between these groups.

In these types of schemes the binary classifier of choice is often the support vector machine (SVM) [24, 11, 13, 20]. Support vector machines are widely used in practice, because of their capabilities to implicitly map initially non-linearly separable data to data that are linearly separable.

The remainder of this paper is organized as follows: Section 2 provides a brief review of the binary support vector machine and discusses the different combination schemes described above. In Section 3 we compare the different schemes based on results reported in literature and discuss the performance of the four different classification schemes. This discussion includes the theoretical and empirical time and space complexity of these schemes. The last section presents our conclusions.

## 2 Description of methods

In this section we give an outline of the theory behind the different combination schemes used to solve multiclass classification problems. In order to do this we first discuss the support vector machine, which is used as the binary classifier.

### 2.1 Binary Support Vector Machines

SVMs were first introduced in 1995 and are a popular solution for binary classification problems [5]. In this section we give a brief review of support vector machines. For more in depth explanations of the concepts introduced below we refer the reader to the original paper [5] and the introductory text by N. Cristianini and J. Shawe-Taylor [6].

We start our discussion of the support vector machine with the most simple SVM, namely the maximal margin classifier. This classifier can classify linearly separable data without errors. An $N$-dimensional data set is considered linearly separable if a hyperplane with dimension $N-1$ exists which divides the space into two half spaces which correspond to the inputs of the two distinct classes [6].

• *Rick van Veen is a Computing Science student at the University of Groningen, E-mail: r.van.veen.3@student.rug.nl.*
• *Laura Baakman is a Computing Science student at the University of Groningen, E-mail:l.e.n.baakman@rug.nl.*

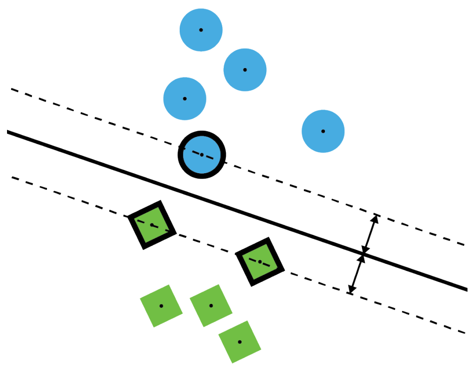


Fig. 1. The separation of two classes, by a maximal margin classifier. The support vectors are denoted as symbols with a border. The two arrows perpendicular to the separating hyperplane, shown as a line, indicate the margin.

Formally, consider a data set $X$ with $\mathcal{L}$ points:

$$X = \left\{ (\vec{x}_t, r_t) \mid \vec{x}_t \in \mathbb{R}^N, r_t \in \{-1, 1\} \right\}_{t=1}^{\mathcal{L}}, \tag{1}$$

where $r_t$ denotes the class to which a data point $\vec{x}_t$ belongs. Since we consider the binary case a label is either $-1$ or $+1$. A separating hyperplane is defined as all points that satisfy

$$\langle \vec{x}, \vec{w} \rangle + b = 0,$$

where the weight vector, $\vec{w}$, is perpendicular to the hyperplane and the bias, $b$, determines the location of the hyperplane with respect to the origin.

Given a linearly separable training data set $X$, as defined in Equation (1), the maximal margin classifier aims to find a bias $b$ and a weight vector $\vec{w}$ that satisfy:

$$\min_{\vec{w}, b} \langle \vec{w}, \vec{w} \rangle$$

under the constraint that for all $t = 1, \ldots, \mathcal{L}$:

$$r_t \left( \left\langle \vec{w}^T, \vec{x}_t \right\rangle + b \right) \geq 1. \tag{2}$$

The margin is the line through the points that satisfy Equation (2). As its name indicates, the maximal margin classifier maximizes the distance between the line perpendicular to $\vec{w}$ and the lines through the support vectors, respectively the solid and the dashed line in Figure 1. The advantage of maximizing the margins is an increase in generality [1].

A geometric representation of a maximal margin classifier is presented in Figure 1. This figure illustrates that the support vectors lie at a distance equal to the margin from the separating hyperplane.

The boundary that maximizes the margin is called the optimal separating hyperplane. The vector that defines the separating hyperplane is a linear combination of some of the vectors used for training. These vectors are called the support vectors, in Figure 1 they are indicated by the bordered symbols.

One important disadvantage of the maximal margin classifiers or hard margin SVMs is that they cannot handle data that are not linearly separable.

The hard margin SVM can fail to produce a classifier for a given training set where a soft margin SVM produces one at the cost of misclassifying data from the data set [5]. If there exists no hyperplane that can separate the positive from the negative examples, the soft margin method will choose a separating hyperplane that splits the dataset as cleanly as possible, while still maximizing the distance to the nearest examples. This is done by introducing a slack variable $\xi_t \geq 0$ that stores the deviation from the margin for pattern $\vec{x}_t$. Two types of deviation are possible: an instance may lie on the wrong side of the hyperplane and be misclassified, or it may be on the right side but not lie sufficiently far away from the hyperplane. Relaxing Equation (2) we now require:

$$r_t \left( \left\langle \vec{w}^T, \vec{x}_t \right\rangle + b \right) \geq 1 - \xi_t. \tag{3}$$

If $\xi_t = 0$ there is no problem with $\vec{x}_t$. However if $0 < \xi_t < 1$, $\vec{x}_t$ is correctly classified, but in the margin. If $\xi_t \geq 1$, $\vec{x}_t$ is incorrectly classified [1].

Both the soft and hard margin SVM are linear classifiers, i.e, they can only partition linearly separable data. To allow support vector machines to classify non-linear data, they map the data to a higher dimension using a kernel [2]. In this higher-dimensional feature space a separating hyperplane is constructed [5]. The linear model in this new space corresponds to a non-linear model in the original space [1]. Denoting the kernel function with $\psi(\bullet)$ Equation (3) becomes:

$$r_t \left\langle \vec{w}^T, \psi(\vec{x}_t) \right\rangle + b \geq 1 - \xi_t.$$

A large number of different kernels exist and which kernel is optimal depends on the application. There is a group of kernels, the universal kernels, that guarantee a globally optimal classifier [22]. The radial basis function (RBF),

$$\psi(\vec{x}_i, \vec{x}_j) = e^{-\alpha |\vec{x}_i - \vec{x}_j|^2}, \tag{4}$$

is one of those universal kernels. The parameter $\alpha$ controls the width of the kernel. One advantage of non-linear kernel functions, such as Equation (4), is the flexibility it provides to the SVM hyperplane calculation. For a discussion of other possible kernels and more details we refer the reader to ‘Kernel Methods for Pattern Analysis’[21].

The output of a support vector machine for a specific pattern is the distance between that pattern and the separating hyperplane. However the methods introduced below often need a posterior probability. A method that fits a sigmoid to map the output of a SVM to a posterior probability is introduced by J. Platt [19]. We refer the reader to the original paper [19] for a thorough explanation and to ‘A Note on Platt’s Probabilistic Outputs for Support Vector Machines’ [14] for an improved algorithm.

### Complexity

The time complexity of training a support vector machine on $\mathcal{L}$ patterns is $\mathcal{O}(\mathcal{L}^\gamma)$, where $\gamma \approx 2$ for algorithms based on the decomposition method [20].Classifying $\mathcal{S}$ patterns with a binary SVM has time complexity $\mathcal{O}(\mathcal{S})$ [10].

The space complexity of a trained support vector machine depends upon the number of support vectors, since its decision hyperplane is implicitly stored as its support vectors. If we assume that a portion $\beta$ of the training data will become support vectors the space complexity of a SVM is $\mathcal{O}(\beta\mathcal{L})$.

Although we use here, and in the rest of out paper, the number of support vectors to measure the space complexity of the trained model one should note that the dimensionality of the vectors influences both the amount of space needed to store the data and the time needed to classify a pattern.

## 2.2 One-Against-All

The One-Against-All scheme is probably the earliest implementation of a SVM multiclass classifier [3]. In this scheme, for a $(K > 2)$-class problem, $K$ binary SVMs are constructed. Every SVM is constructed by training it to distinguish between one of the $K$ classes and the other $K-1$ classes. Consider the set of classes to be $\{C_1, \ldots, C_K\}$, then every $i$th SVM is trained with the data of class $C_i$, which are assigned the positive labels and with the data from the other $\{C_j | j \neq i\}$ classes, which are given the negative labels.

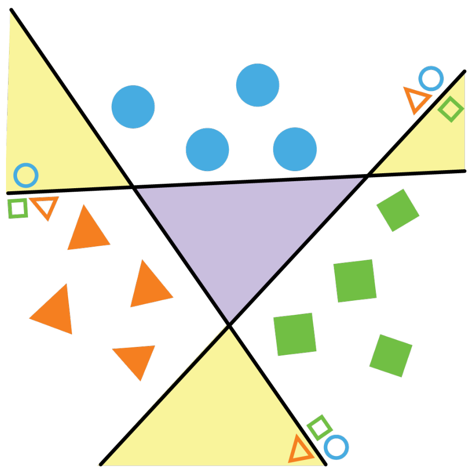


Fig. 2. Decision surfaces of One-Against-All combination scheme for binary SVMs. Different classes are illustrated as different shapes and colours. The shaded triangle in the middle illustrated an ambiguous region where it is also possible to have $0$ or $> 1$ votes from the different binary SVMs.

When the classifier is presented with a novel pattern that has to be classified, all the constructed SVMs are given this input. The SVMs then cast their votes i.e. they give an output which says the input belongs to the single class or to the group of other classes. In order to handle the votes the multiclass classifier uses a method e.g. the method of J. Platt [19] to get a probability, which tells the classifier how certain a SVM is of its vote.

Figure 2 illustrates a $N = 2$ dimension feature space, with $K = 3$ classes. Using a One-Against-All multiclass classifier, these three classes can be separated using three binary SVMs. Three different cases can be distinguished after counting the votes. In the case that a novel pattern gets only one vote, the final class assignment is clear. The second case is when multiple SVMs vote that the novel pattern belongs to their single class, in this case the classifier that is most confident is chosen to be the winner. This case is illustrated by the yellow triangles on the outsides of the image in Figure 2. The purple triangle illustrates the case where none of the SVMs think the pattern belongs to their single class, in this case the pattern is assigned to the single class of the SVM that is least certain about its choice.

A problem of the OAA strategy shown in ‘A Review on the Combination of Binary Classifiers in Multiclass Problems’ [15], is imbalance of the classes among the data. The OAA scheme may show disadvantages when the number of patterns from the single class is too low compared to the number of patterns in set formed by the patterns from all the other classes.

### Complexity

Without loss of generality we can assume that each class has $\mathcal{L}/K$ training patterns [13]. Since we need to train $K$ binary SVMs, the time complexity of training an OAA classifier is

$$\mathcal{O}\left(K\mathcal{L}^{\gamma}\right).$$

The time complexity of classifying $\mathcal{S}$ patterns with a One-Against-All classifier is

$$\mathcal{O}\left(K \cdot \mathcal{S}\right).$$

If we assume that for each binary SVM, a portion $\beta$ of the training data will become support vectors, the space complexity of a model trained according to One-Against-All is

$$\mathcal{O}\left(\beta K \mathcal{L}\right).$$

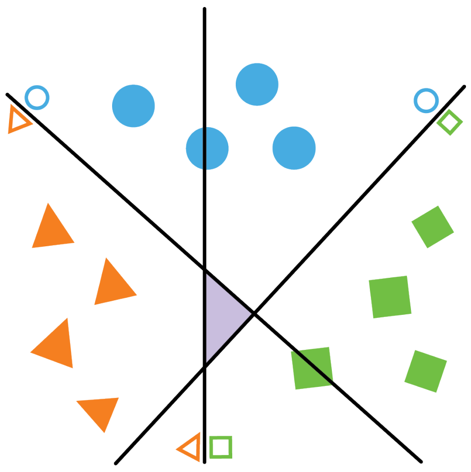


Fig. 3. The decision surfaces of the binary classifiers in One-Against-One, when trained on a $2D$ dataset with three classes. Different classes have different shapes and colours. In the shaded area the classifiers disagree.

## 2.3 One-Against-One

A second simple method to combine binary classifiers is One-Against-One, which trains a classifier for each of the pairs of classes in the dataset. Each of the $K(K-1)/2$ binary classifiers is trained on a subset of the data, i.e. the classifier separating $C_i$ from $C_j$ is trained on a training set containing only patterns belonging to either $C_i$ or $C_j$. We use $P_{ij}(C_i)$ to represent the probability that a pattern belongs to the class $C_i$ as determined by the classifier trained on the classes $C_i$ and $C_j$.

We combine the output of binary classifiers with Friedman’s procedure [8], which assigns a pattern to the class that wins the most pairwise comparisons, formally:

$$\delta = \arg\max_{i} \left| \left\{ P_{ij}(C_i) > P_{ij}(C_j) \mid i \neq j, i < K, j < K \right\} \right|.$$

Where the final class is denoted as $\delta$. Figure 3 shows a two-dimensional dataset that contains three classes, and the three separating hyperplanes as trained by an One-Against-One classifier. A disadvantage of Friedman’s method is that there will be cases where all classifiers disagree, illustrated by the shaded area in Figure 3, or where there are ties in voting, which is only possible if $K > 3$[23]. If the system may not reject any patterns, one can assign the patterns where the classifiers do not agree to the class with the largest prior probability.

### Complexity

Since we train each of the $K(K-1)/2$ binary classifiers with $2 \cdot \mathcal{L}/K$ patterns the time complexity of training a multiclass classifier with OAO is:

$$\mathcal{O}\left(K(K-1)/2\left(\frac{2 \cdot \mathcal{L}}{K}\right)^{\gamma}\right).$$

The time complexity of classifying $\mathcal{S}$ patterns with an One-Against-One classifier is:

$$\mathcal{O}\left(\mathcal{S} \cdot K(K-1)/2\right).$$

Let $\beta$ be the portion of the training patterns that becomes a support vector, the space complexity of an OAO classifier is then:

$$\mathcal{O}\left(\beta \mathcal{L}(K-1)\right).$$

## 2.4 Half-Against-Half

The Half-Against-Half scheme, proposed by H. Lei and V. Govindaraju [13], is the only scheme where the binary SVMs are used to separate two groups of classes. When the data set allows it these groups effectively divide the data after every classification step in half. The

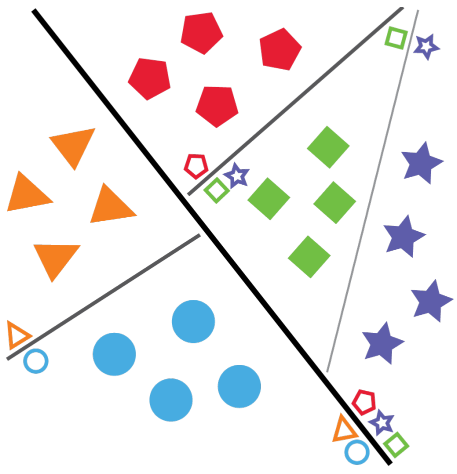


Fig. 4. Decision boundaries of the Half-Against-Half multiclass classifier. Each boundary indicates the groups or single classes it separates. The decision boundaries do not have the same thickness. By following the boundaries from thick to thin one can see the order in which the the recursive binary classification problems are solved.

illustration in Figure 4 shows multiple SVM decision borders. The classification process is ordered in the same way as a binary decision tree, and recursively solves the binary classification problems. The binary classifiers are trained according to this decision process, i.e. from course to finer decisions. Consider a $K = 5$ class classification problem. The root of the decision tree will separate the two largest groups e.g. $\{C_1, C_2\}$ from $\{C_3, C_4, C_5\}$. All nodes below the root divide the remaining classes into two e.g. $\{C_1\}$ from $\{C_2\}$ until the decision process reaches a classifier in that separates gives an answer which is only one class. In the case of the examples the classifier will either vote for $C_1$ or $C_2$.

The hard part and unique problem of this multiclass classification scheme is to determine the optimum division of the data on which the binary classifiers are trained. With smaller data sets the division into groups can be done manually with prior knowledge, where similar or close classes can be grouped together. For larger problems it is preferred to automatically, determine the groups. One proposed strategy is to divide the data into arbitrary groups. The problem with this strategy is that the arbitrary chosen groups are not necessarily separable, resulting in poor classification [13].

As an example see the illustration in Figure 4 and consider a novel data point that belongs to the pentagon class. The multiclass classifier that uses the Half-Against-Half scheme first decides if the novel point belongs to the group of circles and triangles or to the other group, by letting the binary SVM vote that is trained to separate these two groups of classes. Secondly the SVM that separates the pentagons from the squares and stars votes for the class it thinks the novel data belongs to.

### Complexity

Considering a $K$ multiclass problem. The structure of the Half-Against-Half scheme is a binary decision tree and has a depth of

$$2^h - 1, \text{ with } h = \lceil \log_2(K) \rceil.$$

For every node in the decision tree a binary SVM needs to be trained and therefore the the number of SVMs needed is $2^h - 1$ [13]. The training time is the sum of the time it takes to train all the binary classifiers in the $h$ levels. At the $i$th level, there are $2^{i-1}$ nodes and each of the SVMs at the nodes are trained with $\frac{\mathcal{L}}{2^{i-1}}$. The total training time is:

$$\mathcal{O}\left(\mathcal{L}^2\right),$$

as shown by H. Lei and V. Govindaraju [13]. H. Lei and V. Govindaraju [13] have shown the time complexity of classifying $\mathcal{S}$ patterns with the Half-Against-Half scheme to be

$$\mathcal{O}\left(h \cdot \mathcal{S}\right).$$

If we assume that for each binary SVM a portion $\beta$ of the train data will become support vectors, the space complexity of a model trained according to Half-Against-Half is

$$\mathcal{O}\left(h\beta\mathcal{L}\right).$$

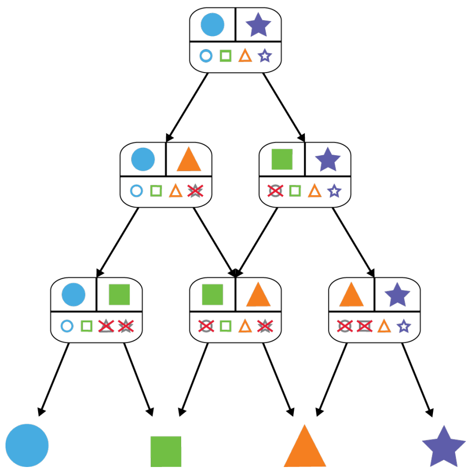


Fig. 5. A decision DAG for finding the best class out of four classes. Each node shows between which classes it compares and which classes are still possible at that node.

## 2.5 Decision Directed Acyclic Graph

Similar to the One-Against-One scheme, the Decision Directed Acyclic Graph (DDAG) method trains a classifier for each pair of classes in the dataset. Each of the resulting $K(K-1)/2$ binary classifiers are placed in a directed acyclic graph (DAG). A DDAG is a decision acyclic directed graph in which each vertex has either two or zero outgoing edges. A DDAG has exactly one root; a vertex which has no incoming edges. A DDAG has $K$ leaves and $K(K-1)/2$ internal vertices. The vertices are arranged in a triangle with the root at the top, two vertices in the second layer, three in the third, and so on until the final layer of $K$ leaves. The $i$-th node in layer $j < K$ is connected to the $i$-th node and node $(i+1)$ in layer $(j+1)$[20]. An example of such a DDAG is presented in Figure 5.

The classification of some pattern $\vec{x}$ starts at the root. The pattern $\vec{x}$ is used as input for the SVM associated with the root vertex. Based on the output of this binary SVM the next vertex to visit is selected. This process is continued until we reach a leaf. The leaf where this path ends is the class that is assigned to $\vec{x}$ by this multiclass classifier. We call the resulting path through the DDAG the evaluation path.

The vertex distinguishing class $i$ from $j$ is represented as $v_{ij}$ and the binary SVM at vertex $v_{ij}$ as $P_{ij}(\bullet)$. If $P_{ij}(C_i) > P_{ij}(C_j)$ the evaluation path is continued at the left child of $v_{ij}$, otherwise it is continued at the right child of $v_{ij}$. If the left child of $v_{ij}$ is chosen, $\vec{x}$ cannot be classified as belonging to $C_j$. In Figure 5 this is represented by the grey crossed-out symbols.

One disadvantage of the DDAG is that its result depends on the sequence of binary classifiers in the graph. This affects the reliability, as different permutations of nodes in the graph may produce different results. Another drawback of the DDAG model is that depending on the position of the actual class of the pattern in the graph the number of evaluations is unnecessarily high, which is not only inefficient but also results in a high cumulative error. If the probability of misclassification is 1% in each vertex this will result in a cumulative error rate of $(1-(0.99)^{(K-1)})$, which becomes critical for a high value of $K$ [15].

The Decision Directed Acyclic Graph scheme has a couple of advantages when compared with the One-Against-One scheme. Firstly, the classifiers cannot disagree on some pattern. Secondly, when classifying some pattern with a DDAG it only needs to be classified by $K-1$ binary classifiers, if OAO is used, $K(K-1)/2$ are used. Thirdly, unless the binary classifiers are carefully regulated, OAO tends to overfit [20].

#### Complexity

The time complexity of training a Decision Directed Acyclic Graph classifier is the same as training a classifier with One-Against-One:

$$\mathcal{O}\left(K(K-1)/2\left(\frac{2\cdot\mathcal{L}}{K}\right)^{\gamma}\right).$$

Classifying $\mathcal{S}$ patterns with a DDAG classifier has time complexity

$$\mathcal{O}\left(\mathcal{S}(K-1)\right)$$

The space complexity of a classifier trained with Decision Directed Acyclic Graph is the same as one trained with OAO:

$$\mathcal{O}\left(\beta\mathcal{L}(K-1)\right).$$

## 3 Discussion

In this section we discuss the differences in complexity between the combination schemes introduced in Section 2 and we review differences in performances as described in the literature. We do this based on a number of papers which we shortly introduce.

H. Lei and V. Govindaraju [13] introduced the Half-Against-Half combination scheme. In their paper they not only introduce HAH models, they also compare its performance with that of OAO, OAA and DDAG models on four different datasets. They concluded that, HAH results in a more compact model and has similar performance to the other three methods they consider.

J. Milgram, M. Cheriet and R. Sabourin [17] compare the performance of a One-Against-One and a One-Against-All classifier on classifying hand written digits and letters. Based on their research they conclude that OAA classifiers are better suited for problems with few classes, whereas with more classes the difference in accuracy between OAO en OAA seems insignificant. Furthermore they conclude that the OAO is better suited for problems with a large number of training samples than OAA, due to its lower training time.

The DDAG was introduced by J. Platt, N. Cristianini and J. Shawe-Taylor [20], they also compare the performance of their newly introduced classifier with OAO and OAA on three datasets. They find that the error rates for the three algorithms are similar, but the DDAG model is faster when classifying a pattern.

J. Manikandan and B. Venkataramani [16] introduce a novel technique for a multiclass SVM classifier, diminishing learning, which we do not consider. They compare the performance of their diminishing learning classifier with that of a HAH, DAG, and OAA models on recognizing isolated spoken digits.

J. Platt, N. Cristianini and J. Shawe-Taylor [20] do not mention the source of the code they use for their experiments. H. Lei and V. Govindaraju [13] and J. Milgram, M. Cheriet and R. Sabourin [17] indicate that they used LIBSVM [4], J. Manikandan and B. Venkataramani [16] use a combination of MATLAB and C code. All of the support vector machines used in the research mentioned in this section used a radial basis function as their kernel.

### 3.1 Complexity

We start this section by comparing the four different combination schemes based on their time and space complexities. The second part of this section is used to verify the theoretical results empirically.

#### Theoretical

If we compare the time complexity of training the different combination schemes, we see that all have the same time complexity, except for OAA which is a factor $K$ higher.

There are however large differences in the time complexity when classifying novel data. DDAG is faster than OAO. HAH is even faster than DDAG because of the depth of the HAH decision tree, $\lceil\log_2(K)\rceil$, which is smaller than DDAG's $(K-1)$, especially when $K \gg 2$ [13].

In practical applications the size of the trained classifier is an important concern. Since the model generally stays in memory. An OAA classifier has the worst space complexity of the mentioned combination schemes. The size of the DAG and OAO models are similar, although strictly speaking the DAG needs a bit more space to also store the graph. HAH has the best space complexity, especially for $K \gg 2$.

Based on the complexities discussed above, the Half-Against-Half classifier seems optimal, with only its training time complexity not lower than its competitors. Although OAA is a simple scheme its theoretical performance is the worst of the four.

#### Emperical

Since the number of support vectors both influences a model's space and classification speed we discuss the complexity empirically by comparing the number of support vectors generated by each model on a number of datasets. All models mentioned below were trained with an RBF kernel.

J. Milgram, M. Cheriet and R. Sabourin [17] found that OAA models have more support vectors than OAO models. Although this is generally consistent with the findings from [9], they found that the OAA model needed less support vectors than the OAO or DAG models on a small dataset with few classes and high dimensionality ($\mathcal{L} = 178$, $N = 13$, $K = 3$). J. Manikandan and B. Venkataramani [16] also found some cases where the OAA model had fewer support vectors than the DAG model. Once again the datasets that resulted in these models were relatively small, with few classes and high dimensionality.

J. Platt, N. Cristianini and J. Shawe-Taylor [20] found that in general the DAG and OAO models need the same number of support vectors. Although in one case ($\mathcal{L} = 16000$, $N = 16$, $K = 26$) the DDAG model had significantly more support vectors than OAO. This fits with the results found by C. Hsu and C. Lin [9] except for one dataset ($\mathcal{L} = 2310$, $N = 19$, $K = 7$) where the OAO model had significantly more support vectors.

According to H. Lei and V. Govindaraju [13] Half-Against-Half models consistently need the smallest number of kernel evaluations when classifying a pattern. Since there are no other comparisons between Half-Against-Half and the other three methods in the literature we could not confirm or deny this finding.

Based on the theory, one would expect that to classify a pattern a DAG model, would need fewer kernel evaluations than an OAO model. The number of kernel evaluations found for OAO and DAG models found by [13, 20] fit with the theory.

In general OAA models need more support vectors than the other three types of models, however there are some exceptions to this rule. Overall the number of support vectors required by a DAG model is the same as the number required by a OAO model.

### 3.2 Performance

H. Lei and V. Govindaraju [13] found that HAH models are more compact and classify faster than OAO, OAA or DDAG models, and have comparable classification accuracy. J. Manikandan and B. Venkataramani [16] confirmed that HAH needed significantly fewer support vectors than its competitors. However they did not find that HAH consistently had the same classification accuracy as the other schemes. On the dataset 'multispeaker dependent' ($\mathcal{L} = 300$, $K = 10$) HAH performed significantly worse than the other options. However, they also found that HAH is the fastest model when classifying inputs. We could not find any other data comparing the speed of classification of a HAH model with that of a DDAG model.

J. Milgram, M. Cheriet and R. Sabourin [17] found that OAA works better on problems with few classes ($K \approx 10$), whereas OAO are OAA are comparable when there are more classes ($K \approx 26$). Others did not find a significant difference in classification accuracy between OAO and OAA when the dataset has ten classes [20, 13]. Nor did they find that OAA performed better than DDAG or HAH on problems with a small ($K \leq 10$) number of classes [16].

J. Platt, N. Cristianini and J. Shawe-Taylor [20] and H. Lei and V. Govindaraju [13] confirmed the finding that OAO and OAA are comparable when the dataset has 26 classes. Strangely J. Platt, N. Cristianini and J. Shawe-Taylor [20] found that OAO performed significantly better than OAA on a dataset with seven classes. The authors did not offer an explanation for this difference in performance.

J. Platt, N. Cristianini and J. Shawe-Taylor [20] found that DDAG performs comparable to OAO and OAA but is faster when classifying patterns. H. Lei and V. Govindaraju [13] found the same differences between OAO, OAA and DDAG. J. Manikandan and B. Venkataramani [16] however found that it is not necessarily the case that DDAG performs comparably to OAA. They even found that on one specific dataset ($\mathcal{L} = 500$, $K = 10$) OAA performed significantly better than DDAG. This fits the conclusion from J. Milgram, M. Cheriet and R. Sabourin [17], that OAA is suitable for problems with few classes.

In conclusion, an One-Against-All classifier works well, but not necessarily better than the other three schemes, on problems with few classes. One distinct disadvantage of the OAA model is that it is generally larger than models generated with any of the other schemes.

Ordering classifiers in a tree results in models that are faster than the other schemes. However the resulting models do not necessarily have the same performance as OAA or OAO. Comparing the classification accuracy of HAH and DDAG based on the data provided by [16] we find that their accuracy is about the same.

## 4 Conclusion

We did not find any hard evidence that one classification scheme is distinctively better than the others. Consequently the choice for a specific scheme comes down to the time the classifier needs to train, classify and the space required to store it.

Based on both theoretical and practical results we can conclude that a One-Against-All model is not a good choice based on the previously mentioned criteria. Although HAH and DDAG need about the same amount of storage as OAO, they classify new patterns significantly faster. Furthermore they cannot result in ties. Consequently it seems that the HAH or DDAG models are the best choice for most applications.

## Acknowledgements

The authors wish to thank B. Marinissen, D. Greveling and, M. Biehl for their helpful reviews.

## References

[1] E. Alpaydin. *Introduction to machine learning*. MIT press, 2014.

[2] B. E. Boser, I. M. Guyon, and V. N. Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the fifth annual workshop on Computational learning theory*, pages 144–152. ACM, 1992.

[3] L. Bottou, C. Cortes, J. S. Denker, H. Drucker, I. Guyon, L. D. Jackel, Y. LeCun, U. A. Muller, E. Sackinger, P. Simard, et al. Comparison of classifier methods: a case study in handwritten digit recognition. In *International Conference on Pattern Recognition*, pages 77–77. IEEE Computer Society Press, 1994.

[4] C.-C. Chang and C.-J. Lin. Libsvm: a library for support vector machines. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 2(3):27, 2011.

[5] C. Cortes and V. Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[6] N. Cristianini and J. Shawe-Taylor. *An introduction to support vector machines and other kernel-based learning methods*. Cambridge university press, 2000.

[7] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern classification*. John Wiley & Sons, 2012.

[8] J. Friedman. Another approach to polychotomous classification. Technical report, Department of Statistics, Stanford University, 1996.

[9] C.-W. Hsu and C.-J. Lin. A comparison of methods for multiclass support vector machines. *Neural Networks, IEEE Transactions on*, 13(2):415–425, 2002.

[10] S. S. Keerthi, O. Chapelle, and D. DeCoste. Building support vector machines with reduced classifier complexity. *The Journal of Machine Learning Research*, 7:1493–1515, 2006.

[11] J. Kindermann, E. Leopold, and G. Paass. Multi-class classification with error correcting codes. *Treffen der GI-Fachgruppe*, 1(3), 2000.

[12] S. Knerr, L. Personnaz, and G. Dreyfus. Single-layer learning revisited: a stepwise procedure for building and training a neural network. In *Neurocomputing*, pages 41–50. Springer, 1990.

[13] H. Lei and V. Govindaraju. Half-against-half multi-class support vector machines. In *Multiple classifier systems*, pages 156–164. Springer, 2005.

[14] H.-T. Lin, C.-J. Lin, and R. C. Weng. A note on platts probabilistic outputs for support vector machines. *Machine learning*, 68(3):267–276, 2007.

[15] A. C. Lorena, A. C. De Carvalho, and J. M. Gama. A review on the combination of binary classifiers in multiclass problems. *Artificial Intelligence Review*, 30(1-4):19–37, 2008.

[16] J. Manikandan and B. Venkataramani. Study and evaluation of a multiclass svm classifier using diminishing learning technique. *Neurocomputing*, 73(10):1676–1685, 2010.

[17] J. Milgram, M. Cheriet, and R. Sabourin. one against one or one against all: Which one is better for handwriting recognition with svms? In *Tenth International Workshop on Frontiers in Handwriting Recognition*. Suvisoft, 2006.

[18] G. Ou and Y. L. Murphey. Multi-class pattern classification using neural networks. *Pattern Recognition*, 40(1):4–18, 2007.

[19] J. C. Platt. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In *Advances in Large Margin Classifiers*, pages 61–74. MIT Press, 1999.

[20] J. C. Platt, N. Cristianini, and J. Shawe-Taylor. Large margin dags for multiclass classification. In S. A. Solla, T. K. Leen, and K.-R. Mller, editors, *Advances in Neural Information Processing Systems 12*, volume 12, pages 547–553. MIT Press, 1999.

[21] J. Shawe-Taylor and N. Cristianini. *Kernel methods for pattern analysis*. Cambridge university press, 2004.

[22] I. Steinwart. On the influence of the kernel on the consistency of support vector machines. *The Journal of Machine Learning Research*, 2:67–93, 2002.

[23] D. M. Tax and R. P. Duin. Using two-class classifiers for multiclass classification. In *International Conference on Pattern Recognition*, volume 2, pages 20124–20124. IEEE Computer Society, 2002.

[24] D. Tsujinishi, Y. Koshiba, and S. Abe. Why pairwise is better than one-against-all or all-at-once. In *International Joint Conference on Neural Networks*, volume 1. IEEE, 2004.

# Hyperconnectivity and Hyperconnected Filters

Rogchert Zijlstra    Bart Marinissen

**Abstract**—In this paper we look at known results for hyperconnectivity, an extension of connectivity that aims to relax the restrictions of connectivity while keeping its strengths. Connectivity itself is already a generalization of the basic 4- and 8-connectivities. The strength of connectivity depends on connected components. These tend to correspond to structures in an image. Thus, operating on these components is tantamount to operating on the very structure of an image. However, general connectivity is restricted by a very strong overlap condition. No two connected components can share a point. Hyperconnectivity loosens this restriction.
We first introduce the concept of connectivity and hyperconnectivity more formally. After that we consider two applications that show the descriptive power of this framework. The first example is the $K$-flat zones filter. A very simple and intuitive hyperconnected filter. It simply states a set to be hyperconnected when the grey value variation is limited. This turns out to perform very well. The second example is a fuzzy measure of connectivity for fuzzy sets. It turns out that conventional 'crisp' methods have their issues. Hyperconnectivity plays a surprising role in solving these issues. These examples, combined with theoretical considerations, show the elegant power of hyperconnectivity.

**Index Terms**—Hyperconnectivity, Connectivity, Image Segmentation, Connected Filters, Morphology, K-flat, Fuzzy Sets

## 1 Introduction

Connected filters, and with them connectivity, are a powerful tool. They allow us to remove unwanted structures from an image without molesting the wanted structures. However, they are also limited by what we consider connected. The aim of this paper is to see what can be achieved by relaxing this limitation. This is done via the framework of hyperconnectivity as introduced by [4]. We will see how this framework gives rise to hyperconnected filters, which retain the strength of their connected counterparts. Furthermore, we will also show how hyperconnectivity can play a role in defining a fuzzy measure of connectivity for fuzzy sets as seen in [1].

In short, a connectivity tells you which sets are considered to be connected. The quintessential examples of connectivity in are 4- and 8- connectivity. These are shown in Figure 1. For these, we say two pixels in a set are connected when we can find a path of adjacent pixels in the set from one pixel to the other. A set is then connected when all of its points are connected to each other. The difference between 4- and 8 connectivity lies in which points are considered to be connected. In 4-connectivity two pixels are adjacent when they share an edge, in 8-connectivity it sufficient from them to share a vertex.

At first sight these connectivities may seem perfectly able to describe all connected. However, it is rather limited. The power of connected filters comes from the correspondence between image structures and connected components. Image structures can be quite clearly present without being 4- or 8-connected. Consider, for example, the two clearly separate clusters seen in figure 2a. We want to treat each

- *BSc. Rogchert Zijlstra, Master student at the University of Groningen, track: Computational Science and Visualisation E-mail: rogchertzijlstra@gmail.com.*
- *BSc. Bart Marinissen, Master student at the University of Groningen, track: Intelligent Systems E-mail: bartmarinissen@gmail.com.*

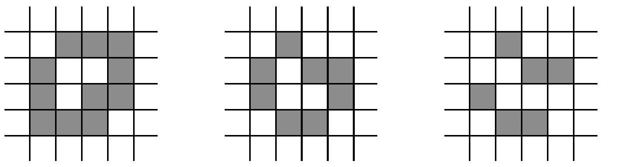

(a) 4- and 8-connected image (b) 8-connected but not 4-connected image (c) Neither 4- nor 8-connected

Fig. 1: examples of 4- and 8-connectivity

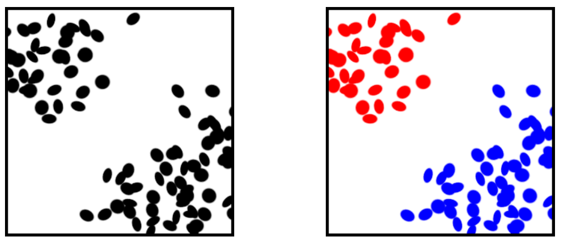

(a) Two clearly seperated clusters (b) The most satisfying segmentation into connected components

Fig. 2: Two clusters and a possible segmentation of these into connected components

of the clusters as a single structure. In other words, we want the connected components to be as seen in figure 2b

Generalized connectivity allows for this, and much more. The key restriction for a connectivity is that connected components cannot overlap. That is, they cannot share a single common point. It is this restriction that is relaxed by hyperconnectivity. It allows for a different definition when *hyper*connected components overlap.

First, we will formalize these notions of connectivity, connected components and connected filters. Then we will do the same for their counterparts in hyperconnectivity. After that, we will show an example of a simple yet powerful hyperconnected filter. Finally, we will show the application in hyperconnectivities when it comes to defining a fuzzy measure of connectedness for fuzzy sets.

## 2 Connectivity

In this section, we will first show how the concept of connected sets gives rise to connected components and connected filters. Only then will we give the formal definition of a connectivity. For now, one can use the 4- and 8-connectivity to guide one's intuition. Throughout this section, we consider $I$ to be a binary image over space $E$. The definitions seen here are largely based on those seen in [2].

Informally, a connected component of an image is a connected part that cannot be extended without becoming disconnected; formally, we have the following:

**Definition 1** (Connected Components)**.** *Given a binary image $I \subseteq E$, we call $C \subseteq I$ a connected component if and only if:*
*$C$ is connected, and there is no connected set $B$ such that $C \subsetneq B \subseteq I$*

The connected components of an image form a partion of that image. Connected filters take this partitioning of an image and keeps or discards each connected component based on some criterion. Formally, we have:

**Definition 2** (Connected Filter)**.** *A selection criterion is a function $\Lambda : \mathcal{P}(E) \to \{0,1\}$. Such a criterion $\Lambda$ gives rise to a connected filter $\psi_\Lambda : \mathcal{P}(E) \to \mathcal{P}(E)$. To define $\psi_\Lambda(X)$, we first need to define:*

$$C_X = \{A \subseteq X \mid A \textit{ is a connected component}\}$$

$$\Gamma_\Lambda(C) = \begin{cases} C & \textit{if } \Lambda(C) = 1 \\ \emptyset & \textit{otherwise} \end{cases}$$

*This finally gives us what we need to define the connected filter:*

$$\psi_\Lambda(X) = \bigcup_{C \in C_X} \Gamma_\Lambda(C)$$

One example selection criterion might be: $\Lambda(X) = 1$ iff $|X| > a$, the so called area filter. Other filters such as those considering the 'roundness' of $X$ also exist.

From this, it is clear that connected components are supposed to correspond to real world objects. It is now clear why we want connected components to correspond to image structures. It are these structures that we operate on. We already saw 4- And 8-connectivity fall short for the clusters. The common definition of general connectivity allows for much better correspondence between connected components and image structures. The core concept of connectivity is that, when connected sets overlap (I.E. have a common point), their union must be connected. This is what causes the connected components to be a partition. The only other criterion is that each singleton set is a connected set. Formally, this is:

**Definition 3** (Connectivity)**.** *We call a family $\mathcal{C} \subseteq \mathcal{P}(E)$ a connectivity if and only if both of the following hold:*

$$\forall x \in E.\ \{x\} \in \mathcal{C}$$
$$\left(\forall \mathcal{A} \subseteq \mathcal{C}.\ \bigcap \mathcal{A} \neq \emptyset\right) \Rightarrow \left(\bigcup \mathcal{A} \in \mathcal{C}\right)$$

*We then call a set $C$ connected if and only if $C \in \mathcal{C}$*

## 3 Hyperconnectivity

The concept of connectivity as seen in definition 3 makes big strides towards dealing directly with image structures. However, the fact that connected components always partition an image is a rather strict limitation. It means no single point can lie in two different connected components. This works when objects are clearly delineable. However, in less clear cases this forces us to assign a point without sufficient information.

Hyperconnectivity was made to address this by loosening the overlap criterion. Specifically, it allows us to choose a custom 'overlap criterion'. The resulting framework is much more powerful. In this section, we first define hyperconnectivity. Then, we will define hyperconnected components and hyperconnected filters analogously to their connectivity basic counterparts of the previous section. The definitions seen here are largely based on those seen in [2].

Before defining hyperconnected components, we first need to define what constitutes an overlap criterion:

**Definition 4** (overlap criterion)**.** *(taken almost verbatim from [2]) An overlap criterion is a mapping $\perp$: $\mathcal{P}(\mathcal{P}(E)) \to \{0,1\}$ such that $\perp$ is decreasing. That is for any $\mathcal{A}, \mathcal{B} \subseteq \mathcal{P}(E)$:*

$$\mathcal{A} \subseteq \mathcal{B} \Rightarrow \perp(\mathcal{B}) \leq \perp(\mathcal{A})$$

We then consider a family of sets $\mathcal{A}$ to be overlapping when $\perp(\mathcal{A}) = 1$. An overlap condition is required to be decreasing to ensure that a non-overlapping family cannot be made overlapping by adding sets. This allows us to define a hyperconnectivity quite analogously to definition 3:

**Definition 5** (Hyperconnectivity)**.** *We call a family $\mathcal{H} \subseteq \mathcal{P}(E)$ a hyperconnectivity under overlap criterion $\perp$ if and only if both of the following hold:*

$$\forall x \in E.\ \{x\} \in \mathcal{H}$$
$$(\forall \mathcal{A} \subseteq \mathcal{H}.\ \perp(\mathcal{A}) = 1) \Rightarrow \left(\bigcup \mathcal{A} \in \mathcal{H}\right)$$

*We then call a set $C$ hyperconnected if and only if $C \in \mathcal{H}$*

It is immediately clear that connectivity is a special case of hyperconnectivity for the overlap criterion:

$$\perp(\mathcal{A}) = 1 \iff \bigcap \mathcal{A} \neq \emptyset$$

With this definition, we can immediately generalize connected components to hyperconnected components:

**Definition 6** (Hyperconnected Components)**.** *Given a binary image $I \subseteq E$ and hyperconnectivity $\mathcal{H}$, we call $C \subseteq I$ a hyperconnected component if and only if:*
*$C \in \mathcal{H}$ and $\nexists B \in \mathcal{H}. C \subset B \subset I$*

Note that the hyperconnected components no longer form a partition of $I$. Instead, they merely cover $I$. Connected filters allow a similar generalization to hyperconnected filters.

**Definition 7** (Hyperconnected Filter)**.** *Let $\lambda$ be a selection criterion and $\mathcal{H}$ a hyperconnectivity. These give rise to a Hyperconnected filter $\psi_\Lambda : \mathcal{P}(E) \to \mathcal{P}(E)$. To define $\psi_\Lambda(X)$, we first need to define:*

$$H_X = \{A \subseteq X \mid A \in \mathcal{H}\}$$

$$\Gamma_\Lambda(C) = \begin{cases} C & \textit{if } \Lambda(C) = 1 \\ \emptyset & \textit{otherwise} \end{cases}$$

*The hyperconnected filter then becomes:*

$$\psi_\Lambda(X) = \bigcup_{C \in H_X} \Gamma_\Lambda(C)$$

Note that, because we use a union here, if a point lies in any hyperconnected filter that passes the criterion, it is not filtered out. In the next section, we will show k-flat zones as an example of a hyperconnectivity and how this gives rise to a connected filter.

## 4 k-Flat Zones

Connected filters have been implemented using tree-based algorithms. One of these is the Max-Tree algorithm [3]. The tree constructed by this algorithm can be used to construct a hyperconnected filter, the $k$-flat filter [2].

### 4.1 Max-Tree

The Max-Tree algorithm, as described in [3], constructs a tree with the grey-value peak components at the nodes of the tree. A peak component $P_h$ is a connected component with the threshold set at $h$. This results in components which have a grey-value of $h$ or higher. Each node points to its parent $P_{h'}$ which has a lower height, $h' < h$. It is easily seen that the parent component is always larger then its child as the threshold is lower. These peak components and their parents form

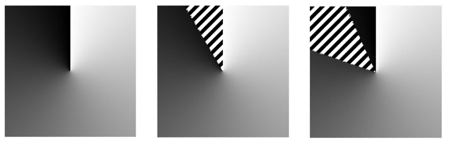
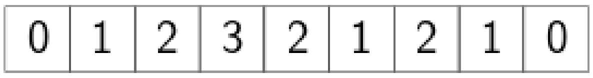
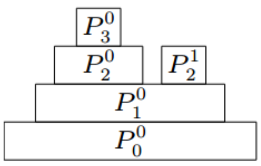


Fig. 3: Two $k$-flat zones which overlap where $k = 16$ [2]



(a) Input signal



(b) Peak components

$$\begin{tikzcd}
C_3^0 \arrow[d] & \\
C_2^0 \arrow[dr] & & C_2^1 \arrow[dl] \\
& C_1^0 \arrow[d] & \\
& C_0^0 &
\end{tikzcd}$$

(c) Max-Tree

Fig. 4: Simple 1D Max-Tree based on [6]

a tree structure as is shown in Figure 4c. To filter using a Max-Tree can be quite simple. There are several ways to construct a filter using the Max-Tree. A simple but effective method is subtractive filtering [5]. Whenever a peak component is filtered out due to the selection criterion $\Lambda$ then subtractive filtering lowers its gray-value to the highest surviving ancestor. Furthermore the grey-value of all its decedent are also lowered by the same amount.

### 4.2 Defining k-flat

A hyperconnected filter which can now be defined is the $k$-flat zone filter [2]. A $k$-flat zone is a region where the total grey-value variation is bounded by $k$. An important thing to note is that these zones may overlap as seen in figure 3.

Because these areas can overlap it is not suited for a connected filter. With a proper hyperconnected overlap criterion we can construct a hyperconnected filter using this principle. A $k$-flat zone can be defined formally as follows:

**Definition 8** ($k$-flat zone)**.** *A $k$-flat zone $F_{k,h}$ at level $k$ and height $h$ is a set of pixels path-wise connected to $x \in E$ with a grey-value from $h$ to $h$-$k$:*

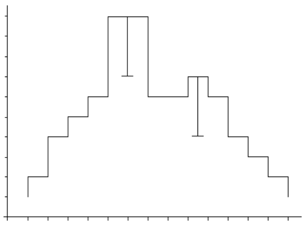
$$F_{k,h}(x) = \{p \in x \mid h - k \leq f(p) \leq h\}$$

Based on this definition an overlap criterion can be contructed:

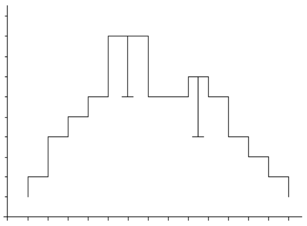
$$\perp_j^k (\{H_j\}) = 1 \text{ iff } \bigcup H_j \neq \emptyset \wedge \max_{p,q \bigcup H_j} \|f(p) - f(q)\| \leq k$$

The corresponding hyperconnectivity is:

$$\mathcal{H}_\tau = \{\emptyset\} \cup \{A \subseteq \mathcal{C} \mid \|f(p) - f(q)\| \leq k \, \forall p, q \in A\}$$

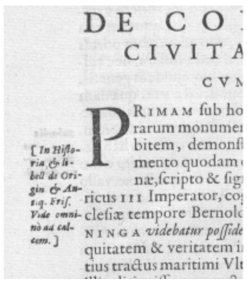


(a) Original signal

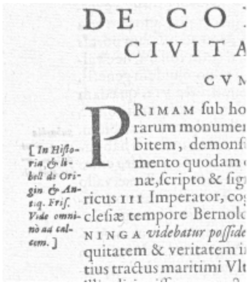


(b) Signal after filtering

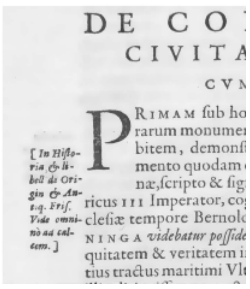
Fig. 5: $k$-substraction filtering on a 1-D signal where $k = 3$



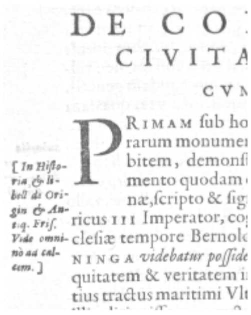
(a) Original image



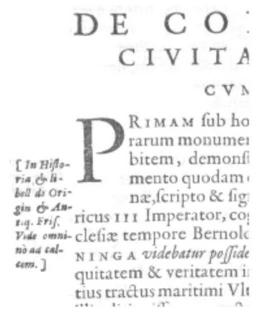
(b) Rolling ball with radius 15



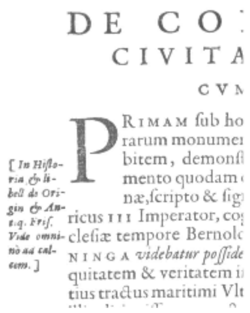
(c) Anisotropic diffusion



(d) Area attribute filter with $10 \leq$ area $\leq 8000$ and $k = 0$



(e) The same area attribute filter with $k = 60$



(f) $k$-absorption rule with $k = 40$

Fig. 6: Digitisation of *De Agro Frisae* by Ubbo Emmius using different filters. Images taken from [2]

With this hyperconnectifity 4.2 we can alter the Max-Tree such that it replaces its peak components with $k$-peak components. Now a regional maximum is needed just like the peak components of the Max-Tree. a $k$-regional maximum $M_{h,k}$ is a $k$-flat at height $h$ zone which has neighbours smaller than $h - k$. One can see that these corresponds to peak components of $P_{h-k}$. As the $k$-peak components corresponds to the normal peak components it can be implemented as a filter over the Max-Tree. This also allows the use of multiple $k$-flat zones filers to be applied by generating one Max-Tree.

### 4.3 Filtering using k-flat

For $k$-flat zones the subtractive filter is not suitable. Applying it can violate the idempotent condition of an connected filter. For this reason we will need a new filter. The $k$-flat zone subtractive filter will need to preserve decisions upwards within a range of $k$ above a preserved ancestor. A way of doing this is adding a propagation range $k'$ to the nodes of the max tree. for any preserved nodes this is equal to $k$. If a node $P_{h'}$ should be discarded based on some criterion we need to check whether it lies within the propagation range of its parent $P_h$. If $\Delta h = h' - h \leq k'_h$ then it lies within the propagation range which means we preserve the node. $P_h$ then gets a different propagation value, $k'_h = k'_{h'} - \Delta h$. However if $\Delta h > k'_h$ then it falls outside of the propagation range. In this case the node has to be removed and $k'_{h'}$ is set to zero. In the subtractive the gray-value would be set to the grey-value of its parent. With $k$-subtractive the grey-value will be set to the grey-value of its parent $P_h$ plus its propagation range $k'_h$. These rules allow $k$-subtraction to be idempotent. The results of these rules can be seen in figure 5. In this signal two components do not satisfy $\Lambda$ and are being considered for filtering. The first component lies outside of the propagation range. For this reason we lower it to the value of his parent plus its propagation rate. However the second component lies withing the propagation range and is preserved.

For some applications using $k$-subtraction is not optimal. An alternative to this method is $k$-absorption. It is able to filter low contrast features which touch high contrast features. It allows $k'$ to be negative when a feature is rejected by the attribute criterion and it is more then $k$ lower then the peak. When a node has a negative $k'$ we need to absorb into the background.

## 4.4 Example using k-flat

$k$-absorption has uses within digitisation of books [2]. When digitising books it can be a problem that the letters on the other side of the page show through. One would like to flatten the background and the letter from the backside of the page. While digitizing *De Agro Frisae* by Ubbo Emmius this problem occurred [2]. In figure 6 shows that the letters from the backside of the page showed through and hindered the character recognizer. Several filters were used to combat this issue. A rolling ball filter 6b, anisotropic 6c diffusion and area attribute filter were used for this 6d. They were not able to suppress all the letters from the backside. Using $k$-flat filters significantly increased the result 6e. Taking a closer look at the title and the side note in 6e reveals some noise which has not been filtered out. For these details $k$-absorption is a better method. Using $k$-absorption with $k = 30$ gave the best result out of all these filters.

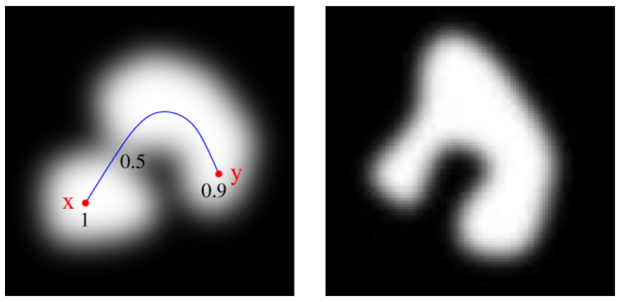
## 5 APPLICATIONS TO CONNECTIVITY FOR FUZZY SETS

Besides allowing us to define hyperconnected filters (like K-flat zones), hyperconnectivity also sees application in defining a connectivity measure for fuzzy sets. Fuzzy sets are a generalization of normal sets. Fuzzy sets allow elements to be partially in the set. The membership of an element to the set lies on a scale rather than being binary. Formally, a fuzzy set on space $X$ is denoted by a membership function: $\mu : X \to [0,1]$. $\mu(x)$ denotes the degree to which an element $x \in X$ belongs to the fuzzy set with 0 indicating no membership at all and 1 indicating full membership. The approach taken in this section is based largely on [1]. However, this section is agnostic with regards to the underlying connectivity, while the article it is based on assumes path connectivity.

One can characterize a fuzzy set $\mu$ entirely by its so called $\alpha$-cuts: $(\mu)_\alpha$. These are defined as all elements that belong to $\mu$ with degree $\alpha$ or more. That is:

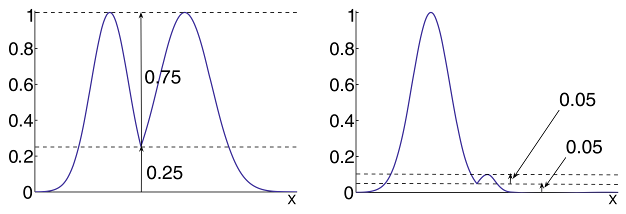
$$(\mu)_\alpha = \{x \in X \mid \mu(x) \geq \alpha\}$$

Like for normal sets, the union and intersection of fuzzy sets also exist.



(a) A disconnected fuzzy set with $C(\mu) = 0.5$ (b) A (fully) connected fuzzy set with $C(\mu) = 1$

Fig. 7: examples of fuzzy sets and how they are connected. Images taken from [1]



(a) A fuzzy set with $C(\mu) = 0.25$ (b) A fuzzy set with $C(\mu) = 0.05$

Fig. 8: Counter intuitive examples of 1D fuzzy sets and their connectivity. Images taken from [1]

These correspond respectively to the maximum and minimum membership of the elements, that is:

$$\left(\bigcup_i \{\mu_i\}\right)(x) = \max_i\{\mu_i(x)\}$$
$$\left(\bigcap_i \{\mu_i\}\right)(x) = \min_i\{\mu_i(x)\}$$

There are many applications for fuzzy sets. One example is simply allowing the expression of uncertainty when it comes to segmentation. $\alpha$-Cuts are the natural way to then move from fuzzy sets to normal (or ‘crisp’) sets. These $\alpha$-cuts will play a large role in our first real connectivity for fuzzy sets.

**Definition 9** (a connectivity for fuzzy sets)**.** *We call a fuzzy set $\mu$ connected for some connectivity $\mathcal{C}$ over $X$ if and only if:*

$$\forall\alpha \in [0,1] : (\mu)_\alpha \in \mathcal{C}$$

This condition is equivalent to $\mu$ not having ‘disconnected’ local maxima. An example can be found in figure 7. In 7a one can see that any $\alpha$-cut with $\alpha > 0.5$ will result in the $\alpha$-cut being disconnected. When looking at $\alpha$-cuts of 7b it becomes clean that any $\alpha$-cut is still connected. This is a ‘crisp’ definition of connectivity for a connected set. Since we are working with fuzzy sets, it would be reasonable to expect a similar fuzzy measure of connectivity. This can be achieved by looking at whether the different $\alpha$-cuts are connected. One might then say that a fuzzy set $\mu$ has a connectivity measure equal to the highest $\tau$ so that all $\alpha$ cuts up to $\tau$ are connected, that is:

$$C(\mu) = \max\{\tau \in [0,1] \mid \forall\alpha \leq \tau : (\mu)_\alpha \in \mathcal{C}\}$$

Examples of how this works can be seen in figure 7.

Hyperconnectivities come in to play when we start considering all $\mu$ with a minimum connectivity of $\tau$. Such a set gives us a hyperconnectivity with overlap criterion:

$$\perp_\tau (\{\mu_i\}) = 1 \text{ iff } \forall\alpha \leq \tau : \bigcap_i (\mu_i)_\alpha = \emptyset$$

The corresponding hyperconnectivity is:

$$\mathcal{H}_\tau = \{\mu \mid C(\mu) > \tau\}$$

This definition does have some issues. These are illustrated in figure 8. Intuitively, one would say that 8b is more connected than 8a and yet, 8a has a much higher connectivity than 8b. The degree of connectivity is determined by the height of the lowest saddle point (or non-edge minimum in the 3D case). It does not take into account how deep a cleft a saddle point creates.

## 6 Conclusion

We set out to see how hyperconnectivity generalizes connectivity. How this generalization improves upon connectivity, and what is lost in this generalization. Very little turns out to be lost. The biggest change is that one can not rely on the hyperconnected components of an image to be a partition. One might also say that hyperconnectivity gives more choices, making the resulting methods less practical to implement. However, the intuitive example of $K$-flat zones and its simplicity should serve as counterpoint to this charge.

What is gained is versatility. This gives access to more powerful filters such as the $K$-flat zones which outperforms many connected filters in certain areas by having the restriction of connected components forming a partition lifted. Furthermore, we saw how hyperconnectivity comes up naturally when trying to define connectivity for fuzzy sets. Whilst the resulting definition still has some flaws, it certainly shows the descriptive potential of the framework of hyperconnectivity.

With regards to fuzzy sets, in [1] the definition presented here is extended to rectify some of these flaws. However, the entire paper uses path connectivity (i.e. 4- or 8-connectivity) as a basis. We would consider it interesting to see if the approach of this paper could be extended based on any general connectivity.

There is much more to connectivity and hyperconnectivity than we have shown here. For example, whilst we have been working with sets, all of this can readily be extended to so called lattices. In fact, our two main sources: [1] and [2] are both formulated for lattices. There is also a much more theoretical result that shows an equivalence between all algebraic openings over lattices and hyperconnections over lattices [7]. One does not need to know what these terms mean to realize that the uses of hyperconnectivity extend far beyond what we have explored in this paper.

That final point sums up our conclusion. Hyperconnectivity is a very broad framework. Within it, both many things already known and things yet to be discovered can be described. As such, this is an interesting and exciting field of research that shows a lot of potential.

## Acknowledgements

The authors wish to thank the expert reviewer Dr M.H.F. Wilkinson for his suggestions and support. They also wish to thank their fellow students whom reviewed this paper. Our paper truly benefited from our reviewers' suggestions and remarks.

## References

[1] O. Nempont, J. Atif, E. Angelini, and I. Bloch. A new fuzzy connectivity measure for fuzzy sets. *Journal of Mathematical Imaging and Vision*, 34(2):107–136, 2009.

[2] G. Ouzounis and M. Wilkinson. Hyperconnected attribute filters based on k-flat zones. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 33(2):224–239, Feb 2011.

[3] P. Salembier, A. Oliveras, and L. Garrido. Antiextensive connected operators for image and sequence processing. *Image Processing, IEEE Transactions on*, 7(4):555–570, Apr 1998.

[4] J. Serra. Connectivity on complete lattices. *Journal of Mathematical Imaging and Vision*, 9(3):231–251, 1998.

[5] E. Urbach, J. Roerdink, and M. Wilkinson. Connected shape-size pattern spectra for rotation and scale-invariant classification of gray-scale images. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 29(2):272–285, Feb 2007.

[6] M. Westenberg, J. Roerdink, and M. Wilkinson. Volumetric attribute filtering and interactive visualization using the max-tree representation. *Image Processing, IEEE Transactions on*, 16(12):2943–2952, Dec 2007.

[7] M. Wilkinson. Hyperconnections and openings on complete lattices. In P. Soille, M. Pesaresi, and G. Ouzounis, editors, *Mathematical Morphology and Its Applications to Image and Signal Processing*, volume 6671 of *Lecture Notes in Computer Science*, pages 73–84. Springer Berlin Heidelberg, 2011.

# Comparing COSFIRE Filters with Neural Networks in Visual Pattern Recognition

Sweta Singh and Niels Kluiter

**Abstract**—Object detection is one of the most prominent applications in computer vision. Keypoint detection is the initial step in many object detection algorithms. Challenges for most existing methods are the wide range of existing shapes and textures, sensitivity to illumination, presence of noise and object repetitions. This is the reason for extensive ongoing research in this field.
In our research we study the two recently developed algorithms in object detection and pattern recognition,namely Combination Of Shifted FIlter REsponses (COSFIRE) filters and deep learning(with convolutional neural networks). COSFIRE filters are inspired by the mechanism of neurons in the human visual cortex, and also convolutional neural networks are inspired by the mechanism of human visual nervous systems.
The goal of this paper is to bring about a comparison between these two algorithms. As there is a broad range of keypoint detection and pattern recognition problems, we will look at a subset of it for which the methods in question are most frequently used, like handwritten character and traffic sign recognition. We will study the inner working of the algorithms and their performance on well known and large datasets like MNIST.
From a detailed study of the two methods we understand that they are two distinct approaches where COSFIRE filters are key feature based detectors and Deep neural networks do not need any of it. The network adapts to have enough output nodes to represent all possible output classes.

**Index Terms**—COSFIRE filters, Convolutional neural networks, Deep learning, Keypoint detection, Pattern recognition

## 1 Introduction

Computer vision can be described as: 'making the computer observe visual information and understand it'. This is rather a complicated and probabilistic task. In 1963 L.G. Roberts, a PhD student at MIT, proposed to let machines distinguish between three-dimensional objects like cylinders, balls and cubes[27]. Dr. Seymour Papert, a mathematician and one of the pioneers of Artificial Intelligence, also at MIT, picked this up and proposed a summer project on this in the year 1966[20]. He later extended this research to identify more complex structures. This was the beginning of research in the area of computer vision. All at once in 90's, there was an overflow of techniques that were proposed to solve computer vision related problems, like object classification, detection and segmentation. This led to a new generation of smart systems. Important successes in computer vision can be contributed to key feature detection and representation algorithms. This included SIFT, Bag of Words and local keypoint extraction algorithms. The introduction of Support Vector Machines further revolutionized the field of Computer vision. This was coincidentally also the time when social media were in phase of revolution, which led to availability of enormous data. This, together with an exponential rise in computing power with advent of stronger CPU's and GPU's, has given rise to heavier methods like deep learning networks, which achieve close to the accuracy of humans in certain tasks.

Today, computer vision is used in object detection, stereo vision and many such applications. In our study, We compare the recently developed trainable COSFIRE filters method and deep convolutional neural networks in the field of object recognition and object detection. We discuss the working principles of these two methods and examine their shortcomings. We also study their practical performances in applications like the recognition of handwritten digits.

Visual recognition can be divided into two parts, image representation and image classification. Image representation includes keypoint detection, feature extraction and feature encoding. Keypoints are the points of interest that represent important aspects of an image. Fig. 1 shows some of the examples of keypoints, these can be for example corners, junctions or crossings of lines. Feature extraction is described by a multi-dimensional feature vector and is called descriptor. It represents the environment around a point of interest so as to capture the characteristic and indicative information about the image content. Examples of descriptors include SIFT and SURF. After application of feature detection and description algorithms, each image is represented by a set of multidimensional vectors, which is transformed into a single vector representation. Classifiers like support vector machines are trained on these data to learn the features that are to be tested. [3]

• *Sweta Singh is with University of Groningen. E-mail: rnsweta@gmail.com.*
• *Niels Kluiter is with University of Groningen. E-mail: nl.kluiter@gmail.com.*

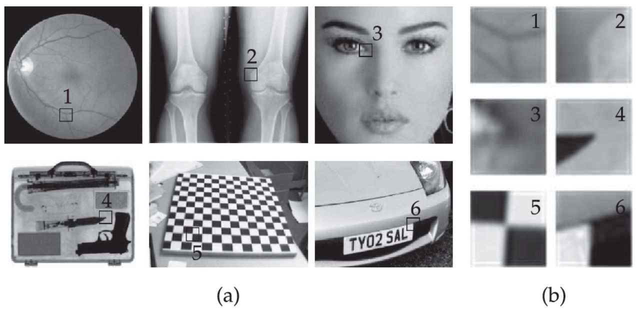


Fig. 1: (a)Examples of keypoints in form of corners and junctions. (b)Keypoints enlarged[1]

Computer vision, like the term suggests, tries to incorporate the vision happening in the human brain into computers. Studies suggest that ventral visual and dorsal visual pathways are the two major cortical systems that process the visual information. The primary function of the ventral visual system is to recognize and identify properties of objects like shape and color. Fig. 1 shows the ventral visual pathway of a rhesus monkey, that is very similar to that of a human being and that explains the manner in which objects are identified at different levels in different parts of the visual system of the brain. In the eyes an object is projected at the retina, this information is processed in different parts of the visual cortex based on the complexity of the identified object[7]. Fig. 2 shows the ventral visual pathway that is involved in object identification. COSFIRE filters are inspired by shape selective

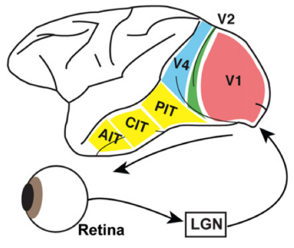


Fig. 2: Locations of the ventral stream cortical area in the macaque monkey brain and the flow of visual information from the retina[8].

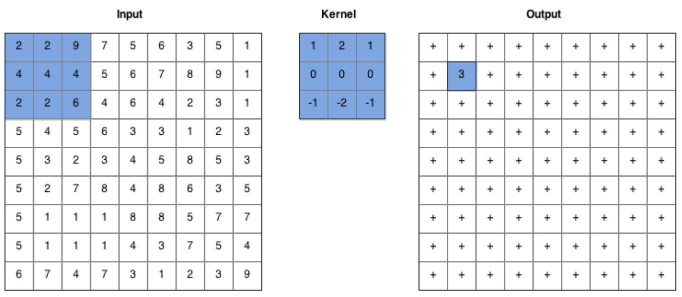


Fig. 3: The process of convolution on images, the kernel is applied to a patch of the input image, here highlighted in blue, the values are multiplied element-wise and then added together. The result is the value of the pixel in the output image that correspond the center of the patch from the input image. Here this value is shown in blue, calculated as $1 \times 2 + 2 \times 2 + 1 \times 9 + 0 \times 4 + 0 \times 4 + 0 \times 4 + -1 \times 2 + -2 \times 2 + -1 \times 6 = 3$. The input patch is then shifted one pixel, to calculate new output pixels with the same kernel, until a complete output image has been calculated.

neurons in area 'V4' that exhibits selectivity of (curved) contours or of combinations of line segments[1][21] [22].

Deep convolutional neural networks were also inspired by the human visual nervous system [9]. Many studies suggest that the human brain recognizes objects starting from simpler objects at the 'V1' region and the most complex objects in the inferior temporal cortex region in a cascading reflexive feed-forward manner[8]. This is emulated by deep learning methods where depth corresponds to number of layers of neurons in the system.

In this paper we compare these convolutional neural networks, because it has proven itself to be the best at several pattern recognition tasks with COSFIRE filters, which show great potential, but have have not yet proven themselves. This to see if COSFIRE filters could be a viable method to use for certain pattern recognition and keypoint detection tasks. We do this by first studying how both methods work, this is explained in section 2. In section 3 we analyze both methods and find their advantages and disadvantages and also look at some practical results, these are then discussed in section 4. Section 5 proposes some future work, after which the paper is concluded in section 6.

## 2 Methods

Many methods for doing keypoint detection and pattern recognition tasks exist today. In this paper we focus on two successful methods: COSFIRE filters and deep learning in convolutional neural networks. In the following sections we will explain how these methods work and how the aforementioned tasks were done traditionally.

### 2.1 Traditional machine learning methods

Many problems in computer vision could have been easily solved if we were able to represent the data in the right form. For example one pixel of an image would give little information on what it represents. But if this data can be transformed into a high level representation, it could tell if the image was a face or a hand et cetera. In the past, people would hand an engineer the mapping from raw input, like pixels, to features that they could use. These features were then fed to machine learning algorithms. Success of algorithms like deep learning show that we do not need to hand engineer and computers can be used for this job in a much better way. A large data set is chosen to learn the process of transforming the raw data into high level representations.

### 2.2 Neural networks

Neural networks have been a field of research for more than six decades now. Throughout their history, neural networks have had a typical architecture; multiple layers of interconnected nodes, representing biological neurons. They generally have an input layer, an output layer and often one or more hidden layers in between. The values of the output nodes are calculated from layer to layer using weights on the inter-node connections. By backward propagation these weights can be updated to train the network[19].

#### 2.2.1 Deep learning

Prior to 2006, research in neural networks mainly focused on architectures with up to two hidden layers, these are called 'shallow architectures'. Researchers preferred to work with these, because architectures with more layers, called 'deep', were hard to both prove and train. This changed due to the work of several researchers [15][16], LeCun et al.[26] and Bengio et al.[2] in the years starting from 2006. In these studies, new training strategies for neural networks were researched, making it easier to train nets with a deep architecture. This opened the door for several neural networks algorithms, which could now be used more effectively for several applications, like speech recognition or image reconstruction.

#### 2.2.2 Convolutional Neural Networks

A method that is especially good for object detection is called 'convolutional neural networks'. This form of neural nets has been introduced in the 1980's by K. Fukushima, at that time called the neocognitron [9]. In the following years it was picked up and enhanced by various research groups, up till 2006, when the new developments in deep learning added new opportunities for convolutional neural networks.

The main field for which these networks have been used is pattern recognition in images. This is why the nodes in the networks are generally aligned in 2D grids, while it could just as well be 3D or 1D for example. Such grids form the initial layers of the convolutional neural network. The layers in these networks can be of three forms: convolutional, sub-sampling and fully connected, below these are explained in detail:

**Convolutional layer** The type of layer from which the network got its name. Each node in the grid (or image) that forms this layer is connected to a square section of the grid(s) in the previous layer. The weights on these connections are the same for each node in the convolutional layer. This causes the weights to have the same effect as a convolution kernel, as you can see in Fig. 3. Convolution is a process that is used in processes like edge detection. In edge detection multiple kernels are usually used to detect edges in different directions. Similarly in convolutional neural networks, the goal is also to detect multiple features (not necessarily edges) instead of one. This is why in the convolutional layer there are usually not one, but multiple grids, being the results of multiple different sets of 'kernel' weights.

Also the previous layer could consist of multiple grids, which is why the nodes in a convolutional layer are connected to *all* grids in the previous layer.

**Sub-sampling layer** A sub-sampling layer does exactly what its name suggests, it sub-samples the grids in the previous layer, which is a convolutional layer, to reduce it in size. Every grid in the convolutional layer is connected to one smaller grid in the sub-sampling layer. Every node in the smaller grid is connected to a small section

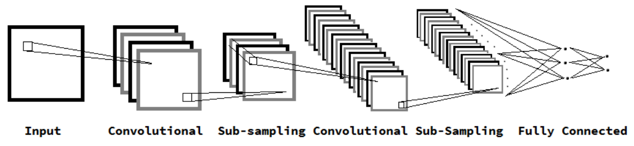


Fig. 4: The structure of a convolutional neural network. It starts with alternating convolutional and sub-sampling layers, then derives a result through fully connected layers.

of nodes from the convolutional layer grid, from which its value is determined. This can be done in multiple ways, like averaging the input values, but in most cases taking the maximum input value is chosen. This particular process is called 'max-pooling'.

Fully connected layer The initial layers of the network are formed by alternating convolutional and sub-sampling layers, this results in a large number of small grids/images. At the final stage there are so called fully connected layers, in which every node is connected to *all* nodes in the previous layer. These layers are used to get from the great amount of nodes/pixels to the desired final result.

Together the layers form a network like in Fig. 4. As happens with all neural networks, the weights between the layers of node are trained by introducing example inputs for which the output is known. The weights can be updated to make the response from the network approach the expected output. In convolutional networks, the weights are only trained in the convolutional layers and the fully connected layers. This is because in the sub-sampling layers the max-pooling process is done, which is a fixed procedure.[12]

### 2.3 COSFIRE Filters

A COSFIRE filter is a trainable filter proposed in by G. Azzopardi and N. Petkov[1] that can be used for keypoint detection and pattern recognition. How the process of training and applying these filters works is explained in the following sections.

#### 2.3.1 Method

Combination Of Shifted REsponses (COSFIRE) filters are contour based detectors. The responses of these filters are calculated as the weighted geometric mean of the shifted responses of simpler orientation selective filters (for example Gabor filters, as explained in section 2.3.2). In order to obtain the shifted responses, the corresponding supports at different locations are combined to obtain a bigger support in form of a sophisticated COSFIRE filter. For this computation, instead of the arithmetic mean, the geometric mean was considered. This is because of two reason. One one hand because it is resistant to contrast variations. And more importantly, by the psycho-physical fact that curved contours are identified by the activity of neurons by multiplication of responses from the sub-units, that are sensitive to different parts in curve patterns[11]. This helps COSFIRE filters to produce the responses only when all constituent parts of the pattern of interest are present[1] . A COSFIRE filter works in different stages. It first needs to apply the selected Gabor filters. The output of Gabor filters goes through Gaussian blurring, whose response is shifted by distinct vectors and the shifted responses are multiplied in order to calculate the weighted geometric mean, which determines the final response. The basic working of COSFIRE filters is explained in Fig. 5. The figure shows an input image with 3 vertices. The vertex that is encircled is considered as a prototype pattern of interest. This is used to automatically configure the COSFIRE filter that responds to same and similar test patterns. The two ellipses represent the dominant orientations in the surrounding region of the mentioned point of interest. These lines are detected by the symmetric Gabor filters. The circle in Fig. 5(b) shows the overlapping supports of a group of such filters.

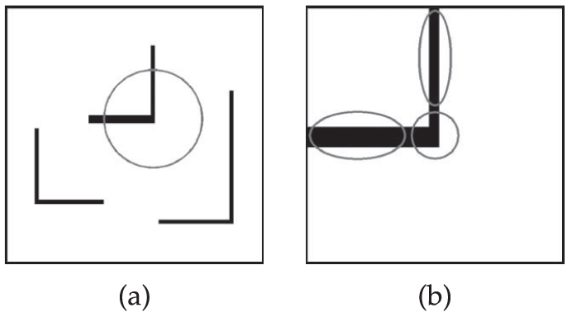


Fig. 5: Working Principle of COSFIRE Filters. Fig. (a) represents a synthetic image of size 256 X 256 pixels. The circle shows a prototype pattern of interest chosen by the user. Fig. (b) is an enlarged version of Fig. (a). It shows ellipses representing the support of line detectors that applicable for the concerned feature.[1]

The response of these filters in the centers of the corresponding ellipses are multiplied to give the response of the COSFIRE filter. The selected orientations and the locations of the response are found by studying the local prototype pattern used in configuring the particular COSFIRE filter. The Gabor filter responses at different locations in the vicinity of the point are shifted by different vectors. This is followed by pixel-wise evaluation of a multivariate function to give the output of COSFIRE filter.

#### 2.3.2 2D Gabor filters

A Gabor filter is a linear filter. Frequency and orientation representations of Gabor filters are comparable to those of the visual system of human beings. They are found to be suitable especially for texture representation and discrimination. A Gabor filter is made by modulating a sinusoid by a Gaussian. The response of a Gabor filter can be represented by $g_{\lambda,\theta}(x,y)$ where $\lambda$ and $\theta$ are the selected wavelength and orientation respectively. In order to increase the object detecting property, the Gabor filters are enhanced with surround inhibition by which the texture edges are suppressed leaving the contours of the object and the region boundaries unaffected. Other aspects of the filter like aspect ratio, bandwidth and phase offset are not considered in the COSFIRE implementation[14][17][23][24][25][28]. The responses of the Gabor filters are normalized in order to keep the sum of all positive responses to 1 and their negative responses to -1. All the responses are thresholded to $t_1$ of the maximum response $g_{\lambda,\theta}(x,y)$ for combination of values $(\lambda,\theta)$ at every point $(x,y)$ of the image.

#### 2.3.3 Process of COSFIRE filter configuration

The responses of the Gabor filters from the process explained in section 2.3.2 form the input to a COSFIRE filter. Each of these Gabor filters are defined by values $(\lambda_i,\theta_i)$ around each of the points $(\rho_i,\phi_i)$ with respect to the center of the COSFIRE filter. These four parameters $(\lambda_i,\theta_i,\rho_i,\phi_i)$ represent the properties of the contour in the region of given point of interest. The response of the bank of Gabor filters along the circle of given radius $\rho$ in the area of chosen point of interest is considered for configuration of the COSFIRE filter (see Fig. 6). At each of the positions along the circle, the maximum of all responses for all the possible values of $(\lambda,\theta)$ that are used in the bank of filters is considered. The positions with values higher than their corresponding values in the nearby positions along an arc of angle $\pi/8$ are chosen as points of dominant orientations in the region of a point of interest. For all these values, their polar coordinates with respect to the center of the filter are computed $(\rho_i,\phi_i)$. For each of these locations $(\rho_i,\phi_i)$, the corresponding Gabor filter responses $g_{\lambda,\theta}(x,y)$ higher than a fraction $t_2 = 0.75$ of the maximum value of $g_{\lambda,\theta}(x,y)$ throughout the combinations of values $\lambda,\theta$ are considered. For each value of $\theta$ that matches the condition, a single value of

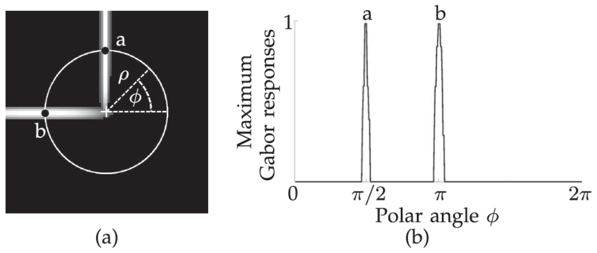


Fig. 6: Configuring a COSFIRE filter[1]

$\lambda$ with maximum response of Gabor filters across all $\lambda$ values is considered. Each unique pair of $\lambda, \theta$ and position $(\rho, \phi)$ constitutes a tuple $(\lambda_i, \theta_i, \rho_i, \phi_i)$. In this way, for each location $(\rho_i, \phi_i)$, there are multiple tuples possible. With this method of selection of parameters, the example considered with the point of interest shown in Fig. 6, for $\rho \in 0, 30$, the four tuples formed are:

$$
\begin{aligned}
S_f = \{ & \\
& (\lambda_1 = 8, \quad \theta_1 = 0, \quad \rho_1 = 0, \quad \phi_1 = 0) \\
& (\lambda_2 = 8, \quad \theta_2 = 0, \quad \rho_2 = 30, \quad \phi_2 = \pi/2) \\
& (\lambda_3 = 16, \quad \theta_3 = \pi/2, \quad \rho_3 = 0, \quad \phi_3 = 0) \\
& (\lambda_4 = 16, \quad \theta_4 = \pi/2, \quad \rho_4 = 30, \quad \phi_4 = \pi/2) \\
\} &
\end{aligned}
$$

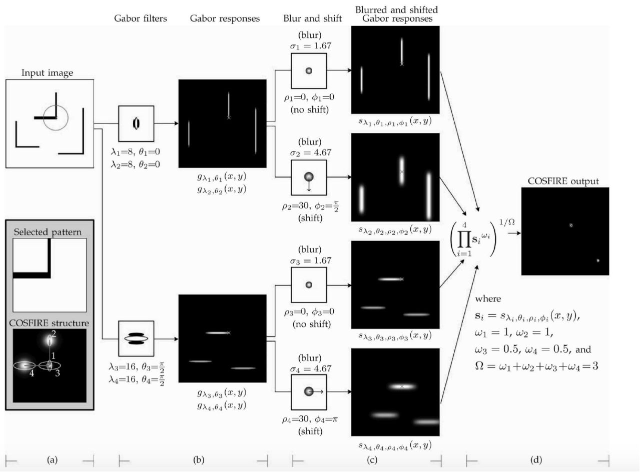

Gabor filters
Gabor responses
Blur and shift
Blurred and shifted Gabor responses
Input image
Selected pattern
COSFIRE structure
$\lambda_1=8,\ \theta_1=0$
$\lambda_2=8,\ \theta_2=0$
$\lambda_3=16,\ \theta_3=\frac{\pi}{2}$
$\lambda_4=16,\ \theta_4=\frac{\pi}{2}$
$g_{\lambda_1,\theta_1}(x,y)$
$g_{\lambda_2,\theta_2}(x,y)$
$g_{\lambda_3,\theta_3}(x,y)$
$g_{\lambda_4,\theta_4}(x,y)$
(blur) $\sigma_1 = 1.67$
$\rho_1=0,\ \phi_1=0$ (no shift)
(blur) $\sigma_2 = 4.67$
$\rho_2=30,\ \phi_2=\frac{\pi}{2}$ (shift)
(blur) $\sigma_3 = 1.67$
$\rho_3=0,\ \phi_3=0$ (no shift)
(blur) $\sigma_4 = 4.67$
$\rho_4=30,\ \phi_4=\pi$ (shift)
$s_{\lambda_1,\theta_1,\rho_1,\phi_1}(x,y)$
$s_{\lambda_2,\theta_2,\rho_2,\phi_2}(x,y)$
$s_{\lambda_3,\theta_3,\rho_3,\phi_3}(x,y)$
$s_{\lambda_4,\theta_4,\rho_4,\phi_4}(x,y)$
$\left(\prod_{i=1}^{4} \mathbf{s}_i^{\omega_i}\right)^{1/\Omega}$
COSFIRE output
where
$\mathbf{s}_i = s_{\lambda_i,\theta_i,\rho_i,\phi_i}(x,y)$,
$\omega_1 = 1,\ \omega_2 = 1$,
$\omega_3 = 0.5,\ \omega_4 = 0.5$, and
$\Omega = \omega_1+\omega_2+\omega_3+\omega_4=3$
(a)
(b)
(c)
(d)

Fig. 7: Different stages of the working of COSFIRE filters, namely configuration (a) and blurring and shifting. An input image has a size $256 \times 256$ pixels. A point of interest is shown by a circle around a vertex. To detect the horizontal and the vertical lines, two anti-symmetric Gabor filters are used. The output of a COSFIRE filter is computed as the weighted geometric mean of all the blurred, shifted and thresholded Gabor filter responses. The two local maxima in the output response of the COSFIRE filter correspond to two similar vertices of the input image.[1]

Let us consider the second tuple in $S_f$: $(\lambda_2 = 8, \theta_2 =, \rho_2 = 30, \phi_2 = \pi/2)$. This describes the contour path with a width of $\lambda_2/2 = 4$ pixels and an orientation angle of $\theta_2 = 0$ which can be detected by the filter with selected wavelength $\lambda_2 = 8$ and orientation $\theta_2 = 0$ at a position of $\rho_2 = 30$ pixels above the point of interest shown by point *b* in the Fig. 6.

### 2.3.4 Blurring and shifting of responses from Gabor filters

As seen in the previous section, for the given point of interest there were four strong responses produced, each at different positions from the center of the filter. These responses are first blurred in order to bring in some tolerance in the position of corresponding contour parts. This, with the help of a Gaussian function $G_\sigma(x,y)$ with standard deviation $\sigma$, is used to compute the weighted responses. The maximum of this value is considered as blurring.

$$\sigma = \sigma_0 + \alpha\rho. \tag{1}$$

In the equation (1), $\sigma_0$ and $\alpha$ are constants. The orientation bandwidth can be increased by increasing the value of $\alpha$. All the responses are shifted to bring them towards the center of the filter.

### 2.3.5 COSFIRE filter response

The response of a COSFIRE filter, as mentioned earlier, is a geometric mean of all the blurred and shifted responses of thresholded Gabor filter responses. It can be represented as in (2):

$$r_{s_f}(x,y) = |(\prod_{i=1}^{|s_f|}(s_{\lambda_i,\sigma_i,\rho_i,\phi_i}(x,y))^{\omega_i})^{1/\Sigma_{i=1}^{|s_f|}\omega_i}|_{t_3} \tag{2}$$

Equation (2) shows that the response is thresholded at a fraction $t_3$ of the maximum value of across all the coordinates $(x,y)$ of the image.

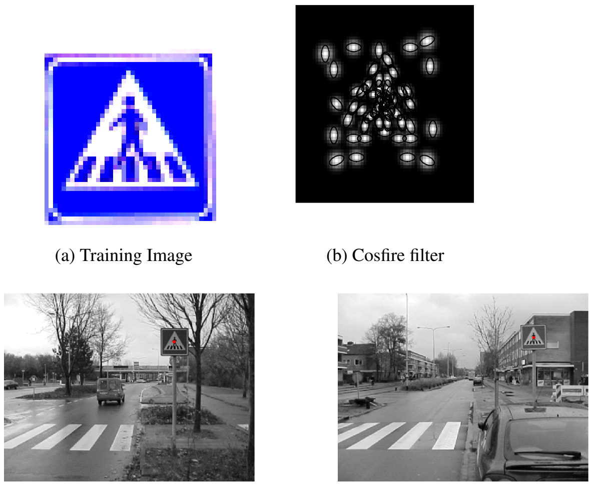
Fig. 7 shows the different stages of the working of the COSFIRE algorithm. It shows that the output response of a COSFIRE filter is determined by the geometric mean of the four blurred and shifted images that are produced as the responses of two Gabor filters. It can be noticed that the filter responds strongly to the horizontal line to the left of selected point of interest, the vertical line and the intersection of the horizontal and vertical line.

In section 2.3.3 it was mentioned that the prototype can be selected by the user. Nevertheless it is possible for the system itself to discover the pattern that is to be used for configuration. We mentioned the use of Gabor filters for detection of lines and edges, nonetheless any other orientation selective filters could be used. As can be seen in the examples above, the number of parameters $\rho$ increases with the complexity of the local pattern that is used in configuration of the filter.

In order to experiment with the algorithm, we executed the openly available COSFIRE algorithm on one prototype pattern and sixteen test images of complex scenes. The code was successful in correctly detecting the given traffic sign in all the complex scenes. A few of the images can be seen in Fig. 8.

## 3 Analysis

The two methods presented greatly vary in the way they work. COSFIRE filters are trained in a defined manner, so they are able to do reliable detection. It was able to detect traffic signs in a publicly available dataset[13] with training images like the one in Fig. 8a and detailed scenes for testing like in Fig. 8c and 8d. On the entire dataset, COSFIRE had a recall rate of 100%, that is, it was able to detect the traffic signs in all scenes. Similarly it had a very high recall rate of 98.50 percent on the DRIVE dataset in another test. The DRIVE dataset is a dataset of retina images, in which bifurcations of blood vessels need to be detected. This result was achieved by training four COSFIRE filters on selected bifurcations in one training image, so a 100% recall rate was achieved. The same four filters were then used to detect bifurcations in the training set. To be able to solve both detection problems, some parameters need to be changed in the algorithm. These include threshold values to suppress some filter responses, parameters regarding blurring of input filter responses and a set of parameters that determine which Gabor filters to use. Besides that the COSFIRE algorithm can be opted to run in different 'modes'. In vascular bifurcation detection the algorithm was used in rotation, scale and reflection invariant mode, while in traffic sign detection it was used in invariant mode. The algorithm can also be utilized to do recognition tasks. This requires some extra steps. Depending on how much a pattern can change, more filters need to be trained to capture all the varieties a pattern can have (for example the varieties in handwritten digits in Fig. 9). When a dataset is too large like the MNIST dataset (Mixed National Institute



(a) Training Image (b) Cosfire filter (c) test image (d) test image

Fig. 8: The algorithm used the prototype image of pedestrian crossing sign to create a COSFIRE filter. This was used in detection of pedestrian crossing sign in a complex scene. In both the scenes (c) and (d) it successfully detected the pattern, as can be seen by a red dot on the sign board.[13]

of Standards and Technology), a popular database for recognition of handwritten characters, a random subset of images from each class with a random local pattern can be used for training. A support vector machine is then used for their classification. Convolutional neural networks are on the contrary designed to do recognition tasks. They need little configuration specific to the object they are recognizing, apart from the configuration of the neurons in the fully-connected layers, that have to lead to a desired amount of resulting classes. They have been proven to be very effective in recognizing patterns like handwritten characters and traffic signs [18][10]. Neural networks are however not able to do detection tasks effectively on their own. For that another algorithm is needed that (finds and) presents parts of the image to the neural network to check whether a pattern is present there. An example of this is a sliding window, a 'window' is move over a scene and the part inside the window is presented to the neural network, the window then moves a few pixels and presents the new image. This is the simplest way to make convolutional neural networks compatible for detection tasks. More sophisticated methods exist however.

### 3.1 Experimental results

Many commercial applications like postal mail sorting and bank cheque processing require handwritten character recognition. This is still a challenging task. Feature recognition is a crucial step in effectiveness of these technologies.
COSFIRE filter attempts to successfully does this job[1]. In this process the trainable COSFIRE filter is first configured to detect specific parts of the image of the handwritten digits. Response of many of such filters is finally combined to form the shape descriptor of the given handwritten digits. The popular MNIST data-set was used to evaluate the performance of this method. The set contains about 60000 handwritten characters that can be used for training and another 10000 for testing. For configuration, a random subset of images of digits was taken from each class. In each of these images, a random location was chosen to configure a COSFIRE filter. After configuring around 500 COSFIRE filters and applying it to a test image, a vector with each element corresponding to a maximum response of a COSFIRE filter is created. This training in total took about ten seconds. The feature vectors of the digit images thus obtained were then used in training an all pairs multi-class support vector machine classifier with a linear kernel.

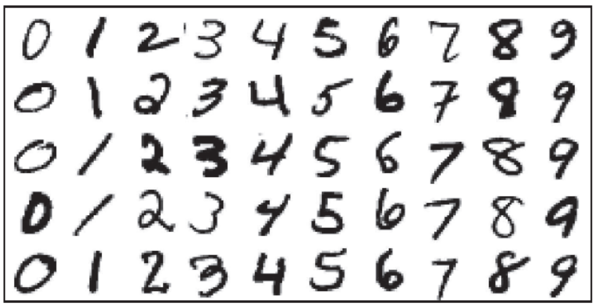


Fig. 9: Handwritten digit examples from the MNIST dataset[1]

Fig. 9 shows examples of handwritten digits from the MNIST dataset that was used to examine the performance of COSFIRE filters. As can be seen that different handwritings of the same digits vary in orientation, therefore the experiment was repeated several times by applying the COSFIRE filters in a partial rotation invariant mode with rotation tolerance angle $\psi$ ($\psi \in -\pi/4, -\pi/8, 0, \pi/8, \pi/4$) in equation (3).

$$\Re_{\psi}(S_f) = (\lambda_i, \theta_i + \psi, \rho_i, \phi_i + \psi) \forall (\lambda_i, \theta_i, \rho_i, \phi_i) \in S_f \tag{3}$$

Using this strategy, COSFIRE achieved a recognition rate of 99.48 percent, so it had a validation error of 0.52%.

Convolutional neural networks also claim to be particularly good at doing tasks like handwritten character recognition. Several convolutional neural networks methods have used this set to test their performance, the best result for such method at this point is a committee of 35 trained neural networks[5],[18] it had a validation error 0.23%. Training of these neural networks however took 14 hours per network, which would be a total of 490 hours if not trained in parallel. The best single network was presented one year earlier by the same researchers[6]. It was a convolutional network with seven hidden layers. Training it took a longer time than comparable networks with fewer layers, but it had a much lower validation error of 0.35%. Another frequently used benchmark in computer vision is for recognition of German traffic signs[10] known as GTSRB. In this benchmark a dataset of more that 50000 images are used with images that belong to more than 40 traffic sign classes. The same convolutional neural network as with the MNIST dataset is most successful with this dataset. It has a success rate of 99.46% which was better than the result of one of the humans that had to do the same tasks[4].

## 4 Discussion

From the previous section it becomes clear that there is not a lot of overlap in application to which the methods have been applied. Both have been applied to the MNIST handwritten character dataset, where convolutional neural networks proved to been more effective. However they required much more time to be trained on more dedicated hardware [5]. Apart from the MNIST dataset, there is no set to which both methods have been tested. This is due to their difference in design. COSFIRE filters are designed to do object detection tasks, while convolutional neural networks do pattern recognition tasks. Both algorithms have however been used to do the tasks for which the other is designed, but for both this requires extra steps. There is however evidence that there are more possible common grounds for the algorithms, is has for example been shown that both methods are able to handle patterns like traffic signs. When the features are well defined, keypoint detectors can be used easily. COSFIRE filters perform very well even when the key features are not defined, as it can automatically configure the filters by choosing random keypoints. Deep neural networks are also useful when the features are hard to be represented, but the only drawback is the computational cost it has to pay for the complexity of features.

## 5 Future work

From the limited amount of benchmarks on which both the methods have been tested we have only been able to get an idea of the relative performance of the two algorithm. To draw reliable conclusions on which algorithm is better both of them need to be tested on additional common benchmarks. We have not been able implement any of the necessary extensions on existing algorithms to do the task for which the other algorithm was designed. This is a task that could be done in future to support the conclusions made in this paper.

## 6 Conclusion

In this article two completely different computer vision methods have been examined, that are able to do similar tasks. The COSFIRE method considers key features (keypoints) as a prototype pattern. These keypoints can be used for detection and recognition problems. The other method, deep convolutional neural networks, consists of multiple layers of 'neurons' that have to be trained to automatically solve problems. We have compared both methods on the MNIST dataset in order to examine which ones perform better. With this dataset convolutional neural networks achieved a validation error as low as 0.25-0.32%. COSFIRE got a validation error of 0.52%. These results are quite close, however convolutional neural networks can sometimes take hours to train, while COSFIRE is often done training within 10 seconds. Using COSFIRE does however require the user to have some knowledge about the method itself, as a lot of configurations have to be made to receive optimal results for a certain task. To conclude, convolutional neural networks are probably the wises choice if accuracy is most important. However if efficiency also plays an important role, COSFIRE is probably more favorable.

## Acknowledgments

The authors wish to thank Toon Albers, Nicolai Petkov, Rogchert Zijlstra for reviewing and re-reviewing this paper, which made it greatly improve in quality.

## References

[1] G. Azzopardi and N. Petkov. Trainable cosfire filters for keypoint detection and pattern recognition. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 35(2):490–503, 2013.

[2] Y. Bengio, P. Lamblin, D. Popovici, H. Larochelle, et al. Greedy layer-wise training of deep networks. *Advances in neural information processing systems*, 19:153, 2007.

[3] E. Chatzilari, G. Liaros, S. Nikolopoulos, and Y. Kompatsiaris. A comparative study on mobile visual recognition. In *Machine Learning and Data Mining in Pattern Recognition*, pages 442–457. Springer, 2013.

[4] D. Cireşan, U. Meier, J. Masci, and J. Schmidhuber. Multi-column deep neural network for traffic sign classification. *Neural Networks*, 32:333–338, 2012.

[5] D. Ciresan, U. Meier, and J. Schmidhuber. Multi-column deep neural networks for image classification. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 3642–3649. IEEE, 2012.

[6] D. C. Ciresan, U. Meier, J. Masci, L. Maria Gambardella, and J. Schmidhuber. Flexible, high performance convolutional neural networks for image classification. In *IJCAI Proceedings-International Joint Conference on Artificial Intelligence*, volume 22, page 1237, 2011.

[7] J. J. DiCarlo and D. D. Cox. Untangling invariant object recognition. *Trends in cognitive sciences*, 11(8):333–341, 2007.

[8] J. J. DiCarlo, D. Zoccolan, and N. C. Rust. How does the brain solve visual object recognition? *Neuron*, 73(3):415–434, 2012.

[9] K. Fukushima. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological cybernetics*, 36(4):193–202, 1980.

[10] I. fr Neuroinformatik. *INI Benchmark*, 2014 (accessed March 2015). `http://benchmark.ini.rub.de/?section=home&subsection=news`.

[11] E. Gheorghiu et al. Multiplication in curvature processing. *Journal of Vision*, 9(2):23, 2009.

[12] A. Gibiansky. *Convolutional Neural Networks*, Feb. 2014 (accessed March 2015). `http://andrew.gibiansky.com/blog/machine-learning/convolutional-neural-networks/`.

[13] C. Grigorescu. *INI Benchmark*, (accessed March 2015). `http://www.cs.rug.nl/~imaging/databases/traffic_sign_database/traffic_sign_database.html`.

[14] C. Grigorescu, N. Petkov, and M. A. Westenberg. Contour detection based on nonclassical receptive field inhibition. *IEEE Transactions on Image Processing*, 12(7):729–739, 2003.

[15] G. Hinton, S. Osindero, and Y.-W. Teh. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554, 2006.

[16] G. E. Hinton and R. R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006.

[17] P. Kruizinga and N. Petkov. Non-linear operator for oriented texture. *IEEE Trans. on Image Processing*, 8(10):1395–1407, 1999.

[18] Y. LeCun. *THE MNIST DATABASE of handwritten digits*, 2012 (accessed March 2015). `http://yann.lecun.com/exdb/mnist/`.

[19] R. P. Lippmann. An introduction to computing with neural nets. *ASSP Magazine, IEEE*, 4(2):4–22, 1987.

[20] S. Papert. The summer vision project. 1966.

[21] A. Pasupathy and C. E. Connor. Responses to contour features in macaque area v4. *Journal of Neurophysiology*, 82(5):2490–2502, 1999.

[22] A. Pasupathy and C. E. Connor. Population coding of shape in area v4. *Nature neuroscience*, 5(12):1332–1338, 2002.

[23] N. Petkov. Biologically motivated computationally intensive approaches to image pattern recognition. *Future Generation Computer Systems*, 11(4-5):451–465, 1995.

[24] N. Petkov and P. Kruizinga. Computational models of visual neurons specialised in the detection of periodic and aperiodic oriented visual stimuli: bar and grating cells. *Biological Cybernetics*, 76(2):83–96, 1997.

[25] N. Petkov and M. A. Westenberg. Suppression of contour perception by band-limited noise and its relation to non-classical receptive field inhibition. *Biological Cybernetics*, 88(10):236–246, 2003.

[26] C. Poultney, S. Chopra, Y. L. Cun, et al. Efficient learning of sparse representations with an energy-based model. In *Advances in neural information processing systems*, pages 1137–1144, 2006.

[27] L. G. Roberts. *Machine perception of three-dimensional soups*. PhD thesis, Massachusetts Institute of Technology, 1963.

[28] N. P. S.E. Grigorescu and P. Kruizinga. Comparison of texture features based on gabor filters. *IEEE Trans. on Image Processing*, 11(10):1160–1167, 2002.

# Similarity metrics for psychological symptom graphs

J.D. van Leusen and S.F. de Bruijn

**Abstract**— Research has shown that comorbidity, a phenomenon where a patient exhibits multiple mental disorders simultaneously, is very common. In the field of psychopathology, the scientific study of mental disorders, the causes of comorbidity are currently a hot research topic. Psychological features, also known as psychological variables, are markers of some state of mind. Examples at any given moment may include cheerfulness, irritability, fatigue, but also degree of worrying. These features can be modeled by using a graph where the features are represented by the nodes and the causal relationships between the features are represented by the edges. The benefit of modeling the features by using a graph is that many algorithms exist to perform analysis on a graph, giving way to a large variety of possible analysis techniques to be used on this data. An example of such an analysis technique is *comparing* two graphs. Being able to compare these graphs is valuable as it provides insight into the similarity of the psychological features of two people. This in turn can help in understanding the causes of comorbidity. In this paper we provide an overview of existing graph comparison methods, while also proposing two new techniques to determine the similarity between two graphs. By theoretically evaluating the methods on both performance and accuracy we conclude that our proposed methods are in theory an optimal trade off between performance and accuracy.

**Index Terms**—Graph similarity, Psychological symptom graphs, Graph theory

## 1 Introduction

Research has shown that comorbidity, a phenomenon where a patient exhibits multiple mental disorders simultaneously, is very common [14]. In the field of psychopathology, the scientific study of mental disorders, the causes of comorbidity are currently a hot research topic. The exact causes of comorbidity are still unknown, but it is suspected to be related to the networked nature of psychological symptoms [4]. ”The networked nature of psychological symptoms” refers to the theory that psychological features are not self-contained. Instead they are thought to be affected by other psychological features.

Psychological features, also known as psychological variables, are markers of some state of mind. Examples of these features may include cheerfulness, irritability, fatigue, but also degree of worrying. Several techniques exist to assess these psychological features and diagnose mental disorders in patients. An example of such a technique is having the patient fill out regular surveys containing questions about their mood, among other things. The answers to these surveys are then used to measure psychological features such as cheerfulness and degree of worrying. These surveys are conducted over some period of time resulting in a time series representation of their psychological features. Having a time series of these features creates a pattern over time which can be used to detect the presence of mental disorders.

As mentioned earlier, these psychological features are not standalone; relationships exist between them. One important relationship is the causal relation where one feature is the cause of another. Modeling the relationships between these features can be done by using a graph. Such a graph represents the psychological features as nodes and depicts the relations between them as the edges between the nodes. One benefit of modeling the features by using a graph is that many algorithms exist to perform analysis on a graph, giving way to a large variety of possible analysis techniques to be used on this data.

An example of such an analysis technique is *comparing* two graphs. As each graph represents part of the psychological features and relationships, being able to compare these graphs is valuable as it provides insight into the similarity of the psychological features of two subjects. These subjects might be the same person at different points in time or two different individuals.

This in turn is valuable to determine the causes of comorbidity, as people may have different combinations of disorders while also having different, or similar, combinations of symptoms. Not only is it potentially useful for comparing the features of two people, if multiple graphs exist of the features of one person then the development of symptoms within a person over time can be studied. Knowing which different causal relationships between symptoms tend to result in similar manifestations of mental disorders is very valuable knowledge for this reason. Having a method to accurately determine the similarity of symptom graphs fills a gap in current knowledge.

In this paper we provide an overview of existing graph comparison methods while also proposing two new techniques to determine the similarity between two graphs. We consider three main categories of graph comparison methods:

- Edit distance
- Feature extraction
- Iterative methods

The edit distance category focuses on the structural shape of the graph while the feature extraction category focuses on certain properties of the graphs and the iterative methods category focuses on the similarity of node's neighbors. Our proposed methods extends one of the iterative methods to determine node similarity to a metric that quantifies the similarity of two graphs.

We start with an overview of the existing graph comparison methods in Section 2. We present our two new methods in Section 3. We analyze the pros and cons of each of these methods by examining the properties of the methods and determining how the methods can be applied in the field in Section 4. The paper is concluded in Section 5.

## 2 Related work

A graph $G$ is a set nodes $V$ and edges $E$. Each edge has a start node and an end node: $E \subset V \times V$. If the nodes of a graph have attributes, then it is an attributed graph [5].

Multiple techniques exist to determine the similarity between graphs. These techniques can be divided in three categories [9]:

- Edit distance
- Feature extraction
- Iterative methods

The edit distance set of techniques state that two graphs are similar if they are isomorphic or they have isomorphic subgraphs. Their main

- *J.D. van Leusen is a student at the University of Groningen, E-mail: jvleusen@gmail.com.*
- *S.F. de Bruijn is a student at the University of Groningen, E-mail: s.f.debruijn@gmail.com.*

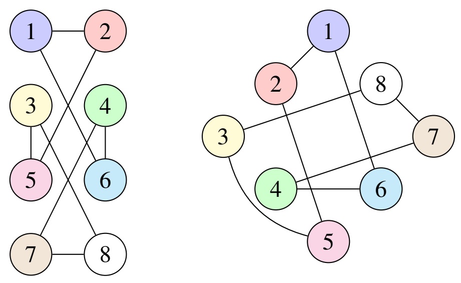


Fig. 1: Equivalent graphs in different configurations, an example of graph isomorphism.

disadvantage is that the only known algorithms to determine whether two graphs are isomorphic have an exponential complexity, making them unsuitable for large graphs.

The second class of techniques, feature extraction, focuses on determining the similarity of a set of statistics derived from the graph. While they scale well due their reduced amount of data, their results are very dependent on the chosen statistics. The consequence is that statistics may not actually reflect the similarity of the graphs.

The third class of techniques is based on the school of thought that two nodes are similar if their neighbors are similar. We discuss each of these classes of techniques in more detail in the next sections.

## 2.1 Edit distance

Graph isomorphism is a property of two graphs where there exists a one-to-one mapping between the nodes of the graphs and the edges between those nodes. When two graphs are isomorphic it is possible to turn one of the graphs into the other graph by moving the nodes without changing changing the nodes or edges of the graphs. An example of isomorphism is shown in Figure 1.

Edit distance is a concept related to graph isomorphism that involves calculating the number of actions required to make two graphs isomorphic. In order to determine the edit distance the following operations can be applied to a graph:

- Insert or delete an isolated[1] node
- Insert or delete an edge
- If the graph has attributes, changing these attributes

The edit distance of two non-isomorphic graphs represents the difference between the graphs. To calculate a similarity score for the graphs the edit distance needs to be compared to the size of the actual graph. This gives a measure of how much the graph would stay the same and how much the graph would need to change to be equivalent to the other graph.

The primary issue with the use of graph isomorphism property and edit distance is that the most efficient known algorithm to determine graph isomorphism has a computational complexity of $\mathcal{O}(2^{O(\sqrt{n \log n})})^2$ for graphs with $n$ nodes [11, 1]. *Graph Isomorphism* is one of the open problems in computational complexity theory as it has yet to be classified as a *P* or *NP-complete* problem.[6]

While edit distance is a very powerful method of describing the similarity of two graphs, the exponential complexity of the algorithms involved in the calculation causes the performance of this method to scale badly with the size of the graphs involved.

## 2.2 Feature extraction

The idea behind feature extraction is that the original graph contains a lot of redundant data. By reducing the input data to a number of values with a smaller size or dimensionality, the relative similarity of the original graphs can be calculated by comparing their derived values. This reduces the problem of comparing the original graphs to a problem of comparing their derivative features, a trade between performance and accuracy.

Koutra et al. [9] defines a number of features that can be used to assess similarity between graphs.

- By turning the graph into an adjacency matrix[3] it is possible to calculate eigenvalues for the graph.
- The degree of the node refers to the number of edges attached to a node. The distribution of node degrees throughout the graph describes the distribution of connectedness in the graph. Distributions with a large number of high degree connections implies a highly connected graph while a distribution with a large number of low degree connections imply a loosely connected graph. Generally the average degree of all the nodes in the graph is used as a feature in the feature comparison.
- The diameter of a graph is defined as the maximum length of all paths between any pair of nodes in the graph. This value describes the presence of outsider values in the graph, where the path between the outermost value and the central cluster of the graph is very long. If the graph is disconnected then the diameter would be infinite. To avoid these infinite values the diameter feature should only be used on connected graphs or subcomponents of the graph.

All of these features discard some of the information contained in the original graph during the calculation of the values. The features have a reduced accuracy when used to identify their original graph. This means error in the determination of similarity is increased proportionally to the amount of data that is discarded.

To improve the accuracy of the comparison when using feature extraction, the comparison is not performed with only a single feature. Instead multiple features are calculated to determine a feature vector that can be used to compare similarity with better accuracy than individual features.

Feature extraction performance depends on the features selected, but the fact that a selection of features is used means that computationally expensive features can be replaced by a set of features that are cheaper to compute. This allows the feature extraction method to be very flexible when it comes to the required performance and accuracy.

## 2.3 Iterative methods

Iterative methods determine the similarity of two nodes within a graph based on how similar the neighbours of these two nodes are. These methods are iterative as for each iteration a similarity score of the nodes is calculated. These scores are then used for the next iteration to compute a more accurate score. This process continues until the scores converge to a state of equilibrium.

We discuss multiple prominent algorithms that belong to this category. Firstly there is the similarity flooding algorithm by Melnik et al. [12], which aims to find nodes between graphs that correlate to each other. For example, in case of feature graphs two corresponding nodes would be the node for ‘fatigue’ in one graph with the node ‘fatigue’ in the other graph. One way that the correlation between nodes can be determined is by looking at the node labels. This algorithm takes two graphs as its input and provides a mapping between correlated nodes as the output. Note that this does not provide a similarity metric for two graphs but rather provides a map of nodes that correspond to each other.

The second example of a prominent algorithm in the class of iterative methods is the SimRank algorithm [8]. This algorithm also does

[1] An isolated node has no edges attached to it

[2] $2^{O(n)}$ is the notation for an exponential time complexity where the exponent is linear.

[3] $n*n$ matrix defined for a graph with $n$ nodes, where the value at $(i,j)$ is 1 if there is an edge between node $i$ and $j$ and 0 in all other cases. If the graph is weighted or directed then the values of the adjacency matrix can be changed to represent these values instead of the binary representation.

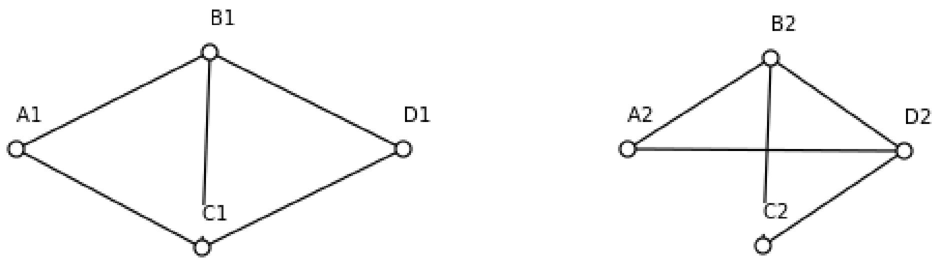


Fig. 2: A simplified example of two graphs with corresponding nodes.

not provide a similarity metric for two graphs, instead it provides a measure of how similar the nodes within a graph are to each other. For example it will provide a similarity score for the nodes 'fatigue' and 'irritability' that exist in the same graph. This is another form of similarity that is based on the similarity of the neighborhood of a node.

The third example is close to what we need, since it is a total graph comparison method. As proposed by Zager et al. [15] this algorithm computes the similarity of two graphs based on how similar their nodes and edges are. This is done by first computing the similarity between the nodes and the similarity between the edges and then determining which nodes from one graph correspond to nodes in the other, as well as which edges in one graph correspond to edges in the other graph. This algorithm is used for graph matching, for example determining whether a subgraph is contained in another graph.

Finally we discuss an iterative method as proposed by Leicht et al. [10]. Like the others it determines the similarity of two nodes based on the similarity of the node's neighbours. They propose the following definition for node similarity:

$$S_{ij} = \frac{2m\lambda_1}{k_i k_j} \left[ (\mathbf{I} - \frac{\alpha}{\lambda_1}\mathbf{A})^{-1} \right]_{ij} \tag{1}$$

Where:

- $S_{ij}$ is the similarity from node i to node j
- $k_i$ and $k_j$ are the degrees of nodes i and j respectively
- $m$ is the total number of edges in the network
- $\mathbf{A}$ is the adjacency matrix
- $\lambda_1$ is the largest eigenvalue of $\mathbf{A}$
- $\alpha$ is a parameter between 0 and 1 that reduces the contribution of long paths relative to short paths
- $\mathbf{I}$ is the identity matrix

Knowing the similarity between nodes could be useful for psychological symptom graphs since the nodes represent symptoms. Knowing how similar two symptoms between two patients are could be a useful metric in determining the causes of comorbidity.

### 2.4 Symptom graphs

The methods that are mentioned above are applicable to general graphs. Symptoms graphs are specific instances of graphs. They are not necessarily connected because not all symptoms need to have causal relationships. For example fatigue may cause concentration problems, while depression may cause suicidal tendencies. These two relationships do not need to be connected at all. Causality is not a two way street, 'when it rains I get wet, but when I get wet it does not have to rain' is a very clear example. As such symptom graphs are *directed graphs*. Symptom graphs are *weighted graphs*, as causality is not all or nothing. The weights represent the degree of causality. Symptom graphs can contain self loops [3].

Symptom graphs exhibit the small world phenomenon [2]. This means that they tend to consist of clusters since multiple symptoms are clustered together by a shared cause. The clustering property of the graph also means the average path length will be very short.

Finally the nodes of the symptom graphs are known symptoms. This property means that there are common nodes between two different symptom graphs, since there are a number of common symptoms that will be shared.

## 3 Proposed similarity metric

We propose two new ways to determine graph similarity. These methods are based on an existing node similarity metric, as proposed by Leicht et al. [10]. We extend this similarity metric of two nodes to a similarity metric of two graphs.

We start the node similarity metric as proposed by Leicht et al. [10] and discussed in Section 2. This method fits symptom graphs best as the neighbors of nodes are very important, given that they represent symptoms. Using this method also has the advantage that is applicable to the research question of what the causes of comorbidity are. As mentioned, it is suspected that symptoms are caused in part by each other. As a result it is valuable to determine a graph's similarity based on how similar the neighbors of corresponding nodes are. This metric in itself does not provide any possibility to extend it to a graph comparison method, as it only focuses on one graph in specific. However since we know that the nodes of the graph represent psychological symptoms we can pair the corresponding nodes of each graph with each other. For example the symptom "irritability" has a node in graph $A$ and is also represented by a node in graph $B$. Even in the scenario where the symptoms differ between graphs it is possible to use the similarity flooding algorithm to map corresponding nodes to each other, albeit with a potentially larger margin for error as the nodes now may not match in a one-to-one fashion. In this case there is a chance that the wrong symptoms are mapped to each other, so it is desirable to have two graphs with the exact same symptoms represented in them, even if nodes are entirely disconnected.

We now have two corresponding nodes in two different graphs and a method to compare these two nodes. What remains is the matter of taking the similarity of corresponding nodes and extending this to account for the similarity of *all* corresponding nodes, resulting in the proposed similarity metric for the two graphs. There are several possibilities to make this extension happen, but the following two will be considered:

1. two graphs are similar based on the average similarity of the corresponding nodes
2. two graphs are similar based on the percentage of nodes whose similarity exceeds some threshold $\alpha$

We will clarify both cases by using a very simplified example scenario concerning two graphs. Figure 2 displays two graphs: on the left graph 1 with nodes $A_1, B_1, C_1$ and $D_1$. On the right graph 2 with nodes $A_2, B_2, C_2$ and $D_2$. The letters in these graphs represent the symptoms and similar letters represent similar symptoms. The edges of the graphs differ while the nodes are similar. Table 1 displays an example of the values that the similarity between nodes may exhibit. Note that these values are made up but do correspond to the graph displayed in Figure 2. The similarity metric we use for the nodes is based on how similar their neighboring nodes are. For example since node $A_1$ has neighbors $B_1$ and $C_1$ while node $A_2$ has neighbors $B_2$ and $D_2$, we say that nodes $A_1$ and $A_2$ are 50% similar. This translates into a similarity score of 0.5. These values lie between 0 and 1, 0 meaning that the two nodes are not similar at all and 1 meaning that the two nodes are practically identical. We can tell from Table 1 that one set of nodes is identical, two sets of nodes are quite similar and one set is not similar at all. Intuitively we might say that the two graphs are reasonably similar.

| Node pair | Similarity value |
|---|---|
| $A_1,A_2$ | 0.5 |
| $B_1,B_2$ | 1.0 |
| $C_1,C_2$ | 0.67 |
| $D_1,D_2$ | 0.67 |

Table 1: Hypothetical node similarity values.

Let us look at the first proposed extension to see how it compares to our intuitive notion of similarity. The first proposed extension states that two graphs are similar based on the average similarity of the corresponding nodes. The average similarity of the sets of nodes is easily calculated:

$$\frac{0.5+1.0+0.67+0.67}{4} = 0.71$$

This measure would say that the two graphs are 71% similar, coinciding pretty well with our intuitive notion of the value.

The second proposed extension states that two graphs are similar based on the percentage of nodes of which the similarity exceeds some threshold $\alpha$. The results are obviously very dependent on the choice of parameter $\alpha$. If we are to set $\alpha = 0.7$, we obtain a similarity score for the graph of 25%. The eventual value for $\alpha$ would have to be carefully considered in a real environment. Considerations to take into account include assessing what values the similarity values for nodes will attain. A test setting where the similarity between nodes is known can help in determining this by providing feedback on what the similarity values are for nodes that are known to be similar.

## 4 Evaluation

To evaluate the usability of the proposed methods to compute symptom graph similarity the performance as well as the accuracy of the results will be considered. The asymptotic runtime[4] will be used to evaluate the performance of the methods. The level of detail as well as information used to calculate the results will be used to evaluate the accuracy of the methods.

### 4.1 Performance

The first aspect to consider when selecting the method to compare graphs is the performance when dealing with the subject graphs. The paper by Borsboom et al. [2] contains a symptom graph containing 439 nodes, but in compositon to other graphs used in computer science this is a moderately sized graph.

As mentioned in Section 2.1, the performance of the edit distance approach does not scale. The most efficient known algorithm has an asymptotic complexity of $\mathcal{O}(2^{O(\sqrt{n \log n})})$. Even given the moderate average size of symptom graphs this approach would require a larger amount of processing power than the other approaches, making it unsuitable for this use case.

The performance of the feature extraction approach heavily depends on the features that are being used to evaluate the graphs. Some of the features only consider the nodes of the graph such as the average degree of the nodes (resulting in a complexity of $\mathcal{O}(n)$). The expensive features will consider each node and all the potential neighbors of that node, such as the diameter of the graph. Since each other node in the graph is a potential neighbor the resulting complexity is $\mathcal{O}(n^2)$.

[4]Big $\mathcal{O}$-notation

Finally the performance of the proposed iterative method approach heavily depends on the specific node similarity calculations that are being used since their complexity can vary wildly among the various options. Since the method involves going through each named pair in the graphs and calculating the similarity of those pairs the similarity calculations generally involve the neighbors of the two nodes which results in a complexity of $\mathcal{O}(n)$. This means that the final complexity of calculating the proposed method is $\mathcal{O}(n^2)$, since the finalizing methods discussed in Section 3 are of linear complexity and and are asymptotically dominated by the quadratic complexity of the similarity calculations.

### 4.2 Accuracy

Section 2.4 lists a number of properties of symptom graphs, these properties need to be considered when evaluating the accuracy of the methods since they represent the unique properties of symptom graphs.

The edit distance approach would be very accurate in determining the similarity of symptom graphs since it would be able to detect isomorphism in shared clusters of symptoms between graphs. Since edit distance determines the smallest amount of changes required to make two graphs isomorphic, it also represents the exact difference between the two graphs. This property means it is the perfect method for determining the similarity between two graphs as well as the exact differences between the graphs.

The accuracy of the feature extraction method depends heavily on the features used in the comparison. To evaluate the accuracy of this method we will consider the features listed in Section 2.2:

- **Eigenvalues** are heavily dependent on the structure of the graph, so they might be a good feature to use to compare the symptom graphs. However there is no theoretical reason why this feature would represent the symptom graphs well or poorly.
- **Node degrees** are a poor way to distinguish symptom graphs since they are by their nature very interconnected. This means the distribution of node degrees within different symptom graphs would be similar, thus making them a bad way to determine similarity.
- **Graph diameter** would only be able to distinguish graphs with outlier symptoms. While these outliers would indicate differences between graphs they would not be sufficient to determine whether two symptom graphs are structurally similar. The graph diameter feature should be used in combination with other features.

The evaluation of the features shows that feature extraction would be an unreliable way to determine similarity of symptom graphs due to the shared properties of these graphs.

Finally the accuracy of the proposed iterative method approach depends on the accuracy of determining the similarity through the nodes of a symptom graph. Since symptom graphs are highly connected and the exact connections between nodes are an important distinguishing factor the proposed method might be a very good method of determining similarity between graphs. The scope of similarity might be too small since a lot of the important structure is contained in the expanded neighborhood of the nodes. However the limited scope might be enough to distinguish between differences in the local structures between the graphs.

### 4.3 Summary

The results of the evaluation can be seen in Table 2. When accuracy is the most important property it is clear that edit distance is the best method to use. Its downside is that the performance will not be acceptable for most purposes, making it a bad choice for the bulk comparison of symptom graphs.

| Method | Accuracy | Performance |
|---|---|---|
| Edit Distance | Most accurately represents the similarity between two symptom graphs, affected by global and local structures. | $\mathscr{O}(2^{O(\sqrt{n}\log n)})$ |
| Feature Extraction | Captures the global structural similarity of the graphs well, but fails to capture the local similarity. | $\mathscr{O}(n^2)$ |
| Iterative Method | Captures the local structural similarity of the graphs well, but fails to capture the global similarity. | $\mathscr{O}(n^2)$ |

Table 2: Results of the evaluation

Feature extraction and the proposed iterative method have a similar performance, but their accuracy depends on the type of similarity that the user is interested in. Feature extraction distinguishes similarity on a larger scale than the iterative approach. However the properties of symptom graphs mentioned in Section 2.4 cause symptom graphs to exhibit similar structures on a larger scale while differing in local structure. Because of this reason the iterative approach has a better accuracy when used to determine the similarity of symptom graphs due to the use of local structure in the algorithm.

## 5 Conclusion and future work

We have seen several different techniques to determine the similarity between two graphs. Aside from existing techniques we proposed two new ways to perform this task. Both methods extend an existing measure of node similarity to a measure of graph similarity.

Considering the performance issues with the edit distance approach and the accuracy concerns with the feature extraction approach, we consider the proposed iterative method as proposed in this paper to be a good middle-ground approach to determining the similarity of symptom graphs. Understanding the causes of comorbidity is aided by a method that is both accurate, while also being able to complete in a reasonable amount of time. Future work will have to show whether theory translates into practice, by testing the different methods with an actual implementation.

Future work will thus involve a practical assessment of the proposed similarity metrics, as well as a comparison of both accuracy and performance with existing graph comparison methods. Another aspect to test is to see how well the different methods scale when the graph sizes increase. The symptom graphs are not extremely large since they normally only have several hundreds of nodes. However since each graph can have a quadratic number of edges relative to the number of nodes the algorithms involved still need to be able to perform efficiently. Being able to scale these methods might make them more applicable to other fields besides psychopathology, such as software analysis where the graphs involved are often much larger.

## Acknowledgements

The authors wish to thank Frank Blaauw, Michael LeKander and Niels Kluiter for their review of the draft paper and their valuable feedback. The authors also wish to thank their colleague students at the University of Groningen for their fruitful discussions and valuable insights.

## References

[1] L. Babai and P. Codenotti. Isomorhism of hypergraphs of low rank in moderately exponential time. In *Foundations of Computer Science, 2008. FOCS'08. IEEE 49th Annual IEEE Symposium on*, pages 667–676. IEEE, 2008.

[2] D. Borsboom, A. O. J. Cramer, V. D. Schmittmann, S. Epskamp, and L. J. Waldorp. The small world of psychopathology. *PLoS ONE*, 6(11):e27407, 11 2011.

[3] L. F. Bringmann, N. Vissers, M. Wichers, N. Geschwind, P. Kuppens, F. Peeters, D. Borsboom, and F. Tuerlinckx. A network approach to psychopathology: New insights into clinical longitudinal data. *PLoS ONE*, 8(4):e60188, 04 2013.

[4] A. O. J. Cramer, L. J. Waldorp, H. L. J. van der Maas, and D. Borsboom. Comorbidity: A network perspective. *Behavioral and Brain Sciences*, 33:137–150, 6 2010.

[5] X. Gao, B. Xiao, D. Tao, and X. Li. A survey of graph edit distance. *Pattern Analysis and applications*, 13(1):113–129, 2010.

[6] M. R. Garey and D. S. Johnson. Computer and intractability. *A Guide to the Theory of NP-Completeness*, 1979.

[7] R. A. Hanneman and M. Riddle. Introduction to social network methods, 2005.

[8] G. Jeh and J. Widom. Simrank: a measure of structural-context similarity. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 538–543. ACM, 2002.

[9] D. Koutra, A. Parikh, A. Ramdas, and J. Xiang. Algorithms for graph similarity and subgraph matching, 2011.

[10] E. Leicht, P. Holme, and M. E. Newman. Vertex similarity in networks. *Physical Review E*, 73(2):026120, 2006.

[11] E. M. Luks. Isomorphism of graphs of bounded valence can be tested in polynomial time. *Journal of Computer and System Sciences*, 25(1):42 – 65, 1982.

[12] S. Melnik, H. Garcia-Molina, and E. Rahm. Similarity flooding: A versatile graph matching algorithm and its application to schema matching. In *Data Engineering, 2002. Proceedings. 18th International Conference on*, pages 117–128. IEEE, 2002.

[13] M. Newman. *Networks: an introduction*. Oxford University Press, 2010.

[14] K. RC, C. W, D. O, and W. EE. Prevalence, severity, and comorbidity of 12-month dsm-iv disorders in the national comorbidity survey replication. *Archives of General Psychiatry*, 62(6):617–627, 2005.

[15] L. A. Zager and G. C. Verghese. Graph similarity scoring and matching. *Applied mathematics letters*, 21(1):86–94, 2008.

# A survey of big data architectures for smart cities

Erik Jager and Stephan Boomker

**Abstract**— The objective of this paper is to survey the current state of smart city architectures. Smart cities are cities that strive to be self-sustainable. The city consists of a set of advanced services that create a citizen friendly, sustainable city. These services often work in near realtime with large amounts of data that can be seen as Big Data. Big Data is heterogeneous unstructured data that is difficult to process in realtime. We review four different approaches which try to find a solution to handle these kinds of data in a smart city setting. We contrast these approaches to find the common purpose within these architectures.

**Index Terms**—Big data, smart cities, data-mining and analyzing, design principles, software architecture

## 1 Introduction

In city landscapes, the investment in Information and Communication Technology (ICT) for enhanced governance becomes more present in urban environments. These technologies provide the basis of sustainability for smart cities of the future [7]. The smart city system can be seen as a set of advanced services to create a citizen friendly, efficient and sustainable city [8]. The ICT tools for a smart city are often applied in different domains like transport-, energy- and environment management. The systems controlling these smart cities have to deal with scalability, heterogeneity, geo-location information, privacy issues and large amounts of continuously incoming data from the city environment. This is where big data comes into play. Big data is defined as a large amount of data that increases exponentially over time [11].

The problem related to the concept of Smart Cities and Big Data is growing, this has to do with the urban population. Approximately 75% of the European population today lives in urban areas and the urbanization of the European population is expected to increase over 80% by 2020 [1]. The continuous increase in urban population strains the limited resources of a city, this affects its resilience to the increasing demand on resources and urban governance faces ever increasing challenges. The innovation in ICT in the urban environment can provide integrated information intelligence for better urban management and governance, sustainable socioeconomic growth, and policy development using participatory processes.

Another important problem includes the fact that data volumes increase exponentially over time, this data tsunami can easily overwhelm traditional analytics tools or platforms designed to ingest, analyze and report. The challenges that are being faced are not only how to store and manage diverse data but also to effectively analyze the data to gain insight knowledge to make smart decisions.

Currently, a wide array of commercial product solutions are already available on the market that support the adoption of Smart Grid technology among energy providers as well as Smart Building technology among building constructors and managers. Although these products are a sound starting point to achieve sustainable, efficient and citizen friendly cities, the vision of a Smart City reaches much farther. This includes broader resource integration, policy sharing, and increased ease of deployment and operation automation. Moreover, the commercially available Smart Cities analytics technologies are known to be cumbersome to setup and reconfigure, making them less likely to get adopted. Because of this complexity, there is no single tool or one-size-fits-all solution that can cope with deep mining and analyzing big data.

- *Erik Jager is a MSC student at the University of Groningen, E-mail: h.j.jager.2@student.rug.nl.*
- *Stephan Boomker is a MSC student at the University of Groningen, E-mail: s.boomker@student.rug.nl.*

The reason for this paper is the fact that the concept of a Smart City is lately been adopted by the research community, companies and public governments to try to come to a solution for this complex problem. The solutions that they came up with are architectures for the Smart Cities of the future that are able to handle and analyze Big Data. In this paper we try to find common ground for different architectural approaches. These architectural approaches have been chosen because they represent the current state of knowledge and are published in well-known journals and discussed on well-known conferences.

The remainder of this paper is structured as follows: Section 2 establishes the formal definition of a Smart City and Big Data and their relation. Section 3 gives a survey of different architectures related to the Smart City and Big Data concept. We conclude the paper by drawing our conclusion and discussing this survey in Section 4.

## 2 Background

This sections provides information on the field of Smart City and Big Data'. First we discuss the subjects in their respected fields and analyze these concepts and how they relate. We conclude by bringing them together as a distributed system.

### 2.1 Smart city

The first question that comes to mind; what is a smart city? A smart city environment can be seen as an environment where the city actually gets to communicate with the people residing within the city. The city would be able to give live status updates on traffic, energy usage, water quality, air pollution and smart lighting. The system can aid public and environmental health but also aid in a more user friendly experience. Figure 1 shows an overview of the main thematic pillars for smart cities. The basis of a smart city is built on the infrastructure which deals with the processing and storage of the data. The next step is to transform these specific types of data into useful information and knowledge[7]. A smart city can be seen as a distributed system where different sources of information provide data to a set of applications to elaborate responses at a strategic and tactical level [10].

#### 2.1.1 Perspectives

There are various perspectives named in the previous section, to get a clearer view for some of these perspectives we are going to elaborate some more on these perspectives. One of the perspectives of a smart city is the energy perspective. From this perspective the power-plants need to interconnect with individuals who produce and consume energy which can be seen as prosumers as described by Girtelschmid et al. [2]. These prosumers use renewable energy sources to produce energy that can be given back to the energy grid. Another perspective is transportation within the city environment. Data is already the fuel that drives intelligent transportation systems, this data comes from smartphones, gps trackers, social media posts and camera's located around the city. Cities already capture and analyze data from all these sources as well as using sensors embedded within the roads. This example has a large impact but there are also solutions which have a

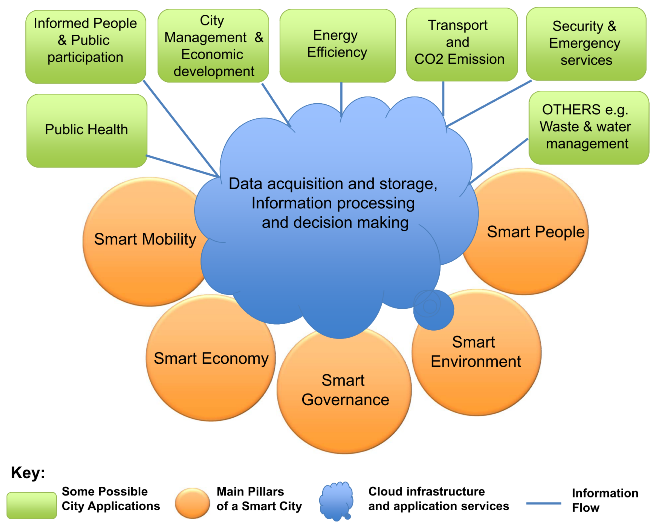

Informed People & Public participation
City Management & Economic development
Energy Efficiency
Transport and CO2 Emission
Security & Emergency services
Public Health
OTHERS e.g. Waste & water management
Data acquisition and storage, Information processing and decision making
Smart Mobility
Smart People
Smart Economy
Smart Governance
Smart Environment
Key:
Some Possible City Applications
Main Pillars of a Smart City
Cloud infrastructure and application services
Information Flow

Fig. 1. Cross-thematic data management and analysis for variety of smart city applications in Cloud environment [7].

much smaller impact. For example in the city of Groningen in the Netherlands, the traffic lights are outfitted with rain sensors that ensures that cyclists get a green light more often when it rains[1]. This solution has as a small influence on traffic flow, but has a bigger impact on citizen satisfaction.

## 2.2 Big Data processing

All of the applications discussed in the previous section produces a continuous stream of data, which can be defined as Big Data. But what is Big Data? The term in itself can be confusing, because it implies that the main challenge is its size. Although size is one of the challenges, it is not the only challenge. One of the main challenges is to have a data platform that leverages the large volumes of data produced into insights. The data that we are talking about comes from small integrated circuitry which can be added to nearly anything. These sensor devices can measure light, temperature, motion or sound. The output of these sensors result in a continuous stream of data.

According to Zikopoulos et al. [11] big data refers to massive, heterogeneous, unstructured data that is difficult to process using traditional tools and techniques. The bottleneck in traditional architectures is the database server on peak workloads [4]. Traditional tools and techniques are useful when the source-data is well understood, and data is relational and can be modeled as such. Whereas Big Data consists largely out of heterogeneous unstructured data which is continuous and of changing quality.

The incoming data streams from all kinds of sources can be unreliable and measurements can be invalid. The data collected can be in different heterogeneous formats and measurements can be in different units of measure. The heterogeneity issues can be addressed by establishing semantics and applying meta-data to the incoming data streams. The streaming of of high volumes of data need reliable streaming platforms that arrive at the system at high velocities [2, 4]. The systems must be able to provide scalability to automatically rebalance the workload when there is to much data to be processed on one point.

Ji et al. [4] states that big data is a challenging concept which calls for a scalable storage index and a distributed approach to retrieve required results in near real-time. Data in systems of this scale is to big to process conventionally. The source of these kinds of data are sensors which are placed for example in smart grids. The systems running on these kinds of data are highly dependent on time critical and reliable data.

[1]http://www.eltis.org/discover/news/groningen-installs-rain-sensors-cyclists-traffic-lights-netherlands-0

After the data has be gathered and stored there have to be some components in the smart city system that perform data analytics. These components are well known in data mining. An overview of how such a set of components would work is depicted in figure 2.

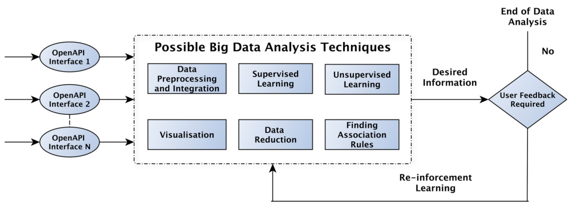


Fig. 2. Big Data analysis architecture [7].

## 3 Architectures

As stated in the introduction this paper compares four architectures related to smart cities and big data from the papers [2, 7, 8, 10].

### 3.1 Architectural overview

The following sections elaborates on four architectures that have been compared with each other. The main idea is to find the common ground on which these architectures have been designed.

1. The architecture of Girtelschmid et al. [2] contributes to the big data and smart city concept by employing ontology reasoning and distributed stream processing on the cloud. Which results in a fully automatic and self-contained decision process, while it remains robust and time efficient. An overview of the architecture is depicted in figure 3.

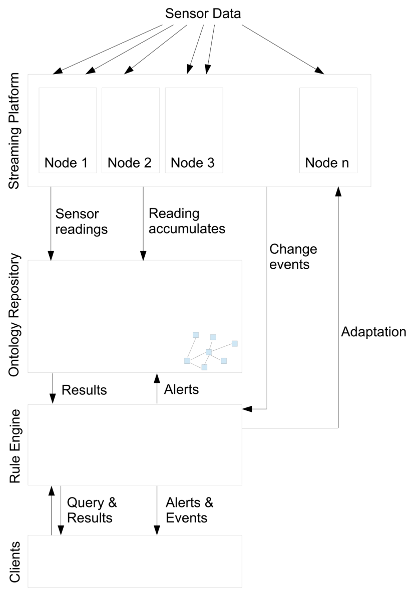


Fig. 3. Smart city data processing architecture high-level overview by Girtelschmid et al. [2].

2. The second architecture from Khan et al. [7] is a design for cloud based big data analysis, their guiding design is to reuse existing, well-tested tools and techniques. Therefore, they use some architectural concepts from their own previous work. They came up with a layered architecture where each layer represents the potential functionality that will meet their objectives as shown in figure 4.

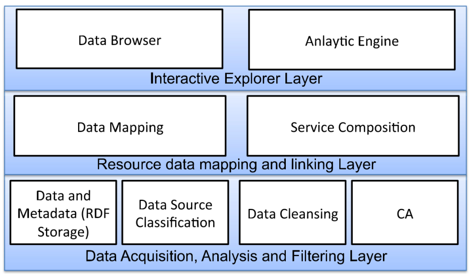

Data Browser
Anlaytic Engine
Interactive Explorer Layer
Data Mapping
Service Composition
Resource data mapping and linking Layer
Data and Metadata (RDF Storage)
Data Source Classification
Data Cleansing
CA
Data Acquisition, Analysis and Filtering Layer

Fig. 4. Proposed architectural design by Khan et al. [7].

3. Villanueva et al. [8] present in their paper the distributed object-oriented middleware, called Civitas, specially designed for smart cities. An overview of the architecture is shown in figure 5. This middleware provides services that range from environmental sensor deployment to the necessary hardware for high performance algorithms devoted to extract information from raw data. This raw data of a smart city is an abstraction that comprises the IT infrastructure deployed by governmental institutions all over the city, such as semaphores, traffic sensors, cameras, and public Wi-Fi networks. They present their architecture on the basis of a case study scenario in which a service for license plate based vehicle tracking is required.

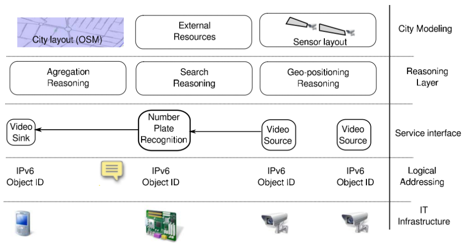


Fig. 5. Civitas architectural overview by Villanueva et al. [8].

4. The architecture of Ye et al. [10] tackles the challenges of big data mining and analysing by integrating different existing tools and technologies. From the fields of infrastructure resource management to rich statistical computation and graphic functions. Their architecture proposed consists of four layers: infrastructure, virtualization, dataset processing and service as shown in figure 6.

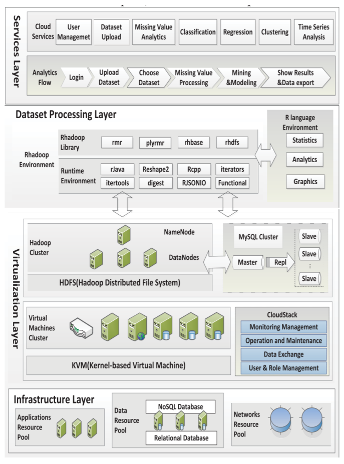


Fig. 6. The Architecture of Cloud-based Big Data Mining & Analyzing Services Platform by Ye et al. [10].

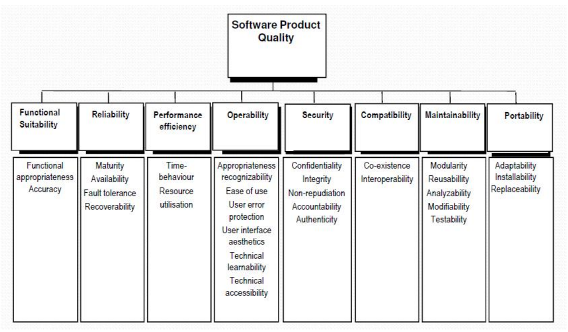


Fig. 7. ISO/IEC 25010:2011 overview [3].

### 3.2 Analysis

This section analyzes the literature by finding common ground on which each of the architectures has been built upon. The architectures are designed by using standard design principles which are specified in Section 3.2.1. The architectures are built by using multi-tier architectures that are composed of out of several layers described in Section 3.2.2. At the end of this section, the tools and techniques used for these architectures are described.

#### 3.2.1 Design principles

This section goes further into the description of the key design principles employed in the literature and how these can help in developing an architecture that is suited for data transaction of these magnitudes. Architectural drivers are the set of requirements that have a significant influence on the architecture of a system. The architectural drivers that we discuss are defined according to the ISO/IEC 25010:2011 [3].

**Standardization** and the open system principle is used throughout the literature as a key design principle [7, 5, 8]. The open system principle means that the system uses standards based technologies, by using international standards for API communication, data modeling and data storage. This prevents lock-in by certain vendors and supports adaptability of a system by open-standards to enable new tools and technologies that increase functionality, and are more efficient. The standards are only used when its utilization does not break any of the other design principles. The standardization enables the system to be extendable with new data sources. Standardization encourages dataproviders and user communities to join the system [7]. Villanueva et al. [8] goes in deeper in specifying actual standards to employ, where geographical data representation is based on Mobile Location Protocol(MLP) from the Open Mobile Alliance.

**Scalability** The actual analytics engine that is going to perform adaptions or predictions based on the sensor data has to scale, where the analysis and search will be parallelized by employing technologies such as MapReduce [7]. These technologies can be used to optimize the processing of large amounts of data. Villanueva et al. [8] tackles the scalability issues of the smart city by using administrative districts to divide the core infrastructure as can be seen in figure 8. The architecture of Girtelschmid et al. [2] adopts the dynamically adding and removing of computing nodes within the analytics infrastructure.

**Portability** and *adaptability* are important for an ever changing smart city environment. Adaptability is important for the deployment and operational services because this works tightly with the dynamics of a smart city system. The intelligence within the system proposed by Villanueva et al. [8] consists of a context aware services and system that is based upon the *adaptability* key-driver for unforeseen situations. Villanueva et al. [8] use a set of simple interfaces within their architecture to improve the portability and maintainability of the system.

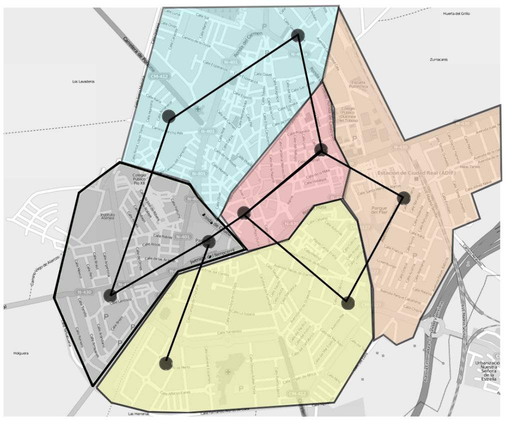


Fig. 8. District-based smart city architecture [8].

**Security** is not a main driver for the architectures analyzed, and is omitted in three of the four papers. The issue of privacy and security is avoided, even though it is not the focus of these papers but still rather important for the development of a real-world system employed in a smart city environment. Any smart city system that can interact with smart grids, traffic management or water management should be built from the ground up with security in mind [7, 10]. The data will also contain sensitive data and privacy and security has to be considered. The Civitas architecture does mention the access to traffic control camera's that can be based on a security level, where police can control the camera's and the traffic control can only observe the feeds. The system uses interfaces that contain security properties. Girtelschmid et al.[2] does mention the lack of security in their architecture and state that this was the result of time constraints, and state that security and privacy are indispensable [9]. Even though security is not one of the key drivers in these archtitectures encryption of sensitive data is still accepted as the default measure to incorporate security in most systems. Khan et al. [6] proposed end-to-end security measures for smart city applications which use open data.

### 3.2.2 Layers

A common property that stands out of each of the architectures is the fact that they all use some sort of layering structure. The number of layers within the multi-tier architectures varies between three and five and the overall purpose of each of the layers is quite similar. In figure 9 you can find a generalized layer architecture which is based on the architectures by [2, 7, 8, 10]. The next section elaborates more on this.

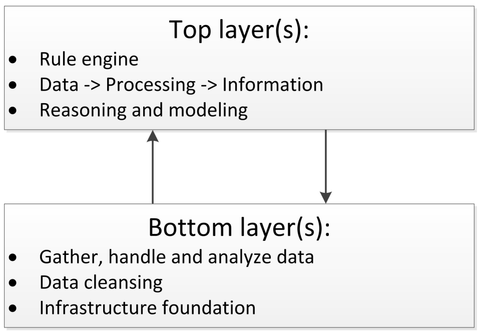

Top layer(s):
Rule engine
Data -> Processing -> Information
Reasoning and modeling
Bottom layer(s):
Gather, handle and analyze data
Data cleansing
Infrastructure foundation

Fig. 9. Layer structure

The bottom layer is the layer that gathers, handles and analyses the data in all of the architectures, but there are some differences in the way they do that. Girtelschmid et al. [2] calls this layer the streaming platform, which contains a cluster of many individual compute nodes. Each of these nodes can handle streams of sensor data from arbitrarily many sensors. It also detects considerable changes in sensor data readings or failures in sensors. The data cleansing processes is also applied in this layer. Khan et al. [7] does not specify the number of computing nodes, but does provide data acquisition and cleansing. However this paper provides only three layers so the data analysis is also done in this first layer. Villanueva et al. [8] takes on a different approach, they give an overview of their architecture on the basis of an example, namely a number plate recognition system. They subscribe their system from a less technical perspective than the other papers, so their first and second layer are the sensors and the logical addressing of those sensors. The third layer receives the video from the traffic surveillance cameras and applies the number plate recognition algorithm. The forth layer is an interface layer and the fifth is a visualization layer. Ye et al. [10] describe their architecture in the most technical fashion of the four. The architectural overview has a lot of detail, including the main techniques they use. The first layer provides the infrastructure foundation for big data processing just as in [2, 7]. The second layer in this architecture is the virtualization layer which the others do not have. This layer contains different virtual machine clusters used to manage the infrastructure resources.

Moving to the top layers we also see some differences and similarities. [2] uses the top layers for the recording of the most recent readings and accumulate information. Any changes within this layer triggers the rule engine, which contains a set of rules that are stored and executed on a timely basis upon user request and whenever the compute cluster is signaling a change event. This is used to raise alerts, change settings and send execution results back to the clients. The same principle applies in [7], the function of their top layers is to establish the mapping between the resources, generating links and making the data semantically relevant and browse-able. So the data is available to help the users submitting queries, algorithms and work flow to find information from the repositories. [8] used the top layers also for reasoning and modeling. Same goes for the more technical architecture in Ye et al.[10]. They have a processing layer for statistics, analytics and graphics and a services layer for computing, cloud services, data mining and a user interface.

The basic functionality of the layers are similar in most of the architectures, the first layers are mainly for collecting and sometimes analyzing the data and the top layers are for the analyzing, modeling and getting useful information out of the data. The differences between the papers depend on the focus of their research, key-drivers, and the requirements they have.

### 3.2.3 Tools and Techniques

The final topic of the architecture survey, are the tools and techniques that they have used. The papers differ from each other in the way they are written, which means that some of the papers describe more technical details than the others. Also [7] does not describe a prototype or concept they created but talks about the techniques they aim to use if they build such a big data analytics system for smart cities. This section describes in detail the techniques and tools on the basis of the category they fit in and the functionality they provide, this means that we are not going to compare the tools or techniques itself.

An important part of this subject is how to deal with and process the continuous stream of (real-time) data. Girtelschmid et al. [2] uses Apache Storm as the basis for the prototype implementation. Storm is a distributed and fault-tolerant real time computation platform. [2] uses a cluster of Storm workers to handle the data. Storm uses spouts to stream data from the sensors to the system, and it uses bolts to consume and emit the data streams. [7] receives the data through repositories by using APIs. The video steam interface from the traffic surveillance cameras of Villanueva et al. [8] are integrated using Slice. This interface provides a compiler which automatically generates a VHDL wrapper for the number plate recognition algorithm. Ye et al. [10] uses a virtualization layer where different management solutions can exist, such as MySQL cluster and Hadoop distributed file system and diverse storage tools. The layer on top of that is the dataset processing layer. It consists of R language runtime environment and RHadoop environment for statistics, analytics graphics and so on. The presentation of the data also differs. Some do not talk about the presentation, while the other papers talk about the possibility for the user to get some information out of the data or let the user do the querying themselves and one paper also provides a layout and external reasoning on the data. So, Girtelschmid et al. [2] is concrete in the techniques and tools they have used because they developed an actual prototype while the others did not.

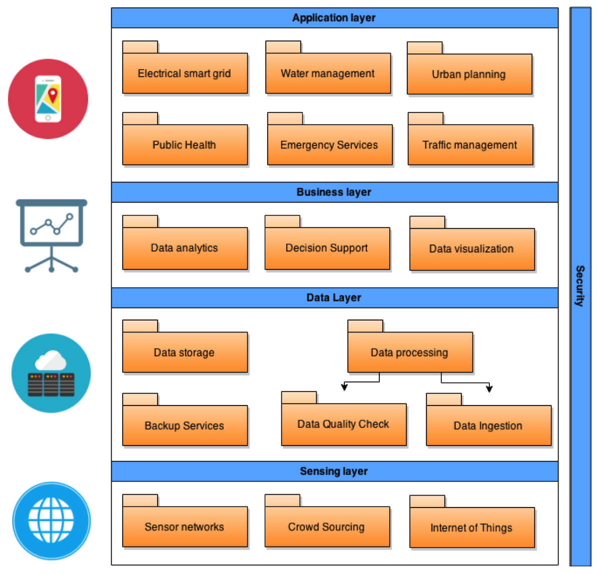


Fig. 10. Layer overview

## 4 Conclusion and Discussion

This paper shows the current state of knowledge on smart city data processing architectures. As we found during our research, developing an architecture that takes into account all of the different architectural issues is not an easy job. This really is a multi-disciplinary field, where the architectures discussed have to be able to process and store data from all types of sources. And finally the issue is that data does not speak for itself, there has to be a top layer that gives meaning to the data for different audiences. One recurring theme in choices for design principles for the smart city architecture is the usage of standardization. Standardization can be decomposed in an open-source and open-standard supporting solution.

We have established a set of components which describes the information system at a high level. This resulted in a layered overview of the architecture which is shown in figure 10. The first layer is the *sensing layer* which can get process data from sensor networks, crowd sourcing, or the internet of things [7]. The next layer is the *data layer* data stores which ingests data checks the quality of source data and storing ultimately storing the data. The *business layer*, which facilitates the transformation of source data. Because when the data is in its original format, it is not usable for the end-user. The source data after being processed can be used for analytics, decision support and visualization. The *application layer* which finds itself in the actual real life application area, which can be an application that helps with city utilities like smart grids and water management, but is also applied in intelligent -transportation, -buildings, urban planning and public safety. All of these layers have an overlaying vertical layer that can be applied on each layer. These are the security challenges and the cloud platform environment on which this layered structure would run.

## References

[1] E. E. Agency. Urban sprawl in europe - the ignored challenge. In *Office for Official Publications of the European Communities*, 2006.

[2] S. Girtelschmid, M. Steinbauer, V. Kumar, A. Fensel, and G. Kotsis. Big data in large scale intelligent smart city installations. In *Proceedings of International Conference on Information Integration and Web-based Applications & Services*, page 428. ACM, 2013.

[3] ISO/IEC 25010:2011. Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) – System and software quality models. Standard, International Organization for Standardization, Geneva, CH, 2011.

[4] C. Ji, Y. Li, W. Qiu, U. Awada, and K. Li. Big data processing in cloud computing environments. In *Proceedings of the International Symposium on Parallel Architectures, Algorithms and Networks, I-SPAN*, 2012.

[5] Z. Khan, A. Anjum, and S. L. Kiani. Cloud based big data analytics for smart future cities. In *Proceedings of the 2013 IEEE/ACM 6th International Conference on Utility and Cloud Computing*, pages 381–386. IEEE Computer Society, 2013.

[6] Z. Khan, Z. Pervez, and A. Ghafoor. Towards cloud based smart cities data security and privacy management. 2014.

[7] Z. Khan, A. Anjum, K. Soomro, and T. Muhammad. Towards cloud based big data analytics for smart future cities. *Journal of Cloud Computing: Advances, Systems and Applications*, 2015.

[8] F. J. Villanueva, M. J. Santofimia, D. Villa, J. Barba, and J. C. López. Civitas: The smart city middleware, from sensors to big data. In *Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS), 2013 Seventh International Conference on*, pages 445–450. IEEE, 2013.

[9] A. Wagner, S. Speiser, and A. Harth. Semantic web technologies for a smart energy grid: Requirements and challenges. In *In proceedings of 9th International Semantic Web Conference (ISWC2010)*, pages 33–37. Citeseer, 2010.

[10] F. Ye, Z.-J. Wang, F.-C. Zhou, Y.-P. Wang, and Y.-C. Zhou. Cloud-based big data mining & analyzing services platform integrating r. In *Advanced Cloud and Big Data (CBD), 2013 International Conference on*, pages 147–151. IEEE, 2013.

[11] P. Zikopoulos, C. Eaton, et al. *Understanding big data: Analytics for enterprise class hadoop and streaming data*. McGraw-Hill Osborne Media, 2011.

# An overview of different techniques and tools to visualize software

Bram Musters, Euaggelos Karountzos

**Abstract**— Nowadays, software may consist of millions of lines of source code which make the maintainability of the code a difficult, expensive and time consuming process. This is why the need of getting insight and understanding these huge structures is higher than ever before. A great way of tackling this problem is through visualization. By getting insight, we answer questions such as “which files should be modified?” and “what is the impact of these modifications?”.
In order to visualize these systems, a measurement for source code has to be defined. Acquiring these metrics alone is not very helpful, since getting insight in a huge amount of numbers is not intuitive for humans, therefore visualization is necessary. This paper also gives a global overview of the different software structures that are around, and how to visualize them. In the visualization part, we zoom in on the call dependency relations. Some techniques to visualize these relations in combination with metrics are analyzed. Finally, tools corresponding to these techniques will be used on test data with the purpose of comparing the different visualization techniques. We will present the drawbacks as well as the benefits of each technique and then we will evaluate the tools according to ISO standards.
The goal of this paper is to guide future users to select the most suitable tool for specific situations.

**Index Terms**—Software, Visualization, Structures, Call graphs

## 1 Introduction

Since programs started to grow in size, the need to evaluate them grew as well. The evaluation of the code can be performed using metrics such as the lines of code that it contains, the number of methods, the number of classes or even the number of comment lines. However, this is not the only way we can understand code. One more method of getting insight in code structures is the visualization of the relationship between the code components, such as the connections between classes or functions. These insights are important when it comes to software maintenance and evolution because they help the developers understand code of thousand lines with very little effort, which can save resources in terms of work hours and money as well as improve the quality of the code itself.

In this paper we will focus on the visualization methods that are available and used on different code structures, such as container and association relations. The aim of that is to provide a solid overview of what kind of options developers have when it comes to visualize the code and assist them on using the most suitable one for every specific situation.

At first we will address this goal by explaining briefly what metrics are and why they are useful in chapter 2. Then we will move on to explain and provide a variety of ways to visualize methods used to represent containment relationships of software components. This is done in section 3. After that the paper moves on to section 4 to explain, analyze and evaluate improvements of association relationships. A survey on tools that implemented the techniques described in the earlier sections is presented in section 5. We will elaborate more on these tools, exposing their weaknesses and reveal their strengths.

## 2 Code measurement

One way to measure software systems is to use source code metrics. A code metric is a quantitative measure of a degree to which a software system possesses some property [5]. This allows us to perform analysis and creates a way to visualize software systems. Some of these metrics are described below.

**Line of code (LOC)** This metric represents the lines of code in a repository. The more LOC a software system has, the more difficult it is for a programmer to remember every part of the code, therefore changes to the code or try debugging it may be time and money consuming. This metric can turn out to be a little bit biased because the lines of code depend on things such as the programming language used. Some languages (usually the low level languages) need more lines in order to perform a task while others (high level languages) need less lines for the same task. On top of that the lines of code also depends on the code style of the developer, some developers tend to use more lines in order to have more clear code structure while others tend to wright in a more compact style.

**Halstead complexity** Halstead complexity[3] works with unique and total operation metrics, vocabulary and program length. Instead of a single complexity value (like the Cyclomatic complexity) it may return a variety of metrics such as the difficulty of the code or the effort one has to put into the code in terms of coding time.

**Code coverage** Code coverage is a measure that describes the degree to which the code of a system is tested. Measuring code coverage can be very useful since high code coverage means that there is a lower chance of that the system contains bugs.

**Cyclomatic complexity** The Cyclomatic complexity[7] measures how many different “paths” the programs can take. For example, every *if, else, while* or *for* statement adds 1 point to the complexity of the code. High levels of complexity are in general undesired since it is harder to get a high amount of code coverage.

**Comment density** Comment density is a a useful metric because comments can make sure that source code is more understandable. It is important for software to be understandable, since it is necessary that developers who are not yet involved in the development can understand what particular lines do.

The combination and proper interpretation of the above information can save resources in both terms of money and man hours. In many cases developers consult these metrics to make decisions about which part of the code they should modify. Furthermore, developers can gain insight in what parts of the system are the most important. A good example would be the following; By extracting the complexity and the LOC metric from a file, developers can estimate how time consuming it will be to patch this particular file. In the remaining part of this paper, it is assumed that the extracted code metrics are correct with regard to the input source code.

• *Bram Musters is a MSc. Computing Science student at the University of Groningen, E-mail: b.t.musters@student.rug.nl.*
• *Euaggelos Karountzos is MSc. Computing Science student at the University of Groningen, E-mail: e.karountzos@student.rug.nl.*

## 3 Code structure

Apart from the metrics we can also visualize the relations of structure components that a project has. This structure is something that we can not measure in a quantitative way, the most common way to visualize structure is by drawing graphs. In software systems, the structure of code is typically split in two different kinds of relations[4].

### 3.1 Containment relations

The first kind of relation is the containment relation, also called parent - child relations. The containment relation describes what software entities are contained in other entities. Examples of this relation are inheritance of objects, variables, or included header files. Since containment relations are directed and do not contain self-loops, they are mostly visualized using trees [4]. These kind of relations - as the name suggests - are relations between methods, classes, files, or packages with the entities to represent the nodes and the relation to represent the edges of the graph.

Some visualization techniques are described in the following sections, and the benefits and drawbacks of each visualization technique are listed.

#### 3.1.1 Top-down tree, Left to right

We group these two visualization techniques together because they are not really that much different. The only thing that changes is the topology in which they are represented. The top down tree is the most widely used technique, the root is placed at the top and child nodes are placed lower according to the depth. The left to right tree is similar, but is rotated 90 degrees. Figure 1 contains a top-down tree.

**Pros**

- The user can easily find the top level nodes or the back bone of the software.
- It is easier for the user to estimate the importance of some nodes. The closer to the root they stand the more important they are.

**Cons**

- Not really informative in terms of visualizing groups of similar ancestors.
- It does not provide any information on code metrics.

#### 3.1.2 Radial tree

The radial tree has quite some similarities with the top-down and left to right tree. With this technique, the root is placed at the center and children nodes are placed on adjective circles according to depth[10]. A radial tree can be seen in figure 1.

**Pros**

- Scales both in width and height when big structures are visualized.
- The user can easily find the top level nodes or the back bone of the software.
- It is easier for the user to estimate the importance of some nodes. The closer to the root they stand the more important they are.

**Cons**

- Not really informative in terms of visualizing groups of similar ancestors.
- It does not provide any information on code metrics.

#### 3.1.3 Balloon tree

The balloon tree is an extension to the radial tree visualization; grouping nodes that have similar ancestors has been added. A balloon tree can be seen in figure 1.

**Pros**

- Inherit pros from the above tree structures.
- Introduce grouping of similar (that can be package, folder or class grouping) which makes it easier to the user to identify groups.

**Cons**

- It does not provide any information on code metrics.

#### 3.1.4 Tree map

Compared to the other tree structures, the tree map can also represent a metric along with the hierarchical structure. As can be seen in figure 1, the tree map uses a square which represents the whole system. The large squares that are formed by multiple smaller squares represent classes, while the inner squares represent the smaller entities of the system, such as methods. The size of each square is related to a chosen metric, examples are the complexity or LOC per entity.

**Pros**

- Provide additional metric information.
- Provide grouping of similar ancestors on all levels (classes, folders and packages).
- Scales well when big hierarchical structures are visualized.

**Cons**

- No root information. The user cannot really tell which file is high up near the root or lower on the leaves.

#### 3.1.5 Sunburst

A sunburst visualization has similarities with the radial tree, since it's root is also at the center and child nodes appear on concentric circles. However, the sunburst also has the advantage of showing a chosen metric along with the hierarchical structure. The length of each arc represents this metric. A sunburst can be seen in figure 2.

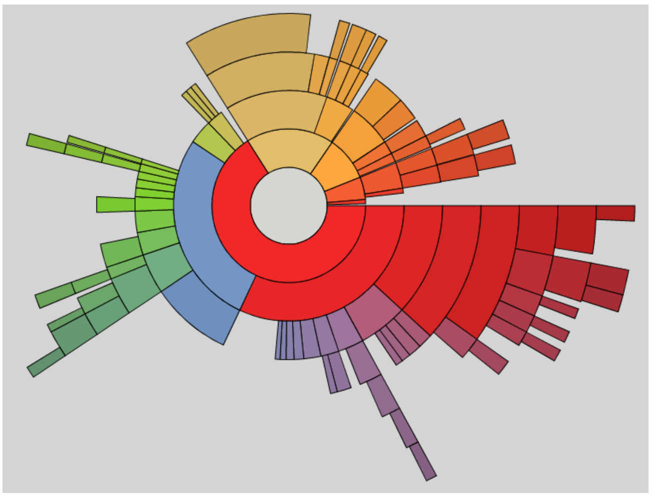


Fig. 2: This is an example of a sunburst. The center represents the root node. Here color coding is used, where the color of an arc represents the file type. (Generated using Disk Usage Analyzer.)

**Pros**

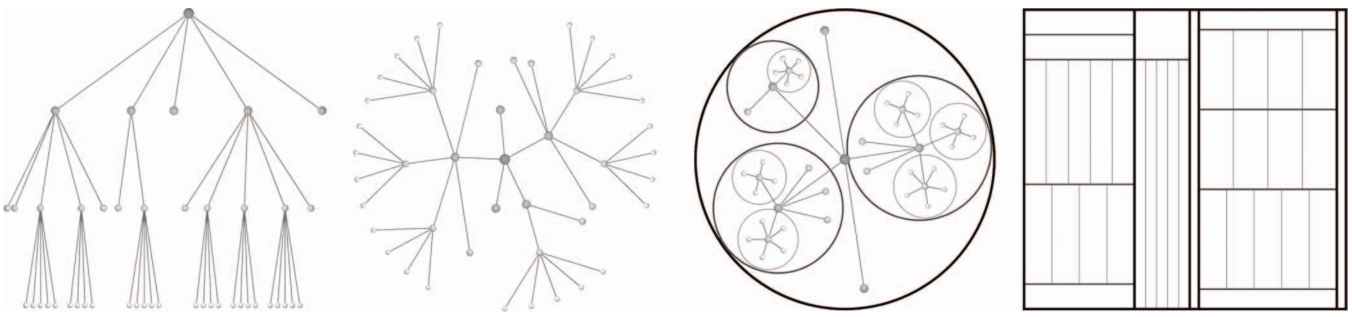


Fig. 1: Different kind of tree visualizations, from left to right: top-down tree, radial tree, balloon tree and tree map. (From "*Hierarchical edge bundles: Visualization of adjacency relations in hierarchical data*, by D. Holten" [4])

- Provide additional metric information.
- Scales well when big hierarchical structures are visualized.

**Cons**

- Not really informative in terms of visualizing groups of similar ancestors.

### 3.2 Association relations

Association relations represent the links between entities of the system. An example of such a relation is a call dependency, where a relation occurs when a method calls another method. A fundamental difference with respect to containment graphs is that they do not have to be directed nor tree-shaped. For example in the case of a recursive method, the call graph would create a loop. The classic way of drawing call graphs is by using node-link diagrams, where relations are represented by edges between nodes.

## 4 CALL GRAPHS

In the next section, we analyze techniques that visualize association relations. Note that these techniques can visualize all kinds of association relations equally well, for example call relations, dependency relations or a data flow relation. However, we focus on call relations for simplicity.

Since call relations occur on software entities - and these entities are containment relations - call graphs consist of both containment and association relations. A visualization technique that is able to visualize both relations has to be introduced. When visualizing software systems, the association relations are usually drawn on top of the containment visualizations. Drawing the call graph on top of a radial tree is an example, this can be seen in figure 3, where $\beta = 0$. When $\beta = 0$, a classical node link diagram is created.

**Pros**

- Since the lines are straight, it is relatively easy to follow the relations.

**Cons**

- Becomes cluttered when the amount of relations increases.

### 4.1 Hierarchical edge bundles

Holten [4] proposed a technique, hierarchical edge bundling(HEB), where edges are bundled in order to reduce cluttering. The proposed technique is a way to visualize containment - and association relations, which is an advantage compared to other techniques. Since call graphs consist of the same type of relations, this method can be used to visualize a call graph.

The main idea of the algorithm is to use control points along the hierarchy that attract the edges to that point. This technique makes sure that edges transform into spline curves and this results in bundling of edges. Relations that traverse through the same parent nodes will be bundled together.

According to Holten, it is a flexible and generic method that can be used on top of existing visualization techniques. Also, visual clutter is reduced when dealing with a large number of edges. Furthermore, the method is customizable by controlling the strength of the bundling of edges. This is very useful since it is easier to gain insight in high-lever information when the bundling strength is high, while low bundling strength emphasizes on low-level connectivity information. An example of different bundling strengths can be seen in figure 3.

Telea et al.[9] performed a comparative study where users were asked to perform certain tasks. They concluded that using HEB is indeed an improvement compared to using classical node-link diagrams. One of the main advantages was that HEB is able to show more data on the same amount of space. A second advantage is that the bundling makes sure that edges become less cluttered. However, classical node-link diagrams also have advantages, one of them is that it is easier to follow a relation compared to HEB.

### 4.2 City

Using cities to represent call graphs is an extension of the tree map in section 3.1.4, it is developed by Wettel and Lanza[11]. The city is a 3D visualization technique that displays object-oriented languages. The main reason that cities are used as metaphors to visualize software systems is that cities provide a clear concept of orientation, since they are familiar for humans.

A city consists of buildings and districts, buildings represent classes and districts represent packages. The height and size of the base of each building is related to chosen metrics of classes. Placement of buildings is done using a modified version of the tree map algorithm. An example of a software system visualized as a city can be seen in figure 5. In this example it is possible to see which classes act like data classes, they are represented as wide but low buildings. The classes that are long and tall have a large amount of methods, but do not contain as much attributes.

**Pros**

- Is able to show multiple code metrics together.
- Scales well when big hierarchical structures are visualized.

**Cons**

- Only takes care of classes and packages, class internals are not displayed.

### 4.3 City using HEB

Since cities are mainly focused on visualizing hierarchical structures, and HEB is mainly focused on improving adjacency relations, Casert et al. [1] proposed a combination of the two techniques. This technique first displays the city and on top of that the HEBs are drawn. The control points are in 3D, since the HEBs are drawn above the city visualization and a 3D scene is required. Each control point, that represents a level of hierarchy, is positioned at a different height. Higher levels of hierarchies are placed on higher altitudes. The goal of this

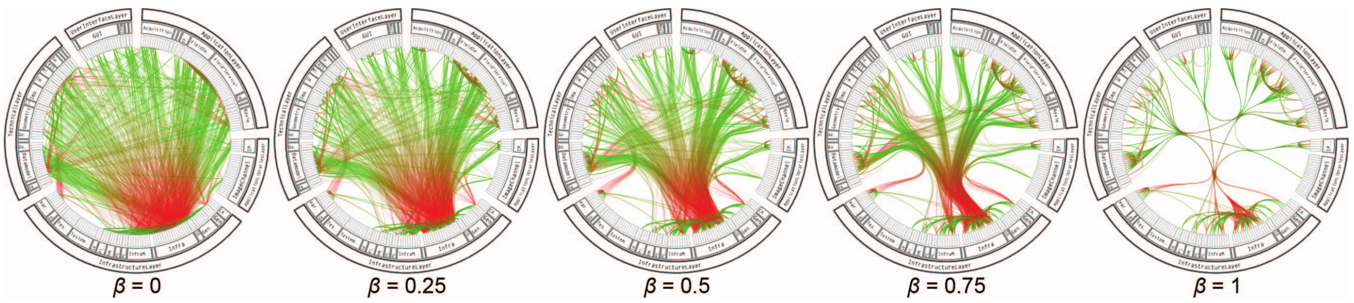


Fig. 3: Call graph visualized on top of a radial tree. The edges are bundled with different bundling strengths($\beta$). (from "*Hierarchical edge bundles: Visualization of adjacency relations in hierarchical data*, by D. Holten" [4])

technique is to use best of both worlds; the results can be seen in figure 4.

**Pros**

- Is able to show multiple code metrics together.
- Containment relations are also visualized.
- Scales well when big hierarchical structures are visualized.

**Cons**

- Only takes care of classes and packages, class internals are not displayed.
- Containment relations details are lost when a lot of association relations are drawn.

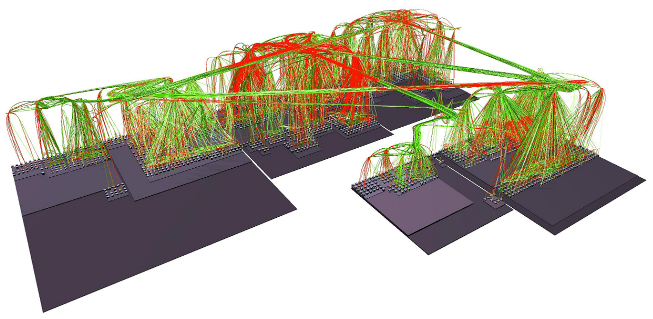


Fig. 4: Call graph visualized on top of a city using HEB, where edges are attracted using 3 control point altitudes. (from "*3D Hierarchical Edge Bundles to Visualize Relations in a Software City Metaphor*, by Caserta et al." [1])

## 5 Tools

There are multiple tools used both for extracting metrics from the code as well as creating the graphical structures of it. In this section we will show a small sample of such tools and describe their cons and pros. The tools were tested by inputting different software code bases to them.

The tools are reviewed on several requirements that are part of the ISO Standard 25010 for software quality [6].

- Functionality
  - Feature set
  - Capabilities
- Usability
  - Human factors
  - Documentation
- Performance
  - Speed
  - Scalability

When we are talking about the tools used to extract metrics or call dependencies from the software systems, there are in general two main categories. The *lightweight* and *heavyweight* extractors:

**Lightweight** extractors such as, GCCXML[1] and MCC[8], only perform a partial extraction of metrics. That means that they will provide *some* information about the code but not a complete insight. As the name suggests, these tools are fast in terms of computation time.

**Heavyweight** extractors, in contrast, perform a full parsing of the software system. They are in general slower than the *lightweight* extractors and they require more computational power. Such *heavyweight* extractors are, Columbus[2] or SolidFX[2].

The following tools are analyzed.

### 5.1 CodeCity

CodeCity[3] is a heavyweight 3D visualization tool that is able to visualize C++, C# and Java languages. However, a separate metric's extractor is required when Java is analyzed. As we can see the visual representations (5) looks very similar to a city (thus the name CodeCity). The working of the CodeCity visualization is described in section 4.2. CodeCity's compatibility levels are pretty good since it supports three different programming languages. The user is able to "navigate through the city", subparts of software systems can be explored this way, and it adds to the feature set of the tool. The visual representations of CodeCity allow the user to show two metrics (the height of the building and its area). When all relations are shown on top of a city, the general overview can be highly disturbed, this is why CodeCity offers an option to filter out certain relations. Figure 5 shows a city without relations, figure 6 shows filtered relations on top of the city.

CodeCity allows the user to save the visualized repository as Smalltalk[4] and Parcel which adds to the functionality of the tool.

CodeCity's usability is mediocre, but it can cover the most fundamental expectations of a developer. CodeCity however does not maintain any kind of documentation other than a scientific paper where the tool is introduced, and some video tutorials. Its performance is very good for small projects. Lastly the functionality of the tool is limited. Other than the city, it does not provide any other kind of views.

[1]http://gccxml.github.io/HTML/Index.html
[2]http://www.solidsourceit.com/products/SolidFX-static-code-analysis.html
[3]http://www.inf.usi.ch/phd/wettel/codecity.html
[4]http://www.smalltalk.org/main/

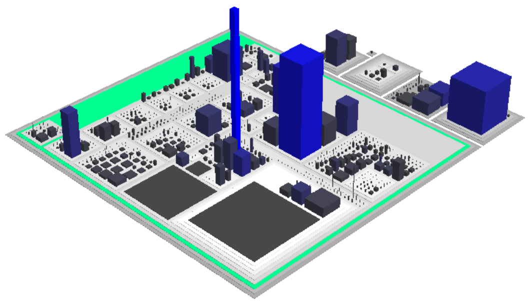


Fig. 5: Visual representation of JEdit (http://www.jedit.org/) source code.

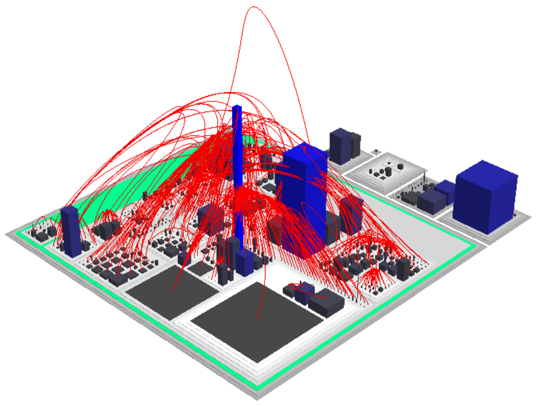
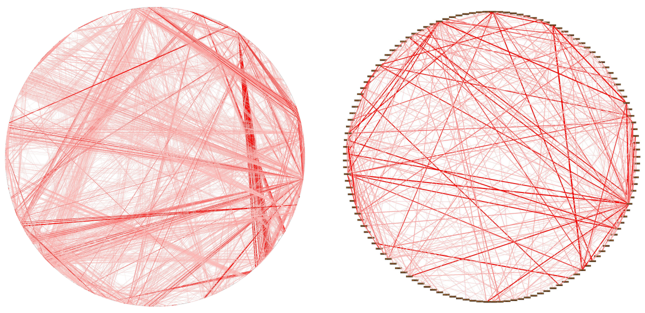


Fig. 6: Visual representation of the JEdit source code along with the relations (red strings).

### 5.2 X-Ray

X-Ray[5] is also a heavyweight Eclipse plugin written in Java, it only supports visualization of Java projects. X-Ray uses the *node-link diagram* to represent the relations between the various classes or packages. In figures 7a and 7b we can observe the results we got from the tool when we applied it to the JEdit repository for the class and package relations respectively. On the negative side, X-Ray only supports saving images as JPG.



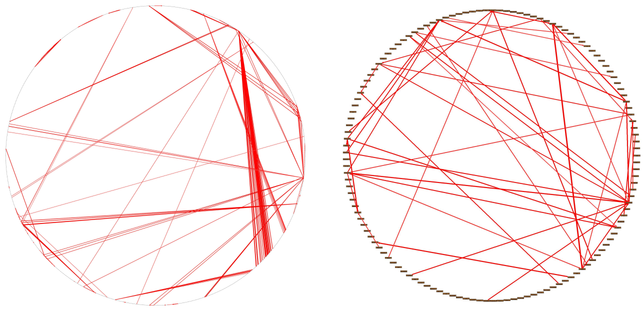
(a) Class dependencies. (b) Package dependencies.

Fig. 7: Visual representation of the JEdit repository.

Figure 7 however is too chaotic, there is too much visual clutter due to the huge amount of relations. A functionality that X-ray provides is filtering out specific relations, this reduces visual clutter. In figures 8a and 8b, the effect of filtering out "weak" relations is shown.

[5]http://xray.inf.usi.ch/xray.php

Since Telea et al. showed that HEB is also able to reduce visual clutter while keeping the amount of relations the same, it would be a good improvement of X-Ray if HEBs are used.



(a) Filtered Class dependencies. (b) Filtered Package dependencies.

Fig. 8: Visual representation of the JEdit repository after filtering out the weak relations.

At this point it is worth mentioning that when experimenting with X-Ray, several problems occurred, the tool was not reliable.

Overall X-Ray did poorly compared to the other two tools. Its functionality is very limited as well as the usability, which is partially supported by documentation. The graphs it produces are static without any dynamic properties. The speed of the tool is also unreasonably small. When we used it, it took a lot of time for project small in size.

### 5.3 Structure101

Structure101[6] is also a heavyweight tool like the previous two but its the one with the highest variety of functionalities. These features include containment and and association relations, metrics and architectural views. The tool however does not stop there since all these different view are dynamic and allow the user to interact with them in an efficient and convenient way. In figure 9 and 10 we can see the interface of Structure101 and some of its views. To create those figures we used the Apache Ant[7] repository. Figure 9 shows an interactive overview of the Apache Ant repository. Structure101 allows the user to interact with this view and expand or collapse various components in order to get a "higher" or a "lower" view. On the top left corner the tool offers a "mini map" where it shows whether the connections inside the repository are structured (bottom left corner) or unstructured (top right corner). Figure 10 shows a dependency graph of packages, it is a node-link diagram without edge bundling. This results in cluttered graphs with many edge crossings. Structure101 also has the option of a textual summary, where explanation on different aspects of the code is given, this includes metrics or architectural structure. Lastly we should mention that the tool employs many more functionalities, such as a matrix overview of the dependencies, or independent views focused on certain components (packages, classes or methods).

Overall, Structure101 has high functionality, usability and performance. It has numerous feature sets, as well as a well structure documentation, which includes textual documentation and tutorial videos. It is highly usable and very straight forward to its use. Its performance is well above medium as well. It loaded and analyzed the Apache Ant in seconds.

### 5.4 Overall evaluation

In table 1, is an overview of our evaluation for each of the tools in this paper. The minimum value is three minus signs, while the maximum three plus signs.

[6]https://structure101.com/products/
[7]http://ant.apache.org/

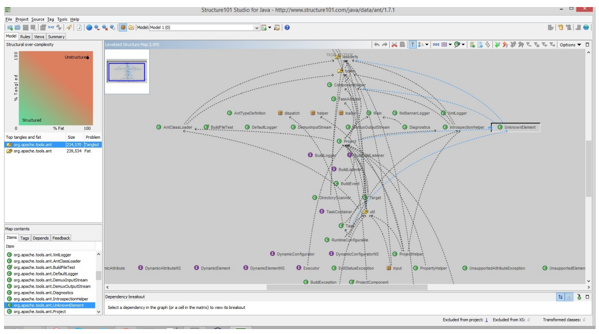


Fig. 9: Interactive overview of the project.

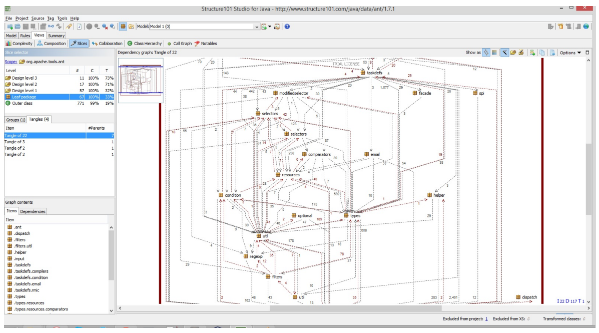


Fig. 10: Interactive dependency graph.

| Tool | Functionality | Usability | Performance |
|---|---|---|---|
| CodeCity | - | + | ++ |
| X-Ray | - - | + | - |
| Structure101 | +++ | + | + |

Table 1: Tools evaluation table.

## 6 Conclusion

In this paper, we described and analyzed different techniques to visualize structures that are present in software. In general, a software system has two kind of structures, containment structures and association relations. Containment structures have many different possibilities to be visualized, an overview is given in section 3. The classical way of displaying association relations is by drawing node link diagrams. In section 4.1, an improvement of classical node link diagrams is shown. However, Telea et al. [9] showed that this improvement also has drawbacks, since relations are harder to follow.

After the description of different techniques that are around, software visualization tools are analyzed and their benefits and drawbacks are listed. The tools are reviewed on some requirements that are part of the ISO standard[6].

## Acknowledgement

We would like to thank prof. dr. A.C. Telea for reviewing our article.

## References

[1] P. Caserta, O. Zendra, and D. Bodenes. 3d hierarchical edge bundles to visualize relations in a software city metaphor. In *Visualizing Software for Understanding and Analysis (VISSOFT), 2011 6th IEEE International Workshop on*, pages 1–8, Sept 2011.

[2] R. Ferenc, Á. Beszédes, M. Tarkiainen, and T. Gyimóthy. Columbus-reverse engineering tool and schema for c++. In *Software Maintenance, 2002. Proceedings. International Conference on*, pages 172–181. IEEE, 2002.

[3] M. H. Halstead. Elements of software science. 1977.

[4] D. Holten. Hierarchical edge bundles: Visualization of adjacency relations in hierarchical data. *Visualization and Computer Graphics, IEEE Transactions on*, 12(5):741–748, 2006.

[5] T. Honglei, S. Wei, and Z. Yanan. The research on software metrics and software complexity metrics. In *Computer Science-Technology and Applications, 2009. IFCSTA '09. International Forum on*, volume 1, pages 131–136, Dec 2009.

[6] ISO, Geneva, Switzerland. Systems and software Quality Requirements and Evaluation (SQuaRE), ISO 25010, 2011.

[7] T. J. McCabe. A complexity measure. *Software Engineering, IEEE Transactions on*, (4):308–320, 1976.

[8] P. F. Mihancea, G. Ganea, I. Verebi, C. Marinescu, and R. Marinescu. Mcc and mc#: Unified c++ and c# design facts extractors tools. In *Symbolic and Numeric Algorithms for Scientific Computing, 2007. SYNASC. International Symposium on*, pages 101–104. IEEE, 2007.

[9] A. Telea, H. Hoogendorp, O. Ersoy, and D. Reniers. Extraction and visualization of call dependencies for large c/c++ code bases: A comparative study. In *Visualizing Software for Understanding and Analysis, 2009. VISSOFT 2009. 5th IEEE International Workshop on*, pages 81–88, Sept 2009.

[10] I. Tollis, P. Eades, G. Di Battista, and L. Tollis. *Graph drawing: algorithms for the visualization of graphs*, volume 1. Prentice Hall New York, 1998.

[11] R. Wettel and M. Lanza. Visualizing software systems as cities. In *Visualizing Software for Understanding and Analysis, 2007. VISSOFT 2007. 4th IEEE International Workshop on*, pages 92–99. IEEE, 2007.

# Hooking up forces to produce aesthetically pleasing graph layouts

Johannes F. Kruiger and Maarten L. Terpstra

**Abstract**—With the advent of social networks as Facebook and Twitter, graph data has taken an even more important stance in the world. Visualising graph data is important as it may lead to new insights in the data, e.g. hidden relations or bottlenecks in a system. At the same time it can be a difficult task, especially when the graph is of substantial size.

Naive methods of graph drawing, e.g. random placement of vertices, typically produce poor results where edges have many crossings and vertices are placed too close to each other. This is not aesthetically pleasing and thus may not produce the prospected insights. Force-directed graph drawing may produce a solution to this problem. By applying constraints to the location of vertices, better results can be obtained. These constraints include dynamic forces inspired by physics, like gravity, spring forces, and electrostatic forces. The weights of these forces can be influenced by vertex measures, as its degree, its closeness to other vertices, or more complex measures.

In this paper, we compare several approaches[7, 1, 2] to create a force-directed graph layout. We perform a comparison of the approaches in running time, compactness, and the number of edge crossings and a comparison of the various results in terms of shape and comprehensibility. These comparisons will mostly be qualitative. Moreover, we try to adapt the approaches to also be applicable to weighted graphs, and propose a new vertex measure inspired by the methods described in[1] and edge weight.

**Index Terms**—Graph visualization, force-directed graph drawing, graph theory

## 1 Introduction

Many real-world problems can be modelled by using graphs. A graph is a set of vertices and edges $G = \{V, E\}$. The vertices represent entities in the model, and the edges represent connections between those entities.

In this paper, we discuss various methods for drawing graphs. The following introductory sections will give some motivation into why drawing graphs is important, and how force-directed graph drawing approaches this problem. Section 2 explains various methods of force-directed graph drawing, and Section 3 introduces some additions to these methods.

### 1.1 Graph layouts

A graph can be laid out in numerous ways, many of which are not insightful. The field of graph drawing tries to devise methods for creating insightful graph layouts. A layout is spatial representation of a graph, where every vertex has a position. The methods we describe focus on producing *good* layouts. That is, if we think of producing a layout as a function that takes a graph as input, it will output the position for every vertex. A good layout fulfils several properties, such as:

- Few edge crossings
- Little occlusion
- Maximised angular resolution[1]
- Compactness
- Uniform edge length
- Spatial distance between vertices is proportional to their graph-theoretical distance.

- *Johannes F. Kruiger, Rijksuniversiteit Groningen, E-mail: j.f.kruiger@student.rug.nl.*
- *Maarten L. Terpstra, Rijksuniversiteit Groningen, E-mail: m.l.terpstra@student.rug.nl.*

[1]The angular resolution is the smallest angle that is formed by two edges that share a vertex.

Of course, some of these properties have conflicting aims. The art is to find a desirable balance. Graph layouts can be in either 2D or 3D, but we will restrict ourselves to 2D in this paper.

Figure 3 shows two layouts of the same graph. The first has been randomly laid out, and the second has been laid out with a more advanced technique.

A graph's *plane embedding* is a 2D layout of the graph where no edges intersect other edges[6]. In practice, this may not always be feasible (or mathematically possible) but aiming for a low number of edge crossings is a good strategy for finding insightful graph layouts.

For small graphs (consisting of, say, $< 10$ vertices), making an insightful layout can easily be done by hand. However, when the graph contains a large number (say, $> 10^5$) of vertices and edges, laying it out in an aesthetically pleasing way becomes non-trivial.

Examples of use cases for large graph data are:

**Social networks** Networks like Facebook, Twitter and Google+ contain many users. The graphs are used mainly to model relations between the users, but also to model relations which denote that a user is fond of a certain entity, or that a user is attending a specific event, etc. Visualising these relations gives insight into the social importance of users, and insight in the influence of certain entities.

**Recommendation engines** Web shops are often interested in maximizing profit. One way of doing that is suggesting which products their users should buy. Web shops often use graph databases to recommend purchases based on certain criteria. Visualising this data may be useful for gaining insight in costumer trends.

**Computer networks** Computer networks are very complex, especially for large companies. Tracking errors in the network can be tedious, but graph visualisation can make this process a lot more convenient.

**Path finding** Finding paths in a graph is a classic problem. To give the user more insight in the paths, the graph can be laid out in such a way to enhance the visibility of the paths.

Naive strategies of producing graph layouts are straightforward, but often produce poor results. An example of a naive method is employing random placement of vertices and drawing their respective relations. The reason these graphs are cluttered is because vertices may be placed too close to each other, causing overlapping vertices, which diminish the visibility. Also, using random placement of vertices, the probability that the edges have many crossings is quite high, especially for larger graphs. This makes the relations hard to follow.

Using naive strategies like this is appealing, because it is straightforward to implement and it has a low time complexity.

### 1.2 Force-directed graph drawing

A more appealing way to produce graph layouts is by using force-directed graph drawing. Force-directed graph drawing is a strategy of drawing graphs that has the potential to produce much better results than naive methods of drawing graphs. It employs several features commonly found in the field of physics - such as electric forces, spring forces, and gravitational forces - to obtain a better layout of a graph. This layout is often better because it applies constraints we also observe in the real world, causing the graph to 'behave naturally'. People experience these layouts as better because they tend to have fewer edge crossings and fewer overlapping vertices. Oft-times, the graphs will also exhibit an angular resolution which is higher than graphs obtained using the naive method.

Typically, vertices are modelled as physical entities with a positive charge, and edges are modelled as springs that connect the vertices. In the following sections, the physical forces will be explained more thoroughly.

These physical forces are iteratively applied to the vertices, and the vertices are moved in the direction of the resulting net force. This is continues until a certain terminating condition is fulfilled. Typical termination conditions are

- The maximal vertex displacement during one iteration is smaller than some threshold $\tau$.
- A maximal number of iterations have passed.

It is desired that, when the algorithm has ended, the layout is in equilibrium, or minimal energy state.

#### 1.2.1 Hooke's law

If we let $k_s$ denote the spring constant, $x$ denote the current spring length, and $x_0$ denote the natural spring length, Hooke's law states that the magnitude of the spring force is approximated by:

$$F_H(x) = k_s\,(x - x_0)\,.$$

It can be seen that the magnitude of this force is positive (i.e. attractive) when the spring is larger than its natural length, and negative (i.e. repulsive) when the spring is shorter than its natural length.

Given two vertex positions $\vec{p}_1$ and $\vec{p}_2$, this results in the following force vector on the first vertex:

$$\vec{F}_H(\vec{p}_1, \vec{p}_2) = k_s\left(||\vec{p}_2 - \vec{p}_1|| - x_0\right)\hat{p}_{12},$$

where $\hat{p}_{12}$ denotes the unit vector in the direction from $\vec{p}_1$ to $\vec{p}_2$, given by $\hat{p}_{12} = \frac{\vec{p}_1 - \vec{p}_2}{||\vec{p}_1 - \vec{p}_2||}$. And of course, by Newton's third law, an opposite force of equal magnitude acts on the second vertex.

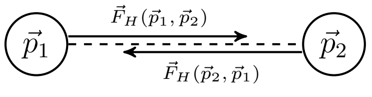

$\vec{p}_1$
$\vec{F}_H(\vec{p}_1, \vec{p}_2)$
$\vec{F}_H(\vec{p}_2, \vec{p}_1)$
$\vec{p}_2$

Fig. 1. The attractive spring forces between two vertices, in the case that the spring is elongated.

This force is desired since it allows compactness, uniform edge length, and prevents occlusion. This is due to the nature of springs, and therefore results in vertices that are close, but not too close.

#### 1.2.2 Coulomb's law

Since every vertex is modelled to have a positive charge, they exert repulsive forces to all other vertices, according to Coulomb's law of electrostatic interaction. The magnitude of the force between two charges $q_1$ and $q_2$ is given by:

$$F_E(x) = \frac{k_e q_1 q_2}{x^2},$$

where $k_e$ is Coulomb's constant, and $x$ is the distance between the charges. It is important to notice that this force does not work in the direction from $q_1$ to $q_2$, but in the direction from $q_2$ to $q_1$. This means that the force is repulsive when $F_E$ is positive, and attractive when $F_E$ is negative, as opposed to Hooke's law, where it is the other way around. Using this, it can be seen that the force is attractive when the charges are of opposite sign, and repulsive when the charges are of equal sign.

For two vertices with positions $\vec{p}_1$ and $\vec{p}_2$ and charges $+1$, this results in the following force vector on the first vertex:

$$\vec{F}_E(\vec{p}_1, \vec{p}_2) = \frac{k_e}{||\vec{p}_1 - \vec{p}_2||^2}\hat{p}_{21}$$

and an equal but opposite force on the second vertex.

Note that the unit vector $\hat{p}_{21}$ is opposite to the one used for Hooke's law, $\hat{p}_{12}$, to make the force be repulsive when the charges are equal, and attractive when the charges are opposite.

$\vec{F}_E(\vec{p}_1, \vec{p}_2)$
$\vec{p}_1$
$\vec{p}_2$
$\vec{F}_E(\vec{p}_2, \vec{p}_1)$

Fig. 2. The electrostatic repulsive force between two vertices.

It is important to realise that this force acts between every pair of vertices. Therefore, it might turn out to be very expensive, $\mathcal{O}(|V|^2)$, when no special care is taken. Walshaw et al. use a grid structure to store vertex proximity, which makes this a lot cheaper[7].

The repulsive Coulomb force prevents occlusion, and undesired cluttering of vertices. This allows for fewer edge crossings, a higher angular resolution, and less occlusion.

#### 1.2.3 Gravity

Oft-times, a global gravitational force is employed to direct the vertices to the center of the layout. This can be simulated by placing an invisible mass $M$ in the center of the layout (position $\vec{0}$), and applying the following gravitational force on a vertex at $\vec{p}$ with mass $m$:

$$\begin{aligned}\vec{F}_G(\vec{p}) &= G\frac{mM}{||\vec{0} - \vec{p}||}\left(-\hat{p}\right)\\ &= -G\frac{mM}{||\vec{p}||}\hat{p},\end{aligned}$$

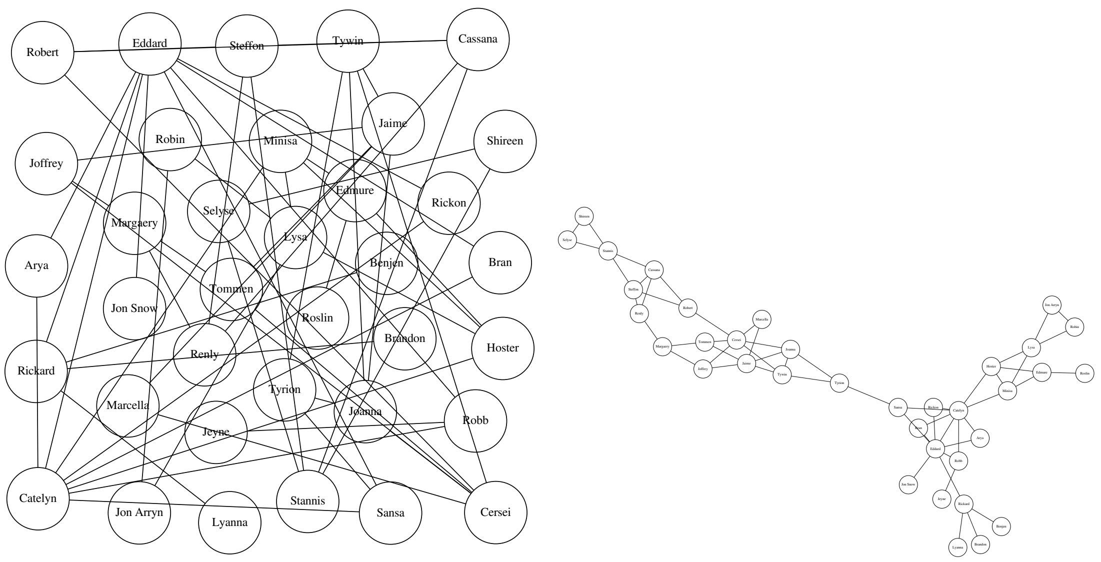
where $\hat{p}$ denotes the unit vector in the direction of $\vec{p}$, and $G$ is the gravitational constant.

Since we only need a single scaling parameter, we can set the global mass to $M = 1$:

$$\vec{F}_G(\vec{p}) = -G\frac{m}{||\vec{p}||}\hat{p}.$$

The vertex mass $m$ can still be used to give more gravitational pull to certain vertices, based on the graph's properties.

The gravitational force is mostly applied to keep the graph centred. It is desired, because it contributes to the compactness of the graph, and it can also be used to minimize edge crossing, as will be observed later.



(a) A poor layout of a graph produced by a naive method. This image was made using the *Graphviz* package.

(b) A better layout produced by a spring embedding technique. This image was made using the *Graphviz* package.

Fig. 3. A comparison of a graph that has been randomly laid out, and the same graph that has been laid out using force-directed graph drawing.

## 2 Methods

In this section, three methods will be discussed. The first being Walshaws's multilevel approach [7], the second the social gravity approach by Bannister et al., and the third pre-positioning approach by Dong et al. We describe what these methods entail and weigh several advantages and disadvantages of the methods.

These methods are approaches to force-directed graph drawing, and aim to provide good layouts for certain situations, by reaching a minimal energy state in short time. These more advanced methods hope to outperform simple Euler integration in terms of speed and layout quality.

### 2.1 Multilevel approach

The multilevel approach, as described in [7], on the most global level, works as follows:

1. Coarsen (see Section 2.1.1) the input graph $G_0$ $L$ times, so there will be a list of increasingly coarsened graphs $\{G_0, G_1, G_2 \ldots, G_L\}$. The graph coarsening decreases the number of vertices and edges in every step by merging *matching* vertices.
2. Assign an initial layout for the coarsest graph, $G_L$. This is not computationally expensive, since the coarsest graph has very few vertices.[2]
3. Take a step back in the level of coarseness, and place the new child vertices[3] on the location of their parent vertices. Perform force-directed placement on this finer graph.
4. Repeat the previous step until the original graph $G_0$ is restored and perform the final force-directed placement on this graph.

This approach is considered an improvement over simple force-directed placement on the original graph, because of the following two reasons:

- The convergence time of these $L$ force-directed placements is shorter, because the vertices are generally initialised on a location close to the location where they have their global energy minimum.
- The quality of the layout is better, because initialising the vertices on their neighbours' location results in fewer edge crossings.

#### 2.1.1 Coarsening graphs

For the coarsening of the graph, a process called matching is used. A matching of a graph is a set of mutually non-adjacent edges. This means that the edges in the matching set cannot share any of the vertices they connect.

So how should we construct this matching? The trivial answer to this question is to simply take a random edge from the graph. By definition, this singleton set has no vertices which are shared with other edges in the set. However, using a matching method like this results in a high number of levels $L$.

On the other side of the spectrum, we can determine the maximal matching, which is the matching that has the highest possible number of vertices. Unfortunately, algorithms that solve this problem are at least of order $\mathcal{O}\left(|V|^{2.5}\right)$. Since this is of too high time complexity, Walshaw used a variant of the heuristic proposed in [3], which works as follows:

1. Create a randomly ordered list of the unmatched *vertices*.
2. Iterate over the vertices, and match every unmatched vertex with a neighbouring unmatched vertex. If a vertex has no unmatched neighbours, match the vertex with itself. Remove the matched vertices from the list, and add the edge that connects them to the matching set.

Such a matching strategy is a convenient compromise that matches a decent number of edges within a reasonable time complexity.

### 2.2 Social approach

Another approach to draw graphs is to take several social measures into account, as described in [1]. On a global level, this approach works as follows:

[2]The number of vertices in the coarsest graph, and the level of coarsening $L$, can be determined by some coarsening threshold.

[3]A child vertex is one of the vertices that are being replaced by their coarser level representation.

1. Calculate the net force on each vertex. Usually, the forces consist of the gravity, the repulsive forces of the other vertices and the attractive forces of the edges. The gravity starts at zero when placing the vertices, but is gradually increased as time progresses. (See Figure 4.) The gravitational force of every vertex is scaled by the social measure derived for that vertex. Typically this social measure equals the degree, closeness (reciprocal of the mean distance of a vertex to all other vertices) or the betweenness (proportion of paths containing the vertex among all shortest paths in the network) but can also be chosen to be a more complex measure, such as PageRank[5] or Katz centrality[4].
2. Move the vertex in the direction of the force. By moving the vertices in the direction of the acting force, the net force on the vertex is reduced, bringing the embedding closer to an equilibrium.
3. These steps are repeated until an equilibrium is reached. When an equilibrium is reached, it should approximate a good layout which is a suitable stopping criterion.

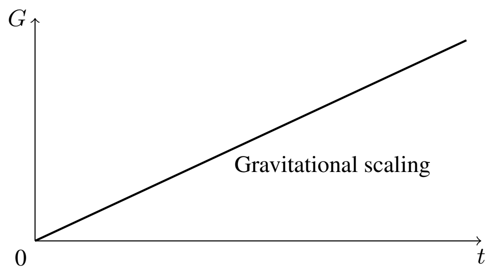


Fig. 4. Scaling of the gravitational force as a function of time in the placement algorithm.

A clear advantage of this method is that it is relatively simple, since it is easy to mentally visualise. Also, it has a low computational cost, because computing this additional gravitational force does not result in a lot of overhead. The real advantage of this method is that high centrality vertices will be guided towards the centre of the layout, and the final result of the placement is a very compact layout, as a result of the gravity. Finally, it is also flexible. The centrality measure can be chosen to be any function the user desires. Also, the scaling function can be manipulated, which could produce interesting results.

A disadvantage is that this method tends to produce circular graph layouts. This might be too suggestive, in the sense that the user might conclude that the graph has a circular structure, even though it has no such structure.

### 2.3 Advanced pre-positioning

Two of the largest problems of force-directed graph drawing are high running times compared to naive methods and the chances of ending up in a poor local minimum.

The high running time is caused by the large number of iterations often required until a layout is found that is in equilibrium. A local minimum is encountered when the algorithm finds a layout where small changes do not necessarily improve the layout, even though it turns out that a better layout is indeed possible.

These two problems can largely be avoided by a good prepositioning prior to force-directed graph drawing. If the prepositioning is good, then a lower number of iterations is required until a proper layout is found as opposed to a random initial placement.

A good pre-positioning to avert the aforementioned problems is what Dong et al.[2] are trying to achieve. Dong et al. describe the following method:

1. Determine the importance of each vertex using VertexRank, which is a modified version of PageRank
2. Group the vertices into layers based on the shortest path from a vertex to the most important vertex (i.e. the one with the highest VertexRank)
3. Determine relationship matrix
4. Place most important vertex at the centre
5. Place vertices connected to already placed vertices with an as large as possible angle at a distance proportionate to the relation with the placed vertex

After this is done, the normal force-directed algorithm is performed for a number of iterations until an optimal placement is obtained.

#### 2.3.1 Vertex importance

Dong et al. have chosen to replace PageRank with their own ranking system. While PageRank seems like a rational choice for ranking vertices, as this algorithm has proven itself to be a good choice for ranking vertices, the authors had good reasons to adapt it to their own ranking system. PageRank in its classical form is denoted as $\text{PageRank}(p_i) = \frac{1-d}{N} + d \sum_{p_j \in M(p_i)} \frac{\text{PageRank}(p_j)}{L(p_j)}$ where $d$ is a damping factor, $N$ is the number of vertices, $p$ represents a vertex of the graph, $M(p_i)$ denotes the set of vertices linking to $p_i$ and $L(p_i)$ represents the set of vertices that $p_i$ links to.

The keen observer notes that PageRank is an operation applied to every vertex in a directed graph, but common force-directed algorithms are generally applied to undirected graphs. This distinction can be circumvented by defining a undirected graph as a directed graph where each undirected edge $(p_i, p_j)$ is represented as two half-edges $\{(p_i, p_j), (p_j, p_i)\}$. This does not have to be implemented explicitly as one can define that for each vertex the in-degree is equal to the out-degree which are both equal to the degree of a vertex. Now this problem may be solved, but now PageRank no longer has a direct connection with the centrality or importance of a vertex, as illustrated in Figure 5.

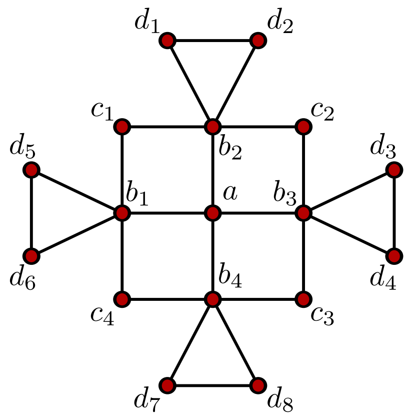


Fig. 5. The most important vertex is $a$ but PageRank will not identify it as such but rather vertices $b_1 \ldots b_4$. VertexRank will identify $a$ as the most important vertex. This image is based on [2].

Instead, Dong et al. describe VertexRank as

$$\text{VertRank}(p_i)_{\text{new}} = (1-d)\text{VertRank}(p_i)_{\text{old}} + d \cdot \sum_{p_j \in C(p_i)} \text{VertRank}(p_j)$$

where $C(p_i)$ is the set of vertices connected to $p_i$. This measure can successfully assign importance scores to vertices.

#### 2.3.2 Vertex layering

In the second step, vertices are classified into two different classes: central vertices and layer vertices. For central vertices, the following criterion holds true: $\forall p_j \in C(p_i), \text{VertRank}(p_j) \leq \text{VertRank}(p_i)$.

This means that central vertices have at least as high a VertexRank as all vertices that it shares a connection with. These vertices are placed first using force-directed methods. In later steps layered vertices are placed. A layered representation of a graph can be seen in Figure 6.

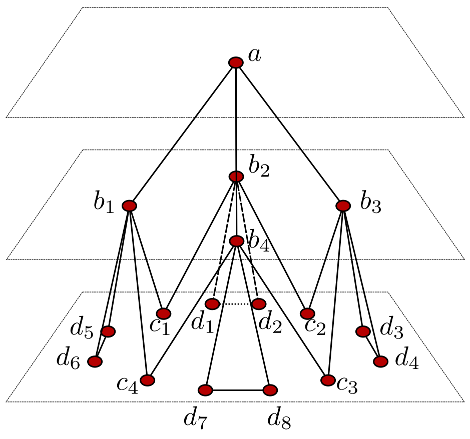


Fig. 6. The layered graph representation of figure 5. It is clear that $a$ is the central vertex. The layer number represents the distance to the central vertex. This image is based on [2].

### 2.3.3 Relation matrix

Relations between vertices can be strong clues where to position vertices as to minimize edge crossings. For example, in Figure 6, vertices $b_1 \dots b_4$ must be placed adjacent to $a$ because they share an edge. However, their mutual order has as infinite number of options in terms of order, swaps or rotations. It is not enforced that vertex $b_2$ is placed between $b_1$ and $b_3$ because they share no direct edge. However, they do share an indirect edge via vertices $c_1$ and $c_2$. These indirect connections can also be clues where to place vertices, possibly increasing the quality of the resulting layout. This information is stored in the *relation matrix*. For every position $(i, j)$ a 1 is stored if $p_i$ and $p_j$ share a direct connection, 0.5 if they share an indirect connection or 0 if there is no connection at all. This information is later used to determine better positions for vertices.

### 2.3.4 Vertex placement

Once the relations are determined, each layer is placed in an as good as possible way. This is done by placing the central vertex at the centre first, and then placing each layer iteratively at the largest possible angle between other vertices, at a distance proportional to the relation between two vertices.

After all layers have been placed, the regular force-directed algorithm is performed for as many iterations as needed before it converges to a proper layout, ideally conforming to the properties stated in Section 1.1.

## 3 New measures

All the measures we have discussed so far are mostly applicable to undirected, unweighted graphs. However, if a directed or weighted graph is considered, useful information will be discarded with the current measures. Therefore, we introduce our own measure which takes information from weighted and directed graphs into account when determining vertex locations.

### 3.1 Weighted graphs

A weighted graph is very similar to an unweighted graph, with the difference that to each edge a number, or weight, is attached. This weight can have a lot of different meanings depending on the purpose of the graph or the intention of the creator. Common examples are distance, time, maximum capacity, closeness of a vertex, or just general cost.

The weight of an edge can play a vital role in creating a comprehensive layout of the graph. For example, suppose a high edge weight indicates that there is a weak relation between two vertices. Intuitively, one would place these vertices further apart than two vertices who have a strong relationship. This potentially important information is lost when the aforementioned measures are used to determine distance between vertices.

We have devised two different measures when weighted graphs are considered, which we will discuss separately.

#### 3.1.1 Centrality scaling

The first measure is a measure which has influence on gravity imposed on vertices by scaling the centrality of a vertex. The original methods as described in [1] merely take into account the number of connections to a vertex, but not the importance of the connections. Therefore, we adapt the centrality of a vertex as computed by the aforementioned measures to be scaled by the importance of every connection. For example, if the measure is the degree of $v$, which is the sum of number of edges containing $v$, it could instead be sum of the weights of the edges containing $v$.

#### 3.1.2 Edge length scaling

Another measure we have devised is edge length scaling based on the weight of an edge. In this measure, the edge length between two vertices is scaled according to the weight of that edge. This length can be chosen to be between a certain minimum and maximum, resulting in the following formula for determining the length of an edge based on the weight:

$$l = L_{\text{min}} + \frac{\omega - \omega_{\text{min}}}{\omega_{\text{max}} - \omega_{\text{min}}} \cdot (L_{\text{max}} - L_{\text{min}})$$

where $\omega$ is the weight of the considered edge, $\omega_{\text{min}}$ and $\omega_{\text{max}}$ are the lowest and highest edge weight in the graph, respectively and $L_{\text{min}}$ and $L_{\text{max}}$ are the minimum and maximum new edge length, respectively. Note that special care has to be taken when $\omega_{\text{max}} = \omega_{\text{min}}$, in other words, when all weights are equal.

This measure has as a result that vertices connected with a low weight are placed closer together, but if desired an analogous formula for the inverse effect can easily be found (which is omitted for brevity). One must consider that the measure effectively reduces the degrees of freedom when placing a graph since vertices are placed at a fixed distance of each other. The only remaining degree of freedom is the angle at which a vertex is placed. For general graph layouts, this measure may be too limiting to produce a proper graph layout but if it is applicable it may certainly help to identify important vertices in relation to other vertices based on the distance between them.

A more forgiving use of this method is to set the natural spring length of the spring force between the vertices to this length $l$, so that the spring contracts when it is larger than the length, and expands when it is smaller than the length.

### 3.2 Directed graphs

An important type of graph is the directed graph. For these graphs, a relation between vertices does not indicate merely the existence of the relation (e.g. vertex A and vertex B share a connection) but also the direction of the relation (e.g. vertex A has a link to vertex B). This type of relation is often found in family trees (e.g. the is-parent relationship) and in many visualisations of computing science phenomena, such as the internet, inheritance of objects in object-oriented programming, and graph databases in general.

Since directed graphs can document not only the existence of a relation but also the direction of the relation, more information can be obtained which could help visualising graphs using force-directed graph drawing.

It is debatable whether it is desirable to use direction information, since it not necessarily always affects the topology of the graph. It is hard to determine at runtime whether the user wants to optimise the layout for this additional information. We believe that there are use cases where one definitely wants to optimise, using this information.

For example, when visualising web pages and their respective links, this information is certainly useful, since in-links are a better indication of importance than out-links.

#### 3.2.1 PageRank

A known measure to determine the 'popularity' of a vertex is to base it on the amount of links to a vertex. This measure is for example used in Google's search engine by the PageRank algorithm[5]. In an undirected graph it is not trivial to successfully use PageRank to aid force-directed graph drawing. However, using a directed graph, the popularity of each vertex can be determined by the PageRank algorithm. This information can be useful at the initial placement of a graph prior to the algorithm of force-direction. The popularity information can act as heuristic for initial placement of a graph by placing the most popular vertex at the centre of the layout. All other vertices should be placed at an offset from the centre of the layout based on how much less popular they are, similar to [2]. Using these initial placements, the placement algorithm generally converges faster.

## 4 Discussion

In this section we will discuss the differences and parallels between the current methods and why our own measures might be an improvement over already existing measures.

### 4.1 Existing methods

We see parallels between the approaches by Dong et al. and Walshaw et al. Both methods apply a so-called 'top-down' approach by placing the vertices in descending order by importance. Subsequently, they both extend the graph by placing less important vertices in a most optimal way.

However, there are also striking differences between the approaches. The method of Dong et al. operates directly on the graph, whereas the method of Walshaw et al. generates a new representation of the graph by grouping vertices together in new vertices that represent multiple lower-level vertices. Moreover, in the method by Dong et al. the vertices are placed optimally by a geometric approach while Walshaw et al. employ a force-directed graph placement approach on every level. There is also another more fundamental difference: the Walshaw et al. approach is an approach that is intended to create a force-directed layout, whereas the Dong et al. approach is a pre-positioning algorithm where force-directed graph drawing is the subsequent step.

The social approach described by Bannister et al. seems to employ a very different approach compared to the clustering or layering approaches as described by Walshaw et al. and Dong et al. However, we do not think of these two fundamentally different approaches as competing aims, but rather as two different approaches which may make more sense to use in a certain context. We suspect that the social approach is more useful on highly interconnected graphs (i.e. there are relatively many edges per vertex) and the clustering and/or layering approach is more sensible in a relatively sparse graph (i.e. there are relatively few edges per vertex).

One observation we noticed with the social approach is that it tends to plot graphs in a circular layout, as a consequence of the gravity. This suggests to the user that every branch of the graph is of equal size, which is not necessarily the case and is thus a minor case of serious deception.

### 4.2 New measures

#### 4.2.1 Gravity scaling based on edge weights

Normally, information about edge weight would be discarded when using traditional force-directed graph drawing methods. By using this information for the social gravity measures a better layout could be obtained. We think that when considering two vertices with the same number of connections, the vertex that has a higher incoming edge weight is more important than the other vertex. Therefore, we think this information should be used to adjust the centrality of a vertex, therefore adjusting the gravity imposed on the vertex. We suspect that this would result in fewer edge crossings and vertices with a higher centrality have a better chance of ending up in the centre of the layout.

#### 4.2.2 Weighted edge length scaling

Usually, every edge is of equal importance in an unweighted graph and therefore obtains roughly the same length in the force-directed placement. When considering weighted edges, using a variable length which is a function of the edge weight a more insightful layout could be obtained. By scaling unimportant edges we hope that fewer edge crossings will be observed in the resulting layout and important vertices could be better recognized.

#### 4.2.3 PageRank

PageRank is a well-known vertex ranking algorithm for directed graphs. We suspect the vertex rank can be used to adjust the centrality of a vertex for using it with the method of Bannister et al., or in the pre-positioning step as described in Dong et al. Dong et al. have researched using PageRank before but were unable to use it due to lacking a directed graph. However, if a directed graph is available it should be possible to use it in their algorithm and may result in a better layout.

In the social gravity approach as described by Bannister et al. it may produce superb layouts, compared with not using direction information. By adjusting the centrality using the ranks of vertices, the vertices with a high PageRank will have a high probability of becoming the most central vertex, which is as intended.

## 5 Conclusion

Force-directed graph drawing has proven itself to be a versatile and robust method for producing aesthetically pleasing graph layouts. There are many methods to improve the process in terms of running time, quality of result and flexibility. We have discussed several methods in our paper, where each method has its advantages and disadvantages. We hope that the readers of this paper have gained insights in the possibilities and options when intending to use force-directed graph drawing.

We have also described several new measures which could improve force-directed graph drawing methods by using additional information available about a graph. However, we have not implemented these measures so we cannot conclude that they can indeed produce better layouts compared to traditional force-directed graph drawing methods. Therefore we imagine that future work includes implementing our measures for weighted or directed graphs so users can benefit from the described methods directly. Moreover, the algorithms could be tested on graphs that are used in scientific visualisation to ensure more insight can be obtained.

## Acknowledgements

We wish to thank our reviewers for their useful comments and suggestions.

## References

[1] M. J. Bannister, D. Eppstein, M. T. Goodrich, and L. Trott. Force-directed graph drawing using social gravity and scaling. In *Graph Drawing*, pages 414–425. Springer, 2013.

[2] W. Dong, F. Wang, Y. Huang, G. Xu, Z. Guo, X. Fu, and K. Fu. An advanced pre-positioning method for the force-directed graph visualization based on pagerank algorithm. *Computers & Graphics*, 47:24–33, 2015.

[3] B. Hendrickson and R. W. Leland. A multi-level algorithm for partitioning graphs. 1995.

[4] L. Katz. A new status index derived from sociometric analysis. *Psychometrika*, 18(1):39–43, 1953.

[5] L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: Bringing order to the web. 1999.

[6] R. J. Trudeau. *Introduction to graph theory*. Courier Corporation, 2013.

[7] C. Walshaw. A multilevel algorithm for force-directed graph drawing. In *Graph Drawing*, pages 171–182. Springer, 2001.

# Choosing between optical flow algorithms for UAV position change measurement

Jasper de Boer, Mathieu Kalksma

**Abstract**—In recent years optical-flow-aided position measurement solutions have been used in both commercial and academic applications. These systems are used for navigating unmanned aerial vehicles (UAVs) in GPS-deprived environments. Movement in sequential images is detected and converted to real world position change. Multiple approaches have been suggested, ranging from using an optical mouse sensor to the use of a stereo camera setup. Our research focuses on single camera solutions.
Previous research has used a variety of optical flow algorithms for single camera solutions. This paper presents a comparison on three algorithms to check if using different algorithms yield different results in terms of quality and CPU time. This paper also provides insight into the general theory behind using single camera optical flow for UAV navigation.
The compared algorithms are the Lucas-Kanade method, Gunnar Farnebäck's algorithm and Block Matching. A testing framework and custom indoor and outdoor datasets were created to measure algorithms flow estimation quality and computation time.
Characteristic differences were found between the performance of the algorithms in terms of both computation time and quality. Choosing between algorithms therefore can increase flow estimation quality or reduce CPU time usage. Also different winners per test set were found in terms of estimation quality.

**Index Terms**—UAV navigation, optical flow, Lucas-Kanade method, Gunnar Farnebäck method, Block Matching.

## 1 Introduction

In recent years the interest in using unmanned aerial vehicles (UAVs) has increased. UAVs can be used for a multitude of applications, for example the inspection of agricultural lands, reviewing annual ditch cleanings, or the inspection of wind turbine blades. An essential part to such systems is position measurement which is needed for point-to-point navigation or for maintaining a position.

One method for navigating UAVs is Global Positioning System (GPS). However, for some applications GPS introduces a significant error to the location [7] and GPS will not work when there is no signal available, for instance in indoor environments.

An alternative to GPS are optical-flow-aided position measurement systems. These systems use sequential images from a camera and computer vision to measure movement. In recent years several single camera solutions have been proposed, which use different optical flow algorithms: SAD Block Matching [4] and Lucas-Kanade [9]. For both algorithms the results are promising.

In 2014 the Twirre architecture for autonomous mini-UAVs using interchangeable commodity components was introduced [14]. The platform is developed for automating inspection tasks. The architecture consists of low-cost hardware and software components. The processing power is positioned on-board the UAV. This allows the platform to run the recognition software on-board. Twirre is able to use optical flow for position measurement.

With architectures like Twirre, CPU time is shared with software that is performing the inspections. Optical flow computation time should therefore be kept to a minimum. This paper therefore investigates if performance in terms of estimation computation time can be improved by choosing the right algorithm. This paper also compares the quality of the algorithms in order to see if optical flow quality can be improved by choosing between algorithms.

The algorithms that will be evaluated in this paper are: SAD Block Matching, the Lucas-Kanade method [8] and the Gunnar Farnebäck technique [5]. The first two algorithms are used in [4] and [9] and the Gunnar Farnebäck technique is used in Twirre. These three algorithms were chosen because they are already applied in the field and because of their availability in the OpenCV library [2]. OpenCV is easy in use and extensively used in the field of computer vision. Its BSD license allows both academic and commercial use [3].

The paper starts with a background chapter which introduces optical flow and how it can be used for measuring position change in terms of real world distance. Section 3 shortly introduces the three used optical flow techniques. The algorithms are compared by looking at their theoretical differences and their efficiency. Section 3 finishes with explaining the test environment, and the data sets that are used for testing the three algorithms. Section 4 provides the results of the tests and section 5 and 6 will give the conclusion and discussion, and some future directions accordingly.

## 2 Background

This section introduces a method for measuring position change with a single camera optical flow system. This section also discusses how the choice of camera and optics affects the optical flow system. Some of the choices made for the research method (Section 3) are based on the method described in this section.

The goal of a optical flow system is to measure x, y and rotational position change on the plane parallel to the ground in a real world measure. The described method assumes the use of a camera that is aimed towards the ground.

### 2.1 Optical flow

An optical flow algorithm is able to track points across two images. Given a set of points or pixels in the first image, the algorithm tries to locate the points in the second image. From the result two corresponding sets of vectors can be constructed. the source vectors

$$\mathbf{v}_{src}[n] = \begin{pmatrix}\mathbf{p}_x[n] & \mathbf{p}_y[n]\end{pmatrix}^T$$

and the destination vectors

$$\mathbf{v}_{dst}[n] = \begin{pmatrix}\mathbf{p}_x[n]+\mathbf{t}_x[n] & \mathbf{p}_y[n]+\mathbf{t}_y[n]\end{pmatrix}^T$$

where $\mathbf{p}[n]$ is the original pixel coordinate and $\mathbf{t}[n]$ contains the translation of the pixel. Figure 1 visualizes this process.

Most optical flow techniques are based on the brightness constancy assumption and the spatial smoothness assumption [1]. With the brightness constancy assumption it is assumed that points keep the same intensity between frames. The spatial smoothness assumption

- *Jasper de Boer is MSc. Computing Science student at the University of Groningen and a Project Engineer at the Centre of Expertise in Computer Vision of the NHL University of Applied Sciences, E-mail: jasper.boer@nhl.nl.*
- *Mathieu Kalksma is a MSc. Computing Science student at the University of Groningen, E-mail: m.kalksma@student.rug.nl.*

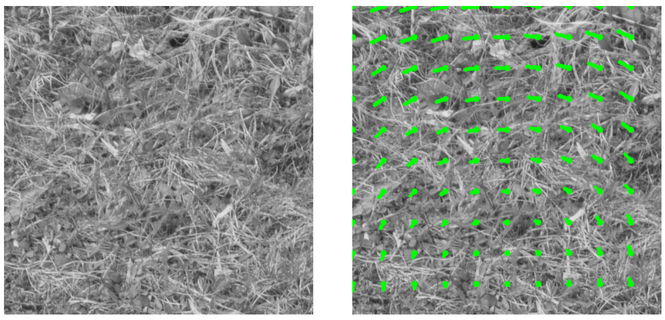

(a) first frame (b) second frame with the detected flow

Fig. 1: Illustration of detected optical flow between two frames. The second frame is a slightly rotated and translated version of the first frame.

comes from the observation that neighboring pixels generally belong to the same surface and therefore have similar motion [11].

### 2.2 Angular correction

An UAV has three principal axis as seen in figure 2. The described method uses similarity transformation estimation, and hence angular correction should be applied for both pitch and roll axis. This correction can be done by using a camera gimbal or by transforming the $\mathbf{v}_{src}[n]$ and $\mathbf{v}_{dst}[n]$ vectors.

A gimbal can be used in order to keep the camera perpendicular to the ground. A gimbal uses an inertial measurement unit (IMU)[1] and electric engines to allow the camera to pivot around one axis (yaw), and therewith blocking other rotational actions (rolling and pitching).

The second approach uses the orientation information from an IMU in order to transform both $\mathbf{v}_{src}[n]$ and $\mathbf{v}_{dst}[n]$ vectors.

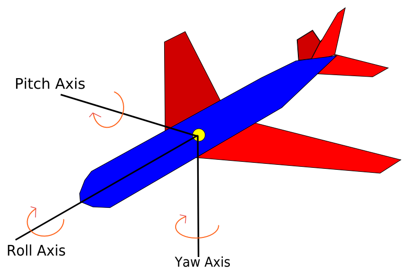


Fig. 2: Principal axis of an UAV. By Auawise (Own work) [CC BY-SA 3.0 (http://creativecommons.org/licenses/by-sa/3.0)], via Wikimedia Commons

### 2.3 Estimating the similarity transformation

After applying angular correction, the transformation between two frames is reduced to translation $\mathbf{t}$, rotation $R$ and uniform scaling $c$. This four degrees of freedom similarity transformation can be estimated by reducing the least squares error as in equation 1 [13].

$$e^2(R,t,c) = \frac{1}{n}\sum_{i=1}^{n} ||\mathbf{v}_{dst}[i] - (cR\mathbf{v}_{src}[i] + \mathbf{t})||^2 \tag{1}$$

### 2.4 Pixel distance to real world distance

To calculate real world position change it is necessary to know the distance between the UAV and the ground plane. The height $h$ can be measured e.g. with an ultrasonic distance sensor or with a barometric pressure sensor. Equation 2 is used for calculating the real world traveled distance, where $\mathbf{s}$ is the pixel size of the camera sensor and $f$ the focal length.

$$\mathbf{t}_{rw} = \frac{1}{f}\mathbf{ts}h \tag{2}$$

From the equation can be seen that a shorter focal length increases the real world traveled distance per moved pixel. Also a greater height and larger size of the camera sensor pixels increase the real world distance. Along with the optical flow algorithm and the frame rate of the camera, these parameters determine the maximum real world speed at which the optical flow system is able to measure position change.

[1]An IMU uses accelerometers and gyroscopes to measure orientation and gravitational forces.

## 3 Materials and Methods

In this section will provide the setup of the research. The section starts with some theoretical information about the used algorithms. After giving the theoretical background the section continues with the developed testing framework. Finally it concludes by explaining how the experiments are performed and by introducing the used data sets.

### 3.1 Description of examined optical flow algorithms

In this section the three different algorithms for computing optical flow are presented. They will be presented by explaining their theory, efficiency and showing whether they calculate sparse optical flow or dense optical flow. The difference between sparse optical flow and dense optical flow is that sparse flow only calculates the flow for certain specified pixels, while dense optical flow calculates the flow for all the pixels. This makes sparse algorithms often faster. Since dense flow results in more flow vectors, more data is used to minimize the error function in equation 1, which can lead to a better estimation of the overall transformation.

In case of a large movement between the two images the algorithms sometimes can not detect the movement. A solution for this, which both Lucas-Kanade and Gunnar Farnebäck use, are pyramids. With pyramids you take $l$ pyramid levels and you run the algorithm for each level. For each level $L$ the image is shrunk $L$ times resulting in a $2^L$ times smaller image. With the image shrunk, the initial too large movement is now detectable.

First the Block Matching algorithm is described, followed by the Gunnar Farnebäck method and then the Lucas-Kanade method.

#### 3.1.1 Sum of absolute differences

The sum of absolute differences (SAD) algorithm is measures similarity between image blocks. It starts by having two images of size $M \times N$. For each pixel in the first image a window of $BxB$ is placed over the image, with the current pixel as middle point. In the new frame the the algorithm searches within a given search area around the position of the original pixel. The algorithm moves the window over every possible position within the search area. For every window position in the new frame the SAD value is calculated. The position with the lowest SAD value is returned as the new position of the current pixel. This process can also be described as minimizing in the following function:

$$\sum_{i=0}^{1-b}\sum_{j=0}^{b-1} |(f_k(x+i, y+i) - f_{k-1}(x+i+u, y+j+v))|. \tag{3}$$

In this equation $f_k$ and $f_{k-1}$ are the two images and $b$ is the block size. By minimizing the result the flow of the current pixel can be estimated. This process is repeated for every pixel in the original image.

This algorithm is very time consuming. The operation will require $O(MNR^2B^2)$ time for a $M \times N$ image, $R \times R$ search region and a $B \times B$ block size. The operation is a dense optical flow method, as it calculates the flow for all the pixels in the image.

#### 3.1.2 Gunnar Farnebäck method

The Gunnar Farnebäck method [5] is a two-frame motion estimation algorithm. Gunnar Farnebäck uses quadratic polynomials to approximate the motion between the frames. This can be done efficiently by using the polynomial expansion transform.

In the case of Gunnar Farnebäck the point of interest is quadratic polynomials that produces the local signal model expressed in a local coordinate system such that

$$f(x) \sim x^T Ax + b^T x_c, \tag{4}$$

where $A$ is a symmetric matrix, $b$ a vector and $c$ a scalar.

Gunnar Farnebäck is a dense optical flow algorithm because it computes the optical flow for all pixels in the image.

#### 3.1.3 Lucas-Kanade method

The Lucas-Kanade method describes an image registration technique using spatial intensity gradient information to search for the best match [8]. It does this by taking more information about the image into consideration. With this, the method is capable of finding the best match using far less computations than other techniques that use a fixed order to search. The algorithm takes advantage of the fact that in most cases the two images are already close to each other. The registration problem is finding a vector $h$ that minimizes the distance between two images $F(x)$ and $G(x)$ so that the distance between $F(x+h)$ and $G(x)$ is minimized in a region of interest R. Lucas-Kanade suggests a generalization to deal with distortions such as rotation of the image.

To find the disparity $h$ Lucas-Kanade uses,

$$\begin{aligned} h_0 &= 0, \\ h_{k+1} &= h_k + \frac{\sum_x w(x)F(x+h_k)[G(x)-F(x+h_k)]}{\sum_x w(x)F(x+h_k)^2}, \end{aligned} \tag{5}$$

where $F(x)$ and $G(x)$ are the input images and

$$w(x) = \frac{1}{|G(x)-F(x)|}, \tag{6}$$

is a weighting function.

Finally this algorithm will converge in $O(M^2 \log N)$ [8] time. As Lucas-Kanade only uses given pixels of the image to measure the optical flow it is a sparse optical flow algorithm.

### 3.2 Testing framework

For conducting the experiments a testing application was developed. The program heavily relies on the OpenCV library [2]. The program can perform optical flow on subsequent images with similarity transformations, using one of the three described optical flow algorithms. The desired algorithm and the corresponding parameters can be set. The application estimates the 4 degrees of freedom (DOF) similarity transformation with the corresponding OpenCV function *estimateRigidTransform()*. The application can also use the *findHomography()* function to find the 8 DOF homography transformation.

The program results the pixel translation of the centre point and the global scaling and rotation between subsequent images.

A second program was written for extracting sequential frames from a large picture. This program was used to generate the image data described in Section 3.3. This section describes the frame extraction process in more detail.

### 3.3 Data sets

The program for extracting frames from a larger image was used for generating test sets. The program moves a window over an image. The pixels within the window are saved as a new image. The advantage of this approach is that perfect ground data can be used to compare the results of the algorithms against. This is because the translation and rotation are known. Pitching and rolling actions are not incorporated in the simulator, because they can be corrected for as described in Section 2.2.

Figure 3 shows the followed path for the dotted floor image. The individual frames were converted to grayscale and white Gaussian noise with a mean of zero and a variance of *0.001* was added to the frames to make the simulation more realistic.

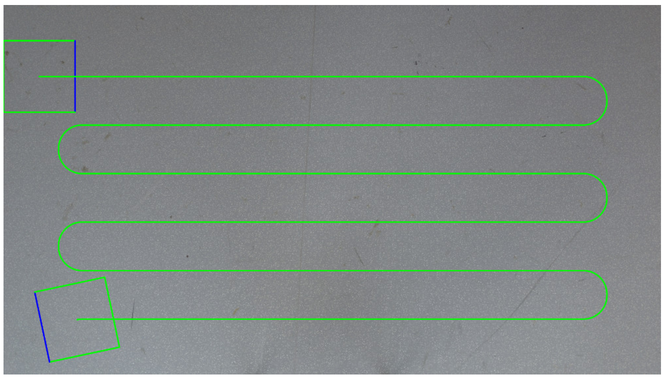


Fig. 3: The path which is followed in order to generate sequential images. The windows of the first and last frame are displayed

Three different images from different types of ground surfaces were used (Figure 4). The first image is an outdoor picture of a grass covered surface. The other two pictures are from indoor floors: a grey floor with small white dots and a floor with red carpet. These three surfaces were chosen in order to have a diversity of data sets where both indoor synthetic floors and an outdoor underground were represented. More data sets could have been generated, but chosen images proved to be diverse enough to answer the research question.

The specifications of the test sets generated from the described images are listed in Table 1. The flow speed can be decreased by binning the frames. By binning an image an area of pixels in the original image is combined into a single pixel for the output image. A two by two binning on an image combines 4 pixels into one pixel and thereby reduces the total number of pixels by a factor 4. The flow speed is then decreased with a factor of 2. The flow speed can be increased by skipping frames.

Table 1: Specifications of the generated and used test sets

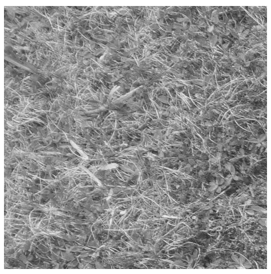
| name | original size | crop size | nr. of frames | flow (px) |
|---|---|---|---|---|
| grass | 4608 x 3456 | 800 x 800 | 1164 | 12.3 - 44.3 |
| dotted floor | 7360 x 4135 | 800 x 800 | 2386 | 12.3 - 45.2 |
| carpet | 7360 x 3856 | 800 x 800 | 2056 | 12.3 - 45.2 |

### 3.4 Experiments

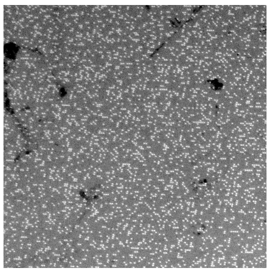
For the experiments, the optical flow algorithm parameters that affect performance were set to the same value for every algorithm. This allows a fair CPU execution time comparison between the algorithms. Other parameters were set according to best practice, as for instance suggested by the OpenCV documentation. Some parameters were adjusted by hand in order to increase flow recognition quality. The range of the scanned neighborhood parameters were set according to known information about the maximum flow range, as listed in Table 1.

The first performed experiment was a CPU processing time comparison. Table 2 lists the parameters used. An Intel i5-4300U based system was used to perform the tests. For the experiment the grass set was used. 50 frames were processed in order to measure the average flow computation time.

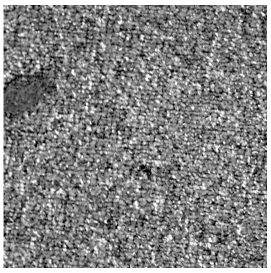
The Lucas-Kanade algorithm requires the locations of features that should tracked. For finding suitable features the Shi-Tomasi corner detector algorithm [10] was used. The execution time of this algorithm was determined for the same binning factors as the optical flow algorithms.



(a) Grass. By Joshua Ezzell (Own work) [CC BY-SA 2.0 (https://creativecommons.org/licenses/by/2.0/)], via Flickr



(b) Dotted floor. The image contrast is enhanced for visibility reasons



(c) Carpet. The image contrast is enhanced for visibility reasons

Fig. 4: First crops of the three ground surfaces

The second experiment was a comparison on the quality of the optical flow algorithms with different types of floor. For determining the quality, the estimated movement of the centre point of the frame was tracked. As error value the average accumulated distance between sequential middle points was calculated, as seen in equation 7. The measure averages out standard normal distributed errors. This measure was chosen since in a real world application the movement of a UAV, in order to get the current location, is accumulated as well.

$$e = \frac{c}{N}||\sum(\mathbf{t}_{et} - \mathbf{t}_{gt})|| \quad (7)$$

In equation 7, $N$ is the number flow estimations, $\mathbf{t}_{et}$ is the estimated translation, $\mathbf{t}_{gt}$ is the ground truth translation and $c$ is the cropping factor.

## 4 Results

This section describes the results and gives an interpretation. First the results for the CPU-time comparison are given followed by the quality comparison results.

### 4.1 Computation time comparison

The results of the computation time algorithm are displayed in Figure 5. The results for Block Matching with a smaller binning size than four were left out, since these computation times were a magnitude higher than applicable in the field. The Shi-Tomansi algorithm was added individually to the graph instead of summing the algorithm together with Lucas-Kanade, since the Shi-Tomansi does not have to be used each iteration when applying Lucas-Kanade. It only has to be used when (too many) current good features move out of frame.

The graph shows that the Lucas-Kanade and Farnebäck algorithm outperform simple Block Matching in terms of execution time. The distinction between Lucas-Kanade and Farnebäck is harder to make because of the logarithmic scaling. Table 3 shows the original timings. From this table the differences in computation times are clearly distinguishable. Farnebäck is faster when binning with atleast a factor of 2 is applied.

Table 2: Several of the used parameters for comparing the differences in CPU processing time

| | Block Matching | Lucas-Kanade | Farnebäck |
|---|---|---|---|
| binning | - | - | - |
| search area | 43 | 43 | 43 |
| block size | 3 | 3 | - |
| pyramid levels | - | 3 | 3 |
| binning | 2 x 2 | 2 x 2 | 2 x 2 |
| search area | 22 | 22 | 22 |
| block size | 3 | 3 | - |
| pyramid levels | - | 3 | 3 |
| binning | 4 x 4 | 4 x 4 | 4 x 4 |
| search area | 11 | 11 | 11 |
| block size | 3 | 3 | - |
| pyramid levels | - | 3 | 3 |
| binning | 8 x 8 | 8 x 8 | 8 x 8 |
| search area | 6 | 6 | 6 |
| block size | 3 | 3 | - |
| pyramid levels | - | 3 | 3 |
| binning | 16 x 16 | 16 x 16 | 16 x 16 |
| search area | 3 | 3 | 3 |
| block size | 3 | 3 | - |
| pyramid levels | - | 3 | 3 |

Table 3: Computation time for several binning factors of Farnebäck and Lucas-Kanade

| | computation time in milliseconds | |
|---|---|---|
| binning factor | Farnebäck | Lucas-Kanade |
| 1 | 384.8 | 327.7 |
| 2 | 53.0 | 70.0 |
| 4 | 12.3 | 22.8 |
| 8 | 3.1 | 5.2 |
| 16 | 0.7 | 1.2 |

### 4.2 Quality comparison

The results of the quality comparison are presented in table 4.

For the dotted floor and carpet data sets the *findHomography()* function was used instead of *estimateRigidTransform()*. The resulting flow for these sets were sometimes too far off for *estimateRigidTransform()*. This occurred specifically for the Lucas-Kanade algorithm and the Block Matching algorithm. Figure 6 shows one of these situations.

All tested algorithms perform reasonably well on the natural grass data set. The errors are reasonably small, since the real average Euclidean flow distance of the grass set is *16.0 pixels*. Figure 7 visualizes the detected movement of the algorithms. The 8 x 8 binning measurements were used to draw the image.

The rotation of the ground truth was used. Further analysis of the results show that rotations are not properly detected, as seen in Figure 8. As soon as a rotation is introduced, the rotational error drastically increases, considering the simulator rotates with 3 degrees per frame, when it rotates. Accumulating the rotation would lead to wrong flight paths. This error is introduced by the functions that estimate the global image transformation. In an real wold UAV application a magnetometer can be used.

The dotted floor proved to be harder to detect for all algorithms. Farnebäck and Lucas-Kanade still perform reasonably well with a binning of 4 and 8, considering the real average Euclidean flow distance is *16.0*. The poor performance of Block Matching can be explained, because most of the dotted floor surface pixels have (near) the same brightness value. Since Block Matching only searches locally, the algorithm will often assign a wrong flow vector.

The Lucas-Kanade algorithm and Farnebäck's algorithm fail when a 16 x 16 binning is applied. Running the Shi-Tomansi separately showed that, without binning, the dots were detected as corners. With

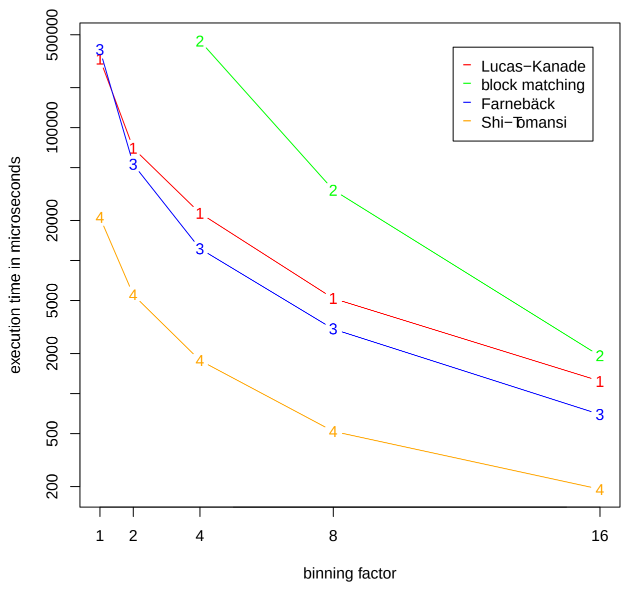


Fig. 5: The computation time according to the binning factor for the three compared algorithms. A logarithmic scaling is used on the execution time axis

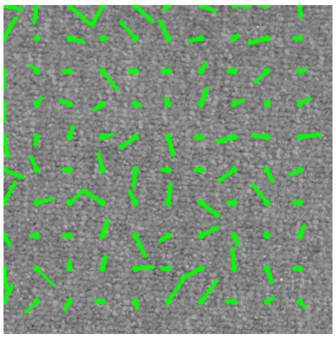


Fig. 6: Wrongfully detected flow by the Block Matching algorithm. The actual movement is a x-axis translation to the right

the highest binning applied however, the dots were to small for the Shi-Tomansi algorithm to detect.

The flow of the carpet data set is well detected at a binning factor of 4. When a binning factor of 8 is applied, Shi-Tomasi does not find enough good points. The Lucas-Kanade algorithm therefore sometimes returns less than 4 flow vectors, which is not enough for estimating the homography transformation. The flow is set to zero in such cases, which leads to high errors in table 4. The same holds true for a binning factor of 16.

Block Matching performs reasonably well with 4 x 4 and 8 x 8 binning, while Farnebäck performs excellent for all binning factors. Figure 9 shows the recognized paths for the algorithms when a 8 x 8 binning is used. The ground truth rotation is used again for the heading of the paths.

## 5 Conclusion and Discussion

This paper shows that choosing between optical flow algorithms for UAV position change measurement can lead to a decrease of CPU-time usage. The more sophisticated algorithms: Lucas-Kanade and Farnebäck are significantly faster than Block Matching, especially for larger picture sizes.

Table 4: Quality measurements of the algorithms on the different floors. For (R), transformation is calculated with *estimateRigidTransform()* and for (H), transformation is calculated with *findHomography()*.

| | binning | | |
|---|---|---|---|
| | 4 x 4 | 8 x8 | 16 x 16 |
| grass | | | |
| Block Matching (R) | 0.65 | 0.24 | 0.19 |
| Lucas-Kanade (R) | 0.17 | 0.18 | 0.47 |
| Farnebäck (R) | 0.17 | 0.16 | 0.14 |
| dotted floor | | | |
| Block Matching (H) | 5.41 | 2.34 | 7.70 |
| Lucas-Kanade (H) | 0.31 | 0.63 | 2.07 |
| Farnebäck (H) | 0.75 | 0.83 | 1.43 |
| Farnebäck (R) | 0.40 | 0.56 | 1.80 |
| carpet | | | |
| Block Matching (H) | 0.17 | 0.72 | 2.42 |
| Lucas-Kanade (H) | 0.21 | 18.14 | 51.44 |
| Farnebäck (H) | 0.49 | 0.49 | 0.12 |
| Farnebäck (R) | 0.27 | 0.28 | 0.10 |

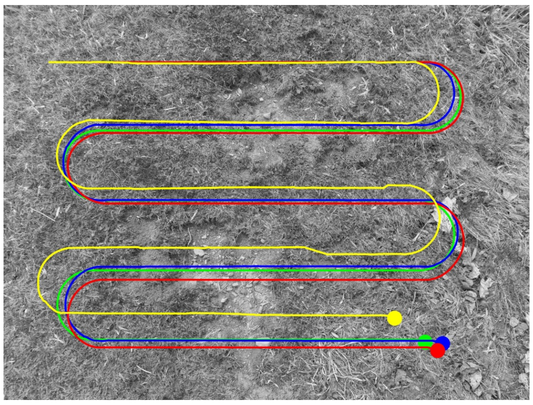


Fig. 7: Visualization of the detected paths for the 16 x 16 binned frames from the grass data set. green is the ground truth, blue is Block Matching, red is Farnebäck and yellow Lucas-Kanade.

Experiment results show that the algorithm's flow recognition quality differ per surface. Also the best algorithm differs per data set and binning factor. Choosing between algorithms can therewith increase optical flow estimation quality. The best optimal flow quality can be obtained by switching between algorithms for different undergrounds.

There is no overall winning algorithm, although in general the Farnebäck method performs best. An surprising result is that the recognition quality does not always degrade by using a higher binning factor. This is particularity interesting because higher binning factors lead to lower CPU-time usage.

During the research a testing framework was developed, which allows fast comparisons between the different optical flow algorithms. Also new data sets for new floor surfaces can be easily generated. The framework can easily be extended in order to test new optical flow algorithms and is therefore also useful for further research.

Due to the low number of examined surface types, the finding that Farnebäck's method performs best cannot directly be generalized to all surface types. Also the simulator did not incorporate scaling. Different height settings may lead to different results.

The compared algorithms were only tested to a certain extent but considering the amount of parameters, performance of in particular Gunnar Farnebäck's algorithm and the Lucas-Kanade algorithm can be further optimized. Also only one feature detection algorithm was used for Lucas-Kanade.

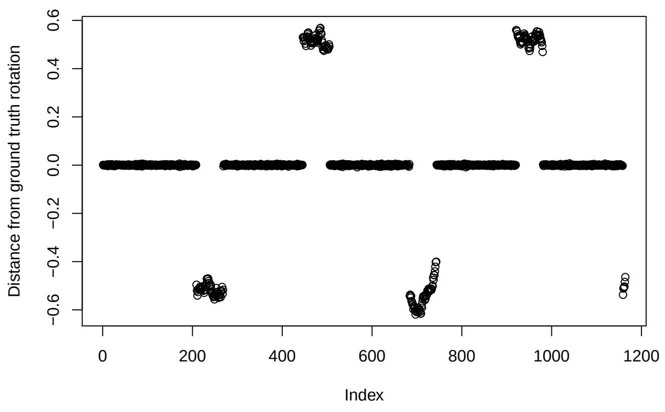


Fig. 8: rotational error in degrees. The index axis corresponds to the index of the frame. The results from Farnebäck with a binning factor of 8 on the grass data set were used

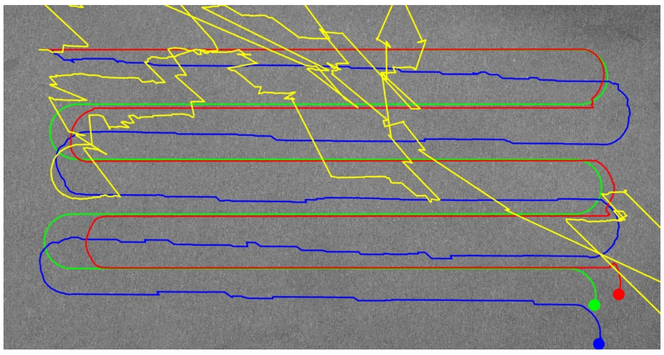


Fig. 9: Visualization of the detected paths for the 8 x 8 binned frames from the carpet data set. green is the ground truth, blue is Block Matching, red is Farnebäck and yellow Lucas-Kanade.

## 6 Future work

The presented research only compared three algorithms. More algorithms can be taken into consideration. Since the developed framework is designed to be extended, new algorithms can be tested. in a short amount of time. Other interesting algorithms might for instance be simple flow [12] and the Horn-Schunck method [6].

The application that takes frames out of a big picture can be extended to support full affine transformations instead of only similarity transformations. More data sets can be generated and tested to be able to further generalize the results of the experiments.

The results show that the used transformation estimation functions were not able to detect rotations correctly. A magnetometer can be used as an alternative, however, in situations where the earths magnetic field is disturbed by the environment, values from a magnetometer are not reliable. Further research is necessary to see if rotation estimation from optical flow can be improved.

## 7 Acknowledgments

The authors would like to thank the expert reviewer, prof. dr. M. Biehl and the anonymous reviewers who reviewed drafts of this paper. This research was partially funded by *Nationaal Regieorgaan Praktijkgericht Onderzoek SIA* under project: *Smart Vision voor UAVs*.

## References

[1] M. J. Black and P. Anandan. A framework for the robust estimation of optical flow. In *Computer Vision, 1993. Proceedings., Fourth International Conference on*, pages 231–236. IEEE, 1993.

[2] G. Bradski. The opencv library. *Dr. Dobb's Journal of Software Tools*, 2000.

[3] I. Culjak, D. Abram, T. Pribanic, H. Dzapo, and M. Cifrek. A brief introduction to opencv. In *MIPRO, 2012 Proceedings of the 35th International Convention*, pages 1725–1730. IEEE, 2012.

[4] H. D., M. L., T. P., and P. M. An open source and open hardware embedded metric optical flow cmos camera for indoor and outdoor applications. *ICRA2013*, 2013.

[5] G. Farnebäck. Two-frame motion estimation based on polynomial expansion. In *Image Analysis*, pages 363–370. Springer, 2003.

[6] B. K. P. Horn and B. G. Schunck. Determining optical flow. *ARTIFICAL INTELLIGENCE*, 17:185–203, 1981.

[7] L. Jayatilleke and N. Zhang. Landmark-based localization for unmanned aerial vehicles. In *Systems Conference (SysCon), 2013 IEEE International*, pages 448–451. IEEE, 2013.

[8] B. D. Lucas, T. Kanade, et al. An iterative image registration technique with an application to stereo vision. In *IJCAI*, volume 81, pages 674–679, 1981.

[9] H. Romero, S. Salazar, and R. Lozano. Real-time stabilization of an eight-rotor uav using optical flow. *IEEE Transactions on Robotics*, 25(4):809–817, August 2009.

[10] J. Shi and C. Tomasi. Good features to track. In *Computer Vision and Pattern Recognition, 1994. Proceedings CVPR &#039;94., 1994 IEEE Computer Society Conference on*, pages 593–600. IEEE, June 1994.

[11] D. Sun, S. Roth, J. Lewis, and M. J. Black. Learning optical flow. In *Computer Vision–ECCV 2008*, pages 83–97. Springer, 2008.

[12] M. W. Tao, J. Bai, P. Kohli, and S. Paris. Simpleflow: A non-iterative, sublinear optical flow algorithm. *Computer Graphics Forum (Eurographics 2012)*, 31(2), May 2012.

[13] S. Umeyama. Least-squares estimation of transformation parameters between two point patterns. *IEEE Trans. Pattern Anal. Mach. Intell.*, 13(4):376–380, 1991.

[14] J. van de Loosdrecht, K. Dijkstra, J. H. Postma, W. Keuning, and D. Bruin. Twirre: Architecture for autonomous mini-uavs using interchangeable commodity components. *IMAV 2014*, 2014.

university of groningen

faculty of mathematics and natural sciences

computing science