SC@RUG 2013 proceedings

Rein Smedinga
Femke Kramer
editors

2013
Groningen

ISBN: 978-90-367-6316-5
Publisher: Bibliotheek der R.U.
Title: SC@RUG 2013 proceedings
Computing Science, University of Groningen
NUR-code: 980

About SC@RUG 2013

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.

In the academic year 2012-2013 SC@RUG was organized as a conference for the tenth time. Students wrote a paper, participated in the review process, gave a presentation and were session chair during the conference.

The organizers Rein Smedinga 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 Janneke Geertsema for her workshops on presentation techniques and speech skills.

Organizational matters

SC@RUG 2013 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 *Nestor*, the digital learning environment of the university. 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. Three 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. Re-reviewers could accept or reject a paper. All accepted papers can be found in these proceedings.

Students were asked to give a 2-minute presentation halfway the period. The aim of this so-called two-minute madness was to advertise for the full presentation and at the same time offered the speakers the opportunity to practice speaking for an audience.

The conference itself was organized by the students themselves. In fact half of the group was asked to fully organize this day (i.e., make 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. We had dual presentations for each paper. The audience graded both the presentation and the chairing and leading the discussion.

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

In her lectures about communication in science, Femke Kramer explained how conferences work, how researchers review each other's papers, and how they communicate their findings by composing a storyline and cleverly designed images. She also taught workshops on writing a scientific paper and on reviewing such a paper.

Janneke Geertsema 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 did the overall coordination, administration and served as the main manager of *Nestor*.

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 and the re-reviewers), the review process itself for 15% and the presentation for 50% (including 10% for the grading of 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 using the new video recording facilities of the University and with thanks to the CIT crew. The recordings were published on *Nestor* for self reflection. Because of a communication mismatch no video recording was available during the real symposium.

On 19 April 2013, the actual conference took place. Each paper was presented by both authors. We had a total of eight presentations this day.

During the symposium Gert-Jan van Dijk, from Target Holding, acted as a keynote speaker and that same company sponsored the symposium as well.

A special catch up meeting was needed (on 22 May) to have three additional presentations because of absence of the presenters during the real symposium.

Hall of Fame

Because we organized SC@RUG for the tenth time we added a new element: the awards for best presentation, best paper and best 2 minute madness. Therefore, since this year's edition, we will have a Hall of Fame:

- **Best 2 minute madness presentation awards:**
  2013:
  Robert Witte and Christiaan Arnoldus:
  Heterogeneous CPU-GPU task scheduling

- **Best presentation awards:**
  2013:
  Jelle Nauta and Sander Feringa,
  Image Inpainting

- **Best paper awards:**
  2013:
  Harm de Vries and Herbert Kruitbosch:
  Verification of SAX assumption: time series values are distributed normally

Thanks

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

- Faris Nizamic
- Andrea Pagani
- Jos Roerdink
- André Sobieck
- Heerko Groefsema
- Ehsan Warnach
- Vali Codrenanu
- Alex Telea
- Alexander Lazovik
- George Azzopardi
- Tijn van der Zant

and all other staff members who provided sets of papers but were not needed in the review process.

Also, the organizers would like to thank:

- the *Graduate school of Science* for making it possible to publish these proceedings and sponsoring the conference,
- *Target Holding* for sponsoring lunch and providing a keynote speaker and
- *Janneke Geertsema* for, again, providing excellent workshops on improving presentation skills.

Rein Smedinga
Femke Kramer

Contents

Hybrid cloud security issues in private enterprises

Aurelian Buzdugan and Tuan Luu Dinh

Abstract—Hybrid cloud computing is the combination of two or more clouds, usually between a private and a public cloud. This combination can bring a huge advantage to private enterprises by delivering necessary computing resources instantly, on-demand and commitment free. Computing services may vary from software such as email handling, office applications to data management or computing resources. Yet data and services are concentrated in clouds, therefore it is much more sensitive from the security point of view. In this paper we will assume a hybrid cloud as the combination between a private and public cloud, and we explain the security risks and benefits associated with cloud implementation. In order to create a full overview of this area, both public and private cloud security aspects are presented. We conclude with the assessment of existing security solutions against main key drivers of cloud computing.

Index Terms—Hybrid cloud security, encryption, virtualization, cloud federation.

1 Introduction

Cloud is an elastic execution environment of resources involving multiple stakeholders and providing metered service at multiple granularities for a specified level of quality [1]. Clouds can be employed in different ways such as hybrid, private, public or community. In this paper, we will mainly focus on hybrid clouds as the fusion of a public and private cloud that remain as unique entities, but still bound together. Applying hybrid cloud to an existing environment raises numerous issues, especially in the security area.

Hybrid cloud offers two options for organizations to store their data and services, both in the public and private cloud. Nowadays the public cloud tends to be used on a temporary basis, until the private cloud will reach the level of security that will satisfy all requirements and needs of the organization. Still, public cloud providers have huge resources dedicated for security, management and infrastructure. This is the main advantage of the public cloud compared to the private one, where the resources are limited and are related to organization's size. On the other hand, the private cloud can ensure that custom security requirements are met and that a full control of data exists.

The concept of cloud computing is linked intimately with those of IaaS (Infrastructure as a Service); PaaS (Platform as a Service), SaaS (Software as a Service) and collectively *aaS (Everything as a Service) all of which imply a service-oriented architecture [1]. These types of clouds functionality can be provisioned individually or multiple at the same time.

Infrastructure as a Service (IaaS) refers to offering resources as services to users, basically virtual hardware on a scalable basis. These resources cannot be offered on a low-level basis, due to the fact that make part of the virtualized environment. Platform as a Service (PaaS), provides a platform on which applications can be developed and deployed. Software as a Service (SaaS), referred also as Application Clouds offers implementation of a certain model of process, coupled with IaaS or PaaS [1].

In this paper we make an overview of security issues for both the private and public cloud, in order to show the best security solutions for hybrid cloud. We begin by describing in section 2 the benefits of implementation of a hybrid cloud in an organization. Section 3 will contain the definition of the selected key drivers for assessing the security solutions. In section 4 we describe the actual security issues of clouds from legal and technical points of view. From the technical point of view security can be assessed during the data or service lifecycle. The existing solutions for overcoming security issues described will be presented in section 5. An evaluation of these solutions against defined key drivers is presented in section 7. We aim to represent notable existing solutions for security problems and evaluate them against the defined key drivers for hybrid cloud: security, performance, extensibility and availability. In the next session we discuss future directions of development towards overcoming security issues. In the end we conclude with a personal opinion and make an overview of the entire paper.

2 Benefits of using hybrid cloud computing

Hybrid clouds make use of the mix from public and private cloud infrastructure. Public cloud is used to achieve a maximum cost reduction through outsourcing whereas the private cloud is used to keep sensitive data in known premises. Therefore, hybrid clouds comprise the general benefits of cloud computing such as:

- Data decentralization, which became inexpensive;
- Storage and bandwidth on a pay-as-you-go system;
- No large upfront capital necessary
- Attractive deploying in enterprises or countries with poor infrastructure.

Cloud computing architecture has highly abstracted resources, is scalable, flexible and uses shared resources such as hardware, memory, CPU or time. These aspects indicate a great advantage for private enterprise, which will be able to save on IT infrastructure cost while having guaranteed stable IT resources for operation.

The hybrid cloud provides great benefits to enterprises because of the flexibility. Owning a private cloud for sensitive data, and making use of a public cloud for archiving is a smart solution for many enterprises, that still need to reach their technological development level for their private cloud.

Public clouds make large scale solutions a smart investment and much more affordable. Data is replicated in many locations, which increases redundancy and independence from failure. Also, threat management, timeliness of response in case of an incident and service reliability are valuable assets which public cloud can offer to an enterprise. On the other hand, private clouds will allow to keep very sensitive data in known boundaries, thus having total control over sensitive information.

3 Key drivers

The key drivers are accessible attributes that can be used to evaluate the effectiveness of the existing solutions when evaluating security issues for the hybrid cloud. The key drivers are chosen based on the general advantages and disadvantages that hybrid cloud can bring to its users. Therefore, these key drivers can be used for verifying any

- *Aurelian Buzdugan is with University of Groningen E-Mail: aurelian.buzdugan@yahoo.com.*
- *Tuan Luu Dinh is with University of Groningen E-Mail: luutuan@me.com.*

solution that relates to security problems in hybrid cloud. A listing and explanation of them is given below:

Security key driver indicates the protection level which the solution can provide to the private enterprises. This is the main key driver and must be approached on a high level. If this key driver is not able to achieved, the solution would beat its purpose and render itself useless.

Performance is the second key driver, and denotes the agility of the hybrid cloud when applying the proposed solutions for security problem. This key driver can affect directly the success or failure of the proposed solutions. The main argument is that private enterprise users would be willing to consider a cloud approach only if it meets their evolving performance requirements.

Extensibility is used to measure how easy the hybrid cloud model can be able to scale up and scale down in term of processing speed and available storage, when proposed solutions are applied to the hybrid cloud system. For enterprises that have a stable operation, in short term this may not be a problem. Yet for long term support, extensibility may become an obstacle if the enterprises want to extend their operations.

Availability key driver indicates the availability of data and services under protecting mechanism from the proposed solutions. The private enterprises often look into this key driver because their operations heavily depend on the availability of data and services.

The proposed solutions for addressing security when adapting hybrid cloud model will be evaluated against above key drivers. The purpose of this evaluation is to validate the effectiveness of the solutions in order to identify which solution can be the promising approach for private enterprises that aim to adapt hybrid cloud model to their IT infrastructure.

4 SECURITY RISKS

In this section we present security issues from technological and legal viewpoints and we explain briefly their impact and affected assets of the organization. The first viewpoint describes risks based on the data and service lifecycle focusing on the technology and functionalities used, whereas the second viewpoint describes security issues in the cloud from legal perspective.

4.1 Data lifecycle

When adapting the hybrid cloud model, private enterprises will move a part of their data and services to the infrastructure of the cloud provider. This can be considered as the root cause of all security problems. The data lifecycle and services in the hybrid cloud model will be represented and they are used to identify the possible security issues from each phrase.

When adapting the hybrid cloud model, private enterprises will move a part of their data and services to the infrastructure of the cloud provider. This can be considered as the root cause of all security problems. The data and services lifecycle in the hybrid cloud model is represented below (shown in Fig. 1), and is used to identify the possible security issues from each phase.

Firstly, the possible security problems that may happen to data will be identified by using provided data lifecycle.

In the data generation phase, the security problem about data ownership is raised. In traditional IT environment related to the private cloud, users or organizations own and manage the data. Yet when data is migrated into the cloud, ownership management gets more complicated. Administrators and auditors can become malicious insiders by misusing their privileges to access confidential data. Even if an attempt to avoid this can be made through a Service Level Agreement (SLA), the organization cannot verify if the SLA terms were fully respected. Other vulnerabilities that are related to this risk include poor patch management, the impossibility to process data in an encrypted form or to manage security policies. This risk is directly associated to keeping private data in a public cloud, but can also be related to the users with high-privileges that administer the private cloud if specialized log tools are not applied.

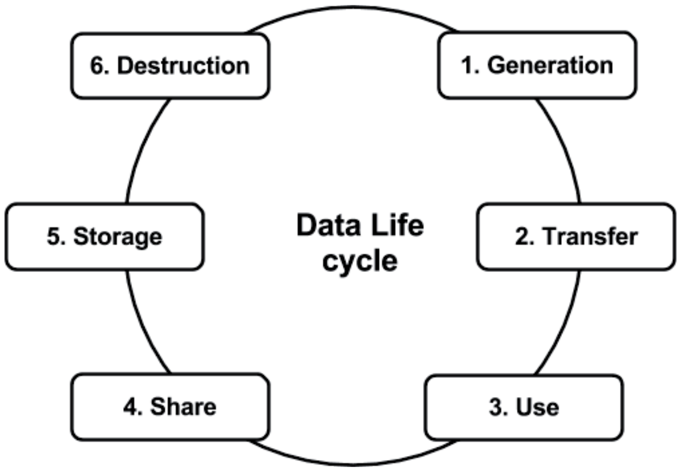


Fig.1 Data lifecycle in a hybrid cloud model

The second phase is data transfer. Within the enterprise boundaries data transmissions do not usually require encryption. Sometimes, a simple data encryption may be used in order to secure data flows within the companies' intranet, which has a certain security level. Yet for data transmission across enterprise boundaries, both data confidentiality and integrity should be ensured in order to prevent data from being tampered and tapped with by unauthorized users. In other words, only data encryption will not be enough. The integrity of data must also be taken into account. Therefore, the transport protocols must be able to provide both confidentiality and integrity. There is a problem with the transport protocols. For transmissions between private enterprise and cloud providers, transport protocols can be controlled by the private enterprise. Yet if cloud providers belong to the cloud federation, which allows all joint cloud providers to share resource with each other, the private enterprise will not be able to control transfer protocols among them which may lead to data leak.

The third phase relates to the actual process of data use. It is not feasible and user friendly if data stored in the cloud or the local environment is encrypted, due to later indexing or query problems. By focusing on usability, there is a loss from the security point of view due to the fact that unencrypted data is highly vulnerable. Thus unencrypted data in the process is a moderate threat to data security. Furthermore, the problem of isolation failure arises. Isolation failure occurs in systems where there are multiple users and shared resource such as network, computing resources of memory. Even if the user does not have access to others' data, the consequences of an SQL injection attack would include data to be exposed to third parties. The chances of this scenario to happen are low in a private cloud and higher in the public one. Taking into account that an organization that uses a hybrid cloud has less human and financial resources dedicated to security of the private cloud, as opposed to the public cloud provider, we want to raise the impact level to high for the private cloud. The assets that can be affected are similar to the loss of governance risk, and include the company's reputation and trust, personal or sensitive data and service delivery.

Data sharing phase expands the use range of data, which renders data permission more complex. The data owners can authorize the access to one party, who may further share data with other parties without the consent of the data owners. Therefore, during data sharing, data owners need to consider whether the third party continues to maintain the original protection measures and usage restriction. This also implies checking sharing granularity and data transformation. The sharing granularity depends on the sharing policy and the granularity of content. The data transformation refers to isolating sensitive information from the original data. These operations make the data not relevant with the data owners.

The data storing phase contains three information security aspects to be considered: confidentiality, integrity and availability. These can be directly linked with loss of governance and compliance checking, because of the direct impact. Loss of governance affects organizational assets such as company reputation, customer trust, personal sensitive or critical data (according to European Data Protection Directive 95/46/EC), and service delivery. Therefore the organization may not be able to have the capacity to reach its goals, and even impossibility to comply with the needed security requirements. When using cloud infrastructures, the client gives permission to the provider to manage a number of issues that affect privacy and security. Moreover, client's techniques of securing data may not be compatible with the provided cloud service, even if stating this aspect in the SLA. This creates a big gap in the security, and can also lead to compliance challenges. Compliance checking affects the most important asset that makes the cloud environment secure - certification. Many organizations strive to be certified in information security in order to gain customer trust. But this might be lost when migrating data to the cloud. Fortunately, by applying a hybrid cloud the above mentioned risks can be minimized. By keeping all important data in the private cloud, as well as services that must be available close to 100% of time, the organization can assure that will not lose acquired certification so far, and that will have a strong image and trust on the market. Another problem which may affect data is that cloud providers may transfer data to data centers which locate in another country. This means data is not kept within national boundary [2] anymore and there is a probability that the data may be used illegal by the government of the country where data centers are located in [3]. For example, to implement a model of public cloud in E-government would be doubtful, due to legislative constraint that data will not stay inside countries boundaries. Therefore, private cloud is desirable when dealing with personal data.

The last phase of data in its lifecycle is data destruction. In traditional IT environment, when data is no longer required, the organizations can securely delete it by using various kinds of secure mechanisms in order to completely destroy it from the storage medium. Deleting sensitive data from a private cloud is easier, as the organization can decide if the disk has to be destroyed in order to permanently delete data, or by using special algorithms. But for public cloud providers it is not cost effective at all, as an ineffective deletion of data could lead to data disclosure, and therefore not comply with sensitive data sanitization. Also, each time there would be resources reallocated, the cloud provider has to make sure that no data can be available beyond the specified lifetime.

4.2 Service lifecycle

Possible security problems that may affect services will be identified by using the service life cycle representation provided below (shown in Fig. 2). Security policies for the cloud include data deletion procedures. Unlike the data lifecycle, the service one has less security issues. There are only three notable problems that relate to cloud services: leaking of business logic, unauthorized access to the services and data leak when accessing cloud services.

The first problem is business logic leaking. This is only applied to the services that are specifically developed for the organization, and usually these services are web applications. There are two possible scenarios that lead to business logic leaking. The first one is that cloud provider receives access to the deployed services of customers and could steal important components which can be decoded by e.g. reverse engineering. The main reason would be information theft, meant for third parties such as competitors or organizations that use same services from the cloud provider. The second scenario is that the cloud provider's protection mechanisms are not good enough and hackers can get access to their systems.

The second problem is that cloud services could be accessed by unauthorized users. As hybrid cloud extends the IT perimeter outside the organizational boundaries, the surface for possible attacks becomes larger and a section of the hybrid cloud infrastructure could be already under the control of the service provider. One possible scenario is that the protection mechanisms are not strong enough, thus unauthorized users can bypass them and be able to gain accesses to the affected services.

The last problem is data leak when accessing cloud services. This scenario happens when data and services are hosted by different cloud providers, and services need to access certain data from outside the cloud's perimeter. By providing those services the ability to access data from other cloud providers means giving indirect data access to other cloud providers. The data leak problem also can happen when data is transferred between cloud providers who host data and cloud providers who host services. Private enterprises as customers have no or limited rights in order to manage the transfer protocols between them.

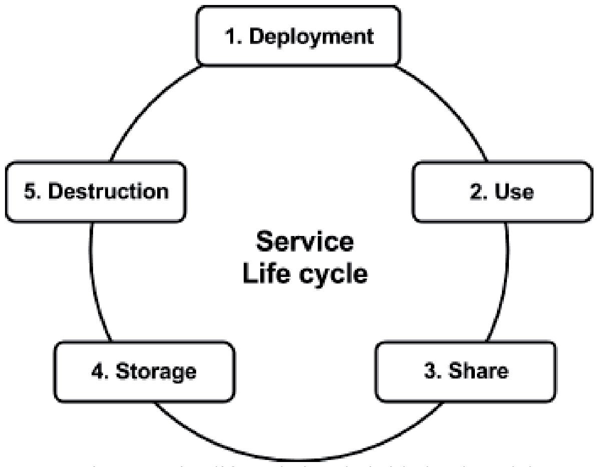


Fig.2 Service lifecycle in a hybrid cloud model

4.3 Legal security issues

Legal security issues are actual in the European Union as the legislation is restrictive and outdated. At present, the legal acts are being reviewed in order to comply with the economic development goals related to cloud computing. It is not clear yet which law is applicable, who is the controller and processor and if data will be stored among cloud providers outside EU. When using hybrid or public cloud in Europe, data protection, licensing or change of jurisdiction risks can appear. It is not possible for the client, which is the controller, to check if its data is stored as agreed, and it is not being changed under different jurisdiction. This is creating a lot of uncertainties in federated clouds, where data can be divided and replicated on multiple clouds from different countries. In the case of a security breach, it is not sure if the client will be notified of the possible risks its data has been exposed to. A jungle of standards [5] is another issue that creates uncertainty in using cloud in Europe. It is unclear which standard is better, and if it allows interoperability and the fulfilling of contract clauses defined by the client. This needs to be changed, as 44% from private sector respondents, stated that data security remains the biggest concern, but certification would help, according to a study done in 2012 by KPMG International [6].

5 PRESENT EXISTING SOLUTIONS

In this section we present existing solutions for protecting data and services that are currently used by private enterprises that already migrated to hybrid cloud model. We mainly focus on the security mechanisms that are public which private enterprises can apply by themselves. The security mechanisms of cloud providers are out of our research scope dues to the limited of detail information about security mechanisms that are used due to confidentiality reasons. The security solutions that are used to protect cloud deployed data and services will be presented from the technical point of view. Solutions focusing on data lifecycle will be named DSol, whereas solutions for the service lifecycle will be named SSol.

Firstly, a list of available solutions for protecting deployed data is presented. The proposed solutions mainly focus on protecting the integrity and the security of data while availability of data is put at lower priority.

DSol 1: The first solution is to encrypt the data before transferring it to the cloud providers' infrastructures. By using encryption methods, organizations can put their data on the cloud providers' infrastructures without worrying about unauthorized accesses to their data. Furthermore, encrypting data before placing it in a cloud may be even more secure than unencrypted data in a local data center. This approach was successfully applied by TC3, a healthcare company with access to sensitive patient records and health care claims, when moving their HIPAA compliant application to AWS [7].

DSol 2: The second solution is to apply user authentication and authorization methods. Authentication is used in order to identify whether the current users have the rights to access to the system or not. Authorization is used to check the current users' permission when they access certain data. By applying these mechanisms, the data's owners can control and limit the accesses to their data. There are five authentication methodologies that are widely used in the reality. These are hybrid solution [8], elliptic curves cryptography [9], public key cryptography with matrices [10], fully homomorphism encryption [11] and attribute based cryptography [12]. The hybrid solution methodology is highly recommended because of its safeness. For authorization, it depends on the architecture of storage system. There are two main types of architecture for authorization. The first one is Access Control List. It supports decentralized access control management by allowing a list of access permissions attached directly to an object). The other one is Access Control Matrix which supports centralized access control management. All objects' names are listed in a table along with their access permissions.

DSol 3: The third solution is by using hash function [13] to protect stored data in the cloud [14]. This solution mostly focuses on data integrity in the cloud by keeping a short hash of uploaded data in the local storage. Whenever the data is used, the hash function will re-calculate the hash string of that data. If new hash string matches the one which stored in the local storage, the integrity of data is guaranteed. In the case of large amounts of data hash tree is the solution [15]. Moreover, numerous storage system prototypes have implemented hash tree functions, such as SiRiUS [16] and TDB [17]. Mykletun et al. [18] and Papamanthou et al. [19] claim that this is an active area in research on cryptographic methods for ensuring the stored data integrity.

DSol 4: The fourth solution is to use the Depsky system [20]. In the DepSky system data is replicated in more than one commercial storage clouds such as Amazon S3, Windows Azure, Nirvanix and Rackspace, and is not relied on a single cloud. This helps to avoid the problem of the dominant cloud which causes the vendor lock-in issue [21]. From the security perspective, spreading data among different cloud providers decreases the chance that data is leaked out. This also helps to prevent data from unauthorized access by the cloud provider and to avoid data to be stolen by hackers when the single security system of the cloud provider is compromised. In addition, storing a fraction of data in each cloud in the DepSky system is achieved by the use of erasure codes because it can ensure that fractions can be combined with error free later. Consequently, exchanging data between one provider to another will result in a smaller cost.

DSol 5: The final solution is to encrypt the exchanged data among organization-cloud providers-end users [22]. This solution requires the involvement of the cloud providers, who already support the mechanisms for a secure data exchange. Options that support these solutions are the usage of cryptographic algorithms or functions such as RSA, AES and SHA. Applying them or not is now the decision of the organizations.

Next, the security solutions for protecting deployed services on cloud are presented. Unlike data, there is less security issues for on cloud services.

SSol 1: The first solution is a type of virtual machine level security which focuses on protecting the business logic of the applications/services, and securing the memory that is used by the applications/services. By applying this solution, the organizations will require to have their private virtual machine cloud instead of directly deploying applications/services to the public cloud. Cloud service provider, e.g. Amazon VPC, will virtualize the cloud in an isolated section of their infrastructure. The notable benefit from this solution is the improvement in security key driver due to data isolation, and performance because it does not share data bus and processor time with other customers.

SSol 2: The second solution aims to protect the exchanged data among the organization, the cloud providers who host the applications/services, and the cloud providers who host data. The main point of this solution is to build applications/services that are deployed in the cloud in a way that they can process the encrypted data as inputs. This solution ensures that the data which is transferred to cloud services will not be leaked out during the transmission and the cloud providers who host those applications and services cannot access the received data.

6 EVALUATION

In this section, the proposed solutions from above section will be evaluated against the defined key drivers in the third section. This aims to give an overview about the effectiveness of the existing solutions and how important key drivers are trade off if the organizations decide to use them. The mark will be given from -2 to +2 as illustrated in the below table (Table 1). The mark is given by basing on the observation of the authors on the solutions. For more objective results, benchmarking tools are required.

Table 1. Range of marks and their description

Value	Definition
-2	Severe negative impact
-1	Small negative impact
0	Non-affected or neutral
+1	Small positive impact
+2	High positive impact

Firstly, the security solutions for protecting data will be evaluated against key drivers. Table 2 contains details in form of marks to each key driver based on the data security solution.

Table 2. Evaluation of data security solutions

Solution	Performance	Security	Extensibility	Availability
DSol1	-1	+2	0	0
DSol2	0	+1	0	0
DSol3	-1	+1	0	0
DSol4	+1	+2	+1	-1
DSol5	-1	+1	0	0

For the DSol 1, the performance key driver is slightly affected because each time data is used, it needs to be decrypted. For large size data, decryption will be a time consuming task. On the other hand, the security key driver is enhanced significantly because the data is protected from transmission phase to storage phase. Data can be leaked in these two phases, and encryption will cause difficulties in accessing the content by unauthorized users. Extensibility and availability stay neutral because encrypting and decrypting processes are executed locally within the organizations and do not depend on the cloud providers' system, thus extensibility and availability key drivers are unaffected.

For the DSol 2, the security key driver is slightly enhanced because it only provides a solution to manage and control the data access. The data itself is not protected. The performance, extensibility and availability stay at neutral. There was no overhead on data processing and transferring between the cloud's providers and the organization, thus performance remains unaffected. Also, there are no extra constraints on data storing therefore extensibility and availability key drivers stay the same.

For the DSol 3, the performance key driver is slightly decreased because each time data is used, it will have to go through the hash function in order to compare with the hash string stored locally. The security is slightly enhanced because it helps to guarantee the integrity of data by taking advantage of hash function. This solution adds overhead to the data receiving process within the organization and does not affect data which is stored in the public cloud, thus the extensibility and the availability stay as unaffected.

For the DSol 4, the performance key driver is slightly increased because data is split into smaller parts and stored in different clouds, thus the speed when retrieving data is fast. The security key driver is greatly enhanced because each cloud provider stores only a part of the data, which reduces the risks of theft. This data accessing constraint also applies to hackers. If hackers want to steal data, they need to know which cloud providers holds different parts of the data and attack all of them in order to get the full data which is almost impossible. The extensibility is slightly increased because there will be no limit to the storage space for the organization. This is like storing data in a big cloud federation. The only key driver that has a slightly negative impact is the availability. This solution implies that all cloud providers' systems are up and running whenever the data is needed. When one cloud provider has a problem which leads to downtime, the data will become unavailable during that downtime.

In DSol 5 the security is increased due to securing data transmission phase. This solution does not aim to protect data while is stored in the cloud providers' storage. The performance key driver is decreased because each time the data is transmitted, it will have to be processed by both senders (encrypting phase) and receivers (decrypting phase). This solution adds overhead to the data transmitting process between the cloud providers and the organization. It does not affect the data storing process thus the extensibility and the availability stay as neutral.

Secondly, the security solutions for protecting services will be evaluated against key drivers. The table below contains assigned marks to each key driver based on the solutions for services security (Table 3).

Table 3. Evaluation of services security solution

Solution	Performance	Security	Extensibility	Availability
SSol1	+1	+2	-1	0
SSol2	-1	+1	0	0

For the SSol 1, the performance is increased because organizations do not have to share the cloud provider's infrastructure with other customers. The security key driver is increased significantly because the applications/services are now protected by the virtual machine layers. Therefore, other applications/services from other users do not interfere between each other, because these are deployed separately on the virtual machine layer. The extensibility is decreased because the organizations will have to deploy their services/applications on cloud providers which support virtualization. The availability remains neutral because this solution does not impact the deployed services in cloud. The availability of deployed services is affected directly by the downtime from the cloud providers.

For SSol 2, the performance key driver is slightly decreased because the applications/services will have to decrypt the data before working. On the other hand, due to encryption during transmission phase the security key driver is slightly enhanced. This solution does not add any extra overhead to the extensibility and the availability of the deployed services because it only works with the data that is used to feed the services, not the services themselves.

7 Discussions

Proposed security solutions have both strengths and weaknesses. They may fit well with some organizations but not for all of them. The usability of the provided solutions mainly depends on the security requirements of each organization. Naturally, organizations will have different security concerns because of their different needs when they implement the hybrid cloud, thus looking for a provided solution that perfectly matches all the concerns is not an easy task. There are two possible ways for organizations to achieve their desired level of security. The first way is to independently build own solutions that can match the security requirements. The second option is to combine existing solutions in order to match their needs. In term of time and cost efficiency, the latter solution is better than the first one.

Many aspects of cloud computing are still in experimental stage, where it is not known the effect of usage or provisioning. Many of the challenges appear when using all cloud capabilities due to the large options of scalability of main resources. We can therefore discuss over technological and legal issues in terms of cloud security. Therefore, research in the technical area is one solution towards reaching the desired level of privacy and security. EU's research estimation to reach desired level of cloud computing use by 2020 shows that 5 to 7 years are needed in order to overcome technical issues, such as data handling, programming models or system and to reach the required level of maturity and progress [1]. On the other hand, legalistic issues need a continuous development and update, in order to make the clouds compliant with the constant changing legislation from EU countries. This is the solution to make clouds secure from the legal point of view, as many of the risks described in section 4 are related to the lack of a clear model in the legislation regarding jurisdiction over cloud data and distribution in other countries. There is a built-in tension between legal and technical availability data placement concerns. [4].

Another way to reach to the point when the clouds will have the desired level of privacy and security is through research. It is known that the European clouds are lagging behind from the American clouds, in terms of development and use. EU's research estimation to develop cloud computing use by 2020, and we can notice that solutions related to technical issues, such as data handling, programming models or system management need 5 to 7 years to reach the required level of maturity and progress [4]. Therefore, we agree to the solution proposed by the EU, and we believe that by improving the research and education area in this field, cloud computing will become a realistic model to be used by any entity.

Finally, the economic impact can be minimized by engaging both public authorities and small-medium enterprises in adopting cloud computing. This would lead legislation update in order to comply and meet requirements for all entities. Nethertheless, it is good to hear the three-cloud specific actions stated by the European Commission: cutting through the jungle of standards, fair contract terms and conditions, and establishing a European Cloud Partnership to drive innovation and growth from the public sector [4]. The last action is a challenge for the economy, as countries will invest a lot of resources in developing a secure cloud environment from the technical point of view in order to reach the proposed economic model, fact which would have a positive effect on the cloud use and security level.

8 Conclusion

Information security is mandatory in order to make use of all the features and resources cloud computing can offer. We looked at the structure of the hybrid cloud, and analyzed its constituents - the public and private cloud. From the security point of view, organizations that use the hybrid cloud can compensate the security gaps from the public cloud by using its private cloud for sensitive data. We explained the legal, technical and policy risks, and we conclude that all these risks are interconnected and need to the approached together.

The possible security risks in the cloud are defined and presented, based on data and services lifecycle in the cloud. Overall, the most important security concern is unauthorized access to deployed data and services in the cloud. Along with that, existing security solutions are introduced and verified for their strengths and weaknesses against important key drivers of hybrid cloud. In the discussion section, some future directions for hybrid cloud security are suggested, based on actual gaps in this area. The reason is that each organization has different requirements regarding security, thus one solution may not be able to fit their needs well.

In the end, public cloud computing can offer robust security solutions due to its size and resources owned by the providers, which can be later applied the organization to the private cloud, therefore switching to a well-known and independently managed private cloud . However, from legal and development aspects, research is the engine towards reaching the desired cloud computing usage. Trust, security and privacy are actual research fields, due to the attackers' interest in concentrated data in the cloud. These are mandatory prerequisites in this industry, in order to make the public cloud as secure as the private one.

References

[1] Lutz, S. "The future of cloud computing", Expert group report, 2010, viewed 15 March 2013, < cordis.europa.eu >.

[2] Armbrust, Fox et.al. "Above the cloud, A Berkeley view of cloud computing", UC Berkeley Reliable Adaptive Distributed Systems Laboratory, 2010, pp. 15.

[3] Satveer, Amanpreet, "The concept of cloud computing and issues regarding its security and privacy", International journal of engineering and research technology, 2012, vol 1, no 3, pp. 5

[4] ENISA, Cloud Computing Security Risk Assessment 2009 www.enisa.europa.eu/, viewed 15 March 2013

[5] European Commission, "Communication from the commission to the European Parliament, the council, the European economic, and social committee and the committee of the regions", Unleashing the Potential of Cloud Computing in Europe, 2012.

[6] John H, Ken C, "Exploring the cloud", A global study of Government adoption of cloud, 2012, viewed 10 March 2013, <http://www.kpmg.com/AU/en/IssuesAndInsights/ArticlesPublications/cloud-computing/ >

[7] Fadadu C, Shrikanth V, Trivedi H, "Cloud security using authentication and file base encryption", International journal of engineering and research technology, 2012, ISSN: 2278-0181,vol 1, no 10, pp. 3.

[8] Sunita R, Ambrish G, "Cloud security with encryption using hybrid algorithm and secured endpoints", International journal of computer science and information technologies, 2012, ISSN: 0975-9646, vol 3, no 3, pp. 4302.

[9] Veerraju G, Srilakshmi I, Satish M, "Data Security in Cloud Computing with Elliptic Curve Cryptography" International Journal of Soft Computing and Engineering, 2012, ISSN: 2231-2307, vol 2, no 3.

[10] Birendra G, Dr.S.N.Singh "Enhancing Security in Cloud computing using Public Key Cryptography with Matrices", International Journal of Engineering Research and Applications, 2012, ISSN: 2248-962, vol 2, no 4, pp.339-344.

[11] Maha T, Saïd E, Abdellatif E, "Homomorphic Encryption Applied to the Cloud Computing Security", Proceedings of the World Congress on Engineering, 2012, ISBN: 978-988-19251-3-8 ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online), vol 1.

[12] Shucheng Y, "Data Sharing on Untrusted Storage with Attribute-Based Encryption", Worcester Polytechnic Institute, 2010.

[13] R.C. Merkle, "Protocols for public key cryptosystems", IEEE Symposium on Security and Privacy, 1980, pp. 122-134.

[14] C. Cachin, I. Keidar and A. Shraer, "Trusting the cloud", ACM SIGACT News, 40, 2009, pp. 81-86.

[15] R.C. Merkle, "Protocols for public key cryptosystems", IEEE Symposium on Security and Privacy, 1980, pp. 122-134.

[16] E. . Goh, H. Shacham, N. Modadugu and D.Boneh, "SiRiUS: Securing remote untrusted storage",NDSS: Proc. Network and Distributed System Security Symposium, 2003, pp. 131–145.

[17] U. Maheshwari, R. Vingralek and W. Shapiro, "How to build a trusted database system on untrusted storage", Proc. 4th Conf. on Symposium on Operating System Design & Implementation, 2000, p. 10.

[18] E. Mykletun, M. Narasimha and G. Tsudik, "Authentication and integrity in outsourced databases", ACM Transactions on Storage (TOS), 2,2006, pp. 107-138.

[19] C. Papamanthou, R. Tamassia and N.Triandopoulos, "Authenticated hash tables", CCS'08: Proc. 15th ACM Conf. on Computer and communications security, 2008, pp. 437-448.

[20] Allyson B, Miguel C, Bruno Q, Fernando A, Paulo S, "Depsky: Dependable and secure storage in cloud of clouds", University of Lisbon, 2011.

[21] H. Abu-Libdeh, L. Princehouse and H.Weatherspoon, "RACS: a case for cloud storage diversity", SoCC'10:Proc. 1st ACM symposium on Cloud computing, 2010, pp. 229-240.

[22] Sudha M, Monica M, "Enhanced security framework to ensure data security in cloud computing using cryptography", Advances in computer science and its application, 2012, vol 1, no 1, pp. 34-35.

Information Security in Service-oriented Smart Grids

Ruurtjan Pul and Brian Setz

Abstract—Smart grids are the electrical grid of the future. They use information and communications technology in order to improve the efficiency, reliability, economics, and sustainability of the production, distribution and consumption of electricity. Smart grids allows a bidirectional flow of energy, this means customers can not only consume energy but also produce energy, for example by use of renewable energy sources such as solar panels or small wind turbines.
In this paper, two smart grid architectures are analyzed and compared in order to uncover differences and similarities between them. The two architectures that are analyzed are the Integration and Energy Management system supported by the NOBEL project and the smart grid architecture supported by the SmartE project.
Recent research indicates that one of the challenges of the smart grid is to secure privacy sensitive data. The research that has been done in this paper identifies privacy sensitive data and investigates strategies and techniques which can be used to increase the information security of this data. These strategies and techniques are evaluated based on their usefulness and applicability in service-oriented smart grid architectures.

Index Terms—Smart grid, service-oriented, architecture, information security, privacy.

1 Introduction

The current electrical grid is a centralized system in which electricity is distributed from a central place. Energy supply companies distribute the energy to their customers using their local distribution network. Transmission grids are used to transport electricity to this local distribution network. They are needed, because vast distances have to be covered in order to reach the customer. An overview of the current electrical grid can be seen in figure 1.

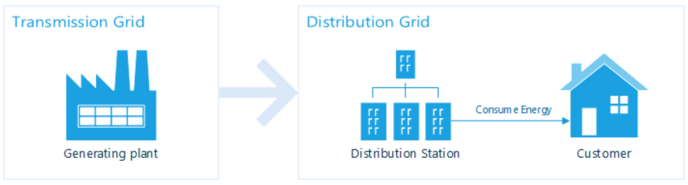


Fig. 1: Current electrical grid

However, this system is changing rapidly due to the fact that the number of distributed renewable energy generators (e.g. solar panels, wind turbines) is increasing, which results in a bidirectional flow of electricity. The current electrical grid does not support these bidirectional flows at the customers end, therefore it has to undergo changes. Furthermore, smart grids are more efficient, for example when dealing with peak loads and distributing the energy accordingly. There is a European policy [5] in place which promotes these changes. It aims for the modernization of the current electrical grid. The future electrical grid is heading towards a decentralized and dynamic architecture, also known as a smart grid.

The smart grid will enable its customers to not only consume energy, but also produce energy. This produced energy can be supplied to the smart grid. For example; when customers produce more energy than they consume, they can supply this excess energy to the grid. In order to fulfill their energy needs in the smart grid, customers shall interact with other customers, as well as the energy supply company. This situation is depicted in figure 2.

In order to support a smart grid environment, an Information and Communications Technology (ICT) architecture has to be in place to control and interact with this future electrical grid. This ICT architecture will have to deal with issues that are part of smart grids. These issues include voltage and frequency changes due to electricity being generated locally and energy security issues because of the bidirectional flow of energy. It also includes increasingly difficult supply & demand prediction, since the production of energy by renewable energy sources is inherently unstable.

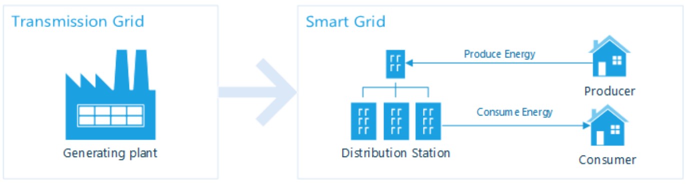


Fig. 2: Smart electrical grid

A smart grid also solves issues that play a role in the current electricity grid. The smart grid shall be able to deal with peak loads more effectively, due to its decentralized nature. It is able to distribute the energy more efficiently across the electricity grid, since it is more aware of the situation on the local power grid due to its monitoring capabilities. Furthermore, it prevents overloading of the grid by means of efficient distribution and decentralization.

The ICT architecture also has to enable customers to interact with the smart grid and its related entities. Energy has to be bought and sold, demands have to be predicted, energy loads have to be balanced, and the smart grid has to be monitored. At the same time customers also want insight in their electricity usage and production. This means services will have to be in place to allow these interactions. A commonly proposed approach is to develop a service-oriented architecture to deal with these issues.

In this paper, we will compare two existing service-oriented smart grid architectures in order to answer the question: what are the differences and similarities between the two existing service-oriented smart grid architectures? Furthermore, the paper covers the topic of information security in smart grids. By looking at the state of information security in smart grids we try to answer the question: what data in smart grids is privacy sensitive and how can this data be protected in order to maintain the privacy?

This paper is organized as follows: in section 2 service-oriented smart grid architectures are explained, followed by a comparison of two existing architectures in section 3. Section 4 will discuss security and privacy issues with regards to smart grids and techniques to counter these issues. The conclusion of the paper can be found in sec-

- *Ruurtjan Pul is a MSc. Computing Science student at the University of Groningen, E-mail: r.pul@st.rug.nl.*
- *Brian Setz is a MSc. Computing Science student at the University of Groningen, E-mail: b.setz@st.rug.nl.*

tion 5. The final section, section 6, covers further work that could be done in the future with regards to this field of research.

2 Service-oriented smart grid architectures

A service-oriented architecture is a type of software architecture that focuses on developing inter-operable services that can be reused throughout the software. This results in loosely coupled services, usually made accessible over a network to allow other parties to interact with the services and build their own applications around them. Applications reusing and combining existing services are also known as mash-up applications.

Web services are a popular choice for networked services. Figure 3 displays the architecture which is often implemented by web services. Service providers register themselves with the service broker. The service consumer can search for services using the service broker. Once the service has been found the consumer can invoke the service directly.

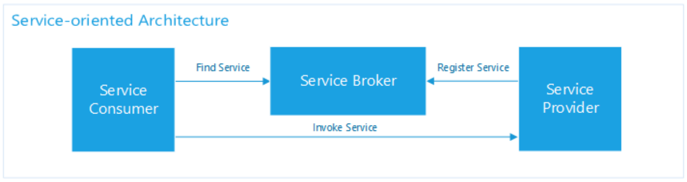


Fig. 3: Service-oriented Architecture

The service-oriented architecture approach seems like a promising architecture to be used in combination with the smart grid. The hardware in the smart grid can be abstracted by a generic interface, which can be published to the outside world by using services. This results in an information-driven system, in which information is exchanged between services and their consumers. Examples of services are billing services, market services which allow the buying and selling of energy, and monitoring services which monitor the energy consumption.

Smart grids are dynamic environments, therefore the system needs to be able to adapt. Using a service-oriented architecture will allow for a pluggable system where services can register and unregisters themselves at any time. This makes it easy to add and remove functionality to and from the service.

A possible scenario is that end-users connect their smart devices to the system, turning their smart devices into services which can be interacted with. These services are volatile; an end-user can connect and disconnect these devices at any moment in time. The service-oriented smart grid makes this possible by providing a pluggable and dynamic environment.

The entities in the system make use of the smart grid services. Together these services form the middle-ware of the system. All smart grid information is accessed via the middleware. Different entities make use of different services. Entities that are part of the smart grid are displayed in figure 4.

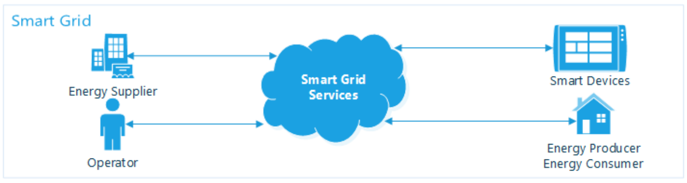


Fig. 4: Entities in the smart grid

The energy supplier can use services such as billing services to bill its customers. The operator of the network can monitor the network usage and, for example, decide to increase the capacity of the network based on data provided by services. Smart devices such as smart meters can communicate with services to optimize energy consumption profiles and reduce costs for the end user. The energy producer and energy consumers can use the services to broker energy, purchasing additional energy or selling excess energy.

3 Comparison of existing architectures

Research has already been done to verify the viability of this service-oriented architecture; the rest of this section will discuss recent research as well as their findings. In particular, the Integration and Energy Management system supported by the NOBEL project and the smart grid architecture supported by the SmartE project will be discussed.

These two smart grid projects were chosen because both architectures are based upon the service-oriented approach, yet vary in their implementation details as we will see in the rest of this section. Also, both projects have been implemented and tested in a real environment, it is interesting to look at the the results of these projects and compare the outcome in order to see how one project may benefit from the other.

3.1 SmartE

The first project we will discuss is the SmartE project. This project was started on april 1st 2010 and ended on march 31st 2012 [6] and was supported by the Flemish Government [11, 12]. The main aim of this project was to design a service-oriented architecture and implement a proof-of-concept of it. This was done to evaluate the two main research questions of the SmartE project:

- How much will the peak load reduce by performing load shifting?
- Will end users change their energy consumption patterns based on the information provided by the new system?

Before these questions can be answered however, an architecture needs to be designed. As noted before, this architecture that was designed by the researchers of the SmartE project is a service-oriented architecture.

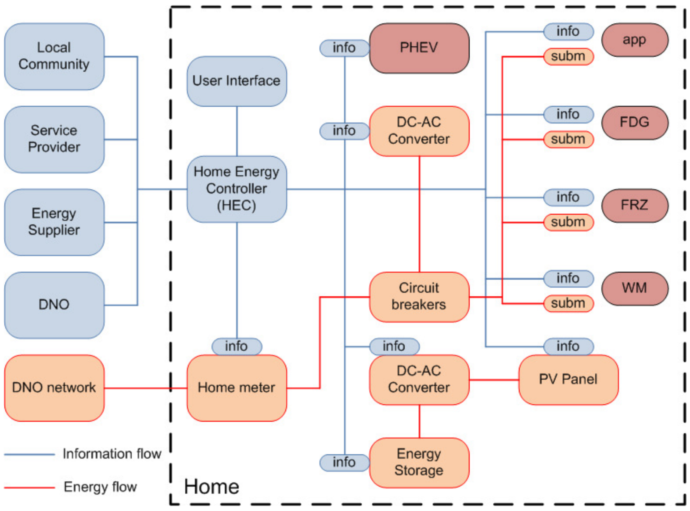
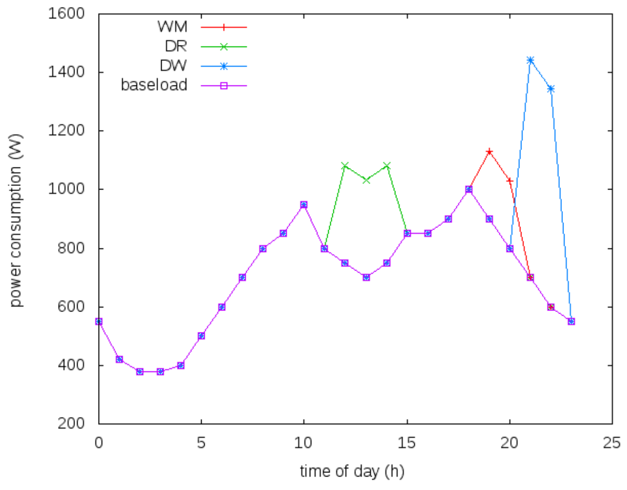


Fig. 5: Information- and energy flow within the SmartE architecture [11]

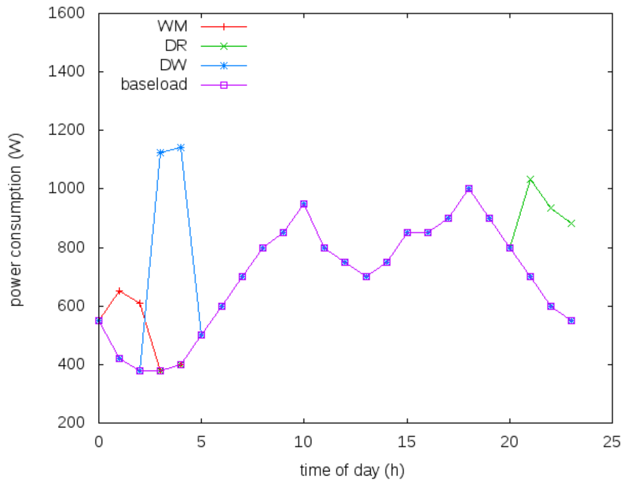
Figure 5 shows a detailed overview of the information- and energy flow within an example configuration of the SmartE architecture. The Home Energy Controller (HEC) is at the heart of the system. User interfaces connect to it in order to gather current information about energy consumption. The HEC communicate directly with all the appliances to gather information about energy consumption. It is also able to interact with devices that externally publish an interface. For example, some devices are directly controllable by the HEC, allowing it to switch them on and off at any time. This is required for automatic load shifting. Appliances include, but are not limited to: FDG (fridge), FRZ (freezer), WM (washing machine), PHEV (plug-in hybrid electric vehicle). Each of them consumes energy and provides information about it to the HEC. The home meter is also connected for overall energy usage information.

The architecture allows for external services, some are technical services that are required for the stability and security of the grid, others are commercial services that offer additional functionality to end users. These services communicate with the HEC and are depicted on the left hand side of the figure.

To evaluate the effectiveness of this architecture in different scenarios, the researchers designed a simulation. This simulation can also be used to evaluate how the architecture will react with regards to various smart grid control algorithms, failures and scalability. The simulation implemented by the researchers used OMNeT++ [3], an open extensible, modular, component-based C++ simulation framework and library. This gave the researchers the flexibility to interchange components and algorithms in order to find an optimal configuration. The built-in scripting language enabled the simple simulation of multiple scenarios. The configuration of the simulation, like the smart grid, consisted of two main components: the power grid and the ICT grid. The first transporting energy; the latter information. This way various fault scenarios, like incorrect energy consumption measurements, can be simulated.



(a) Peak load with no intelligent coordinator



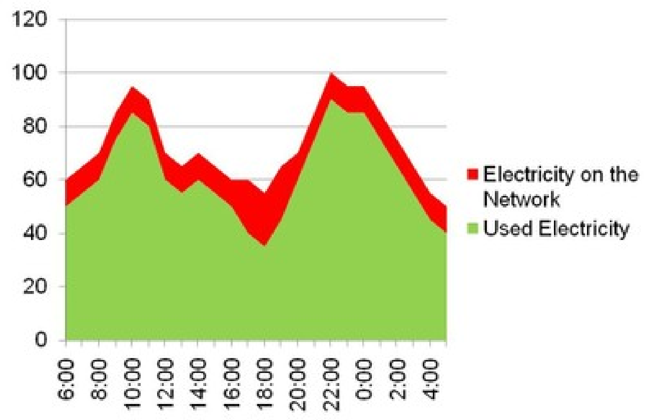
(b) Peak load with an intelligent coordinator

Fig. 6: Comparison of peak loads [12] (SmartE)

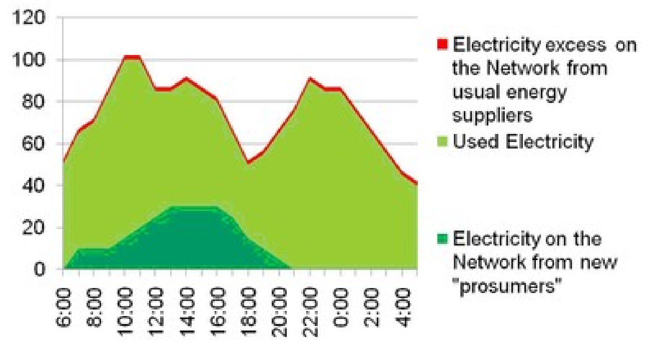
Figure 6 shows the results of this simulation as published by Tom Verschueren et al. [12] It shows that automatic scheduling of the washing machine (WM), dryer (DR) and dishwasher (DW) can reduce peak load of a single household by 22%. An intelligent coordinator scheduled the devices in such a manner that the most energy consuming devices would be active when the baseload is the lowest. Combining areas or neighbourhoods could increase this reduction for an even lower peak load. We can conclude that the research question “how much will the peak load reduce by performing load shifting?” can be answered with “at least 22%”.

In order to answer the second research question, a field test was required. A field test with 21 participants was run for about 5 months. The field test included a proof-of-concept of the architecture with an Android application for end users which featured historical data about energy usage per logical group (e.g. “living room”) or device. It also showed an overview of the overall consumption as well as the price of energy at every point in time.

Questionnaires and reviews were taken in order to measure the impact the system had on the participants. Participants indicated that they were influenced by the information provided to them by the application. Most participants indicated that they compared their usage to previous days. Some were considering replacing appliances that consume a lot of energy or taking tariff information into account and waiting with turning on the dishwasher and washing machine in order to save money. This clearly shows that the second research question “Will end users change their energy consumption patterns based on the information provided by the new system?” can be answered with “yes”.



(a) Current best-case scenario



(b) NOBEL scenario

Fig. 7: Energy used and energy spent [2] (NOBEL)

3.2 NOBEL

The second project that we will discuss is the Integration and Energy Management (IEM) system [8, 9] as supported by the NOBEL interest group which is co-funded by the European Commission. The project was started in 2010. Its mission is to develop, integrate and validate ICT enabling a reduction of the currently spent energy, by providing a more efficient distributed monitoring and control system for distribution system operators (DSOs) and prosumers [2].

NOBEL focuses on designing a new Neighbourhood Oriented Energy Monitoring and Control System which allows operators to improve the distribution energy by allowing bi-directional information and energy flows by turning customers into sources of both energy and information. This will allow for monitoring and control processes that were previously not possible because detailed customers behaviour was not available.

The efficiency improvement that NOBEL wants to achieve is shown in figure 7. Figure 7a shows the present day scenario. Not all energy that is available on the network is used; this results in waste (red area in the graphs). Figure 7b shows the scenario that NOBEL wants to achieve; less energy waste by means of more efficient distribution and monitoring, as well as allowing customers to produce energy and deliver this energy to the network.

To achieve its mission, the NOBEL project developed and designed the IEM. The IEM system provides the service-oriented smart grid architecture for the NOBEL project. This system is depicted in figure 8. There are multiple layers visible in the architecture of the system. First, there is the device layer which consists of the smart devices. The public services layer contains the public services that can be used by smart devices and applications. There is also an enterprise layer, containing services for the enterprise users such as energy supply companies. The applications are present in the application layer. They communicate with the public services.

There are two commonly used methods of communicating between components: Representational State Transfer (REST) and Simple Object Access Protocol (SOAP). The NOBEL project uses REST to communicate between services, devices and applications. REST is more lightweight and allows for rapid application integration, more flexibility and provides a higher performance.

When it comes to the implementation details, the NOBEL project uses Java REST services which use mySQL for data storage purposes, deployed on a Glassfish Application Server. Communication with the IEM is done via encrypted channels using HTTPS in combination with a security framework based on Apache Shiro.

Currently, all IEM services as seen in figure 8 have been designed and implemented. The system has been demonstrated and shows potential, enabling developers to build applications on an abstract level by use of these services.

The NOBEL project recognizes that security, trust and privacy are important issues in smart grids. Yet they did not focus on the security aspect of their system, further research is needed with regards to this subject.

3.3 Differences and similarities

When we compare the two projects described in section 3.1 and 3.2, it is clear that the aim of both projects is the same: reducing global energy production and supporting small scale renewable energy sources. In order to do so, a new information infrastructure was added in addition to the existing energy grid, creating a smart grid. Both projects designed this infrastructure in a service-oriented manner that required energy consumption information to minimize energy production. In both cases, this information was gathered from smart devices. Services and applications were designed in order to achieve the aim of both projects. Some inform end users about their energy usage, while others lower peak loads by automatically turning appliances on and off based on energy consumption information.

Despite all these similarities, the two projects differentiate in their focuses. The SmartE project is focused around the HEC, located in every house. It functions as a gatekeeper, gathering information from smart devices and using external services. The smart grids intelligence lies not primarily in the services, but in the HEC. It informs users about their energy consumption patterns and intelligently switches devices on and off. The NOBEL project however, does not focus on the Information Concentrator, which is the equivalent of SmartEs HEC. Instead, NOBEL focuses on services aimed to optimize neighbourhoods. It uses neighbourhood wide services that are able to coordinate energy consumption patterns. For example, charging the PHEVs of a neighbourhood in a controlled manner will decrease the peak load of this neighbourhood.

Though there are fundamental differences in the focuses of the two projects, the core ideas are combinable. It is possible to take the HEC as the basis of the architecture and publish a public service that balances energy consumption, which rewards users that participate in it.

4 Privacy in smart grids

Research done for the NOBEL project indicates that: “security, trust and privacy are challenging issues, especially associated with the emerging smart grid capabilities”. The SmartE project also recognizes this issue: “energy measurements are very sensitive information. They reveal a lot of the behavior of the inhabitants and could violate their privacy if this data would become publicly available”. When we look at the security techniques used in the existing service-oriented smart grid architectures it becomes clear that security has not received much attention during the design phase of the architecture.

Work done by A. Bekbatyrova and S. B. Signorjnsdttir, “Cyber Security in the Smart Grid” [7], shows possible solutions to enhance security in smart grids. However, the focus of their research lies on cyber security and not specifically on securing privacy issues that service-oriented smart grid architectures introduce.

The use of services implies that there is data being transferred to and from services. This data transferring can be either on a local area network, or over the Internet if the services are hosted in the cloud. In the case of service-oriented smart grid architectures, this data can be privacy sensitive data.

Smart devices will make use of the services provided by service-oriented smart grid architectures. This means smart devices will transmit data to these services in order to make use of them. The privacy sensitive data that is being transmitted is data that belongs to the owner of the device and should not be accessible by everyone.

If the data generated by smart devices is unprotected then the privacy of the user is compromised, because the user’s data will be accessible by anyone. This means anyone would be able to, for example, view the user’s power consumption, types of smart devices used in the user’s home and any other type of information that the smart devices may send to the smart grid services. Furthermore, with some analysis, attackers could potentially know if someone is at home, allowing them to physically break into a house at a time no one is home.

The data also reveals every activity that requires electricity, since electrical consumption is tracked. In short, attackers could know when and how much television is watched, how many times clothes are washed and at what time the inhabitants go to bed. This goes to show that smart grid data is highly privacy sensitive data.

In order the secure this privacy sensitive data we have researched four different security techniques: security frameworks, public-key infrastructures, whitelisting and anonymization. By looking at these techniques and how they can be applied within smart grid architectures, we try to answer the question of how privacy sensitive data can be protected in service-oriented smart grid architectures.

4.1 Security frameworks

Security framework is a general term for frameworks which provide authentication, authorization, cryptography, and session management. These frameworks are often used for their authentication and authorization functionalities to make sure that the entity requesting or sending data has the right to do so and is in fact authenticated. An example of a security framework is Apache Shiro [1], used by the NOBEL project.

The security framework acts as a middleware between the service interface which is exposed and the actual implementation of the service, also known as the service provider. This is shown in figure 9. When a service is called, the request will have to pass through the middleware, the security framework in this case, before it is delivered to the service provider. The security framework decides whether or not the request is valid. For example, it can check if the entity that sent the

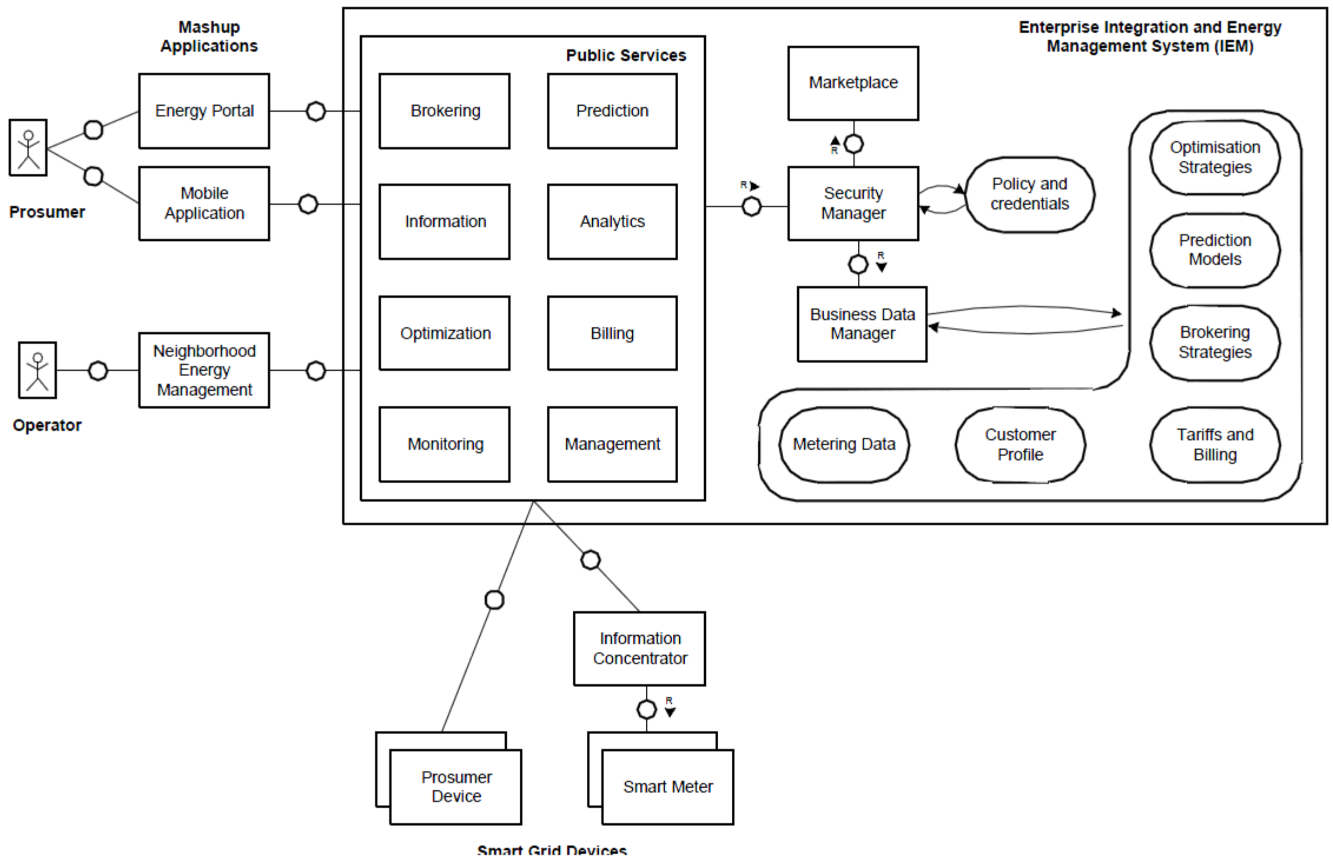


Fig. 8: The IEM system and architecture [9]

request if authenticated with the server and authorized to perform this request on the service provider.

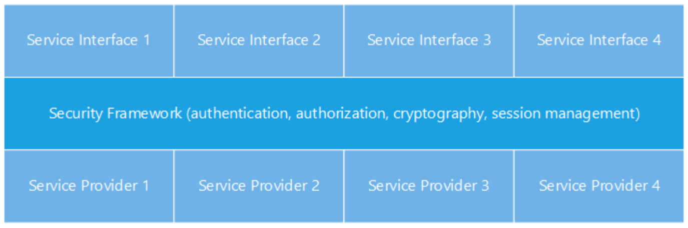


Fig. 9: Security framework as middleware in service-oriented architectures

An example scenario would be a customer requesting his billing information from the energy provider by contacting the billing service interface. The customer would first have to supply valid credentials before he is authenticated. If the credentials are indeed valid, a session will be created for this customer, giving him access to certain services. The customer has to be authorized to use certain services. If the customer is both authenticated and authorized, then the request will be passed to the service provider, in this case the billing service provider.

Security frameworks increase the information security of service-oriented smart grids, by making sure that an entity requesting information from a certain service is indeed allowed to access this data by enforcing authentication.

4.2 Public-key infrastructure

Public-key infrastructure (PKI) is an infrastructure used for creating, managing, distributing, using, storing, and revoking digital certificates [4]. PKI is often incorporated in smart grid architectures in the form of protocols: Transport Layer Security or its predecessor, Secured Socket Layers. Both protocols are used to provide secure communication over the Internet by means of encryption. A. Metke et. al. [10] have proposed the following additions on the PKI specific to smart grids:

- Smart Grid PKI standards
- automated trust anchor security
- certificate attributes
- Smart Grid PKI tools

An in-depth explanation of each of these additions can be read in ”Security technology for smart grid networks” [10].

With a PKI, it is possible to encrypt all messages between services as well as verifying the source of the messages. This way, authenticity and integrity are guaranteed. However, this solves only part of the privacy problem: messages can no longer be intercepted, but malicious services can still receive privacy sensitive information. Other measures are required to be implemented to prevent this from happening.

An example of PKI in action is the following: when an end user requests a bill from a billing service, it is important that the end user can be certain of the authenticity of the message. After the end users application and the service have finished the handshake, the end user knows for sure that the message comes from the billing service and that it has not been tempered with.

4.3 Whitelisting

Privacy is subjective; what some consider privacy invading, others might not. In order to address this, the end user should have control over their own data. One of the methods to ensure this is whitelisting. Before a service or application can access any data, the end user first needs to approve this. The access control can be coarse grained, for example by giving an application access to data from all available smart devices. But it can also be fine grained, this would give an application access to data from a specific smart device or even a limited subset of this data depending on the configuration.

In addition, this method ensures that the end user decides what application developer he or she trusts. This greatly decreases the potential exploitation done by malicious applications. This trust based approach is already implemented by two large platforms: Android and Facebook. User acceptance of this approach is therefore likely to be high, since users are already familiar with the concept.

An example clarifies the concept: suppose an end user wants to use a new service for comparing his energy usage to the average. Upon installing the application, it will ask for permission for usage data from smart devices. If the user does not trus the application, the installation can be stopped before any data was accessed.

4.4 Anonymization

Anonymization can be used in order to increase the end users privacy by anonymising the data before sending it over the internet. To achieve this, any data that could possibly identify the user has to be removed before being transmitted over the Internet. If this sensitive data cannot be removed in any way, it should be encrypted before transmission. These actions will preserve the anonymity of the user to which this data belongs, even if the data is intercepted during transmission to or from a service.

For example, when collecting the neighbourhood statistics, data is retrieved from all the users in the neighbourhood. If this data would be sent unprocessed, it will include information that compromises the users privacy, such as information that could identify the smart devices used at the users home. Therefore, this information has to be removed before being collected or restructured in such a way that it can not be linked to a user, in order to preserve the privacy of the user.

5 Conclusion

In this paper, we have analyzed and compared two service-oriented smart grid projects; SmartE and NOBEL. One of the two research questions posed in the beginning of this paper is: what are the differences and similarities between the two existing service-oriented smart grid architectures? This question can now be answered: although SmartE and NOBEL are very similar architecturally, they differ in their focus. SmartE focuses on in-house solutions, while NOBEL focuses on neighbourhood centered services. By combining the core concepts of the two projects, the advantages of both projects can be utilized.

Looking back at service-oriented smart grid architectures, we think that they offer significant advantages over non service-oriented smart grid architectures due to the use of services which can be accessed by different entities. Due to the open nature of services we expect that application developers are going to take advantage of this, they may find interesting ways to process, combine and present data accessible via the smart grid services. However, in our opinion this openness and ease of access to services poses an important security issue.

The second research question stated in the beginning of the paper is: what data in smart grids is privacy sensitive and how can this data be protected in order to maintain the privacy? Our research shows that when left unchecked, the privacy of the user is easily violated by means of intercepting messages or by malicious applications which collect data. This data includes data generated by smart devices, as well as data with regards to energy consumption and production. This data can be protected by using the security techniques proposed in this paper: the anonymization of transmitted data and applying whitelisting to applications before granting them access to sensitive data.

By applying the techniques proposed by us, anonymization and whitelisting, in combination with the techniques that are already being used in smart grids, the privacy of the smart grid user should be at an acceptable level. In our opinion, the disadvantages that come with service-oriented smart grids do not compare to the large amount of opportunities that they offer. Therefore, we think the service-oriented approach is indeed the correct way for smart grids architectures to be designed.

6 Future Works

The performed research suggests a number of valuable follow-up efforts.

Though we are confident that we have identified the vulnerabilities that pose the highest privacy risks, this needs to be verified. A white hat hacker is required in order to try to infiltrate a service-oriented smart grid architecture. This might also identify unforeseen vulnerabilities.

Since this new approach to energy management is not incorporated in the current law, research needs to be done to the extent that the law supports this new architecture.

Furthermore, a social study is necessary to evaluate the support the new architecture can expect from citizens. Some might not value the advantages of smart grid over the risk of potential privacy violation.

Acknowledgements

The authors of this paper would like to thank Femke Kramer and Rein Smedinga for the guidance they provided in the weeks leading up to the 10th student colloquium of the University of Groningen. We would also like to thank the reviewers of this paper for providing us with their valuable feedback. In particular, we would like to thank Andrea Pagani for sharing his insights on this topic and proof reading our paper.

References

[1] Apache shiro — java security framework. `http://shiro.apache.org/`, 2013.

[2] Objectives - nobel. `http://web.ict-nobel.eu:91/project/objectives`, 2013.

[3] Omnet++. `http://www.omnetpp.org/`, 2013.

[4] Public-key infrastructure. `http://en.wikipedia.org/wiki/Public-key_infrastructure`, 2013.

[5] Smart grids european technology platform. `http://www.smartgrids.eu/`, 2013.

[6] Smarte - iminds. `http://www.iminds.be/nl/onderzoek/overzicht-projecten/p/detail/smarte`, 2013.

[7] A. Bekbatyrova and S. B. Signorjnsdttir. Cyber security in the smart grid. In *SC@RUG, 2012*, pages 17–22. RuG, 2012.

[8] S. Karnouskos. Future smart grid prosumer services. In *Innovative Smart Grid Technologies (ISGT Europe), 2011 2nd IEEE PES International Conference and Exhibition on*, pages 1–2. IEEE, 2011.

[9] S. Karnouskos, P. G. Da Silva, and D. Ilic. Energy services for the smart grid city. In *Digital Ecosystems Technologies (DEST), 2012 6th IEEE International Conference on*, pages 1–6. IEEE, 2012.

[10] A. R. Metke and R. L. Ekl. Security technology for smart grid networks. *Smart Grid, IEEE Transactions on*, 1(1):99–107, 2010.

[11] M. Strobbe, T. Verschueren, K. Mets, S. Melis, C. Develder, F. De Turck, T. Pollet, and S. Van de Veire. Design and evaluation of an architecture for future smart grid service provisioning. In *Network Operations and Management Symposium (NOMS), 2012 IEEE*, pages 1203–1206. IEEE, 2012.

[12] T. Verschueren, W. Haerick, K. Mets, C. Develder, F. De Turck, and T. Pollet. Architectures for smart end-user services in the power grid. In *Network Operations and Management Symposium Workshops (NOMS Wksps), 2010 IEEE/IFIP*, pages 316–322. IEEE, 2010.

Qualitative Comparison of 2D Digital Inpainting Techniques

Jelle Nauta & Sander Feringa

Abstract—Digital inpainting is a technique for filling in missing or damaged regions in images. Different methods have been developed for this, using a wide range of techniques. In this paper we compare three of such methods that focus on geometric smoothness: one that uses iterative smoothness-based propagation along isophotes based on Navier-Stokes fluid dynamics, one that uses connected filters, and one that uses a Fast Marching Method. To give a broader view of the field, we also explain one self-similarity-based technique, but since its application differs so much from that of geometric smoothness, it is not included in our comparison. The criteria on which our comparison is based are the domains of application, visual results, computational performance, simplicity and the ease of implementation. We conclude that among the geometric smoothness methods, there is a tradeoff between speed and visual quality. We observe that results are in general only good for very narrow inpainting regions, so finally we briefly discuss combinations of geometric and self-similarity methods.

Index Terms—image inpainting, image restoration, fast marching, Navier-Stokes, fluid dynamics, isophotes, geometric smoothness

1 Introduction

Image inpainting is the process of filling in missing information in images in such a way that the result looks genuine to a human observer. This is useful in restoration of damaged images (paintings, films) and for removing unwanted occlusions from images (e.g. tourists blocking the view in a holiday picture, or logos and subtitles in images or films). As such, inpainting is not a new field: in the pre-digital era, pictures where often retouched using airbrush or standard fine painting techniques. The digital inpainting techniques developed over the past years attempt to imitate this craft, and often follow similar principles. Adobe Photoshop is a famous example of software that incorporates inpainting algorithms.

Digital inpainting techniques typically first require the user to designate a region in the image to be inpainted (we call this the "missing region"). In this paper we always assume that the region is known and focus on the inpainting algorithms themselves. An important concept to which we often refer is that of isophotes. In grayscale images these are simply lines of equal grayvalue, but the concept is easily generalized to colour images by treating the three colour channels separately. For a good inpainting result, the isophotes should continue as smoothly as possible into the missing region according to Telea [13].

According to Bugeau et al. in [3], the three main paradigms into which inpainting can be grouped are based on:

- Geometric smoothness
- Self-similarity
- Sparse representation

The first two groups can be thought of as preserving respectively structure (i.e. isophotes arriving at the border of the missing region) and texture (the pattern of the image surrounding the missing region). The third group, using sparse representations for inpainting, like Maikal uses in [11], is relatively new and uses a dictionary of basis images which have been created from a database. In recent years, impressive results have been achieved by using databases. One example is a technique by Whyte et al. [15], which is geared towards inpainting photos of objects that are readily available online (e.g. tourist attractions) and infers 3D information from multiple viewpoints. Another example is a method by Hays and Efros [7] that uses millions of photographs to find a suitable inpainting, in which the intriguing challenge of semantics is addressed as well (i.e. you would not expect an area of your backyard to be inpainted with an elephant, despite the surrounding trees).

- *Jelle C. Nauta is a Master student at the RUG, E-mail: jcnauta@gmail.com.*
- *Sander O. Feringa is a Master student at the RUG, E-mail: sander.feringa@gmail.com.*

In this paper we focus on the pioneering work in geometric smoothness. We also address self-similarity, and discuss the combination of both approaches. By comparing several methods, we give an impression of the problems, possible solutions and their properties. Our comparison is based on the following criteria:

- Domain of application and visual results
- Conceptual simplicity and ease of implementation
- Computational performance

In doing so we provide an overview of the field, point out the limitations of each technique, and outline the importance of combining methods.

2 Descriptions of algorithms

In this section, we give a short explanation of different techniques in order to give the reader some insight into the problems and some of their solutions. First we treat three methods based on geometric smoothness, followed by an example of a self-similarity based method. For an in-depth understanding of the technicalities we refer to the papers throughout the text.

2.1 Geometric smoothness algorithms

Methods focussing on geometric smoothness use the information of pixels close to the missing region. Most of them are formulated as Partial Differential Equations (PDE's) or in terms of neighbouring pixel-value differences. The first method we explain in this section was developed by Oliveira et al. [12] and is much akin to regular image smoothing. As such it is not specifically geared towards preserving structure, but nevertheless a good introduction to the geometric smoothness methodology. The second method we explain was developed by Bertalmio et al.. The initial algorithm [10] determines the direction of isophotes (the gradient rotated by 90 degrees) and attempts to minimize differences in smoothness in this direction. The method was shown in [9] to be analogous to the well-studies Navier-Stokes equations. Finally, we treat a method by Telea [13] that uses a fast marching method to anisotropically transport the information of pixels in a small region around the boundary into the missing region.

The interested reader may also have a look at other examples of PDE methods, like the Total Variational (TV) model [4] and the Curvature-Driven Diffusion (CDD) [5] model, both by Chan and Chen.

2.1.1 Convolution filter method

Oliveira et al. use in their paper "Fast Digital Image Inpainting" [12] an inpainting method using the convolution of the image with a 3x3 filter over missing image regions to isotropically diffuse known image information into the missing pixels. Let the image region to be inpainted be denoted by Ω. The 1 pixel wide boundary outside that regions is defined as $\delta\Omega$. The inpainting itself can be seen as an isotropic diffusion process that uses the information in $\delta\Omega$ to fill in Ω. To save computing time and to keep the visual quality of the rest of the image we only convolve the filter with the image data at Ω instead of the complete image. The convolutions are done in iterations with the number of iterations defined either manually or depending on a certain threshold function that is evaluated periodically (possibly after every iteration). The algorithm in its general form is discribed in the following pseudocode:

```
Algorithm FILTERINPAINING($I$, $\Omega$, $M$)
Input. An image $I$, a selected region $\Omega$, a mask M.
Output. The inpainted image $O$
1.  initialise $\Omega$ with data in $I$
2.  for (iter = 0; iter < num_iteration; iter++)
3.      convolve $\Omega$ with kernel $M$
```

The kernel used is a weighted average Gaussian kernel that only uses neighbouring pixels and has a zero weight in the centre. This kernel is known as the mask. Masks can be seen in fig. 1 together with often used maskvalues. Oliveira states that the most time-consuming process of creating this model was creating the correct masks. This because the values for the masks had to be found emperically.

$$\begin{bmatrix} a & b & a \\ b & 0 & b \\ a & b & a \end{bmatrix} \qquad \begin{bmatrix} c & c & c \\ c & 0 & c \\ c & c & c \end{bmatrix}$$

Fig. 1. Two weighted average kernels used with the algorithm. Often used values are: a = 0.073235, b = 0.176765, c = 0.125

As can be seen in the top left image of fig. 2, the standard implementation of the connected filter method has problems with hard contrast rich edges. This edge reconnection problem can be solved with diffusion barriers. A diffusion barrier is a two pixels wide block that is placed manually on a specific spot (often on a hard contrast edge) where the user wants to block the diffusion to preserve the contrast. When the convolution process hits such a barrier, it stops iterating and continues to fill in the colour value from $\delta\Omega$, until Ω is completely filled in.

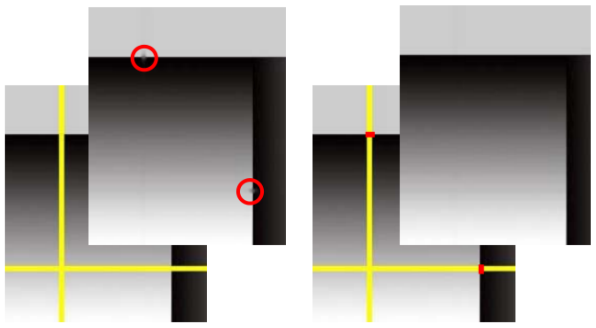


Fig. 2. Yellow crossing lines that need to be removed (back left) and the result after applying standard connected filter algorithm (top left). User-added diffusion barriers (right back) and the result produced with the connected filter and the diffusion barriers [12].

2.1.2 Smoothness-based propagation along isophotes

This method was first described by Bertalmio et al. in [10] and elaborated upon by Bertamlio himself in [9] and by Bornemann in [2]. The problem that inspired this method was the smooth continuation of lines of equal brightness (for greyscale images) into the missing region. Bertamlio [9] made the connection to the well-studied Navier-Stokes equations from fluid dynamics, giving the method a solid theoretical foundation and efficiënt implementation.

The method works by iteratively adjusting all the pixel values in the missing region and is explained very well in [10]. The adjustment is based on the difference in smoothness between a pixel itself and its neighbours as illustrated in fig. 3, and is explained below. Note that the figure shows a one-dimensional case, but we are dealing with two-dimensional images.

To still use this idea on a 2D image, a vectorfield is defined over the image which determines the direction in which the method is applied for each pixel. The vectors in the field are chosen in the direction of least variance (a 90-degree rotation of the gradient) to preserve the direction of incoming structures (isophotes) and must be recalculated after a few iterations. The border pixels of course remain unchanged throughout the iterative process, and the pixels in the missing region may initially have any value (they can be completely white as in for example image restoration or glare removal).

To determine the value by which pixel i is adjusted in an iteration, the second (discrete spatial) derivative is calculated with respect to the direction of least variance. For pixel i:

$$\begin{aligned} \frac{\partial^2 p}{\partial i^2} &= (p(i+1)-p(i))-(p(i)-p(i-1)) \\ &= p(i+1)+p(i-1)-2\cdot p(i) \end{aligned}$$

where $p(i)$ denotes the value of the i^{th} pixel (the pixelvalue p is treated as a discrete function of spatial coördinate i). In each iteration, $p(i)$ is adjusted as follows (also see again fig. 3):

$$p(i,t+1) = p(i,t) + \frac{\partial^2 p(i,t)}{\partial i^2} - \frac{\partial^2 p(i+1,t)}{\partial i^2}$$

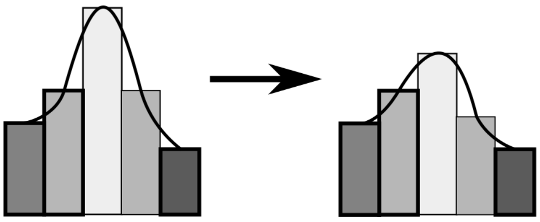


Fig. 3. One iteration of smoothness based pixel adjustment. The bars represent greyscale pixels, the fat-bordered bars represent the pixels bordering the (very small) "missing" region.

The kind of images that are produced iteratively by this method are shown in fig. 4.

2.1.3 Fast marching method

Telea defines in his paper [13] a method of inpainting that uses the simplest interpolation technique combined with a special technique to fill in the region Ω using the Fast Marching Method (FMM). Fig. 5 shows the situation as seen from the perspective of a single point p on the boundary $\delta\Omega$ that needs to be painted in. We can take a small area $B_\varepsilon(p)$ with size ε of the known image Img and use that to define the new colour value of p. A first-order approximation $I_q(p)$ of what the value of point p should become is given in eq. 1. I is the image itself, point q is a point in I that is used for defining p's value using the gradient $\nabla I(q)$.

$$I_q(p) = I(q) + \nabla I(q)(p-q), \tag{1}$$

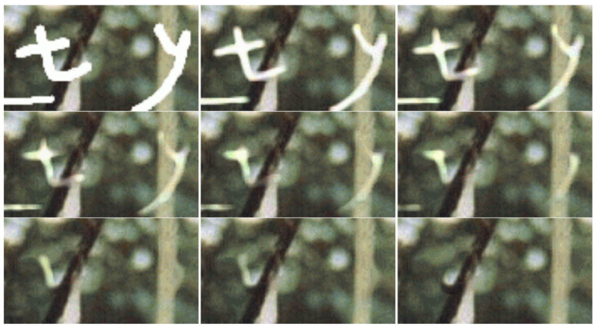


Fig. 4. Image taken from [10] showing the progression of the inpainting along isophotes.

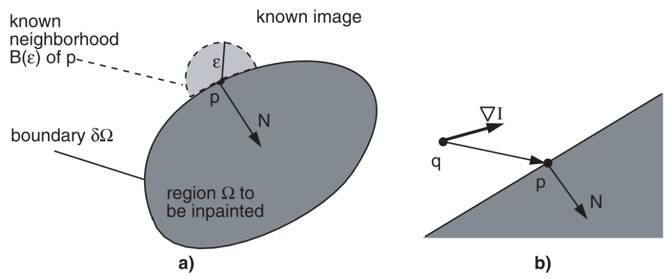


Fig. 5. The inpainting principle as discribed by Telea [13].

$$I(p) = \frac{\sum_{q \in B_{\varepsilon(p)}} w(p,q) I_q(p)}{\sum_{q \in B_{\varepsilon(p)}} w(p,q)} \tag{2}$$

The final new value of p, $I(p)$ is calculated by eq. 2 by taking the sum of all points q in the area $B_\varepsilon(p)$, thereby making it weighted by the function $w(p,q)$ as seen in eq. 3. The directional component $dir(p,q)$ (eq. 4) controls the contribution of the pixels in relation to the normal direction N (as seen in fig. 5: the closer to N, the higher the contribution. The geometric distance component $dst(p,q)$ (eq. 5) decreases the contribution of the pixels that are in distance farther away from p. And the level set distance component $lev(p,q)$ (eq. 6) ensures that pixels that are closer to the boundary to p contribute more then the pixels that are farther away. d_0^2 and T_0 are normally set to 1, the interpixel distance. Controlling the contributionlevels of these three components gives the user some influence over the outcome of the inpainting process.

$$w(p,q) = dir(p,q) \cdot dst(p,q) \cdot lev(p,q) \tag{3}$$

$$dir(p,q) = \frac{p-q}{||p-q||} \cdot N(p) \tag{4}$$

$$dst(p,q) = \frac{d_0^2}{||p-q||^2} \tag{5}$$

$$lev(p,q) = \frac{T_0}{1+|T(p)-T(q)|} \tag{6}$$

But all these interpolation calculations are only for the inpainting of one pixel and we need to fill in Ω completely. Since one inpainting step treats a boundary layer of only one pixel thick, we iteratively apply the inpainting step. This way the boundary is systematically pushed inward until the entire region is filled. To create the one-pixel wide boundary layers, an FMM is used. Telea [13] explains that in short, FMM solves the Eikonal Equation (eq. 7) in which T is the distance map from Ω to δΩ.

$$|\nabla T| = 1 \qquad on\ \Omega, \qquad with\ T = 0\ on\ \delta\Omega, \tag{7}$$

The following pseudocode describes this whole process:

```
Algorithm FMMINPAINTING( I, Ω)
Input. An image I, a selected region Ω
Output. The inpainted image O
1.  initialise $\delta\Omega_i$ as the boundary of Ω in I
2.  $\delta\Omega = \delta\Omega_i$
3.  while δΩ is not empty
4.      p = pixel of δΩ closest to $\delta\Omega_i$
5.      inpaint p using eq. 2
6.      advance δΩ into Ω
7.      add Ω to O
```

T consists of level sets, or isolines (as in height maps), which are equivalent to the different δΩ's we need. FMM ensures that all the isolines are only 1 pixel wide and that they are always processed in the correct order. When the FMM operation is finished and the inpainting algorithm is completed for every isoline, Ω is completely filled.

2.2 Self-similarity based

All the above inpainting methods are used for painting in small and often thin and long regions in an image. There are however also othertypes of Ω for which inpainting is needed. Criminisi defines two in [1]: (i) "texture synthesis" algorithms for generating large image regions from sample textures, and (ii) "inpainting" techniques for filling in small image gaps. One of the methods that use texture synthesis is the one developed by Cheng et al [14] (which is an enhanced version of the work done by Criminisi et al [1]). Chengs method uses exemplar based inpainting algorithm based on patches of the image with texture details that are used to paint in Ω. The order in which Ω is filled is determined by a predefined priority function. This is done to preserve the connectivity of object boundaries. Cheng states that the patch-based technique itself works fine but that the predefined priority function of Criminisi's method becomes unreliable when the number of iterations grows.

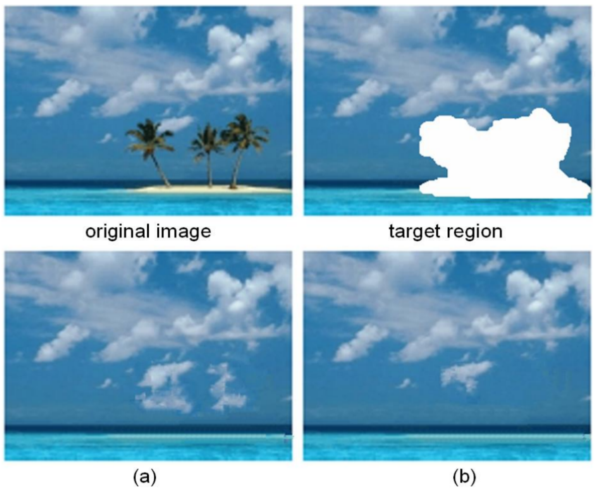


Fig. 6. Exemplar based image inpainting, (a) shows the result of Criminisi's method, (b) shows the result of Chengs method [14].

The original priority function $P(p)$ for a point p from Criminisi [1] is defined as in eq. 8 with $p \in \delta\Omega$. $C(p)$ is the so called confidence term that holds texture information and D(p) is the data term that holds structure properties.

$$P(p) = C(p)D(p), \tag{8}$$

$C(p)$ and $D(p)$ are defined in eq. 9 and eq. 10, where α is the normalization factor (α = 255 for a standard 8 bit image), $\nabla \frac{1}{p}$ is the isophote vector and n_p is the unit vector that is oriented orthogonal to δΩ of p.

$$C(p) = \frac{\sum_{q \in \Psi_p \cap \Phi} C(q)}{|\Psi_p|},\ 0 \leq C(p) \leq 1, \tag{9}$$

$$D(p) = \frac{|\nabla \frac{1}{p} \cdot n_p|}{\alpha},\ 0 \leq D(p) \leq 1, \tag{10}$$

According to Cheng, the problem lies with the calculation of the confidence term, namely: as incremental fillings go by, its value drops to zero too rapidly. He calls this phenomenon the "dropping effect". This effect produces patches that look too much alike the samples. Cheng notes that multiplication is known for its sensitivity to extreme values and that it overamplifies the input. A multiplication is used in the equation for $P(p)$ and the solution should be to replace it by an addition. However, after this change $C(p)$ still goes to zero too fast and another solution must be found. $C(p)$ should be regulated by a factor ω which Cheng emperically sets to 0.7. ω can be seen as a lower threshold value to $P(p)$. The $C(p)$ term is now replaced with $R_C(p)$, shown in eq. 11.

$$R_C(p) = (1-\omega) \times C(p) + \omega,\ 0 \leq \omega \leq 1, \tag{11}$$

Better control should be available over how much influence $R_C(p)$ and $D(p)$ should have on the final priority function. To accomplish this, the values α and β are added with $\alpha + \beta = 1$. All these changes combined give the new equation for a robust priority function $RP(p)$ as seen in eq. 12.

$$RP(p) = \alpha \cdot R_C(p) + \beta \cdot D(p),\ 0 \leq \alpha, \beta \leq 1, \alpha + \beta = 1, \tag{12}$$

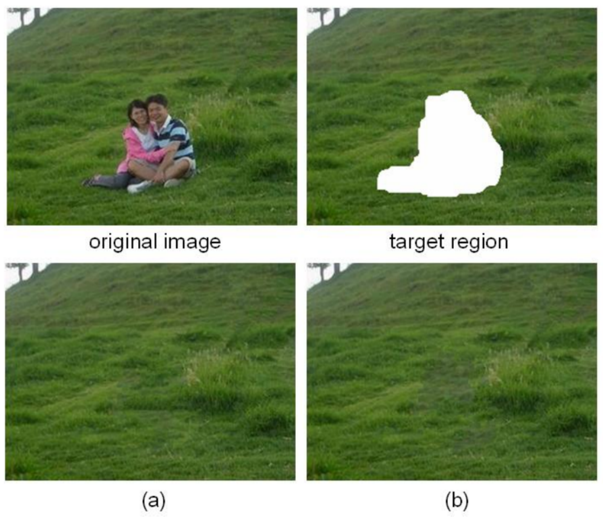


Fig. 7. Both Chengs method, (a) and (b) are made with different component weights. Source: [14].

As can been seen in fig. 6 and fig. 7: the quality of inpaining is very impressive, especially considering that inpaining large regions is much harder compared to inpainting thin lines. When larger regions need to be inpainted it is very important that neighbouring texture data from the image is used. The geometric smoothness-based methods only look at a very small area around a point on $\delta\Omega$ that needs to be painted in. As the width of Ω becomes larger, the chances of repetion in the inpainted result become higher as well. Texture-based inpainting gets important in such cases, since repetition is less visible with the use of larger textures.

3 Comparison of methods

The methods introduced in the previous section need to be compaired. Our comparison is split up into subsections concerning visual results, computational performance, and ease of implementation.

3.1 Domain of application and visual results

The different approaches we consider in this report have specific areas of application. In this section we explain the type of images the algorithms perform well on, and the ones they perform poorly on. Texture and structure play an important role in this.

3.1.1 Convolution filter method

Since the convolution filter method by Oliveira et al. is essentially the extension of a regular noise removal method. As such, it is only usable for small missing regions since neither structure nor texture is preserved. The reason the algorithm still achieves acceptable results is stated (with citation) in [12]: "the human visual system can tolerate some amount of blurring in areas not associated to high contrast edges". The obvious advantages of this method are its speed and simplicity. An example where the algorithm works well is shown in fig. 8

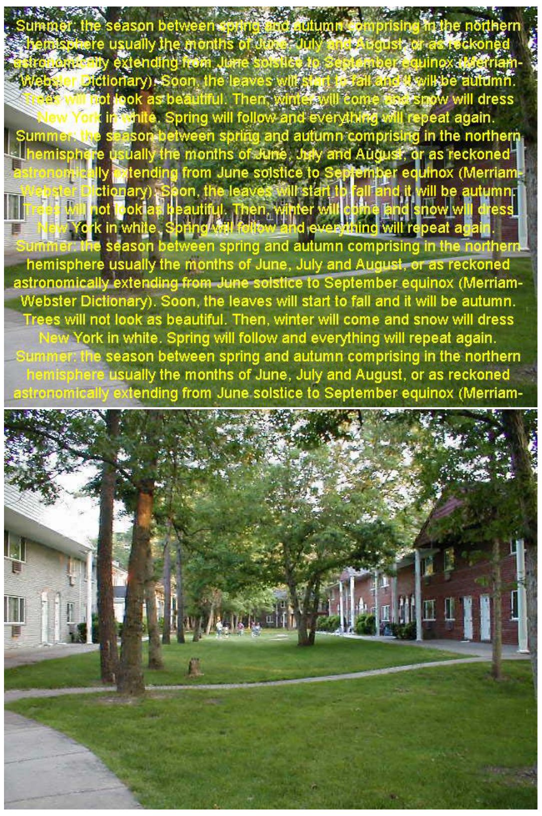

Summer: the season between spring and autumn comprising in the northern
hemisphere usually the months of June, July and August, or as reckoned
astronomically extending from June solstice to September equinox (Merriam-
Webster Dictionary). Soon, the leaves will start to fall and it will be autumn.
Trees will not look as beautiful. Then, winter will come and snow will dress
New York in white. Spring will follow and everything will repeat again.
Summer: the season between spring and autumn comprising in the northern
hemisphere usually the months of June, July and August, or as reckoned
astronomically extending from June solstice to September equinox (Merriam-
Webster Dictionary). Soon, the leaves will start to fall and it will be autumn.
Trees will not look as beautiful. Then, winter will come and snow will dress
New York in white. Spring will follow and everything will repeat again.
Summer: the season between spring and autumn comprising in the northern
hemisphere usually the months of June, July and August, or as reckoned
astronomically extending from June solstice to September equinox (Merriam-
Webster Dictionary). Soon, the leaves will start to fall and it will be autumn.
Trees will not look as beautiful. Then, winter will come and snow will dress
New York in white. Spring will follow and everything will repeat again.
Summer: the season between spring and autumn comprising in the northern
hemisphere usually the months of June, July and August, or as reckoned
astronomically extending from June solstice to September equinox (Merriam-

Fig. 8. A good performance by the convolution filter method as given in [12].

The shortcomings of the algorithm are illustrated in fig. 9 (a fragment from an image in [12]): the plant is blurred to the point of being discontinuous in the masked regions.

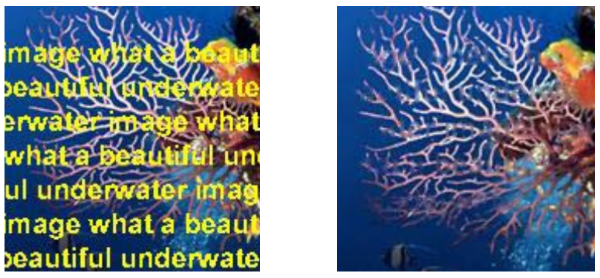


Fig. 9. Illustration of the limitations of the convolution filter method as given in [12].

3.1.2 Navier Stokes fluid dynamics method

The method by Bertalmio et al. performs well on images where a curved region is masked. A prototypical example of this is given in fig. 10

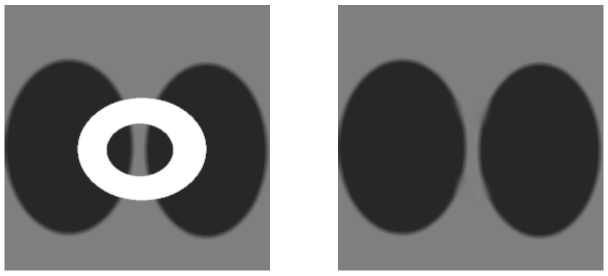


Fig. 10. A schematic example in which the algorithm by Bertalmio et al. from [10] works very well. The paper did not directly use the Navier-Stokes equations as in [9] but the visual results are very similar.

Pure PDE-methods try to create a smooth transition in the missing region, completely disregarding the texture of the surrounding area. An example showing how this can lead to unwanted results is shown in fig. 11, where the texture of the street is not imitated when filling in the region of the microphone. For larger missing regions, the problem becomes worse.

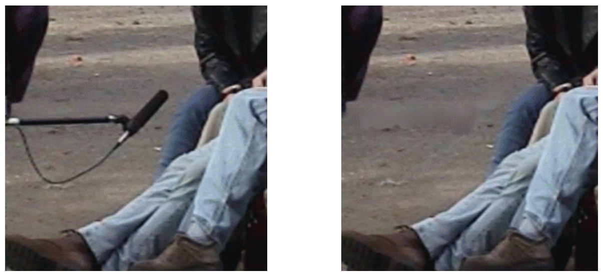


Fig. 11. Illustration of Navier-Stokes limitations as given in [9].

3.1.3 Fast marching method

Telea's Fast Marching approach [13] achieves similar results to those of Bertalmio et al. Telea provides a comparison in his paper, as shown in fig. 12. In both cases, the result of the fragment is not very satisfactory, although the result of Bertalmio et al. appears slightly better. Despite this, the inpainting results on the overall image appear quite good at first sight.

A notable improvement by Bornemann that was inspired by Telea's method is given in [2]. The results as reported in the paper are shown in fig. 13. Note that Bornemann's method is five times slower than Telea's, but the result is also much better.

3.2 Computational performance

Since we are doing a literature comparison without testing any code on a single machine, we can only relay information from the papers and rough estimations on computational performance. We divide the inpainting algorithms into three classes: fast, medium and slow. An example of a fast algorithm would be the inpainting of 15% scratched regions on a image of about 1 megapixel in a few seconds or faster on a typical 2009-2013 period PC. Medium speed would be about 20 to 40 seconds on that same operation and slow would be around 1 minute or slower for that same operation. Applications usefull for people who work on a daily basis with digital inpainting tools will only be considered usable if the inpainting operation is done in a few seconds at most.

Oliveira's connected filter has fast computational performance, taking only a few seconds on average for our example test. Note that this

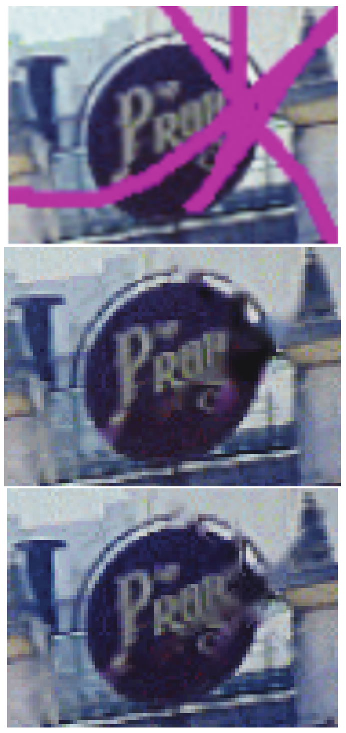


Fig. 12. A masked image fragment (top), inpainting by Bertalmio et al. [9](middle) and Telea's result (bottom)

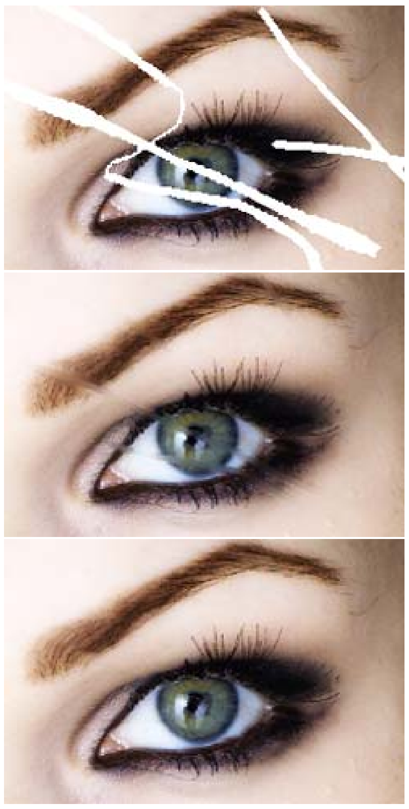


Fig. 13. A masked image fragment (top), inpainting by Telea's method [13](middle) and the result by Bornemann and März [2](bottom)

was even achieved on a PC with a 450 MHz Pentium II CPU, 128 MB RAM and Windows 98, done with 100 iterations. Oliveira states that the cost of inpainting for his algorithm is linear to the size of Ω and that operations are cache intensive. Standard scratch removal of the famous scratched Lincoln portait is 0.61 seconds. The example image of the ”Three Girls”, fig. 6 in Oliveira’s paper, needed only 1.21 seconds with his method on that same 450 MHz Pentium II PC. Bartalmio’s first implementation in [10] needed 7 minutes in its standard version and about 2 minutes with a two-level multiresolution approach, using an Pentium 300 Mhz, 128 MB RAM and Linux. Oliveira’s algorithm can be qualified as fast and should be extremely fast on modern PC’s since CPU power, cache sizes and RAM size has increased dramatically in that timespan.

The other authors are less specific about their computational performance figures but we still try to give an overview. Telea states in his paper that for his FMM method a performance can be achieved of 3 seconds on a 800x600 image with a 15% region needing to be painted in on an 800 Hhz PC. So his method seems to be about 4 times as slow as Oliveira’s method. Bertalmio’s Navier-Stokes solution from [9] only states: ”The results shown here are obtained in a few seconds of CPU time of a standard PC under Linux.” The images he uses seem to be, at least partially, the same as Telea’s. On Chengs examplar based method of [14] must be noted that it specialises in large areas, not in inpainting standard scratches or lines of around 15 pixel wide lines as the other papers do. Criminisi’s method took 2 seconds to inpaint a 40x40 pixels region in a 200x200 pixels image on a 2.5 GHz Pentium IV with 1 GB RAM. The removal of a bungee jumper region of 12% in a 205x307 image was completed in 18 seconds. We note that his version takes more time to complete the inpainting operation and should be classified as medium speed.

When we consider how Oliveira’s method works we can conclude that it is logical that his version is the fastest since it cleary has the lowest complexity by applying only a 2D-Laplacian filter on Ω as the compelete operation. Since the others use much more complicated algorithms and perform more complex operations we are overall impressed with the performance of these inpainting algorithms and expect them to have good performance on present-day PC’s, laptops and even high-end mobile platforms such as smartphones and tablets.

3.3 Simplicity and ease of implementation

According to Bertalmio [9], his inpainting algorithm is “programmed in tens of lines of C++ code”. Nevertheless, the mathematical complexity of this model and its lack of intuitivity make this the hardest to understand of the three techniques we studied, (together with Bornemann’s use of a structure tensor in a fast marching method).

Olivera’s method of isotropic diffusion is by far the easiest to implement and understand, but not very powerful visualy.

Telea’s FMM is in-between as far as simplicity is concerned. His paper gives a link to the location of the C++ source, but unfortunately this file was not found when we requested it. Regardless, the algorithm seems easy enough to implement once the concepts are clear.

We did not find clear indications in Cheng’s paper in his self-similarity method that his solution is very difficult to implement but since no clear information is given on this subject we cannot conclude anything.

4 Conclusion

In this paper we made a comparison of three pioneering inpainting methods from some years ago that are based on geometric smoothness, and we considered the importance of combining these with a self-similarity approach.

Oliveira’s connected filter method is very simple and achieves the lowest visual quality, but is also one of the fastest possible inpainting solutions. Under some circumstances (e.g. low contrast) low detail images where only thin lines need to be painted in, it may suffice. Bertalmio’s method gives the highest visual inpainting quality, but it is also by far the slowest algorithm. In comparison, Telea’s method is a tradeoff between computational and visual performance, being in-between Oliveria and Bertalmio in these respects.

None of the geometrical approaches are adequate for inpainting larger areas, but a self-similarity based method like Cheng’s is better for these cases, since it preserves texture. Since the advent of the first inpainting methods around 2000, results have gradually been getting better, largely due to evolution of techniques and by combining methods. For example, Li [8] obtains good results by combining completion of salient structures (the most defining edges of an image) with subsequent texture propagation. In [6], Du uses hierarchical image segmentation to prioritize smoothness, and using texture filling improves the results notably. This combination of structure and texture is a recurring theme.

For general inpainting solutions we conclude that the best results are obtained by combining the geometric smoothness and self-similarity techniques in a general inpainting solution. This way, one can both remove scratches and fill in larger areas. One possible improvement in future work would be to automatically analyze the image and missing region, and select appropriate inpainting algorithms based on this. Current computer hardware is so fast that speed issues are of less importance.

References

[1] P. P. A. Criminisi and K. Toyama. Region filling and object removal by exemplar-based image inpainting. *IEEE Trans. Image Processing*, 13(9):1200–1212, 2004.

[2] F. Bornemann and T. März. Fast image inpainting based on coherence transport. *Journal of Mathematical Imaging and Vision*, 28(3):259–278, 2007.

[3] A. Bugeau, M. Bertalmío, V. Caselles, and G. Sapiro. A comprehensive framework for image inpainting. *Image Processing, IEEE Transactions on*, 19(10):2634–2645, 2010.

[4] T. F. Chan and J. Shen. Mathematical models for local deterministic inpaintings. *Technical Report CAM 00-01*, 2000.

[5] T. F. Chan and J. Shen. Non-texture inpainting by curvature driven diffusions (ccd). *Technical Report CAM 00-35*, 2000.

[6] X. Du, D. Cho, and T. D. Bui. Image inpainting and segmentation using hierarchical level set method. In *Computer and Robot Vision, 2006. The 3rd Canadian Conference on*, pages 52–52. IEEE, 2006.

[7] J. Hays and A. A. Efros. Scene completion using millions of photographs. In *ACM Transactions on Graphics (TOG)*, volume 26, page 4. ACM, 2007.

[8] S. Li and M. Zhao. Image inpainting with salient structure completion and texture propagation. *Pattern Recognition Letters*, 32(9):1256–1266, 2011.

[9] A. B. M. Bertalmio and G. Sapiro. Navier-stokes, fluid dynamics, and image and video inpainting. *Computer Vision and Pattern Recognition, 2001. CVPR 2001.*, 1:355–362, 2001.

[10] V. C. M. Bertalmio, G.Sapiro and C. Ballester. Image inpainting. *In Proceedings SIGGRAPH 2000, Computer Graphics Proceedings Annual Conference Series*, pages 417–424, 2000.

[11] J. Mairal, M. Elad, and G. Sapiro. Sparse representation for color image restoration. *Image Processing, IEEE Transactions on*, 17(1):53–69, 2008.

[12] M. M. Oliveira, B. Bowen, R. McKenna, and Y. sung Chang. Fast digital image inpainting. pages 261–266, 2001.

[13] A. Telea. An image inpainting technique based on the fast marching method. *Journal of Graphics Tools*, 9(1):23–34, 2004.

[14] S.-K. L. C.-W. W. Wen-Huang Cheng, Chun-Wei Hsieh and J.-L. Wu. Robust algorithm for exemplar-based image inpainting. *Proceedings of the International Conference on Computer Graphics, Imaging and Visualization*, 2005.

[15] O. Whyte, J. Sivic, and A. Zisserman. Get out of my picture! internet-based inpainting. In *British Machine Vision Conference*, pages 1–11, 2009.

A Study about Middleware for Smart Environment

Fatimah Alsaif

Abstract—Ubiquitous computing, and specifically a smart environment, starting from a smart home to more complex one like a smart city, is an emerging paradigm for interactions between people and computers. Its aim is to break away from desktop computing to provide computational services to a user when and where required. Large numbers of heterogeneous computing devices provide new functionalities, enhance user productivity, and ease everyday tasks. In home, office, and public spaces, ubiquitous computing will unobtrusively augment work or recreational activities with information technology that optimizes the environment for people's needs. These smart environments demand efficient interoperation mechanisms among different heterogeneous sensors including the discovery and the management of these devices. The diverse domains of applications also require interoperation among themselves. The middleware plays a key role to achieve this interoperation. From the previous works there are various kinds of devices and different integration methods. The rationale behind this paper is to study the existence or lack of a suitable middleware infrastructure at smart homes for the development of applications and services at ubiquitous computing environments. This is done by surveying the state of the art in the area, and illustrating the requirements of ubiquitous computing in middleware which are dynamicity, scalability, dependability, security and privacy. After that, discuss the current middleware practices which are RUNES, ANGEL, GTSH, Gaia, Serenity, and SM4ALL. Comparing them considering the mentioned ubiquitous requirements, the result is that there are a few smart homes infrastructures and practices (two out of the six mentioned practices) that consider most of the ubiquitous computing needs. Where most of those projects such as RUNES, ANGEL, GTSH, and Serenity touched the topics of dynamicity, scalability, and dependability at the middleware-service level. Also, a few of them like ANGEL, Gaia, and Serenity filled the security and/or privacy needs. The SM4All project was the only one that cover all the mentioned requirements. Since, the area of networked embedded systems at smart homes grows at a rapid pace, several industrial and academic research activities are underway. This paper is ambitious to help investigate the works that have been done befor for smart environment, and to show the requirements for the future smart homes.

Index Terms—Device discovery, middleware framework, smart home,hetrohenious device discovery, domotics, embeded systems, smart home middelware, smart home practices.

1 Introduction

Smart environments have many examples in our life like home automation and assistive living. Everyone wants to live in comfortable places where all of their requirements depend on technology, usability, stability and longevity. This smart environment means that the individual user has his own smart space supported physically with the devices and services that are enabled to him in the real life automatically. The provision of middleware is the main idea of smart spaces. The most proprietary feature of the smart devices is to be hidden and easy to move and deploy at smart spaces [1].

To consider the home as smart home, it must first support and supply the people who live in it with their daily needs. It has to meet their needs correctly to the home events, and change if necessary. For stakeholder inhabitants, the smart homes must be easy to use for their behaviour and be ready for any change. For new devices and services, smart homes must be ready to apply them and make them easy to use without forcing the inhabitants into unwanted vendor or technology lock-in. Also without requiring significant physical, administrative or cognitive effort.

However some automatic pet systems are available now in the markets, they are also used to compare the standards and protocols that can block the creation of smart homes where we can find a large variety of devices and services which can easily inter-communicate. It is necessary that developers can abstract away from the detailed topography, protocols, data formats and control of sensors, actuators and other information devices, and instead of that they have to focus on information processing from many sources. This makes a flexible architecture and has the possibility to produce responses to information which are uncertain and noisy which exactly meet sensor-rich environment. A common approach to addressing such issues is through the use of middleware [2].

• *Fatimah A. Alsaif, Distributed Systems Research Group, E-Mail: F.A.S.Alsaif@org.nl.*

Nowadays, the systems of networked embedded are growing at a rapid speed and many industrial and academic study activities are well underway. As a reason of growing attention is the availability of small, cheap devices, with high networking abilities. These new technologies improvement are the target today to be used easier at home than before. The cheap devices are not sufficient by themselves, the interoperation and creation of dynamic adaptable programs for their interoperation will be the keystone of success. To be clear, any action towards this point of view needs important change in the embedded middleware development strategies of the past, with many respects such as functionality and operational environment.

This paper tends to study the existence or lack of suitable middleware infrastructure practices for the development of applications and services in ubiquitous computing environments at smart homes. For doing this study, I choose one of the latest projects that focus on middleware at smart homes called Smart Homes for All (SM4All). SM4All project aims to design and implement a new middleware platform for inter-working of smart embedded services in immersive environments and person-centric. The dynamic service reconfiguration at SM4All tend to use the semantic techniques and composability. The most significant characteristics of such a middleware are its tolerance to faults, rapid scaling of the network, fulfilling security requirements, dynamicity of its components, and operating on embedded devices [3].

This paper offers an overview of the current state of the art in the fields related to the SM4All project. During the overview of the state of the art related to SM4All, I consider two aspects. First, looking at the concept of middleware, the methodologies, and technologies which are relevant for the SM4All project. Second, reviewing a number of existing and recent research projects which share with SM4All the focus on pervasive computing and embedded systems with some comparisons. Smart homes projects and ubiquitous computing is the focus area in this study with the specific requirements. Also, this study finds out the common reached middleware technologies and gives the needs for the future's point of view in the area of smart homes according to the requirement of ubiquitous computing. The rest of the paper is organized as follows. Section 2 starts with general introduction into middleware. Section 3 illustrates the recent protocols and standards applied at middleware. Section 4 shows the categorization of middleware technologies according to the context of the study. Section 5 provides illustration for the main requirements of smart spaces with regard to middleware technology for services in ubiquitous computing. Section 6 gives some practices of current smart homes projects. Section 7 studies the middleware of SM4ALL and the rest of the practices described in section 6, according to the characteristics given in section 5. Section 8 describes the future directions of research work within the middleware for smart homes. Section 9 is the research conclusion.

2 MIDDLEWARE IN SMART HOMES

Smart homes interaction components like complex communication and data management gradually become impossible to deal with, because of their popularity and increasing numbers instead of support from a mediating infrastructure. Building abstraction layers with common services made available to developers manage the complex systems in traditional distribution. These abstractions are called middle-ware. In other words, middleware is the software layer which abstracts and provides an uniform interface to the application and the user of the system from the distribution of the resources of an information system.

A layer between application and system software is the definition of middleware which supports incorporation between many products and platforms, while keeping the integrity of all solutions in terms of accuracy. For example, the developer's point of view, we can use the middleware to introduce services to form new devices and sensors. The new devices which enter the smart home needs to be installed with the environment. An essential question has to be answered: where do the new devices sit in the network: are they a master, slave or a peer? What does the sensor do? How fast can it release and absorb information? What options does it perform? these questions need to be answered. To address these issues in an efficient way, it is desirable to design a system that can seamlessly integrate new devices without user interference, and this is what we call a middleware [3].

The system and application's decoupling through middleware introduces lots of different advantages: introduction of new devices, components can reuse services, modules and other systems can be transparent; and a uniform view of the world simplifies the development process [2][4].

3 MIDDLEWARE PROTOCOLS

Building automation was focused for one purpose by technologies and standards which take a more device centric and protocol-based view of the domain to manage many kinds of controllable modules by a common abstraction presented a unified interface. By this, it can make the operations automatic through many kinds of controllable modules. Examples of used protocols are:

- The Building Automation Control network (BACnet) and Local operation netWork (LonWorks) are examples of systems which are designed to become the building of automation and control networks. BACnet was specially built in an attempt to make a united protocol for improving interaction between many systems and products in commercial building [5].
- Open Building Information Xchange (OBIX) defined Extensible Markup Language (XML) standard and the service of the web that makes the information exchange easier and smoother to transform information between smart homes and intelligent buildings and applications [5].
- Universal Plug and Play (UPnP) is a protocol that has an important role in device-neutral technology that's used by the abstraction layer of the device. UPnP helps the program independence because it depends on standard technologies such as the Transmission Control Protocol which is of the Internet Protocol suite (TCP/IP), User Datagram Protocol (UDP), Hyper Text Transfer Protocol (HTTP), XML and Simple Object Access Protocol (SOAP). It offers support for control, discovery, presentation and event notification [6][3].
- The framework of Open Services Gateway initiative (OSGi) which is a service platform for Java and also it is a dynamic module system, service gateways in the sever-centric architecture were built by the main purpose. OSGi is the central combine point to manage the network of the house with various technologies of communication, and to make service from many devices to load and set on the main service gateway. The meaning of the framework is to make it easy to create the use of applications and services on a gateway area [7][8][3].
- ProSyst has improved a program that called mBedded Server, compliant with the OSGi dynamic services program which is completely able to interconnect and has the control on smart devices and be able to remotely repair at home network [8].
- The basic service network infrastructure that runs on top of Java is called Jini, that makes the service easy to join and exit without problems on the network or the users of the network. Jini provides the solution for the developing ubiquitous computing needs required [1].
- Open industry standard called (X10) can work as a controller to communicate devices in smart environment. Generally X10 enabled the work with power line wiring to signal and control, and the work with commands like on/off [5].
- The most well known standards are the wireless technology protocols, such as Bluetooth, ZigBee, Radio Frequency Identi-fication (RFID), Wireless Fidelity (WiFi), and cellular technolo-gies. These standards are combined to be used to build a smart home. There are many wireless technologies that work effectively and also support remote data transfer, remote control and sensing devices that are candidate for enclosure in the smart home portfolio [9].

4 MIDDLEWARE CLASSIFICATION

Various classifications of middleware exist, in the context of the present review, I classified middleware in service based, event based and object-oriented.

1. Object-Oriented middleware

Object-Oriented middleware is based on the notion of remote method invocations, which typically are synchronous. Many transactions can be supported by object middleware that can group in each transaction the similar requests between the objects [10]. The programming language Java has its own flavors of object-oriented middleware e.g., OSGi is a recent initiative also based on Java.

2. Service based middleware

A service is an abstract functionality that can be invoked ignoring implementation details and simply focusing on the service signature. This type of middleware has become popular with the increasing adoption of standards known as Web services [11][12]. The Web services standards comprise a great number of protocols and technologies, though only some of them have gained widespread use e.g., SOAP which is the basic protocol for binding two services and is widely used on the Web.

3. Event based middleware

Event based communications are export interfaces abstract functionality that can provide an intuitive reactive communication paradigm that can be exploited to implement asynchronous interactions. An event based interaction assumes that the interacting parties either play the role of publishing or subscribing. Client can refer to their interest in a subset of all the produced events by issuing subscriptions [13]. An example of popular standard for event-based communications in middleware platforms is Jini.

5 UBIQUITOUS COMPUTING AND MIDDLEWARE REQUIREMENTS

These days, we can find few ubiquitous computing environments tend to be extremely specialized and depends on application-specific program. These programs improved to interactive environments sh-ould be able to connect and control lots of components in both hardware and software. They have to work in real-time, dynamically add and clear components to a working system without interrupting its process, control allocation of resources, and represent a means to find out the persistent-state of the information. These components are not instructed to extremely cooperate, because of that they don't only have to be connected, but there's also a call to show the logic of this interconnection. On the other hand, these connections aren't just protocols, but they are also consist of explicit knowledge of the way to use these protocols. So, to simply view the connections as appli-cation programming interface is not enough. It is difficult to reach goals without a common program to cooperate among different applications. To form an application in this domain it is required to define a popular design methodology that depends on new models which are independent from the technology. In order to form the improvement of interactive environment applications, this will require a model that summarizes the main components of an ubiquitous computing environment [1]. We can classify these into three layers:

- Physical layer which deals with technological constrains.
- Middleware layer which structure the cooperation of abstract services.
- Application layer that concerns the user interfaces.

Middleware will show an access abstraction for many ubiquitous devices, which allows them to interact and cooperate. That will let the applications writing gradually scale both in services offered, and on devices composing the system. It is intended that the model will present a standardized view of main interactive environment functionality [14].

Middleware which connects the distributed components together has to address lots of needs of ordinary systems, like heterogeneity, mobility, scalability, and tolerance for the component problems. Moreover it has to protect the user's information, like location and preferences, with their private preferences, and make sure that the applications take the automatic actions and enable the users to understand and control it. At last, many of distributed components which are available in smart home systems show a need of simple techniques for deploying, and the need of sensors which have configuring and managing networks, actuators, processing components, and repositories. The most important Ubiquitous Computing Re-quirements (UCR) or challenges which have to be presented by a middleware infrastructure to serve the ubiquitous computing could be under the following headings [14][15].

- Dynamicity: as in classical networks, the sensors and services are no more static, e.g., the monitoring of the environment and its management, although the overall distributed system which is consisting of all the sensors and devices and appliances requires constant dealing with the user's context, practice, etc., by adding, removing, and composing on-the-fly basic elements like sensors' services, devices, and appliances.
- Scalability: to submerge the users in the system, the sensor's numbers, devices, and applications must be large. The current situations less than the order of magnitude e.g., the current best-in-class smart houses count for tenths of sensors, devices, and appliances. The subsequent generation smart houses for all will tend to add up hundreds of devices.
- Dependability: the users heavily depend on the environment/ system itself, when they are at the centre of it, and payment on the environment or the system itself, so it has to be highly dependable.
- Security and privacy: the security of the overall hidden environment is definite; in addition, an environment, if hacked, may prove any sensible information on the users, so the structure of a system like that will pay special attention to protect privacy, that has to be built-in the system, not added-on recently, as in existing design practices.

6 SMART HOMES PRACTICES

Supporting a wide range of household devices are focused by many of research and industrial projects over heterogeneous network environments that have been performed. Most of the solutions examine the combination and integration of technology and services through home networking to have a better life. The existing solutions or projects are kinds of what we call embedded systems. Embedded systems are specified computers that are used in big machines or systems to have control of devices like communication, machined of the offices, automobiles and devices at home. These Embedded system in immersive realities, i.e., scenarios in which invisible embedded systems need to continuously interact with human users to supply sensed information and to react to service needs from the users themselves. The main cause which is having the user at the centre controlling the home, has many new challenges to the middleware these days and the technologies of services for the embedded systems, in terms of considering scalability, dependability, dynamicity, security and privacy [16].

These considerations subjects need the techniques of novel and the technologies of middleware aimed to person-centric fixed system. One of the most common scenarios that need person centric fixed systems, home-care and domotics assistance are specially remarkable in Europe nowadays context. Fields as home-care and domotics where housing meets technology in its various forms such as informatics, communication, robotics, mechanics, and ergonomics in order to improve new homes from the theory of safety, comfort, and the care of old people. Users need to have the ability of having a unique view on all the hardware in their homes. So, a challenge these days for homes is total interoperability and cooperation. From another point of view, people want to interact with their devices at home, independently of their status and abilities. [16].

Many projects in Europe, recently e.g., DEFIE, DOMOH, MOSAIC-HS, and HOMETALK made their own domotic concerns for particular end-user. So, no one domotic middleware infrastruc-ture is really for all, i.e., they don't enough take into account handicapped and elderly people [27]. Recently there was an important project in the field of SM4ALL. SM4ALL wants to achieve studying and improving an innovative middleware platform for inter-working of smart fixed services in immersive and person-centric environments, by using the techniques of semantic and compos-ability, to make the dynamic guarantee, working alone and gradual improvement, during protecting the security and privacy of the platform and the people who use it. This presents the challenging scenario of home building in the presence of users with a variety of abilities and needs e.g., young able bodied, aged and disabled. SM4ALL satisfies the requirements for interoperability and ways of dynamics, during preserving human intervention to the low level. The design of SM4ALL middleware allows the automatic discovery and connec-tion of devices with various network protocols, e.g. Bluetooth, ZigBee, etc. using UPnP and OSGi standards. From the technical point of view, the suggested solution is an applicable and an effective one [3][17].

In Europe, some addressed projects with concerns that are partially related to SM4ALL are RUNES, ANGEL, GTSH, Gaia, Serenity.

1. Reconfigurable Ubiquitous Networked Embedded Systems (RUNES) [18].
   - Project description: project RUNES makes it easy to create large-scale, widely distributed and heterogeneous systems of networked embedded which deal with and adapt to their environments. RUNES introduce an adaptive middleware program, well known language which make the application creation process simple.
   - Project user's needs: RUNES project helps innovative user applications for networked embedded systems. Its main objectives to supply an adaptive middleware program and application improvement tools which allow programmers to be flexible to interact with the environment where necessary, during affording a level of abstraction which makes the application easy to construct and use.
   - Project interaction: the approach of RUNES is based on a small and well-organized middleware kernel that supports greatly customizable and modularized services that are component-based middleware. These services can be tailored to definite embedded environments, and support adaptivity by reconfiguration at run time. Users can be able to interact with the system throughout normal interfaces such as Personal Digital Assistant (PDAs) or computers.

2. Advanced Networked embedded platform as a Gateway to Enhance quality of Life (ANGEL) [19].
   - Project description: the ANGEL project's objectives to provide ways and tools for instruct complex heterogeneous systems in which Wireless Sensor Networks (WSNs) and ordinary com-munication networks cooperate to monitor and increase the life's quality in common practices such as at home, car and city environment.
   - Project user's needs: the user scenarios addressed by the ANGEL project were personalized indoor environmental moni-toring and control for wellbeing, post-acute and chronic disease management, and personal wellness and interrelated enabling services.
   - Project interaction: users can communicate with WSNs throughout a allocated node called gateway. This node is ac-countable for supplying queries into the network, also meeting responses and introducing those responses to the users. The gateway interacts with the user directly or remotely through wired or mobile communication networks and communicates with the WSN by short-range wireless links, consequently providing the link to a service center.

3. Gator Tech Smart House (GTSH) [20].
   - Project description: GTSH project is a laboratory house specifically instructed to help older people in enlarging independence and improving the life's high quality. The objective of the project is to make helpful environments like homes which can sense themselves and the people who live in and enact mappings between the physical world, intervention services, and remote monitoring.
   - Project user's needs: the project is interested in older persons. It gives solutions to user's traditional life needs, starting from food preparation to home security and management of emergency.
   - Project interaction: the project provides programmable pervasive smart spaces. This smart space works as a runtime environment and a software library. The user can interact with the system transparently in a continuous and ubiquitous way.

4. Gaia [21].
   - Project description: Gaia presents the operating system functionality to physical spaces. To extend the reach of traditional computing systems to include the devices and the surrounding physical space, so both physical and virtual may be allowed to interrelate. Consequently, physical spaces become interactive systems, or in other terms, Active Spaces. The aim of Gaia is to construct and perform a middleware operating system which manages the resources contained in an active space.
   - Project user's needs: it was intended that the Gaia project will improve the capability of small cooperative teams by merging hardware and software resources according to different user context. In this scope, the major significant user needs covered by Gaia are the resources dynamic adaptation.
   - Project interaction: user can interact and manage Gaia applications by a regular computer interface such as PDA, and a voice interface. Gaia also offeres a service that can directly detect user situation as location by using log-in information and sensors.

5. Security and dependability in Ambient Intelligence Systems (Serenity) [22].
   - Project description: Serenity is one of the projects supported by the European Commission. Serenity provides a private and dep-endable solution for the heterogeneous and dynamic architectures.
   - Project user's needs: the project scenario tends to support the ambient intelligence, which is the idea that people will be enclosed by intelligent and interactive interfaces embedded in everyday objects around us. This will help raise the effective-ness and increase productivity of individuals.
   - Project interaction: the main concepts that are used by serenity are ubiquitous communication, ubiquitous computing, and intelligent user interfaces, by means of a more emphasis on the privacy and security issues.

7 SM4ALL Versus Smart Homes Practices

In particular, comparing RUNES, ANGEL, GTSH, Gaia, and Serenity projects with the SM4ALL project, the SM4ALL project defines a general architecture for embedded middleware. This embedded middleware is targeted to immersive scenarios, through which the home-care and domotics are chosen as showcases. Specific features are offered by the SM4ALL platform e.g., dynamicity that takes care of the capability to manage the new devices insertion and removal, and scalability that considers the number of embedded services to be managed. In addition, the SM4ALL platform design focuses on ontologies for telling service capabilities. These service capabilities are used for gaining the dynamic configuration and structuring the services, while maintain the users privacy. The security of the smart environment gained specific emphasis, considering to who and which can do what, also to probable intrusion of spy services and devices [16][23].

Finally, the SM4ALL project is an embedded pervasive platform developed for smart houses truly for all, in which users with different needs and abilities (such as aged, disabled, and young able bodied) can interact through basic and advanced interfaces with the services provided by the diverse domotic devices and appliances [16][23].

To make things more specific, the main areas SM4All involve are middleware, networking and embedded operating systems. Con-sidering the main areas relevant to the SM4All project, in this paper the focus is on the middleware area. The middleware area is further subdivided into object-oriented, service based, and event based as described before in section 4. Tables 1,2,3,4, and 5 show what each project focuses on at the three areas of the middleware (Service based, Event based and Object-Oriented), this is done in terms of the five fundamental aspects that we already defined in section 5, namely, dynamicity, scalability, dependability, security, and privacy.

Table 1. The Middleware Platform of RUNES Project and UCR

Middleware Requirements	Object-Oriented Based	Event Based	Service Based
Dynamicity	×	×	√
Scalability	×	×	√
Dependability	×	×	√
Security	×	×	×
Privacy	×	×	×

Table 2. The Middleware Platform of ANGEL Project and UCR

Middleware Requirements	Object-Oriented Based	Event Based	Service Based
Dynamicity	×	×	×
Scalability	×	√	×
Dependability	×	√	×
Security	×	×	×
Privacy	×	√	×

Table 3. The Middleware Platform of GTSH Project and UCR

Middleware Requirements	Object-Oriented Based	Event Based	Service Based
Dynamicity	×	×	√
Scalability	×	×	√
Dependability	×	×	×
Security	×	×	×
Privacy	×	×	×

Table 4. The Middleware Platform of Gaia Project and UCR

Middleware Requirements	Object-Oriented Based	Event Based	Service Based
Dynamicity	×	√	×
Scalability	×	×	×
Dependability	×	√	×
Security	×	√	×
Privacy	×	√	×

Table 5. The Middleware Platform of Serenity Project and UCR

Middleware Requirements	Object-Oriented Based	Event Based	Service Based
Dynamicity	×	×	×
Scalability	×	×	×
Dependability	×	×	×
Security	√	×	×
Privacy	√	×	×

Table 6. Smart Homes Related Projects Middleware platforms and UCR

Middleware Requirements	RUNES	ANGEL	GTSH	Gaia	Serenity	SM4All
Dynamicity	√	×	√	√	×	√
Scalability	√	√	√	×	×	√
Dependability	√	√	×	√	×	√
Security	×	×	×	√	√	√
Privacy	×	√	×	×	√	√

Table 6 provides a summary of the middleware area covered in the surveyed projects in general according to the findings in the tables 1,2,3,4, and 5. The notice is that most projects (four out of the six projects) touched the topics of dynamicity, scalability, and dependability at the middleware-service level. Also, a few of them (three out of the six projects) filled the security and privacy needs. But, from figure 1 we can easily notice that the only project that covers all defined requirements is SM4All project.

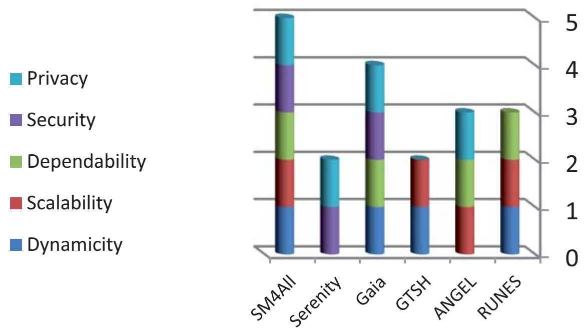


Fig. 1. Smart Homes Related Projects Middleware platforms and UCR.

Moreover, the visions for all the surveyed practices (RUNES, ANGEL, GTSH, Gaia, Serenity, SM4ALL) tended to satisfy some or most of the following smart environment characteristics:

- Humans should be able to manage and control the new immersive environments, and all efforts to improve any smart environment has to be approached and designed according to the requirements and needs of the users to present novel experience to users. These environments and systems would synthesize all available information about each user, personaliz-ed treatment may be offered by individuals or groups depending on their profiles. This needs technologies of novel for data dissemination, integration, user profiling, and context comput-ation.
- The provided infrastructural services such as storage and retrieval of service descriptions, communication, etc. And the middleware itself would be managed widely in a distributed manner; so that, a peer-to-peer paradigm, should be enforced, so as to guarantee dynamicity, scalability and dependability. The nativity of the middleware has to be service-ready, to supply the service deployment and on the fly installation.
- The economic scale, reusability, and the ability to extend, middleware has to be developed to contain all main aspects of the immersive scenarios. To supply any further environment-specific needs on content or the interfaces of the users has to be improved in a adapted fashion on top of it.
- Openness of the middleware is important too. The technologies of web service, and service-oriented access, are a promising solution to this, when adequately complemented with certain solutions for their scalability, dependability, dynamicity and security.

8 Future Directions

The major last ubiquitous computing environments such as Active Campus [24], Oxygen [25] and Easy Living [26] have their own independence improvement with low consideration of interoperab-ility [1]. It would be a good and helpful effect if these ubiquitous environments were improved from a popular middleware infrastructure and leave the varied nature of these ubiquitous environments transparent to the managing system.

The current practices are not reasonable, the point of view demands that the following characteristics are enforced [15]:

- Heterogeneity supplying: the devices which are combined from resource-poor sensors, actuators and mobile client devices to servers all have to be supplied and supported by networking interfaces and programming languages. Also, legacy devices might be presented.

- Mobility supporting: all devices specially sensors sets and applications are able to mobile and being movable, the protocols of communication that underpin the system has to supply specific flexible forms of routing. The information of context may require to combine with context-aware components. The components which are flexible are needed.
- Controlling and traceability: the devices and all the information flows between components of any smart system must be open to inspection and manipulation, so as to present adequate unde-rstanding and system control for the users, and to make it possible to debug.
- Tolerance for component failures: in the ordinary process, sensors and other components may fail, disconnection sometimes happen, but the system must continue operating, without needing any other resources to detect and handle failures.
- Ease of deployment and configuration: the hardware and software components must be easy to use, deploy and configure to be suitable to the non-expert user and environmental needs.

9 Conclusion

Since the main goal for this paper is to study whether the existing middleware practices at smart homes support the requirements of ubiquitous computing environments or not. This paper concludes that there are good smart homes infrastructures and practices e.g., SM4All and Gaia projects, that consider most of the basics of the ubiquitous computing needs (e.g., dynamicity, scalability, dependability, security and/or privacy). This done by first introducing the concept of middleware and its protocols and standards which are relevant for the projects e.g., UPnP and OSGi. Then, I classify middleware according to three types: object-oriented, service based, and object oriented base. Following this, a definition of requirements for smart spaces with regard to middleware technology are highlighted. These requirements are kept generic (e.g., dynamicity, dependability, scalability, security, and privacy) in order to allow the identification of criteria for the evaluation of middleware technology.

After that, I choose SM4ALL, RUNES, ANGEL, GTSH, Gaia, and Serenity as examples of the current practice middleware technologies projects at smart homes, discussing them with their comparisons for the middleware part according to the mentioned classification and the given ubiquitous requirements. The findings are that some projects touched the topics of dynamicity, scalability, and dependability at the middleware-service level. Also, a few of them filled the security and privacy needs. The only project that covers all defined requirements is SM4All.

However, the horizons of the present practices middleware will have to expand more and more to cater for the new requirements of ubiquitous computing (e.g., support of heterogeneity, mobility, tolerance for component failures, controlling and traceability, and ease of deployment and configuration) if they want to survive in the emerging smart space environments. This study help giving a look of the existing works of middleware at smart homes, and showing the requirements for the future vision. Moreover, it is not clear at this point exactly what smart space middleware will consist of. What is likely, however, is that various smart space architectures will emerge independently of each other which greatly increases the need for middleware to provide interoperability between heterogeneous systems.

References

S. Cummins, A. Davy, J. Finnegan, and R. Carroll, “State of the Art: Middleware in Smart Space Management,” *M-Zones,* http://www.m-zones.org/deliverables/d1_1/papers/4-03 middleware.pdf. 2003.

L. Coyle, S. Neely, G. Stevenson, M. Sullivan, S. Dobson, and P. Nixon, “Sensor fusion-based middleware for smart homes,” *International Journal of ARM,* Vol. 8, no. 2, pp. 53–60, June 2007.

E. Warriach, E. Kaldeli, A. Lazovik, and M. Aiello, “An Interplatform Service-Oriented Middleware for the Smart Home,” *International Journal of Smart Home,* vol. 7, no. 1, pp. 115-142, Jan. 2013.

Rowstron and P. Druschel, “Pastry: Scalable, Decentralized Object Location, and Routing for Large-Scale Peer-to-Peer Systems,” *IFIP/ACM International Conference on Distributed Systems Platforms Heidelberg,* pp. 329- 350, 2001.

N. Vlug, F. Fouchal, T. Hassan, S. Firth, B. Fies, K. Ellis, V. Lappalainen, M. Hannus, K. Lindow, T. Buchert, " Deliverable 3.4 Recommendations for New Standards to Overcome Interoperability Barriers," Technical Report TR-3.4 (1.0), Loughborough University, UK., Dec. 2011.

Upnp, “Upnp device architecture version 1.1,” *Upnp Forum,* http://www.upnp.org/specs/arch/UPnParch-DeviceArchitecture-v1.1.pdf. 2008.

OSGi Service platform core specification Official Website, http://www.osgi.org/Specifications/HomePage.

Wu, , C. Liao, L. Fu, “Service-oriented smart-home architecture based on osgi and mobile-agent technology,” *IEEE Trans. Systems, Man, and Cybernetics, Part C: Applications and Reviews,* vol. 37, pp. 193 –205, Mar. 2007.

S. Al Mehairi, H. Barada, and M. Al-Qutayri, "Integration of Technologies for Smart Home Application," *Computer Systems and Applicat*ions, *IEEE/ACS International Conference,* pp. 241,246, 2007.

W. Emmerich, “Software Engineering and Middleware: A Roadmap,” ACM Proc. on the Future of Software Engineering, pp. 117 - 129, 2000.

G. Alonso, F. Casati, H Kuno, and V. Machiraju, *Web Services*. Springer-Verlag, pp. 1-354, 2004.

M. Papazoglou, *Web Services: Principles and Technology*. Pearson–Prentice Hall, pp. 1-782 pages, 2007.

P. Pietzuch, " Hermes: A scalable event-based middleware," Technical Report TR-590 (1476-2986) 590, University of Campridge, Computer Laboratory, United Kingdom, June 2004.

S. Maffioletti, “Requirements for an Ubiquitous Computing Infrastructure”, *unifr.ch*, http://diuf.unifr.ch/~maffiole/Documents/Papers/ParadigmForUbiComp.pdf, 2001.

B. Wuest, O. Drogehorn, K. David, "Framework for middleware in ubiquitous computing systems," Personal, Indoor and Mobile Radio Communications, IEEE, vol. 4, no. 16, pp. 2262 - 2267, Sept. 2005.

S. Carro, P. Antol´ın, F. Olmos, M. Bel, V. Cruz, "SM4ALL: D8.1.d Project Exploitation & Dissemination & Clustering Activities," Technical Report TR- 224332 (2.0), European Commission, Seventh Framework Programme FP7- ICT, July 2012.

E. Warriach, E. Kaldeli, J. Bresser, A. Lazovik, M. Aiello, "Heterogeneous device discovery framework for the Smart Homes," *IEEE*, *GCC Conference and Exhibition (GCC),* pp. 637-640, 2011.

RUNES IST Project Official Website, http://www.ist-runes.org/.

ANGEL project Official Website, http://www.ist-angel-project.eu.

Gator Tech Smart House project Official Website, http://www.icta.ufl.edu/gt.htm.

Gaia project Official Website, http://gaia.cs.uiuc.edu/.

Y. Law, P. Havinga, "Security and dependability for Ambient Intelligence: Informative but busy," *Journal of Ambient Intelligence and Smart Environments*, vol. 3, no. 4, pp. 373-374. Dec. 2011.

R. Baldoni, M. Mecella, "Portfolio of Wireless Sensor Networks Wireless Sensor Networks & Cooperating Objects & Cooperating Objects FP7 Projects," *WSN&CO,* FP7 Projects, ftp://ftp.cordis.europa.eu/pub/fp7/ict/docs/necs/booklet-necs-wsnco_en.pdf. 2008.

Active Campus UCSD Official Website, http://activecampus.ucsd.edu/.

MIT Oxygen project Official Website, http://oxygen.lcs.mit.edu/.

Microsoft’s Easy Living project Official Website, http://research.microsoft.com/easyliving/.

R. Baldoni, C. Aiello, M. Mecella, " D1.1 Project Presentation," Technical Report TR- 1.0, University of Groningen, Groningen, Netherlands, Dec. 2008.

Opportunities for Heterogeneous CPU–GPU Task Scheduling

Christiaan Arnoldus and Robert Witte

Abstract—It is common to exploit the co-processors of modern computer systems to speed up computations which were traditionally done on the CPU. While this is already very common for computer graphical and scientific applications, there is no reason why this cannot be extended to many different kinds of applications.
In this paper we study the current state of general-purpose computing using accelerators, with an emphasis on the everyday user. We discuss several aspects of heterogeneous task scheduling, which becomes a concern when you have many different processors to execute a task on. We also show that there are several frameworks in development to support processor heterogeneity, but most of these are still unsuited for mass adoption due to their experimental or low-level nature.
Besides conjecture we also did performance measurements on our everyday hardware, in order to find out if the promised performance increase is met. We conclude that this is indeed the case. We also take a look at power consumption and show that the fastest solution may not be the most energy-efficient one when heterogeneity is involved. Finally, we discuss the future work necessary to turn heterogeneous task scheduling into a mainstream programming paradigm.

Index Terms—Heterogeneous computing, CPU, GPU, scheduling, task-centric programming.

1 Introduction

Since the beginning of computer development, reaching peak performance has been a goal. In the last decades, we have seen manufacturers increase the computational potential of their hardware by increasing clock rates and combining multiple processors in one chip. Computers were also equipped with specialized co-processors in addition to the traditional central processing unit (CPU), such as: graphics processing units (GPUs), general-purpose graphics processing units (GPGPUs, for example AMD Fusion) and the Cell processor. Systems that contain multiple types of processors are called heterogeneous systems. Heterogeneous systems are common nowadays and include not only PCs, but also smart phones for example. With current development different types of processors can be combined on a single die, which saves space and allows memory sharing between the processors, making them suitable for such mobile devices. In our research we focus on PCs with a CPU and a GPU. Heterogeneous systems can harness a lot of computational power and the challenge is how to use all that power efficiently.

While most applications today only use one type of processor, peak performance can only be achieved when all available processors are used at the same time. This gives rise to the field of task scheduling on heterogeneous systems, that is, scheduling tasks over different types of processors. With heterogeneous scheduling new ways of increasing performance emerge and great results have been booked with heterogeneous scheduling in scientific applications. However, without a standard framework, porting applications to multiple platforms is an issue; rewriting large parts of code severely impairs productivity and development. To tackle this issue several frameworks have been created: the Barcelona Supercomputing Center has created a programming model called OmpSs, Augonnet et al. [2] have created a high-level scheduling framework called StarPU and on a lower level OpenCL and OpenACC exist to support heterogeneous task scheduling. These frameworks are a nice starting point when it comes to heterogeneous scheduling.

We envision that heterogeneous scheduling will become even more common in the future and will be embedded in the operating system or kernel, just like CPU scheduling, thus making heterogeneous scheduling readily available for every application. Our research is aimed at the advantages of heterogeneous scheduling and to test its potential in everyday applications. By everyday applications we understand applications that can be used by all people to support their regular lives and thus excludes specialistic applications that require technical knowledge.

The paper is a review paper and is structured as follows: the following section takes a more detailed look at heterogeneous task scheduling and its challenges. In section 3 we describe the frameworks mentioned to achieve heterogeneous scheduling and take a more detailed look at them. In section 4 we discuss the scheduling algorithms that can be used and present some results that have been booked using these algorithms. In section 5 we present our own experimental results and in section 6 we discuss the platform-dependant development challenges that occur. In the conclusion we answer the following research questions:

1. Is heterogeneous task scheduling a good fit for everyday applications?
2. What future work is necessary to make heterogeneous programming mainstream?

2 Heterogeneous Task Scheduling

Here we take a look at heterogeneous scheduling in a task-driven environment, as is done in StarPU and OmpSs. A task is a part of a computation that can be executed by a processing element (PE). A processing element is a processor or core that can execute tasks. An accelerator is a co-processor specifically designed for one type of computation. Assuming that each task can be computed on every PE, the goal of heterogeneous scheduling is to use all available PEs in such a way that maximum performance is achieved. A few challenges that occur when using heterogeneous scheduling are: distribution of tasks over available PEs, load balancing, task analysis, energy consumption, responsiveness of the system and different hardware architectures.

Distribution of tasks over all available resources is done by using a scheduling algorithm. The heterogeneous scheduler will, depending on the algorithm used, distribute the tasks to the individual PE schedulers. Multiple proven algorithms exist and are in use; more on these algorithms in section 4. At runtime, after distribution of tasks, load balancing should occur; the tasks may not have been distributed fairly regarding complexity and data size or perhaps a certain PE simply performs a lot better than the other. To illustrate this, highly parallel tasks are in general executed a lot faster on the GPU, while tasks with large critical sections are executed faster on a CPU. Data locality should also be taken into account; when the size of the input data is so large that transfer to an accelerator incurs great overhead, or when I/O operations are involved, it may be preferable to execute the task on the

- *Christiaan Arnoldus is a computing science student at the University of Groningen, email: c.arnoldus@student.rug.nl.*
- *Robert Witte is a computing science student at the University of Groningen, email: r.witte.1@student.rug.nl.*

CPU. Hints could be provided by the programmer to increase performance, however it would be better if the system could analyse and evaluate the tasks its own.

When not only the maximum processing capacities of the available PEs are taken into account, but also the ‘nature’ of a task, more efficient scheduling can be done. Heuristic evaluation of previously executed tasks, code analysis and smart compilers all could have a role in analysing tasks for future distribution. But since analysis takes time and processing power, a stand-off has to be made to find an optimum. There is no clear answer yet to what will become the standard scheduling algorithm, but the FCRM results in section 4.4 look promising.

The previously addressed challenges mainly involved performance; to execute the tasks in such a fashion that minimal time is needed. However, this may not be the only motive for scheduling. Energy consumption and overall responsiveness of the system could be goals as well. As is pointed out later in the paper, in section 5, not every PE needs the same amount of energy to execute a certain task, which might lead to an argument from the environmentalists to distribute for energy efficiency. Also, depending on the urge of the task or the needs of the user, maximal load on every PE might may not be needed or wanted. for example, some tasks could be executed on the GPU to save computational power on the CPU for context switching.

And to top it off, besides software challenges, hardware challenges arise as well. Because of the differences in architecture compared to CPUs, accelerators require special programming languages. The most popular type of accelerator is currently the GPU. Because GPUs were initially intended to be used solely for graphical applications, the first general-purpose applications for GPUs were implemented using graphical APIs and programming languages, such as Cg, GLSL and HLSL [4]. The Compute Unified Device Architecture (CUDA) included the first general-purpose programming language for use with GPUs. Since CUDA is only available for NVIDIA GPUs, a demand for a common, open standard emerged. An open standard, managed by the Khronos Group, was eventually established with the name Open Computing Language (OpenCL) [4]. Section 3.1 includes a description of OpenCL. CUDA is omitted, as it does not support heterogeneity by itself, for it is only used to program one specific kind of accelerator. OpenCL and CUDA are compared in section 6.2. For these different kinds of hardware, different implementations are needed, something the programmer has to provide. Besides that, performance may depend on the hardware brand as well, another challenge in the field of heterogeneous scheduling.

There also exist libraries to implement data parallelism directly in traditional programming languages. Besides OpenACC these include: C++ AMP from Microsoft, Bolt from AMD (C++) or Aparapi (Java) [1]. However, these libraries are outside the scope of our research.

3 Frameworks for Heterogeneous Computing

Before taking a look at the frameworks for heterogeneous scheduling, we should take a look at the aim of such a framework, in order to have a better view of each framework. There are several development challenges that a task-centric framework for heterogeneous computing should address. Based on implied requirements in the literature, these include:

1. Support heterogeneity: the framework should support running tasks on different processors, ideally with a common programming language. Most accelerators have their own execution model, which makes writing portable code a major challenge [2].

2. Coherent data management: the framework should provide a way to get a coherent view of all hardware memory banks used by the different processors. As with the previous point, most accelerators have their own data manipulation interface, which is not portable to different systems [2].

3. Scheduling: the framework should be able to dynamically divide tasks over the different processors, so that each of them gets used efficiently.

3.1 OpenCL

A popular framework for heterogeneous computing is OpenCL, which provides a common programming language for CPUs, GPUs and other specialized co-processors. It also provides APIs for explicitly moving data and tasks between co-processors [2]. One of OpenCL’s greatest strength is its broad hardware support [13], even having been adopted by Intel [6] and AMD [1] as their main accelerated programming language, as well as being compatible with NVIDIA hardware [14]. The downside of OpenCL is that its APIs, which are designed to support a large range of hardware, are rather low-level; it does not provide support for task scheduling or global data management [2, 13]. Another disadvantage is that is has a steep learning curve and many performance pitfalls over the different architectures [13].

3.2 StarPU

A high-level framework for heterogeneous computing is StarPU, introduced by Augonnet et al. [2]. StarPU is a tasking API that provides developers with a way to execute parallel tasks over heterogeneous hardware. Tasks are specified as so-called codelets, an abstract representation of a task that can be executed on a PE. A codelet is a description of the task, the data included and whether it requires read and/or write access. Because these tasks are executed asynchronously, StarPU can reorder them to increase performance [2]. The implementation can be provided separately for each architecture which makes StarPU code very portable [2]. In the future, middleware tools, such as programming environments and high performance computing (HPC) libraries, could be built on top of StarPU. This allows programmers to exploit the benefits of heterogeneous scheduling with limited effort [2].

The scheduling of tasks over the different PEs is done with proven algorithms. In addition to choosing a built-in scheduling algorithm, the developer can provide scheduling hints to improve performance and even implement new scheduling strategies where necessary. More details about relevant scheduling algorithms and the performance of those algorithms will be provided in section 4.

In addition to a scheduling library and its portability, StarPU also provides a coherent view of all available memory in the system, relieving the developer of manually moving data between different processors. It uses a caching method to minimize the number of data transfers needed as well as partitioning functions and eviction heuristics to overcome memory limits on co-processors [2].

A disadvantage of StarPU is that is does not take the size of the input data into account when scheduling [11]. It is also the responsibility of the developer to provide implementations for all architectures that a certain task can be executed on, which means that it does not provide a common programming language (although later versions do support OpenCL [14]). Unfortunately, we were not able to get StarPU up and running on our systems due to both library and driver issues and thus have no original results.

3.3 OmpSs

Another approach at easing heterogeneous programming is the OmpSs programming model, created by the Barcelona Supercomputing Center (BSC), as an integration of their previous systems. OmpSs is an extension to OpenMP, which enriches the OpenMP syntax by allowing the developer to define data dependencies between tasks. This allows tasks whose data dependencies have been fulfilled to execute, even if a previously queued task is still waiting for input [11]. OmpSs also introduces directives for specifying on which processors a task can run. The OmpSs programming model is used by Podobas et al. [11] to build a framework for scheduling tasks over heterogeneous hardware, similar to StarPU. Implementations for tasks have to be provided manually for each supported processor, as with StarPU [3]. The performance model of OmpSs is different from StarPU; whereas StarPU assumes that execution time is independent of the content of the data for a certain task, OmpSs does not. OmpSs uses information from the user (whatever that may be) to predict the complexity of tasks that have not been run before [11]. Their argument is that the complexity of a task does not correlate well with the size of the input data for a task.

Unfortunately, we did not get OmpSs to work on our systems either, as there seems to be a bug in OmpSs' MCXX compiler, which caused us to be unable to compile any program. We did contact the BSC for this issue, but they were unable to reproduce it. Using other/earlier versions for MCXX and the runtime library Nanos++ did not resolve the issue.

3.4 Accelerated OpenMP

There is an initiative for extending compilers to support accelerators using compiler directives, in a fashion similar to OpenMP. This initiative is named OpenACC. The OpenMP and OpenACC working groups intend on merging their work to create an extension to OpenMP that supports accelerators [10]. This extension is named Accelerated OpenMP and provides, among other things, the ability to spawn threads on accelerators. The advantage of using and extending OpenMP is that developers use existing knowledge and code [13].

Scogland et al. [13] propose a set of extensions to Accelerated OpenMP to make scheduling in a heterogeneous context easier. One of these extensions allows work to be assigned to specific accelerators. It is also possible to define the performance ratio between different processors, for use by the scheduler. These extension make Accelerated OpenMP similar to OmpSs, although the syntax is incompatible.

4 Algorithms for Heterogeneous Task Scheduling

In this paper we take a look at the following scheduling algorithms:

1. Random
2. Weighted random
3. Work-steal
4. Forecast regression model scheduling (FCRM)

4.1 Random

Random scheduling is an algorithm based on, as the name suggests, random distribution of tasks over the available PEs. This algorithm is very fast and simple to implement. Its simplicity and the fact that it is well-known makes it a good reference point [11].

4.2 Weighted Random

Weighted random is an algorithm in which every PE is randomly given tasks. The difference with the previous algorithm is that the scheduler takes a weight into account for every PE. The weight of a PE is a number that corresponds with how much work can be loaded to a PE with respect to the other PEs. The weight of a PE could be calculated by benchmarks; this is done prior to any scheduling. Benefits of this algorithm are that it is still simple, initialization is very fast and it takes the performance difference of PEs into account.

4.3 Work-steal

Work-steal is an algorithm in which every PE has a separate queue that stores tasks that have to be executed. A PE starts executing these tasks and whenever its queue is empty, it will try to steal a task from the end of another queue and start executing that task. This is a more dynamic approach to scheduling as whenever a PE has finished executing tasks, it will look for new tasks, resulting in load balancing during execution and all threads will finish at around the same moment. The implementation of this algorithm is also simple and initialization is fast.

4.4 FCRM

Forecast regression model scheduling (FCRM) is a scheduling algorithm presented by Podobas et al. [11]. It uses segmented regression to estimate trends concerning the execution time of a task on different PEs. The algorithm will try to estimate how much time a certain task will take on the available PEs and uses roughly same weight system as with weighted random to divide tasks. The main difference here is that this algorithm does this at runtime, unlike weighted random which calculates the weights pre-execution. It is done this way, so the algorithm can also look at how well a certain PE performs at a certain task, instead of only looking at the PE's overall performance. Thus, not only the processing capabilities of a PE are measured, but also which PE is best suited to execute a certain task. Also, FCRM will take the data locality into account, something that the weighted random scheduling algorithm does not [11].

The advantages of this algorithm come from the dynamic learning part and data locality. The algorithm will learn from previously executed tasks and can better estimate the time needed to execute a future task and thus distribute the upcoming tasks in a better fashion when it comes to overall performance (as can be seen in the figures that follow). Because of these heuristics FCRM is a more complex algorithm; it takes more time to initialize and is harder to implement.

The following images are test results taken directly from Podobas et al. [11]. Figure 1 illustrates the performance of FCRM. In fig. 1 it can be seen that the FCRM-preload algorithm performs best overall and weighted-random is a close second. FCRM-preload is a condition in which FCRM has seen every task and can distribute tasks the optimal fashion. This is clearly a highly parallel task as the GPU finishes at the third place. Figure 2 illustrates the learning curve of FCRM. FCRM-preload again performs best, but as long as FCRM still has to evaluate a lot of tasks and execution times, it performs significantly worse. More detailed information about test setup, environment and results can be found in the paper of Podobas et al. [11].

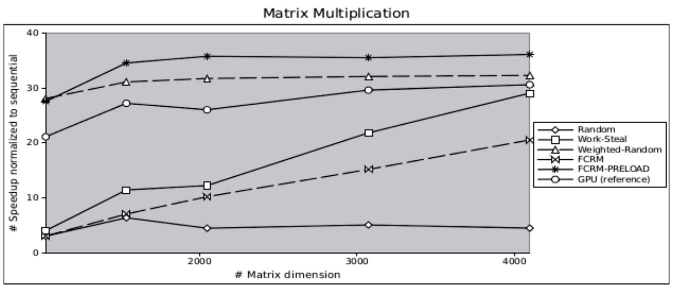


Fig. 1. FCRM performance, by Podobas et al. [11]

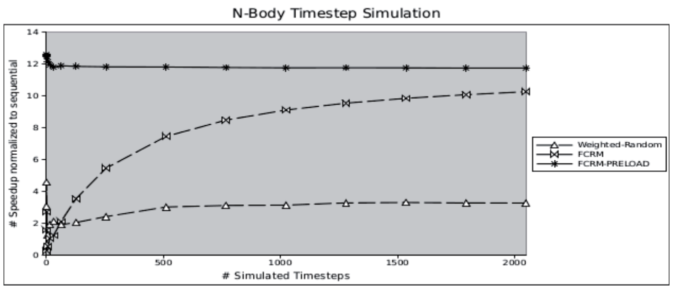


Fig. 2. FCRM learning curve, by Podobas et al. [11]

4.5 Comparing Algorithms

The four presented algorithms used to implement scheduling policies have its benefits and pitfalls. The first three algorithms are very simple to implement, perform reasonably well and have fast initialization, as seen fig. 1. However, they do not take the nature of tasks or data locality into account. FCRM is the most complete and complex algorithm presented, but it comes at a cost of initialization speed.

As this is a field that is still very new, not many applications are available that we can use for testing heterogeneous scheduling algorithms. What we can say about the use of these scheduling algorithms in everyday applications, is that many everyday tasks will be repeated; take applications for school, games, graphical editor programs, movie editing or converting, and so on. These are applications that could benefit from learning, as with FCRM [11]. Take into account though that applications that have a very short computation time will not benefit much from this; the overhead of sending data from main memory to the GPU or another PE might take more time than the actual calculation. However, we do see a benefit for more time-consuming applications or applications in which other qualities can be improved by heterogeneous scheduling. Especially for highly parallel tasks; heterogeneous scheduling can significantly increase performance of the system. The test results of the following section support this.

5 Experimental Results

In order to find out whether adding support for heterogeneity to our everyday applications is worth it, we have executed several performance tests on our hardware, which are described in this section. These tests do not fit our definition of everyday applications, as accelerated implementations of such applications are still rare. What we are most interested in is how these benchmarks perform on everyday hardware, like integrated graphics processors (IGPs).

While time is an important aspect of performance, often *the* most important aspect of performance, it is not the only aspect. Power consumption may also be a concern in a world where awareness of environmental issues increases. Mobile devices like laptops and smart phones increasingly feature accelerators as well and these devices benefit from lower energy consumption, as this increases battery life. Energy measurements have been included in section 5.3 to show how energy consumption can influence scheduling decisions.

5.1 Experimental Setup

For our performance measurements we used two different setups, one AMD machine and one Intel/NVIDIA machine. These names are chosen for convenience and do not imply that Intel requires NVIDIA, AMD excludes NVIDIA or some such. These machines are low- to mid-range and may very well correspond to what an everyday user has on his desk.

The AMD machine has an AMD Phenom 9650 CPU and an ATI Radeon HD 4850 GPU. The tests run on this machine focus on OpenCL, as CUDA is an NVIDIA-only technology. Several OpenCL example kernels are available from the AMD Accelerated Parallel Processing (APP) SDK [1], which can be run on both CPUs and GPUs.

The Intel/NVIDIA machine is a relatively old notebook (2009) which runs a dual core 2.0 GHz processor and has a GeForce 9200M GS graphics card, equipped with eight CUDA cores. Unless specified otherwise all available resources were used.

5.2 Simulating Physical Bodies

Although NVIDIA comes with a large set of example applications in their demo/sample suite, very few of the applications are runnable on both the CPU and GPU, which would support our research about heterogeneous scheduling. One application that is runnable on both the CPU and the GPU is a simulation of *n* physical bodies, called nbody. Nbody numerically approximates the evolution of a system of bodies in which each body continuously interacts with every other body. A graphical representation of the example is available, which shows that simultaneous computation and visualization is feasible with general-purpose GPU computations.

The tests were performed in an everyday situation; multiple applications were running, including a browser, a text editor and a music player. Three different input values were used and for each 10 iterations were calculated and the built-in benchmark tool was used to measure performance. While we would have liked to test the double-precision performance benchmark as well, this CUDA device does not support that.

Table 1. Results of running a single-precision nbody simulation on our INTEL/NVIDIA machine.

	CPU		GPU	
Input size	Total time	Perf.	Total time	Perf.
1024	0.807364	0.260	0.015285	13.720
2048	3.225992	0.260	0.060828	13.791
4096	12.920963	0.260	0.240246	13.967

Although nbody is not an everyday application, but it is very useful to show that the speed-up achieved when using the GPU at the right time can be enormous, even on such a weak GPU. This can be seen in table 1, which compares the computation time and performance. Timings are in seconds and performance is measured in GFLOP/s. Table 2 shows the speed-up of the GPU when compared to the CPU.

Table 2. Speed-up of running nbody on our INTEL/NVIDIA machine.

Input	Speed-up
1024	52.82
2048	53.03
4096	53.78

5.3 Ray Tracing

Our second benchmark is provided by ratGPU [12], which is a standalone renderer which uses OpenCL and optionally a C++ implementation on the CPU. RatGPU benchmarks the system by ray tracing four scenes. RatGPU gives us the option to use the CPU, GPU or both at the same time to do the ray tracing. We show the computational speed differences between using merely a GPU (a very simple one in this case) and using both the CPU and the GPU; thus using some form of heterogeneous scheduling. While ratGPU lacks documentation on its inner workings at the time of writing, the visualization suggests that the image is decomposed in rectangular areas, which are scheduled over the CPU and GPU in a greedy fashion.

We were surprised by the results on our INTEL/NVIDIA machine. We expected the time needed by the GPU to ray trace the images to be close to the time needed by a combination of CPU and GPU, as ray tracing is a highly parallel task and thus very much suited for execution on a highly parallel processor. However, this was not the case, as can be seen in table 3. This is probably the result of using a weak GPU. Timings are in seconds and we used the timing mechanism provided by ratGPU.

Table 3. Results of running ratGPU ray tracing on our INTEL/NVIDIA machine.

Mode	Time	Speed-up
GPU	3257.99	1
CPU (C++)	1908.93	1.707
GPU + CPU (C++)	1820.32	1.790

As mentioned, speed may not be the only relevant performance metric. We also measured the average power consumption of our INTEL/NVIDIA machine using a tool called powerstat [5] and included the results in table 4. The total energy consumption during the whole computation was calculated and included as well. While stressing the GPU consumes less power than stressing the CPU and using both results in the highest speed, it turns out that using just the CPU costs the least energy, so that would be the smartest choice if conserving your laptop battery is the highest priority. Note that, while this particular example does not favour scheduling tasks to the GPU, in the previous benchmark the GPU performs better on every level and scheduling tasks to the GPU would likely save power *and* increase performance.

Table 4. Energy consumption of our INTEL/NVIDIA machine while ray tracing.

Working state	Power (W)	Energy (10^4 J)
Idle	16.92	NA
GPU under load	35.75	11.647
CPU under load	41.10	7.846
GPU + CPU under load	48.56	8.839

We did the same ray-tracing benchmark on our AMD machine. The results of this benchmark are included in table 5, where the speed-up is relative to the time needed by just the GPU. In this case, using a more powerful GPU, the results are more akin our expectations. It is interesting to note that running the OpenCL implementation on the CPU is not much slower than using the C++ implementation. We did find a strange slowdown when using two OpenCL devices at the same time; since this does not occur when using the C++ CPU implementation we suspect this is the result of a software bug, either in ratGPU or AMD's OpenCL driver. No power consumption results are available for the AMD machine, as the tools available did not work for a desktop PC (battery versus net-power).

Table 5. Results of running ratGPU ray tracing on our AMD machine.

Mode	Time (s)	Speed-up
GPU	275.773	1
CPU (C++)	1470.54	0.19
CPU (OpenCL)	1557.3	0.18
GPU + CPU (C++)	256.65	1.07
GPU + CPU (OpenCL)	711.034	0.39

5.4 Image Processing

One possible everyday application which can be supported by heterogeneity is image processing, which people may use from their photo editing package. Image processing often entails applying filters to images and the AMD APP [1] contains several OpenCL kernels which can be used for that purpose. We have measured the performance of these kernels using our AMD machine discussed in section 5.1, in particular kernels for applying Gaussian noise, Sobel edge detection and recursive Gaussian blur.

The results for these measurements are included in tables 6 to 8, which were acquired using the timing mechanisms that were shipped with the samples. When run on the CPU, the kernels make use of all available cores. Three different image sizes were tried, the other settings retained their default values. The time values are given in seconds and are the result of the average of 100 iterations. The kernel time is the time taken by running the actual kernel and moving data back and forth between the device, while total time includes the *setup time*, which is the time required to compile the kernel and allocate memory on the device.

Table 6. Results of running a Gaussian noise kernel on our AMD machine.

	CPU		GPU	
Input size	Kernel time	Total time	Kernel time	Total time
1024^2	0.0440572	0.327617	0.0217584	0.488877
2048^2	0.187817	0.46065	0.082306	0.548544
4096^2	0.741743	1.55724	0.313115	0.747155

Table 7. Results of running a Sobel edge detection kernel on our AMD machine.

	CPU		GPU	
Input size	Kernel time	Total time	Kernel time	Total time
1024^2	0.0524023	0.271538	0.0199078	0.265988
2048^2	0.227918	0.462058	0.0748007	0.326518
4096^2	0.93972	1.18023	0.283620	0.499277

The tables 6 to 8 show that the GPU is consistently faster in the chosen set of algorithms, although the speed-up may not be significant when the image size is small, especially when one factors in the setup time. The setup time may not be relevant though, since a compiled kernel and allocated memory may be re-used if the same kernel is executed multiple times during the same run of the program. As one can tell from the speed-up matrix of table 9, the speed-up is more or less independent of the input size. The speed-up is not independent of the algorithm however, as the Gaussian noise and Sobel edge detection algorithms only gain a modest speed-up, while the speed-up for the recursive Gaussian blur is much larger. We cannot explain why the speed-up is so much larger in this case, but it may be the result of an OpenCL performance pitfall, as noted in section 3.1 or may be because the other algorithms are less suited for execution on a GPU.

Table 8. Results of running a recursive Gaussian blur kernel on our AMD machine.

	CPU		GPU	
Input size	Kernel time	Total time	Kernel time	Total time
1024^2	1.27465	1.97877	0.0331012	0.282886
2048^2	5.15031	5.37706	0.129927	0.410904
4096^2	21.2629	21.4885	0.531589	0.763503

Overall, the absolute time values are not so impressive, as most are below a second and it is unlikely that an everyday user would notice the difference. But it should be noted that the speed-up becomes more important when the input data becomes larger; even if the speed-up is only two, waiting one hour is certainly preferable over waiting two hours. Enabling the GPU for everyday applications would also be good for exploration, as the computer can remain responsive while the GPU is busy, while responsiveness may suffer if the CPU is busy. It is also worth noting that, in some cases like our recursive Gaussian blur experiment, the GPU may be able to provide near-real-time feedback while the user adjusts the parameters at an image size where the CPU may not be able to do so. When both computation and visualization happen on the GPU, it may no longer be necessary to move data between main and video memory [9].

Table 9. Speed-up matrix of our AMD machine.

Input size	Noise	Edge detection	Recursive blur
1024^2	2.02	2.63	38.5
2048^2	2.82	3.04	39.6
4096^2	2.36	3.31	40.0

6 Technical Discussion

When it comes to accelerators, including but not limited to GPUs, several technologies exist on both the hardware and the software side. On hardware side, there are solutions offered by Intel, AMD and NVIDIA. On the software side, the most common technologies to program accelerators are CUDA (favoured by NVIDIA) and OpenCL (favoured by the rest). This section discusses the experimental results and the influence of manufacturers and their technologies on the field of heterogeneous task scheduling.

6.1 Intel versus AMD versus NVIDIA

There are several manufacturers who ship GPUs; Intel currently holds the largest market share, followed by AMD, while NVIDIA comes third [7]. It should be noted that these figures include IGPs, which are integrated in the motherboard or CPU of the computer. IGPs are more common than discrete graphics processors, but generally less powerful and have less features, making them somewhat less attractive for the applications discussed in this paper. Since Intel does not manufacture discrete graphics processors and NVIDIA has stopped manufacturing IGPs [7], the market shares for discrete GPUs are very different, with NVIDIA taking the lead and AMD coming second [8]. Other GPU manufacturers (VIA and Matrox) only have a negligible market share [7].

While NVIDIA may not have the largest total market share of graphics processors, they do have the largest market share of discrete GPUs, which may be the reason that they and their CUDA execution model gained the favour of the scientific community. Both OmpSs [11] and StarPU [2] preferably use CUDA as a back-end for GPU support; in both cases OpenCL support was only added as an afterthought [3, 14] and is still experimental in nature, even more so than the rest of the frameworks.

So, right now, NVIDIA hardware seems to be the safest choice if one wants to do general-purpose GPU computing. But, in order to become truly mainstream, heterogeneous computing technologies have to support a wide range of hardware, including AMD hardware and possibly the Intel HD series IGPs, which support OpenCL [6]. For this to happen there must be a programming standard shared between all processors (for example OpenCL) or frameworks must be able to transparently switch between programming models.

6.2 OpenCL versus CUDA

Both OpenCL and CUDA are used to implement small programs to be run on an accelerator, known as *kernels*. The main difference between OpenCL and CUDA is their hardware support, as noted in section 3.1. Other than that, the architectures assumed by OpenCL and CUDA are very similar, as are the syntax, keywords and built-in functions of the two languages. As a result, it is not too difficult to port a kernel from CUDA to OpenCL [4].

Fang et al. [4] made a comprehensive performance comparison on CUDA and OpenCL. They found that CUDA may be up to 30% faster than OpenCL, but also noted that these performance gaps are usually the result of a programming pitfall or because the CUDA compiler is older and therefore better optimized. There is no reason why OpenCL should perform worse than CUDA and in fact it does not when kernels are properly optimized. These optimizations are often platform-dependant though, which diminishes OpenCL's advantage of broad hardware support a little.

7 Conclusion and Future Work

In the introduction we asked ourselves two questions. The first question was whether heterogeneous task scheduling is a good fit for everyday applications. In section 5 we showed that the accelerators that the everyday user has available can already be used to do certain tasks and achieve at least similar, but usually much better, performance compared to the CPU. Whether scheduling will play a major role in future everyday heterogeneous systems remains to be seen, but we envision it will. As described in section 2, choosing the most suited PE for a certain computation is a complex problem and it is unlikely that the developers' choice results in the best performance in all circumstances.

As for the second question, we must conclude that there is still a lot of work to do before heterogeneous task scheduling can become mainstream. The frameworks discussed in section 3 are still too experimental in nature for mass adoption, as their support for different hardware and software configurations seems limited. If two computing science students cannot get your software to work, it is unlikely that the average computer user can, so future work in this area is warranted. The frameworks also have a bias towards NVIDIA technologies, as discussed in section 6, which is also undesirable. If the frameworks are adapted to support more open technologies, developers can integrate heterogeneous task scheduling in our everyday applications.

Acknowledgements

We wish to thank Dr Valeriu B. Codreanu for his feedback and willingness to help with technical issues. We also thank our fellow students that reviewed our draft paper.

References

[1] Advanced Micro Devices. Accelerated Parallel Processing SDK. URL http://developer.amd.com/tools/heterogeneous-computing/amd-accelerated-parallel-processing-app-sdk.

[2] C. Augonnet, S. Thibault, R. Namyst, and P.-A. Wacrenier. StarPU: a unified platform for task scheduling on heterogeneous multicore architectures. *Concurr. Comput.: Pract. Exper.*, 23(2):187–198, February 2011. ISSN 1532-0626. doi: 10.1002/cpe.1631.

[3] Barcelona Supercomputing Center. The OmpSs Programming Model. URL http://pm.bsc.es/ompss.

[4] J. Fang, A. L. Varbanescu, and H. Sips. A Comprehensive Performance Comparison of CUDA and OpenCL. In *Parallel Processing (ICPP), 2011 International Conference on*, pages 216–225, 2011. doi: 10.1109/ICPP.2011.45.

[5] 'Gayan'. powerstat: Power Consumption Calculator for Ubuntu Linux. URL http://www.hecticgeek.com/2012/02/powerstat-power-calculator-ubuntu-linux.

[6] Intel Software. Intel Developer Zone: Intel SDK for OpenCL Applications. URL http://software.intel.com/en-us/vcsource/tools/opencl-sdk.

[7] Jon Peddie Research. AMD, Intel, and Nvidia all down in Q4 with negative 8% overall quarter-to-quarter, 2013. URL http://jonpeddie.com/press-releases/details/amd-intel-nvidia-q4-graphics-gpu-shipments.

[8] Jon Peddie Research. Graphics add-in board shipments crash from last quarter, 2013. URL http://jonpeddie.com/press-releases/details/add-in-board-report-Q4-2012-crash-down.

[9] Khronos Group. OpenCL 1.2 Reference Pages: clCreateFromGLTexture. URL http://www.khronos.org/registry/cl/sdk/1.2/docs/man/xhtml/clCreateFromGLTexture.html.

[10] OpenACC working group. OpenACC: Directives For Accelerators. URL http://www.openacc-standard.org.

[11] A. Podobas, M. Brorsson, and V. Vlassov. Exploring Heterogeneous Scheduling using the Task-Centric Programming Model. 2012.

[12] Santiago Orgaz. ratGPU standalone renderer. URL http://www.ratgpu.com.

[13] T. Scogland, B. Rountree, Wu-chun Feng, and B. de Supinski. Heterogeneous Task Scheduling for Accelerated OpenMP. In *Parallel Distributed Processing Symposium (IPDPS), 2012 IEEE 26th International*, pages 144–155, May 2012. doi: 10.1109/IPDPS.2012.23.

[14] University of Bordeaux. StarPU. URL http://runtime.bordeaux.inria.fr/StarPU.

Curve Skeletonization of 3D Shapes

P.M.D. Otterbein, M.C.P.M. Scheepens

Abstract—The extraction of curve skeletons of 3D shapes is a fundamental problem with many different applications. These include for example virtual navigation, animation or visualization improvement. Because there is such a broad interest in this field, a lot of different techniques have been developed over the years. Although skeleton extraction techniques have roughly the same purpose, their fundamental idea can be very different. To make a comparison we study the different properties of the techniques and compile a list of criteria. Based on that list we have evaluated each method to create a comparison between them. With the comparison it is possible to distinguish between the different techniques in such a way that a appropriate technique for specific needs can be identified.

Index Terms—Curve skeletonization, 3D shapes, 1D representation, transformation, volumetric methods, geometric methods, surface and object representation, comparison.

1 Introduction

A curve skeleton is a 1D structure which represents a 3D object in a simplified way. They are used in many applications which require a representation of objects geometry and topology. These applications are typical for the field of computer aided design, medical imaging, computer graphics, scientific visualization, computational fluid dynamics, and remote sensing.

Because there is such a broad interest in this field, a lot of different techniques have been developed over the years. Although skeleton extraction techniques have roughly the same purpose, their idea and design can be very different. For example they can either work with geometric methods or with volumetric methods. Geometric methods work with polygon meshes or point sets and volumetric methods have regularly partitioned voxelized representations or discretized field functions as input data. Also the algorithmic complexity has increased to a point that analytical reasoning is not enough to relate the results of a method to desirable skeleton properties. This leads to the question: how can the different methods be compared and which method should be used for a specific problem?

To solve this, we studied existing literature to find out what criteria for skeletonization there are, like robustness, reconstruction properties or connectivity [1]. This resulted in a list of important criteria which are discussed in section 2. There we compared these criteria by identifying which important properties they represent and classify them in contradicting properties or similar properties. Based on this identification a list can be created with relevant criteria for skeletonization method comparison.

The next step was to study existing skeletonization techniques, how they work and what characteristics the results show. We also investigate advantages and disadvantages of them, like computation complexity. This is described in section 3.

Based on our criteria we created a comparison of the different skeletonization techniques. This comparison is shown and discussed in section 4. The discussion shows that is possible to distinguish between the different techniques in such a way that a appropriate technique can be identified with a set of criteria which are important for a specific application.

Finally the conclusions of the findings are discussed in section 5.

• *P.M.D. Otterbein is student at the Rijksuniversiteit Groningen, E-mail: P.M.D.Otterbein@student.rug.nl.*

• *M.C.P.M. Scheepens is student at the Rijksuniversiteit Groningen, E-mail: M.C.P.M.Scheepens@student.rug.nl.*

2 Criteria

In this section we will present different properties of curve skeletons. These properties indicate the quality of the method on the one hand and the significance for a comparison on the other hand. After a general description of each property, they will be discussed under these two aspects. Not all properties which will be discussed are on the final criteria list. Whether a property is a final criterion or not and why it is or isn't is also part of this section.

The following criteria are final comparison criteria: Homotopy (2.1), invariant under isometric transformations (2.2), thinness (2.4), centered (2.5), junction detection (2.7), robustness (2.9), smoothness (2.10), detail preserving (2.11), hierarchic (2.12) and efficiency (2.14).

2.1 Homotopy

The curve-skeleton should be topologically equivalent to the original object [2]. An object is topologically equivalent to another object if it has the same number of connected components, tunnels and cavities. Because a curve skeleton is a 1-D representation of the original objects it cannot preserve cavities. That is why they are not a relevant indicator for homotopy in our case.

The homotopy of a skeleton is a direct indicator for the quality. If a skeleton is topologically preserving, it is also a good representation of the original figure. This can be seen in figure 1. As a main quality indicator it is also a good aspect to compare skeletons. No matter which method is used, the result can always be compared to the original under this aspect.

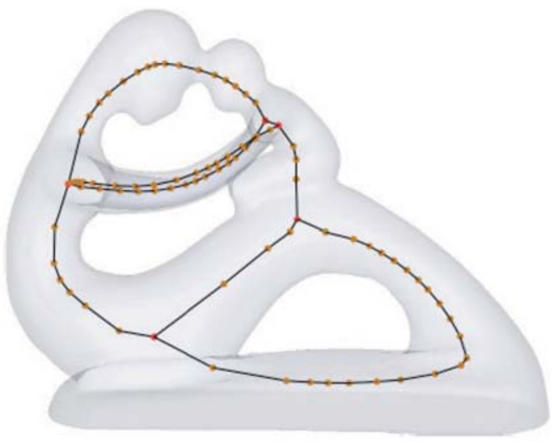


Fig. 1. Example an homotopic skeleton which has been produced by the 'Telea and Jalba' method. The skeleton is topologically equivalent to the original object which shown light grey in the background. Image from [3]

2.2 Invariant under Isometric Transformations

If a methods yields the same result after the figure has undergone an isometric transformation, for example a rotation, as the transformed result of the original figure, it is invariant under isometric transformations. This is very important for skeletonization as the orientation of the figure should not alter the skeleton itself.

This property can easily be checked for each method, so it is very applicable for comparison.

2.3 Reconstruction

Reconstruction of the original object from the curve-skeleton is possible for example by computing the union of the maximal inscribed balls at each curve-skeleton point [4]. The results of the reconstruction then can be compared for each method. This is based on assumption that the reconstruction result says something about the quality of the skeleton. This however is not the case since reconstruction is impossible for some the best shape descriptors [5].

This fact excludes reconstruction from the comparison factors.

2.4 Thinness

Curve skeletons should be one dimensional. Or in other words as thin as possible. The thinness depends on the extraction method. For voxel-based methods the thinness of the result should be one voxel. For mesh-based methods the result should be polyline skeleton [3].

This is the general principle of skeleton extraction and a important property of skeletons. That makes thinness a good comparison criterion.

2.5 Centered

Whether a skeleton is centered or not is a important characteristic of curve-skeletons. The centeredness shows how representative a skeleton is with respect to the original object surface. The problem is the definition of it. When is a skeleton centered and when not? A very loose definition would be if the skeleton lies within the object boundaries it is centered. A very strict definition would be if skeleton lies centered on the medial surface of the corresponding surface patch [6][7]. Since in most cases exact centeredness is not required the approximated middle of the two definitions is appropriate.

Centeredness can be checked visually and differences can be compared by that, as seen in figure 2. Although the definition is vague and most applications do not require exact centeredness, the property is a useful indicator whether a skeleton is roughly centered or not.

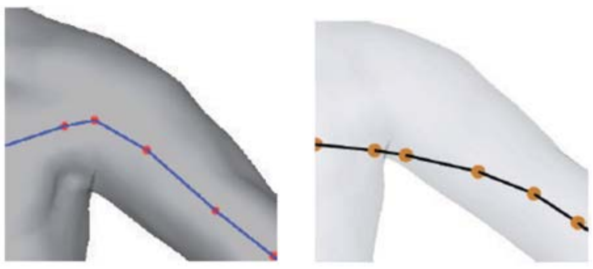


Fig. 2. Left: Example of a centered skeleton which has been produced by the 'Au et al.' method. The skeleton is almost perfectly in the middle of the shoulder. Right: Example of an off-centered skeleton which has been produced by the 'Au et al. (surface skeleton)' method. The skeleton is constructed lower than in left picture. Images from [3]

2.6 Reliability

A method is reliable if every point on the object surface is visible from at least one point on the curve-skeleton. This property is application specific and important for example for virtual navigation [8].

It can be tested for example by checking for every surface point whether it is visible or not and then compare the amount of visible points. But as high reliability is only needed in very specific applications it does not qualify as a general comparison criterion.

2.7 Junction Detection

Methods can be divided into two classes: Methods that can detect junctions before or during the extraction [9] and methods that detect joints after the extraction [10]. Detecting joints after the extractions is trivial because junctions are skeleton points which have more than two neighbors. However these junction might have no significance with respect to the original shapes. If the skeleton is used for animation it can be the case that the skeleton joints should correspond to object joints to animate the object based on the skeleton.

Given the fact that junction detection is such a dividing factor, it is added to the list of criteria.

2.8 Connected

Whether there the resulting skeleton has the same connectivity properties or not, is a consequence of the homotopy of the method. So a single connected object should yield a single connected skeleton, in that case the skeleton is connected.

This property can be derived from the homotopy so it is not added to the list.

2.9 Robustness

A method is robust if it is insensitive to noise. That means a noise-free object and the same object with noise should yield a similar result. Robust methods should also produce equivalent skeletons for different resolutions, an example for this is shown in figure 3. The robustness partly counters the centeredness as perfectly centered skeleton only depends on the medial surfaces which would make it very sensitive to noise on the boundary surface.

Robustness can be very important, for example in medical applications and it is important to know whether a method is robust or not. For these reasons it appears on the list.

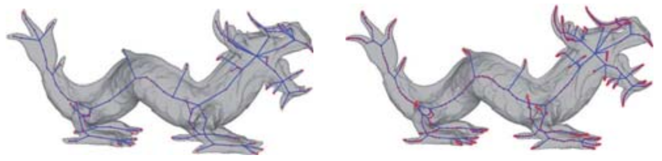


Fig. 3. Left: Skeleton which has been produced by the 'Au et al.' method with 14k image points. Right: Skeleton with the same method with 231k image points. The two skeletons do not differ very much thus the method is robust. Images from [3]

2.10 Smoothness

If the resulting curve-skeleton of a method is smooth, it does not contain any discontinuities or extreme changes in curvature which are introduced by the sampling of the input or the representation of the skeleton itself. In other words, smoothness is a indicator of how neat the skeleton is produced, as seen in figure 4.

This is not only useful from an aesthetic point of view but can also carry importance in applications that need high quality skeletons. Because it introduces an additional quality aspect it is added to list of final criteria.

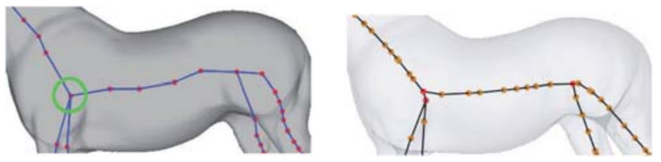


Fig. 4. Left: Example of a skeleton with an introduced change in curvature which has been produced by the 'Au et al.' method. The green circle indicates the part where a curve has been falsely introduced into the skeleton. Right: Example of a smooth skeleton which has been produced by the 'Au et al. (surface skeleton)' method. The skeleton does not have any false curves. Images from [3]

2.11 Detail Preserving

A skeleton extraction method is detail preserving if it captures details of the model like bumps and small cavities. It is desirable that the amount of details captured can be adjusted by the user so it matches the requirements of the application. Theoretically this can be also used as a noise filter [3] because if almost no details are captured, the noise in that regions is also ignored. A good configuration is shown in figure 5.

This property makes the skeleton extraction more adjustable and it is important to know if a method is capable to do it or not. That is why it is on the list of final criteria.

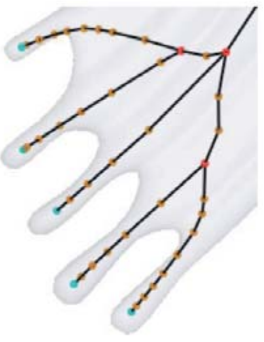


Fig. 5. Example of a detail preserving skeleton which has been produced by the 'Telea and Jalba' method. All details are captured well and the skeleton reaches into all toes of the frog foot. Image from [3]

2.12 Hierarchic

If a method supports hierarchic relations, relations between skeletons and/or joints are defined [11]. So for example if the shoulder joint is rotated all the joints of the arm are translated to the corresponding position. Or maybe with multi-resolution rendering the highest resolution skeleton should have the property that it contains all the points of the lower resolution skeletons.

It is very important for a lot of applications in computer animation and visualization, so whether hierarchy is supported or not is selected as a criterion.

2.13 Point Sets

Points sets are a description of a 3D object which do specify anything about the connectivity or any other internal or external information. This depends on the method itself, so if a method needs such informations, for example all thinning methods, it cannot handle point sets.

We can conclude that if a method requires any internal or external information of the original object it cannot handle point sets as an input. Whether this property holds or not can be derived from the method, that is why it does not appear on final list.

2.14 Efficiency

Not only the quality of the skeleton is important also the time which is needed to extract it can play an important role. If real-time computation is needed methods with a low time complexity are more likely to be chosen over methods with a high complexity, even if the results are not as good as they could be.

This is an important property which should be known for any program or algorithm, just to get an indication of computation time which will be needed. We consider it is also as a comparison criteria.

3 METHODS

The methods to extract curve skeletons from a 3D shape can be classified into four groups: thinning and boundary propagation, distance field methods, geometric methods and general field methods. The four groups are explained detailedly in this section.

3.1 Thinning and Boundary Propagation

Thinning is an iterative and discrete process, in every step voxels from the boundary are removed. The process stops if the required thinness is obtained. To determine whether a voxel is to be removed, the algorithm determines if the voxel is a simple point. A simple point has the property that removal of a simple point does not affect the topology of the object [2]. Another important property is that the calculation for determining a voxel as simple point is done locally, thus thinning algorithm are fast.

The process of thinning starts at the boundary of an object and stops when all simple points in the object are removed. To determine whether a voxel is a simple point, masks are applied to the boundary voxel. The mask is usually of size 3x3x3 and contains conditions for the topology. Figure 6 shows the process of thinning for a 2D object.

A minor problem are the end points of curve skeletons. These points are classified as simple points, thus it is necessary to implement additional conditions to preserve the end points.

There exist several variations on the thinning method based on the detection and removal of simple points.

In Directional Thinning the algorithm is only sensitive to voxels at a particular site, the algorithm is than repeated for other directions. Directional thinning is sensitive to the order of these directions and therefore not rotation invariant. Also the resulting skeleton may not be centered within the object.

Subfield sequential thinning methods create a set of subspaces (or subfields), these subfields are iterated independently.

The last variation are fully parallel algorithms. These algorithms inspect more than just the 26 local neighbors and can remove all unnecessary boundary points in one iteration.

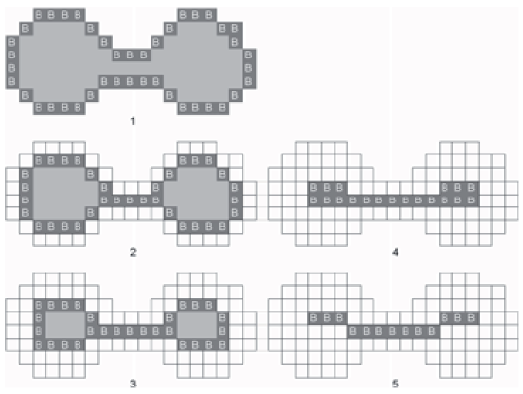


Fig. 6. The thinning process on an example 2D shape. Boundary points are marked "B" at the beginning of each iteration and then removed if they are simple. Image from [1]

3.2 Distance Field

The distance field is the smallest distance from interior point P to the boundary B(O) of a 3D object O: $D(P) = \min_{Q \in B(O)}(d(P,Q))$. An example for the distance metric d is Euclidean distance. Most algorithms based on a distance field use the following steps to calculate a skeleton: find ridge points (local maximums, saddle points) in the distance field), prune insignificant extreme points and finally re-connect disconnected points.

The distance field helps to find locally centered voxels. The locally centered voxels are candidate voxels for the skeleton and can be found on the ridges in the field. Several methods exist to find the ridges: distance ordered thinning, gradient searching, geodesic front propagation, divergence computation, parameter controlled thinning or surface shrinking.

The set of candidate voxels obtained by these methods can be quite large and contains many insignificant voxels. During the pruning step several methods can be used to sift the collection: thinning, sphere coverage, boundary visibility or clustering.

Pruning results in disconnected voxels. In order to create a set of 1D curves we need to reconnect these voxels. Mostly used are well-known graph algorithms like minimum spanning tree or shortest path.

The distance field methods can extract the medial surface accurately, but can have difficulties in finding the curve-skeleton. The most important advantage of distance field based methods are the low computation time and the possibility to recycle the distance field for other applications.

3.3 Geometric Methods

In the case that an object is represented by a mesh or a sets with points geometric methods can extract the skeleton from the object.

The *Voronoi diagram* needs a mesh vertex description of the object. Based on the mesh vertex the Voronoi algorithms calculates regions that are closest to the generator element. The edges of the Voronoi diagram are an approximation of the medial surface of the edge. By using a special erosion on the obtained Voronoi diagram we get the curve skeleton.

It is also possible to fill the object with *cores*. The spread of a core is maximal, the location and the radius are stored. The locations of the center points can be used for the skeleton.

The methods described above are classified as medial-axis based methods. Medial axis-based methods are very sensitive to noise and need much computational time. Furthermore these methods do not generate a skeleton but a medial surface. However, when a curve thinning algorithm is applied, it is possible to derive the curve skeleton.

The *Reeb graph* is a method where the resulting graph has to be mapped into 3D space, since the result is not a curve-skeleton and is not in object space. The 1D structures in a Reeb Graph correspond to critical points in the function that describes the gradient of the structure.

3.4 General Field Functions

General field functions are similar to distance field functions. Instead of a distance measure field function a field measures from physics are applied: potential field, electrostatic field or repulsive force functions. The potential force is usually determined as the sum of potentials generated by the boundary of the object.

The curve skeleton is extracted similar to the distance field methods: detect the local extreme points and connect. The extreme points can be found in the calculated vector field for the force. In order to connect detected extreme points a force following algorithm can be applied.

General field algorithms produce smoother curves for the skeleton than distance based algorithms. Distance field only determines the distance to the smallest distance to the boundary of the object, general field receives input from more points on the boundary. The smoother lines are considered an advantage over distance field algorithms. Furthermore general field is less sensitive to noise, it uses more points as input and thus can blur noise. A disadvantage is that general field is computationally more expensive than distance field.

4 Discussion

In this section we discuss the performance of the various methods based on the ten properties we chose in the criteria section.

4.1 Thinning and Boundary Propagation

Thinning methods are homotopic. Simple points are defined as points which do not affect the topology and thinning only removes voxels with this property.

As mentioned earlier, directional thinning is sensitive to transformations. That means that the order of the directional transformations in the method influences the result.

Thinning does not guarantee that a thin (1D) result is created. In the case that an even number of points exist, most variants stop when the skeleton has a width of two voxels. For example with two voxels removal the removal of the last simple points would mean that no voxels would remain, so two voxels are the end-state.

Centeredness is not guaranteed by thinning methods.

Some thinning methods can directly detect junctions, others use a post-processing step.

Thinning is not a robust method since it very sensitive to noise.

Implementations of that method cannot deliver smooth skeletons, because of the discrete input and the sensitivity to noise.

Thinning methods remove lots of fine details, thus do not preserve details. Furthermore the discretization can create artifacts in the reconstruction.

Thinning methods cannot store hierarchical information, they either remove or keep a pixel.

The time complexity of thinning depends on the number of input voxels, it is almost linear. Most voxels are removed as simple points, remaining voxels will return in every iteration.

Criterion	Method performance
Homotopy	yes
Invariant	most variants
Thinness	no
Centered	some variants
Junction Detection	some variants
Robustness	no
Smoothness	no
Detail preserving	no
Hierarchic	no
Efficiency	good

Table 1. Performance of thinning methods on the criteria

4.2 Distance Field

Distance field methods themselves have no positive or negative homotopic property or guarantee thinness, since they do not extract a curve-skeleton. The next steps in the process, pruning and re-connecting, decide whether the result is homotopic and thin or not.

The invariance under isometric transformations is shown for distance fields methods.

Centeredness of the skeleton is achieved only by some variants of the algorithms, it depends on the re-connecting of the candidate points.

Junction detection is only possible when special post-processing is applied, the results from this process are sensitive to noise.

Also due to the sensitivity to noise, all Distance field methods are not considered robust.

The distance field methods do extract smooth skeletons, appropriate post-processing may introduce smoothness.

Details are preserved well by distance field algorithms, but the low robustness can also introduce noise in the reconstruction.

Distance field methods can built a hierarchy of skeletons during the pruning step, the number of selected voxels during pruning can be adjusted, thus enabling to store the hierarchy.

The complexity of the distance field algorithm is linear. The next processing steps can have a higher complexity, but work with a small set of points.

Criterion	Method performance
Homotopy	not applicable
Invariant	yes
Thinness	not applicable
Centered	some variants
Junction Detection	some variants
Robustness	no
Smoothness	some
Detail preserving	yes
Hierarchic	yes
Efficiency	good

Table 2. Performance of distance field methods on the criteria

4.3 Geometric Methods

All presented geometric methods are homotopic.

The Reeb graph is not invariant under isometric transformations, it uses for example a unidirectional height function. All other variants are isometric.

Reeb graphs produce a 1D thin skeleton, Voronoi diagrams need a post-processing step to acquire a 1D skeleton.

Centeredness is achieved by all algorithms of this group, but require sufficient sampling.

Junctions are identified immediately by the geometric algorithms, these are often used as centroids.

Geometric algorithms are not robust. The algorithms are sensitive to noise.

Smoothness is usually lost during the post-processing step when centroids are connected.

Geometric methods cannot preserve details well, as their focus is on detecting shapes (Reeb graphs) or centroids.

Voronoi diagrams store hierarchical information by adjusting the pruning step. Reeb graphs however do not contain hierarchical information.

The complexity of the geometric methods is in most cases $O(n^2)$.

Criterion	Method performance
Homotopy	yes
Invariant	most variants
Thinness	some variants
Centered	yes
Junction Detection	yes
Robustness	no
Smoothness	some variants
Detail preserving	no
Hierarchic	some
Efficiency	okay

Table 3. Performance of geometric methods on the criteria

4.4 General Field Functions

Field algorithms do not always create homotopic skeletons. The reason is that numerical errors can occur.

General field algorithms are not sensitive to isometric transformations.

The algorithms for general fields force 1D thickness by following forces or contours.

Centeredness is not a property of general field.

Joints are detected by the algorithms directly, since the joints are critical points in the vector field.

General field is not very sensitive to noise, but problems because of the numerical resolution of the space can occur.

Smoothness of the skeleton is given by the high number of averaging steps in the algorithm.

General field methods use local critical points in their calculations, which enables a high detail level.

General field algorithms create strict hierarchical skeletons, this is achieved by variations of the number of seed points in the algorithm.

The computation time for general field methods is high. The complexity is $O(n^2)$, but the data set is much larger as in other methods.

Criterion	Method performance
Homotopy	no
Invariant	yes
Thinness	yes
Centered	no
Junction Detection	yes
Robustness	yes
Smoothness	yes
Detail preserving	yes
Hierarchic	yes
Efficiency	bad

Table 4. Performance of general field methods on the criteria

4.5 Final Discussion

We treated the skeletonization algorithms in a naive way. We only compared the algorithms for their most important method. There are many examples where ideas from the presented basic groups are combined to a specialized skeleton extraction method. The advantages and disadvantages of these combined algorithms were not investigated.

5 Conclusion

In this paper we have presented a comparison of 1D skeleton extraction methods. The methods were categorized by what criteria they represent. Those criteria were:

1. Homotopy (2.1)
2. Invariant under isometric transformations (2.2)
3. Thinness (2.4)
4. Centered (2.5)
5. Junction Detection (2.7)
6. Robustness (2.9)
7. Smoothness (2.10)
8. Detail preserving (2.11)
9. Hierarchic (2.12)
10. Efficiency (2.14)

We discussed the meaning of the criteria and their effect on the skeletonization process. We presented the four main categories of skeletonization algorithms and briefly showed their way of operation. The last step was that we compared the methods with our criteria and listed advantages and disadvantages for them. Figure 7 shows how the different methods perform on different objects and what a typical skeleton of each method group looks like.

We concluded that there is not a “best” performing algorithm, the performance depends on the criteria that are important to the user. Thus it is possible to determine a suitable method for skeletonization for your needs from our comparison, but we cannot recommend a single algorithm which would perform best for all the different applications.

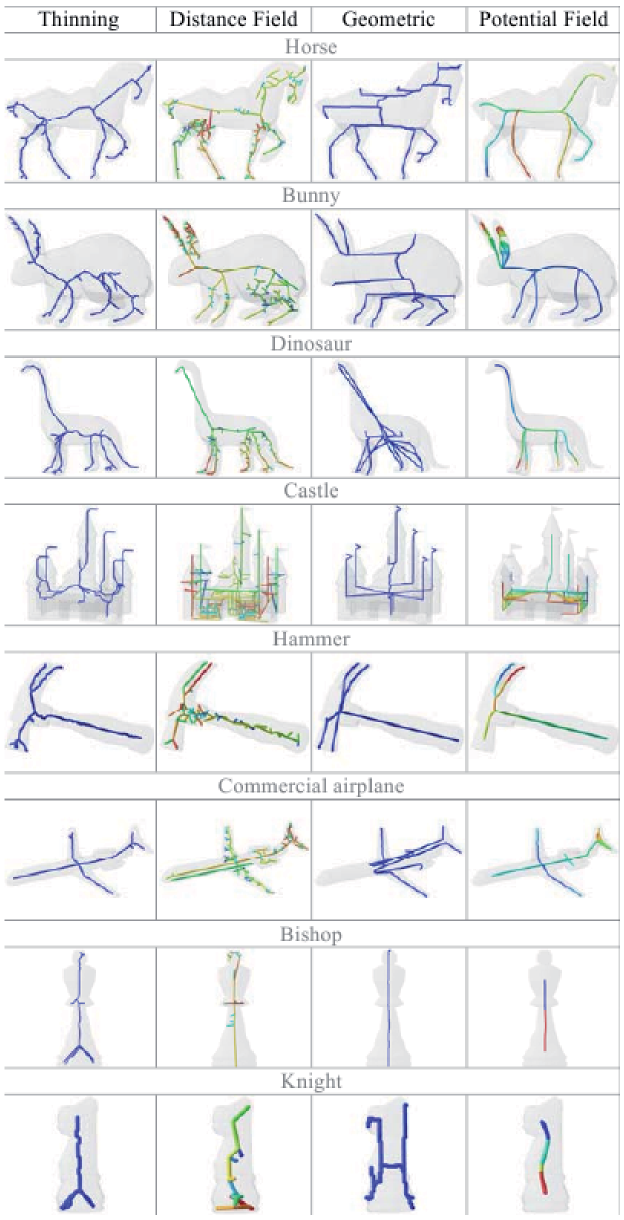


Fig. 7. Performance of the different algorithms visualized for various shapes. Post-processing is not applied to the skeletons. Image from [1]

Although our comparison is not exhaustive, it establishes a basic classification of the methods. A good extension of our work would be to categorize every single major algorithm instead of general methods.

Finally we hope that our findings may stimulate the development of efficient algorithms that exploit the desirable properties of each method group.

Acknowledgements

The authors wish to thank Alexandru Telea for his valuable input as an expert. Furthermore we would like to thank the staff and reviewers of SC@RUG 2013 for the helpful input they provided.

References

[1] N. D. Cornea, D. Silver, and P. Min, "Curve-skeleton properties, applications, and algorithms," *Visualization and Computer Graphics, IEEE Transactions on*, vol. 13, no. 3, pp. 530–548, 2007.

[2] T. Y. Kong and A. Rosenfeld, "Digital topology: introduction and survey," *Computer Vision, Graphics, and Image Processing*, vol. 48, no. 3, pp. 357–393, 1989.

[3] A. Sobiecki, H. C. Yasan, A. C. Jalba, and A. C. Telea, "Qualitative comparison of contraction-based curve skeletonization methods." unpublished.

[4] N. Gagvani and D. Silver, "Animating volumetric models," *Graphical Models*, vol. 63, no. 6, pp. 443–458, 2001.

[5] P. Shilane, P. Min, M. Kazhdan, and T. Funkhouser, "The princeton shape benchmark," in *Shape Modeling Applications, 2004. Proceedings*, pp. 167–178, 2004.

[6] G. Sanniti di Baja and S. Svensson, "A new shape descriptor for surfaces in 3d images," *Pattern Recognition Letters*, vol. 23, no. 6, pp. 703–711, 2002.

[7] T. K. Dey and J. Sun, "Defining and computing curve-skeletons with medial geodesic function," in *ACM International Conference Proceeding Series*, vol. 256, pp. 143–152, 2006.

[8] D.-G. Kang and J. B. Ra, "A new path planning algorithm for maximizing visibility in computed tomography colonography," *Medical Imaging, IEEE Transactions on*, vol. 24, no. 8, pp. 957–968, 2005.

[9] S. Svensson, I. Nyström, and G. S. di Baja, "Curve skeletonization of surface-like objects in 3d images guided by voxel classification," *Pattern Recognition Letters*, vol. 23, no. 12, pp. 1419–1426, 2002.

[10] G. J. Brostow, I. Essa, D. Steedly, and V. Kwatra, *Novel skeletal representation for articulated creatures*. Springer, 2004.

[11] N. D. Cornea, D. Silver, X. Yuan, and R. Balasubramanian, "Computing hierarchical curve-skeletons of 3d objects," *The Visual Computer*, pp. 945–955, 2005.

Simplified graph representation by bundling

Jurgen Jans and Ralph Kiers

Abstract—Visualizing large graphs which contains many nodes and edges can be very difficult. Since many data sets contain both hierarchical as non-hierarchical relations between the data items many edges need to be drawn. Drawings of such graphs generally suffer from visual clutter caused by edges that intersect or overlap each other making it difficult to read the relationships between nodes. In this paper we will discuss the following different edge bundling techniques: Hierarchical edge bundles[2], Force directed edge bundling[3], winding roads[6] and Kernel density estimation[4]. We will compare these techniques based on the quality of their results, the scalability (for large graphs), the computational speed and the simplicity of implementation. In the end we will see that the Hierarchical edge bundling technique[2] is the best method to use when the dataset contains hierarchical relations, in all other cases the Kernel density estimation[4] is the fastest, the winding roads[6] is the best for large datasets and Force directed edge bundling[3] for node-link diagrams with low level relations,

Index Terms—graph bundling, hierarchical edge bundles, force directed edge bundling, winding roads, kernel density estimation, edge bundling, curves, graph visualization, tree visualization, node-link diagrams, hierarchies, treemaps, comparison.

1 Introduction

Graphs are widely used to visualize big sets of data in many research areas such as: biology, microelectronics, social sciences, data mining and computer science. Many data sets contain both hierarchical as non-hierarchical components. The hierarchical components are for example parent-child relations between the data items while the non-hierarchical components are additional relations between the data sets. Examples of such datasets are:

- A hierarchically organized software system, e.g., source code divided into directories, files, and classes, and the relations between these elements, for instance, dependency relations.
- Social networks comprised of individuals at the lowest level of the hierarchy and groups of individuals at higher levels of the hierarchy. Relations could indicate if (groups of) people are acquainted and what the nature of their relationship is.
- A hierarchically organized citation network consisting of publications at the lowest level of the hierarchy and departments and institutes at higher levels of the hierarchy. Links between publications indicate one publication citing the other.

Since both the hierarchical as non-hierarchical components are visualized as edges between the nodes (data items) in a graph, many edges need to be drawn. Especially since improvements in the data acquisition techniques are only increasing the size and complexity of the acquired graphs. This will cause problems when we want to visualize these graphs in a straight-forward way by just drawing straight lines between the nodes. Many lines will intersect or overlap each other resulting in much cluttering and thus decreasing the readability of such graphs. Fig. 1 shows an example of the visual clutter that will occur when the edges are drawn in a straight-forward manner. Many methods have been developed in order to reduce the cluttering of the edges in the graph. They all aim to bundle spatially groups related, close, edges of the graph into curved, dense, bundles, thereby reducing clutter and making the overall graph structure easier to see. The results of the different algorithms seem to produce similar results when been used on similar datasets however differences between these algorithms exist in terms of the computational speed, the simplicity of implementation, and the quality of the drawings output.

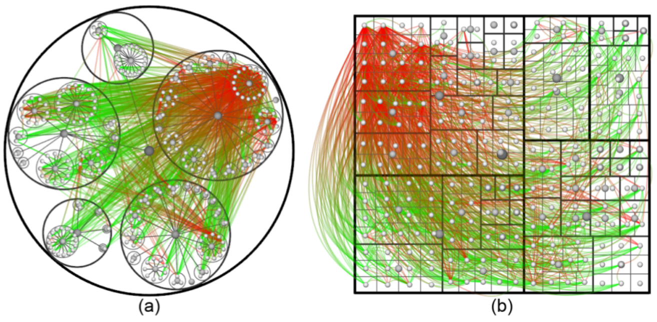


Fig. 1: Displaying adjaceny relations of a call graph(caller = green, callee = red) using (a) a balloon tree, (b) a treemap.[3]

In this paper, we will focus us on the following four edge bundling algorithms: Hierarchical edge bundles[2], Force directed edge bundling[3], Kernel density estimation[4] and winding roads[6]. We will compare and discuss them from the perspective of desirable quality criteria such as: the generation of uncluttered drawings, the scalability (for large graphs), the computational speed and the simplicity of implementation. In this paper we will describe the biggest and most evident differences between the four above mentioned algorithms. We will also look at what the different algorithms have in common.

The remaining part of this paper is organized as follows. In section 2 we give an overview of the existing techniques for tree visualization and edge bundling. Section 3 we describe the four graph bundling algorithms on which we have focused us in detail. Section 4 gives a comparison of the graph bundling techniques as described in the previous section. Finally, section 5 presents the conclusions and explains for which specific case each technique is most suited.

2 Previous works

Since the different edge bundling algorithms all require a graph as input in order to bundle the edges, we will first give an overview of the different techniques that are commonly used for visualizing graphs. After this we will have a look at related methods that reduce clutter in graphs.

- *Jurgen Jans is a MSc. Computing Science student at the University of Groningen, E-mail: jurgenjans480@gmail.com.*
- *Ralph Kiers is a MSc. Computing Science student at the University of Groningen, E-mail: r.s.kiers@student.rug.nl.*

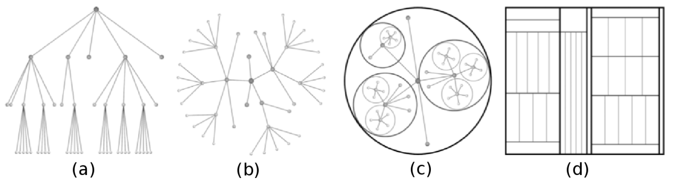


Fig. 2: Common tree visualization techniques. From left-to-right: rooted tree, radial tree, balloon tree and treemap layout.[3]

2.1 Tree Visualization Techniques

The rooted tree as shown in Fig. 2a is one of the most well-known tree visualizations. It is a tree in which one node is distinguished from the other nodes and is called the root. It is best used for showing the hierarchy in a data set. The relationship between parent and child nodes is shown by lines (edges) that connect the nodes. The rooted tree shown in Fig. 2a has a top-down layout were the root is located in the top of the tree other variants are for example a left-to-right layout. Another variant is the radial layout which shown in Fig. 2b. In this layout the root node is placed in the center of the tree, the other nodes are placed in concentric circles according to their depth in the tree. The balloon tree(Fig. 2c) is a special type of the radial tree in which the sibling subtrees are included in circles attached to the parent nodes. The last type of graph shown in Fig. 2d is the treemap layout. This representation aims to make more efficiently use of the available space by using a more space filling layout technique that displays a tree structure by means of enclosure. Although it makes more optimal use of the available space it also makes it more difficult for viewers to perceive the hierarchical relationship between nodes.

2.2 Edge bundling Techniques

One of the first attempts to reduce clutter in graphs visualizations in order to increase the readability of a graph was done by the graph drawing community. They suggested that one should try to bound the number of edge crossings and also to avoid node-edge overlaps. This resulted in multiple edge routing techniques as described by Dobkin et al in [1] or Dwyer and Nachmanson in [8]. Although these approaches efficiently reduce edge clutter by avoiding node-edge overlaps, they do not help to identify high level edge patterns.

Another way to reduce clutter is by simplifying the graph prior to layout by creating metanodes of nodes and edges that are strongly connected as proposed by J.Abello, F. van Ham, and N.Krishmann in [5]. But an undesirable effect of this simplification process is that it also changes the node positions which is critical when the positions encode information.

3 Bundling methods

In this section we will have a look at the different edge bundling algorithms as they are described in the corresponding papers. For each of the algorithms we will describe their working, provide some examples and describe their advantages and disadvantages. We will discuss the algorithms in chronological order, this means in the order that the papers in which they have been published were released. Thus we start with Hierarchical edge bundles, followed by the Force-Directed Edge Bundling and Winding roads and we will conclude with the Kernel Density Estimation.

3.1 Hierarchical Edge Bundles

The hierarchical edge bundle technique was first published by D.Holten in [2]. One of the biggest advantages of this approach is that it is usable in conjunction with existing tree visualization techniques. This means that the adjacency edges can be bundled without needing to change the hierarchical layout of the graph. This is done by using the path along the hierarchy between two nodes that have an adjacency relation as the control polygon of a spline curve. An example

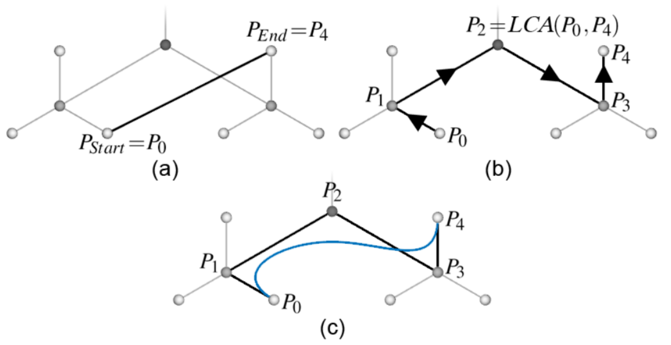


Fig. 3: "Bundling adjacency edges by using the available hierarchy. (a) Straight line connection between $P0$ and $P4$; (b) path along the hierarchy between $P0$ and $P4$; (c) spline curve depicting the connection between $P0$ and $P4$ by using the path from (b) as the control polygon."[3]

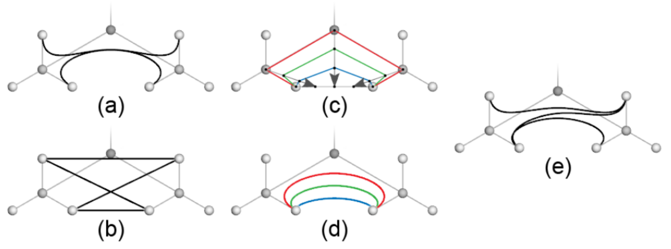


Fig. 4: "The bundle in (a) might contain each edge depicted in (b). (c) and (d) show how different values of β (red = 1, green = $\frac{2}{3}$, and blue = $\frac{1}{3}$) can be used to alter the shape of spline curves. As shown in (e), a fairly high bundling strength (= 0.8) can be chosen to retain visual bundles while still resolving ambiguity."[3]

is shown in Fig. 3 Fig. 3a shows the adjacency relation between the nodes P0 and P4. A path along the hierarchy between the two nodes can be made by making use of the least common ancestor as shown in Fig. 3b. Now that we have a path between the two nodes, the adjacency relation can be turned into a spline curve using the path as the control polygon (Fig. 3c). The strength with which the edges are bundled is controlled by a parameter β with $\beta \in [0,1]$. This parameter controls the amount of bundling by straightening the spline curve. An example is shown in figure Fig. 4. Fig. 4a shows that bundling ambiguity can occur when trying to bundle the adjacency edges shown in Fig. 4b using a wrong value of β.Fig. 4c and Fig. 4d show how different values of β (red = 1, green = $\frac{2}{3}$ and blue = $\frac{1}{3}$) can be used to alter the shape of the spline curves. Fig. 4e shows the best solution in this case by using a bundling strength of $\beta = 0.8$. Fig. 5 shows how increasing the value β has influence on the bundling strength of the curves. The higher the value is set the smaller the bundles get since the curves are stronger bundled together. This way a trade off can be made between showing the low-level and high-level adjacency relations.

3.2 Force-Directed Edge Bundling

Force-Directed Edge Bundling (FDEB)[3] is a self-organizing, force-directed graph bundling technique designed to bundle general graphs without the use of a hierarchy or control mesh. The main advantage of FDEB is that there is no need for a hierarchical-clustering scheme or spanning-tree generation method, which makes sure all the high-level edge data is reflected in the graph. Furthermore, The behavior of the algorithm is easy to understand due to the straightforward physics model, and the layout criteria of the graph is easily modified, which will be explained later.

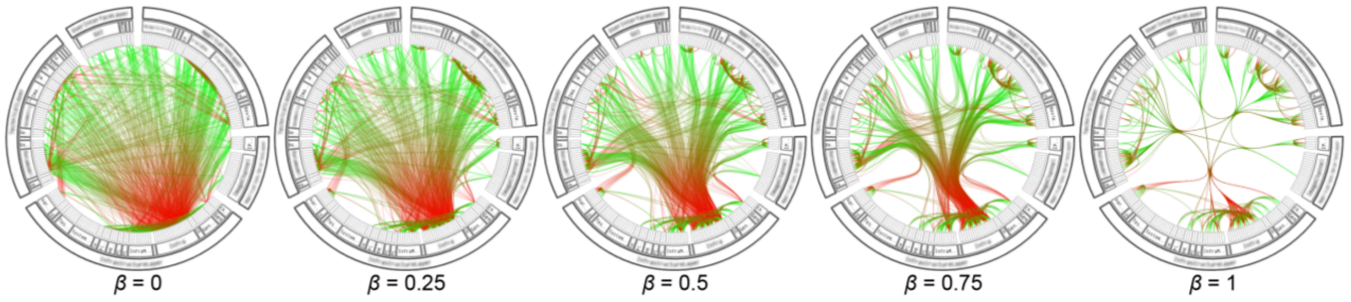
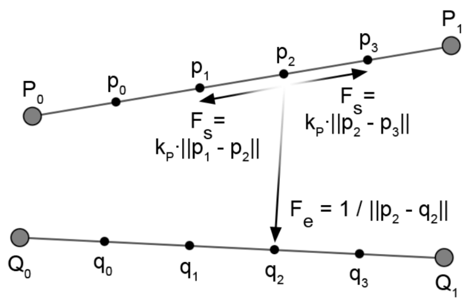


Fig. 5: Radial layout of a call graph(caller = green, callee = red), with an increasing bundling strength β(from left to right). A high β provides high-level the adjacency relations, while a low β provides low-level adjacency relations. [3]



P_0
p_0
p_1
p_2
p_3
P_1
$F_s = k_P \cdot \|p_1 - p_2\|$
$F_s = k_P \cdot \|p_2 - p_3\|$
$F_e = 1 / \|p_2 - q_2\|$
Q_0
q_0
q_1
q_2
q_3
Q_1

Fig. 6: Two interacting edges P and Q, with the spring force F_s and electrostatic force F_e shown for p_2.[3]

FDEB makes use of spring forces and edge compatibility measures in oder to make a bundled graph. FDEB initially starts of as a node-link diagram with straight edges. In order to change these straight edges into a shape, such that they represent a bundled graph, the edges are subdivided into segments. Fig. 6 shows the subdivision of two edges that influence each other along with their subdivision points and the spring forces acting upon point p_2. When changing the shape of an edge all the subdivision points can move, but the end points stay at a fixed position. E.g. the points P_0, P_1, Q_0 and Q_1 in Fig. 6 stay fixed and points $p_0 \dots p_3$, $q_0 \dots q_3$ can move. The attracting spring force $F_s = k_p \|p_i - p_{i-1}\|$ is calculated for each pair of neighboring points $p \in \{p_0 \dots p_{n-1}\}$, with n being the number of subdivision points. The local spring constant k_p used in the calculation of F_s is calculated by $k_p = \frac{K}{\|P\|}$, where K is a global spring constant used to control the amount of bundling, $\|P\|$ is the initial(straight) length of edge P. The other spring force working on the subdivision points is the electrostatic force $F_e = \frac{1}{\|p_i - q_i\|}$, where $0 \leq i < n$. The electrostatic force works between two corresponding subdivision points, e.g. p_2 and q_2 in Fig. 6, an alternative to this is to calculate F_e for each pair (p,q), with $p \in P$ and $q \in Q$.

An other force on the subdivision points comes from edge compatibility measures. Edge compatibility measures determine how much two edges interact with each other based on a certain criteria. Suppose we have k such criteria, then the total edge compatibility for two edges P and Q can be defined as

$$C_e(P,Q) = \prod_{i=0}^{k-1} C_i(P,Q)$$

where $C_i(P,Q)$ is a compatibility measure with $C_i(P,Q) \in [0,1]$ and C_e is the total edge compatibility. An example of a compatibility measure is the angle complexity $C_a(P,Q)$, which gives a measure for the angle between edges P and Q. $C_a(P,Q)$ is defined as

$$C_a(P,Q) = \left|\cos\left(\arccos\left(\frac{P \cdot Q}{\|P\|\|Q\|}\right)\right)\right|$$

with $C_a(P,Q) \in [0,1]$. The following edge compatibility measures are used: angle compatibility, scale compatibility, position compatibility and visibility compatibility. For a definition of the compatibilities refer to [3].

Now that all the forces are defined it is possible to define the combined force F_{p_i} as

$$F_{p_i} = k_p\left(\|p_{i-1} - p_i\| + \|p_i - p_{i+1}\|\right) + \sum_{Q \in E} \frac{C_e(P,Q)}{\|p_i - q_i\|}$$

with E the set of edges interacting with edge P, but excluding edge P itself. Using F_{p_i} the algorithm to compute the bundled graph is as follows: Perform a fixed number of cycles, each cycle performs a number of iterations. During an iteration p_i is moved in the direction of F_{p_i} with a step size S. The algorithm starts with the initially with P_0 subdivision points, step size S_0, number of iterations I_0 and number of cycles C. After each cycle the number of subdivision points is doubled, the step size is halved and the number of iterations is decreased. The paper uses the initial values $P_0 = 1$, $S_0 = 0.04$, $C = 6$ and $I_0 = 50$, each cycle the number of iterations was decreased by a factor of $\frac{2}{3}$.

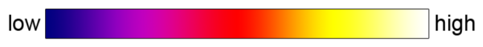


Fig. 7: Color mapping for edge density.[3]

Before rendering the graph is smoothed by convolving the subdivision points with a Gaussian kernel. Rendering of the graph was done using the color map shown in Fig. 7, with low to high representing the amount of edges going through a particular pixel. A final result of the edge bundling algorithm is shown in Fig. 8.

3.3 Winding Roads

The winding roads edge bundling technique was first published in [6]. The main idea of this approach is to use edge routing in order to bundle edges. It basically consists of two steps. First a grid is used to compute the shortest routes for each edges, when this is done frequently used paths can be used to bundle the edges.

In order to compute the shortest paths of each original edge it is necessary to create a grid graph which connect all the original nodes. This is done by dividing the graph plane into cells using nodes positions. Different grid graphs that can be used are shown in Fig. 9 which is generated on the 2000 air traffic network.

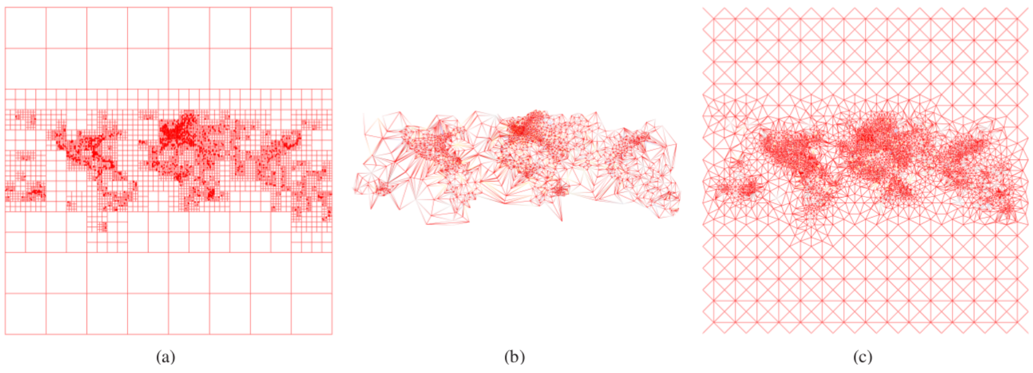


Fig. 9: "Grid graphs generated on the 2000 Air Traffic (AT) network with (a) a quad tree (37395 nodes/69102 edges), (b) a Voronoï diagram (4531 nodes/13558 edges) and (c) the hybrid quad tree/Voronoï approach (10146 nodes/30315 edges)."[6]

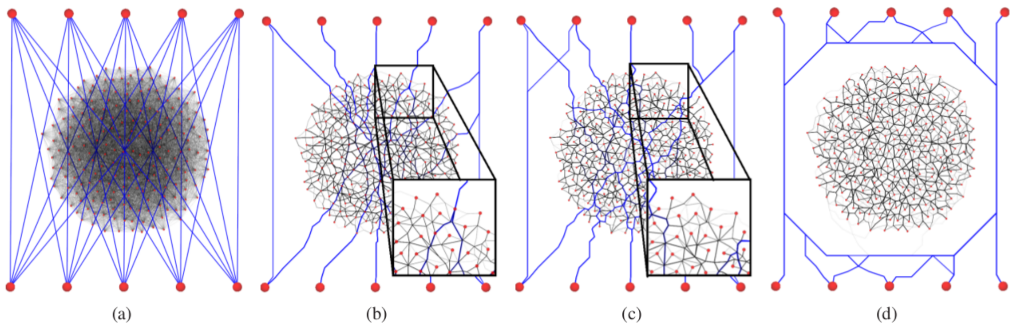


Fig. 10: "(a) Different clutter reductions of a graph representation, (b) using edge-edge clutter reduction method, (c) avoiding node-edge overlap and (d) uncluttering dense zones."[6]

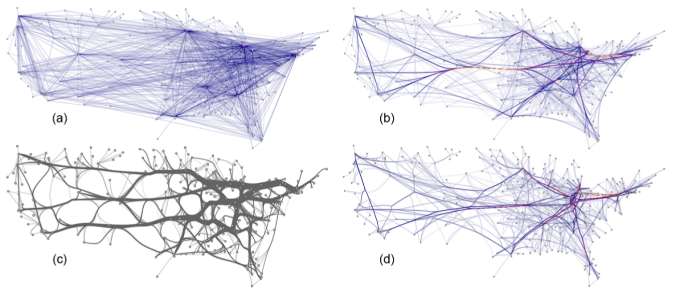


Fig. 8: "US airlines graph (235 nodes, 2101 edges) (a) not bundled and bundled using (b) FDEB with inverse-linear model, (c) GBEB, and (d) FDEB with inverse-quadratic model."[3].

The first grid graph as shown in Fig. 9a is a quad tree. In a quad tree, the plane is divided into four parts until it contains at most one element. This approach is efficient in terms of computation time because the complexity is $O(\|V\|\log(\|V\|))$ but it generates a large grid with 37, 395 nodes and 69, 102 edges for the 2000 AT network example.

Another grid graph that can be used is the Voronoï diagram (Fig. 9b). In a Voronoï diagram, cells are regions of the plane in which the points of a cell are closer to the cells node than to any other node. In the example image a constrained Voronoï diagram is used instead of the classical Voronoï diagram since the classical diagram does not necessary avoid edges that are overlapping nodes.This method generates a small grid graph (4, 531 nodes/13, 558 edges for the 2000 AT network) and can be computed in $O(\|V\|\log(\|V\|))$ time. But the downside is that it generates large cells for sparse regions, this will result in large detours during the routing process.

Therefore the best solution is to use a hybrid algorithm based on both quad trees and Voronoï diagrams (Fig. 9c). In this algorithm, the size of quad tree cells are parametrized to generate different levels of clutter reduction and then then the Voronoï diagrams are used to construct the final grid graph. Since the quad tree adds $O(\|V\|)$ nodes, the $O(\|V\|\log(\|V\|))$ time complexity is preserved. Therefore, the resulting grid graph is of reasonable size (10, 146 nodes/ 30, 315 edges on 2000 AT network).

The next step is to route the edges in the original graph onto the grid that we have discussed in the previous step. This idea can be best explained using a metaphor of roads and highways. Regular roads will be transformed in to highways if they are highly used. This can be done by first computing all the shortest paths between the linked nodes in the original graph using Dijkstra's shortest paths algorithm. Next we can adjust the weights of every edge according to the number of shortest paths that pass through this edge of the grid. Reducing the weight of an edge is equivalent to transforming it into a highway, since using that edge enables it to go faster from one point to another. Finally a shortest path for every edge of the original graph can be computed and the weights can be changed accordingly. This results in new edge bundles since the new distance matrix of the graph promotes frequently used edges. Since Dijkstra's algorithm is used for finding the shortest paths the time complexity is $O(\ |Vgrid\||\|Egrid\| + \|Vgrid\|2\log(\|Egrid\|))$.

With the obtained weighted shortest paths it is possible to define different levels of clutter reduction by either adapting edge weights or avoiding a particular path. Fig. 10 shows different levels levels of clutter reduction. Fig. 10b shows the edge-edge clutter reduction. This reduction only reduces the clutter that is caused by edges crossing other edges. The image clearly shows that blue edges are routed through nodes of the network. To obtain this level of clutter reduction, each edge of the grid graph needs to be considered when routing the edges. In particular, edges linking original nodes to nodes of the grid graph can be used when computing the shortest paths on the grid graph.

As mentioned before edge-edge clutter reduction only allows us to reduce the clutter due to edge crossings but still makes it possible for edges to overlap other nodes including the original graph nodes. This can be reduced by using the node-edge clutter reduction as shown in Fig. 10c. This reduction simply forbids the edges to be routed through the original graph nodes. When comparing this example with the edge-edge clutter reduction, it is easy to notice that the blue edges this time are routed around the original nodes in the graph and can only cross the grid nodes.

The last clutter reduction method shown in Fig. 10d is the uncluttering highly dense zones reduction. This cluttering reducing method improves the clutter reduction by promoting paths to pass through sparse regions. When looking at the example, it can be seen that all blue edges have been routed around the dense subgraph (in the middle of the figure). This makes the graph more readable since the density in this part of the graph is already high since the graph contains many cells in this region.

3.4 Kernel Density Estimation Edge-Bundling

Kernel density estimation edge-bundling (KDEEB)[4] is an image based graph bundling technique for general graphs. KDEEB works by re-sampling the edges into edge points, calculating the density map and then moving the edge pixels in the directing of the gradient. KDEEB's biggest advantages are: that it is an oder of magnitude faster than state-of-the-art techniques, and the algorithm works on general graphs.

KDEEB makes use of a density map ρ created through the use of kernel density estimation (KDE)[7]. The density map will contain large values where the edge density is high, and local maxima are located roughly in the center of local areas with a lot of edges.

Using the density map ρ, a KDEEB operator B is defined as the solution to the following differential equation:

$$\frac{dx}{dt} = \frac{h(t)\nabla\rho(t)}{\max(\|\nabla\rho(t)\|, \varepsilon)} \tag{1}$$

for all points x in the graph drawing, where $h(t)$ is the kernel bandwidth and $\varepsilon = 10^{-5}$ to prevent devision by zero. The gradient of the density map $\nabla\rho$ is normalized, this limits the movements of $\|dx\|$ to $h(t)$. If $h(t)$ decreases in time the movement of points $x \in G$ where G is a graph drawing converges. Equation (1) is solved by computing ρ iteratively and by moving the points $x \in G$ in the direction of $\nabla\rho$, if x is not an edge endpoint.

The chosen kernel K for KDE, which is used to compute ρ is $K(x) = 1 - \|x\|^2$. The bandwidth h used for KDE changes each iteration step, such that $h = \lambda^i h_{max}$ where i is the current iteration step and h_{max} is the initial kernel bandwidth. h_{max} is set to the average inter-edge distance in the input graph. λ is a reduction factor set such that $\lambda \in [0.5, 0.9]$. Because $\lambda < 1$ it is easy to see that $h(t)$ decreases with time. Fig. 11 shows some iterations of ρ as a normalized height plot and the corresponding bundles graph images. As can be seen the density maps get sharper with each iteration. An other example of the KDEEB algorithm can be seen in Fig. 12. The graph bundling was done on a graph containing 100k randomly generated edges. As can be seen, KDEEB still creates a nicely bundled graph even though the graph consisted out of a lot of edges that had no underlying connection.

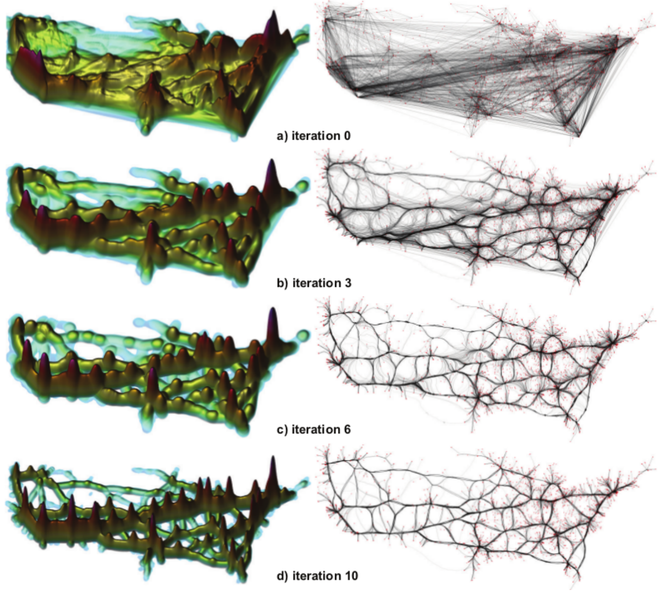


Fig. 11: "Evolution of density map and corresponding bundling for the US migrations graph."[4]

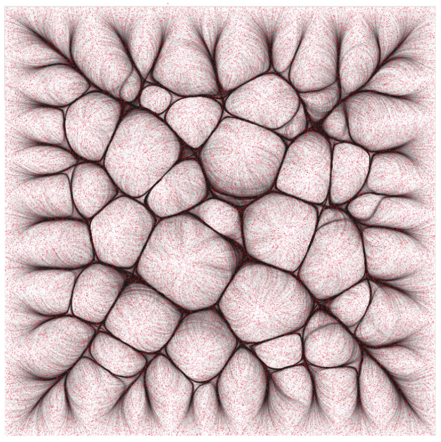


Fig. 12: Edge bundling on a graph with 100k randomly generated edges[4]

4 Comparison

Now that we have described the four different edge bundling techniques, we can make a comparison between them. In this section we will consider the different techniques in the same order as we described them and discuss the pros and cons of each algorithm and what makes them unique compared to the other methods.

The hierarchical edge bundling technique[2] is the fastest algorithm of the four that we have discussed in this paper. This is due the fact that this technique makes use of the existence of the hierarchy tree which tells where the edges should be bundled, therefore it is the only single pass edge bundling method. All the other bundling methods are iterative and thus by definition slower and less easy to control. Another big advantage of this technique is that it does not have any requirements about the used tree layout, it can be applied on different tree layouts without the need to change them thus keeping the hierarchical layout in touch. It also offers the user a trade off between visualizing the low-level and high-level adjacency relations between the nodes by using the β strength bundling value. The disadvantage however is that since this method extensively makes use of a hierarchy tree, it is only suitable for compound graphs. That is the graph should both have hierarchical and association relations. Another disadvantage is that bundle overlap may occur in cases where the layout contains a large number of collinear nodes.

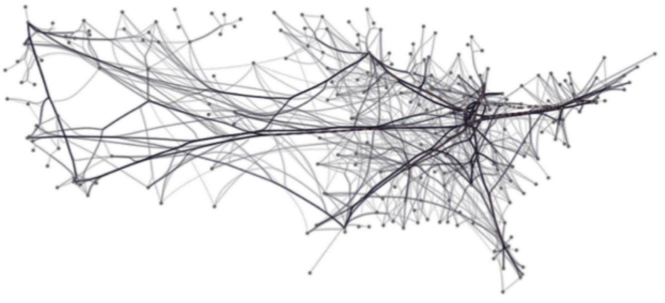
The force-directed edge bundling technique[3] is an iterative method but easy to implement. Customizable behaviour can be added by adding or removing edge compatibility measures. This method is probably the slowest method of the four discussed in this paper but offers a good opportunity to make it faster since each edge pair interaction can be calculated in parallel. Therefore this method can be parallelized although this is not a trivial thing to do. The downside of this method is that it can only operate on node-link diagrams which makes it less flexible as for example the hierarchical edge bundle technique. Another downside is that this technique is not very suitable to be used with real world graphs due to the high data complexity.

The winding roads edge bundling technique [6] offers a good level of clutter reduction combined with a good computation performance. It is one of the few methods which is also very suitable for drawing scale free networks (a graph whose degree distribution follows a power law) and good for identifying relationships and high level edge patterns even in real world graphs with a high data complexity. Another advantage of this method is that it guarantees that no edge-node overlaps will occur in the resulting graphs and moreover that the node positions wont be changed which makes it very suitable for graphs in which they position of the nodes are important because they contain important information like for example geometric graphs. It is also able to remove edge cluttering in high density areas by bundling the edges in sparse regions instead. A downside however is that this method requires the use of a grid graph which therefore adds additional nodes and edges to the graph, increasing it's total size.

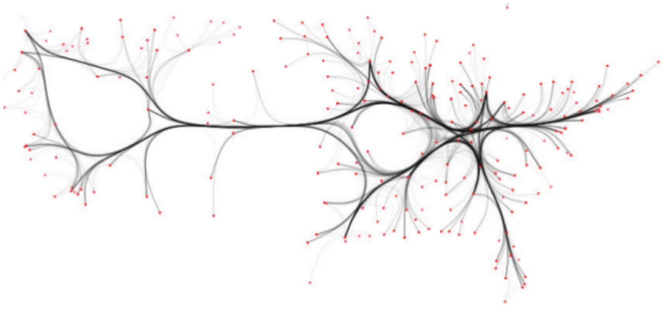
The Kernel density estimation edge-bundling technique [4] is more similar to the winding roads technique [6] but this method is parallelized which makes it faster then the other iterative methods. According to [4] this method is 4 times faster than the force-directed edge bundling technique[3] and 2 times faster than the Winding roads technique [6]. When comparing the results of this method with the other methods it's clearly noticeable that the bundling is a lot tighter than the force-directed edge bundling technique[3] and the hierarchical edge bundling technique[2]. Fig. 13 shows a comparison of the result obtained with this method with the result of same graph using the force-directed edge bundling technique[3]. There is also no need for extra information, such as the edge compatibility measures used by force-directed edge bundling technique[3].

5 Conclusion

In this paper we have discussed four different edge bundling techniques: Hierarchical edge bundling[2], Force directed edge bundling[3], winding roads[6]and Kernel density estimation[4]. We have started with explaining how each technique is working and later on elaborated further by making a comparison between them where we discussed the different properties of each method and in which way they differ from each other. From this we can conclude that in case you have a compound graph containing both hierarchical and association relations, the only method that is suitable to use is the Hierarchical edge bundling technique[2] which is besides also the fastest method. In case of a general graph one of the three other techniques can be used. The best choice in this case depends on the type of graph, the performance and the amount of detail that the resulting graph should contain (high level or low level relations). When the graph to be bundled is a real world graph or a scale free network to best choice would be the winding roads technique [6] since this method is the best at visualization high level patterns in graphs with a high data complexity. If the graph is a node link diagram and you want to show the low level relations than the force-directed edge bundling technique[3] would be the best choice. In all other cases the Kernel density estimation edge-bundling technique [4] should be used since this algorithm is parallelized and therefore has the best performance of the three iterative methods.



(a) Force-directed edge bundling



(b) kernel density estimation

Fig. 13: Comparison result US airlines[4]

References

[1] D.Dobkin, E. Gansner, E. Koutsofios, and S.North. Implementing a general-purpose edge route. *Proc. Graph Drawing 1997 (GD97) (1998)*, 2:262271, 1998.

[2] D. Holten. Hierarchical edge bundles: visualization of adjacency relations in hierarchical data. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):741–748, 2006.

[3] D. Holten and J. J. van Wijk. Force directed edge bundling for graph visualization. *Comp. Graph. Forum*, 28(3):983–990, 2009.

[4] C. Hurter, O. Ersoy, and A. Telea. Graph bundling by kernel density estimation. *Comp. Graph. Forum*, 31(3):865–874, 2012.

[5] J.Abello, F. van Ham, and N.Krishmann. Askgraphview: A large graph visualisation system. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):669676, 2006.

[6] A. Lambert, R. Bourqui, and D. Auber. Winding roads: routing edges into bundles. *Comp. Graph. Forum*, 29(3):853–862, 2010.

[7] B. Silverman. *Density Estimation for Statistics and Data Analysis*. Monographs on Statistics and Applied Probability 26. 1992.

[8] T.Dwyer and L. Nachmanson. Fast edge-routing for large graphs. *Proc. Graph Drawing 2009 (GD09)*, 2, 2010.

Push techniques and alternatives in the mobile phone browser

Erik Bakker, and Mark Kloosterhuis

Abstract— Focussing on websites for the mobile phone browser is also part of web development. Without the need to download and install a mobile phone app it is possible to create a mobile phone website where people are able to chat live with eachother. Displaying new information on a mobile phone website without the need to reload the entire webpage is researched in this paper. This is also called bidirectional communication.

To create a chatbox mobile phone website websockets can be used to achieve this bidirectional communication. There are several alternatives for websockets. In this paper we discuss websockets, Ajax and Comet as alternatives for websockets. These three techniques will be reviewed and compared based on performance, how easy it is to implement them on client side and mobile phone browser support.

Not every mobile phone browser is compatible with websockets without the need to download and install third party libraries. Every mobile phone browser which is able to handle the XMLHttpRequest object is able to use the Ajax and Comet techniques, which is almost every mobile phone browser.

Three mobile phone browsers are used to test the three different techniques: Chrome for Android, Firefox for Android and Opera Mobile for Android. The Android operating system is used as a mobile phone test environment.

Index Terms—Mobile phone browser, mobile web development, push messages, websocket, Comet, Ajax, no reload, webpage

1 Introduction

Web development has rapidly evolved during the last few years. Static web pages which are linked together causes web browsers to reload web pages. Nowadays the user wants a dynamic web page (WEB2.0[12]) which displays information without much waiting time for the user. [8]
Almost everybody owns a mobile phone with a standard mobile phone browser on it to search for information on the internet. Web development is therefore also focussing on mobile web development for the mobile phone browsers, like firefox for Android, chrome for Android, Opera Mobile, etc.
To create something like Google Talk[5] without downloading and installing a mobile phone app, a website can be used. Messages should then be displayed on the webpage right after they are sended by another user. This without the need to reload the whole web page. Considering the cost of internet bandwidth that comes with mobile phones it is handy to only send the information that is needed.
Our research is focussing on mobile web development for the mobile phone browser and dynamic web pages which uses techniques to show information on a mobile phone web page without the need to fully reload the mobile phone web page.
Before we can start to compare the techniques to show information on a web page without the need to reload it, we start making a selection of the techniques. Then we select the most popular mobile phone browsers which are compatible with the chosen techniques. Once we have selected the techniques and the browsers we are able to make a comparison between the techniques in respect to the browsers. The comparison consist of an explanation of the techniques, which technique performance best, which technique is the easiest to implement and platform compatibility. Then an implementation is given of the techniques. Finally a conclusion is given.

- *Erik Bakker, MSc student at University of Groningen, E-mail: E.M.Bakker.4@student.rug.nl.*
- *Mark Kloosterhuis, MSc student at University of Groningen, E-mail: markkl@gmail.com.*

2 Selection

A study examined the effects of five web design features customization, adaptive behavior, memory load, content density, and speed based on user preference for web-based services.
The 2009 study by a HCI study [14] tested site designs for online flight reservations. The results of this study are valuable because insights into the relative importance to users of interface attributes can help web developers increase adoption and retention rates, and boost online revenues. As seen in figure 1 they found that the most preferred feature was high speed, followed distantly by minimal memory load, adaptive behavior, low content density, and customization features.
The selection that we are going to use consist of selecting the techniques which are able to show information without the need to reload the web page to increase the overall speed of the web pages.
Also selecting the mobile phone browsers which are compatible with all the selected techniques: websockets, Ajax and Comet.

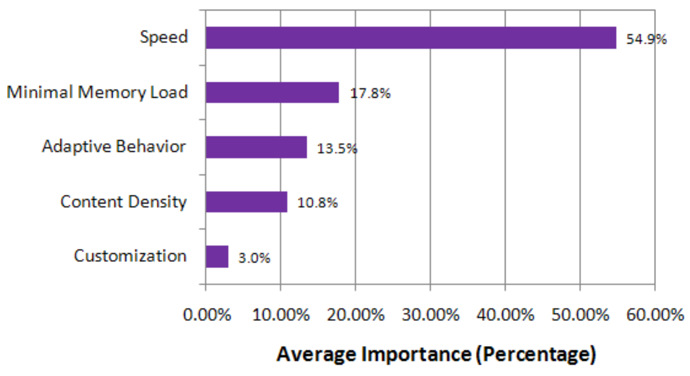


Fig. 1. Relative importance of interface features. [14]

2.1 Techniques

The techniques we are going to select need to meet the following requirements.

- The technique must be compatible on a mobile phone browser without needing to download and install other software to function properly.
- The technique must be used by many not mobile phone WEB2.0 web pages. Thus a popular technique.

One of the first techniques that can be considered is Ajax. Ajax which uses javascipt, can be executed on a mobile phone browser without the need to download and install other software. Ajax is used by many not mobile phone WEB2.0 web pages.
The second one is Comet, also known as reverse Ajax. This technique also uses javascript and can be executed on a mobile phone browser without the need to download and install other software.
The last one is websockets. This is a relative new technique within mobile phone browsers. It is not yet supported by all modern mobile phone browsers, but since the end of the year 2012 several browsers are compatible with websockets by default.

There are much more techniques available, but we will discuss only these three.

2.2 Mobile phone browsers

As explained in the previous chapters the mobile phone browsers are selected on popularity and compatibility with the techniques. The browsers that are popular are Android browser, Blackberry browser, Opera Mobile, Chrome for Android, Firefox for Android and iOS Safari. The most recent versions of these browsers do not all support websockets. [4] We have selected the following mobile phone browsers:

- Chrome for Android, version >25.0
- Firefox for Android, version >19.0
- Opera Mobile, version >12.1

2.3 Operating System

There are several different operating systems available for mobile phones. The most popular ones are iOS, Android and Blackberry. We have selected Android as our operating system for testing environment. The version of Android that we are using is 4.1.2.

3 Comparison

In the previous section we discussed the selection of the several techniques, the selection of the several testing mobile browsers and the operating system. In the next section we will compare the techniques on how they work, performance, compatibility and how easy it is to implement it on clientside.

3.1 Techniques

Three of the main techniques for web developers when developing an interactive web appplication are WebSockets, Comet and Ajax. The next section will discuss the working of the techniques.

3.1.1 Ajax

The classic web application uses HTTP requests to communicate with the web server. When the user requests a web page, the server sends the HTML code, CSS style sheet code, and JS code at once. When the user fills in a form and submits it, the information is send to the web server and the web server rebuilds the web page. The web server then sends the web page back to the user.

The first steps to set up an Ajax webpage are equal to a classic webpage. Request a web page via a HTTP request, the Javascript code, HTML code, and CSS code are returned from the web server. Additional Javascript code is send from the web server to the web browser, which include the javascript code that Ajax is using. Ajax is able to generate HTTP requests without refreshing the web page. Javascript sets up a XMLHttpRequest with the web server. The web server then executes Application logic and returns the response to the web page. Javascript then processes the response and updates the DOM/CSS. See Figure 2

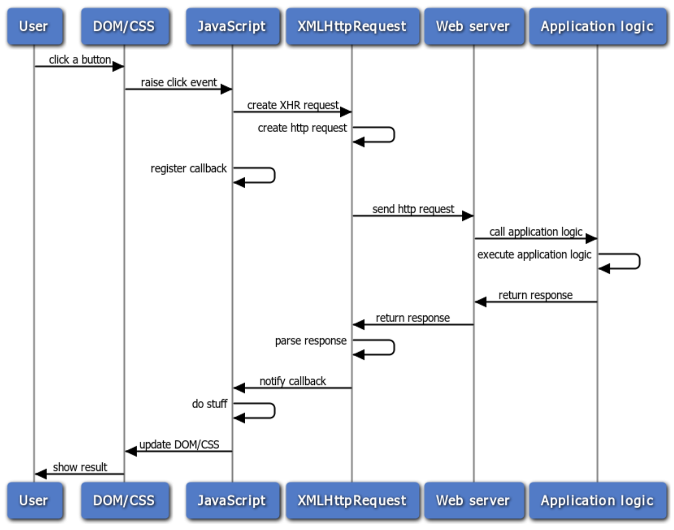


Fig. 2. Ajax technique. [10]

3.1.2 Comet

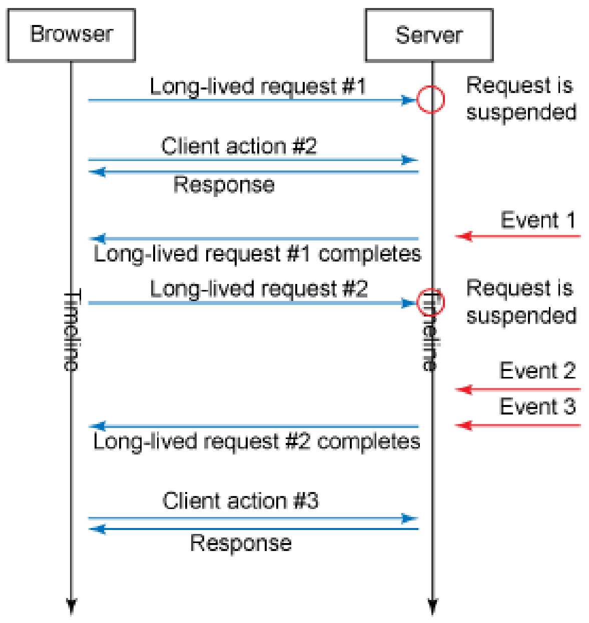


Fig. 3. Comet: how it works.

Comet also know as reverse Ajax is a Web application model that enables web servers to send data to the client without having to explicitly request it. Developers can utilize Comet techniques to create event-driven Web apps. Comet is actually an umbrella term for multiple techniques for achieving client-server interaction. All methods have in common that they rely on browser-native technologies such as JavaScript, rather than on proprietary plug-ins.
Figure 3 is a representation of how comet will work in practice. The

browser request a long-lived request number 1. The server than suspend the request untill an server-side event is triggered. The outcome of the event is then send to the browser. Long-lived request number 1 completes. During the request is suspended a client is able to communicate with the server.

3.1.3 Websocket

The WebSocket Protocol[3] is designed to supersede existing bidirectional communication technologies that use HTTP as a transport layer to benefit from existing infrastructure (proxies, filtering, authentication). Such technologies were implemented as trade-offs between efficiency and reliability because HTTP was not initially meant to be used for bidirectional communication. The WebSocket Protocol attempts to address the goals of existing bidirectional HTTP technologies in the context of the existing HTTP infrastructure; as such, it is designed to work over HTTP ports 80 and 443 as well as to support HTTP proxies and intermediaries, even if this implies some complexity specific to the current environment.
However, the design does not limit WebSocket to HTTP, and future implementations could use a simpler handshake over a dedicated port without reinventing the entire protocol. This last point is important because the traffic patterns of interactive messaging do not closely match standard HTTP traffic and can induce unusual loads on some components.

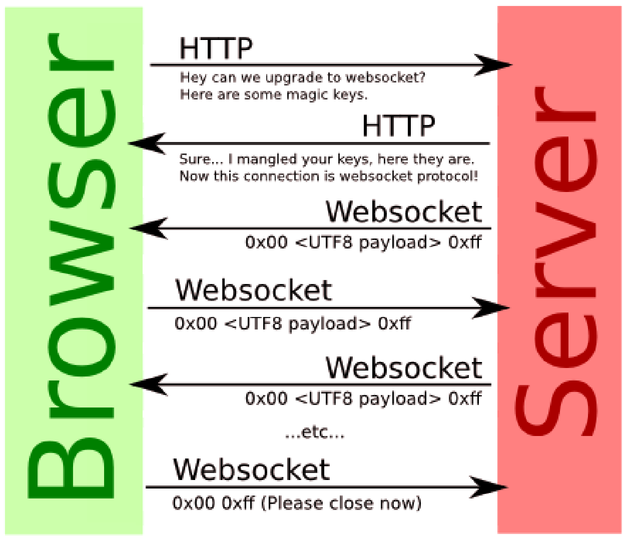

Browser
Server
HTTP
Hey can we upgrade to websocket?
Here are some magic keys.
HTTP
Sure... I mangled your keys, here they are.
Now this connection is websocket protocol!
Websocket
0x00 <UTF8 payload> 0xff
Websocket
0x00 <UTF8 payload> 0xff
Websocket
0x00 <UTF8 payload> 0xff
...etc...
Websocket
0x00 0xff (Please close now)

Fig. 4. Websockets: how it works

3.2 Mobile browser

We will only be testing and comparing the browsers that support all three techniques: WebSockets, Comet and Ajax. There are no popular Android browsers that do not support Comet or Ajax. But there are a lot of browser/version that do not support WebSockets including the stock browser in Android.
One of the main downsides of using Websockets on Android devices is that the stock browser does not accommodate WebSockets. When developers decide to make a web application wrapped inside a real application like the Facebook app in Android, the developers rely on the Android stock browser. This makes it for web developers impossible to use websockets inside a webview without implementing third party libraries in their applications.

3.2.1 Firefox for Android

Developed by the Mozilla Foundation, the open-source Firefox for Android was previously known under its codename of Fennec - and it's come a long way since the early days. Designed to offer a familiar interface to those who use Firefox on the desktop, it's a powerful - but memory-hungry - browser going through a rapid development cycle.

3.2.2 Chrome for Android

It may seem strange for Google to offer two distinct browsers for Android, but Chrome is a lot different to the stock Android browser. Based on the same open-source Chromium engine as the desktop version, Chrome for Android provides a familiar environment for anyone used to the software on other devices. Better still, it synchronises with Chrome on desktops and laptops, providing access to saved tabs, passwords and browsing history. Untill April 2013 the Chromium team worked together with Apple on Apple's Webkit layout engine, In April 2013 the Chromium announced [2] that they will be forking the Webkit layout engine into Blink.

3.2.3 Opera mobile for Android

Opera was originally a paid-for browser, with a free advertising-supported version. The firm made the decision to go completely free some time ago, in the face of dropping market share against the likes of Firefox and Chrome. Its mobile variants have long been popular options for smartphones and featurephones, however, and continue to perform well on Android. In February 2013 Opera announced that they will stop developing there own Presto layout engine in favour of Google's Webkit fork Blink. [11]

3.3 Performance techniques

Performance is an important aspect of web development. Users are very impatient and expect websites to load quickly. On mobile devices, network usage is one of the biggest aspects concerning performance. Carriers usually charge network usage per megabyte and mobile broadband is still not accessable to most mobile users. This is why we decided to focus our performance analysis on network usage, but we did not limit ourselves to it. Some techniques have inherrent performance limitations and are therefor discussed in the corresponding sections.

3.3.1 Ajax

Since Ajax uses polling[1] to enable bidirectional communication, the overhead is consistent over time. According to kaazing [9] the overhead of a header is 871 bytes, depending on the configuration. This includes the request en response header. See listing 1 and listing 2

```
GET /PollingStock/PollingStock HTTP/1.1
 Host: localhost:8080
 User-Agent:  Mozilla/5.0 (Windows NT 6.2; WOW64)
     AppleWebKit/537.31 (KHTML, like Gecko) Chrome
     /26.0.1410.64 Safari/537.31
 Accept: text/html,application/xhtml+xml,application
     /xml;q=0.9,*/;q=0.8
 Accept-Language: en-us
 Accept-Encoding: gzip,deflate
 Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
 Keep-Alive: 300
 Connection: keep-alive
 Referer: PollingStock/
 Cookie: showInheritedConstant=false;
     showInheritedProtectedConstant=false;
showInheritedProperty=false;
    showInheritedProtectedProperty=false;
showInheritedMethod=false;
    showInheritedProtectedMethod=false;
showInheritedEvent=false; showInheritedStyle=false;
    showInheritedEffect=false
```

Listing 1. HTTP request header[9]

```
HTTP/1.x 200 OK
 X-Powered-By: Servlet/2.5
 Server: Sun Java System Application Server 9.1_02
 Content-Type: text/html;charset=UTF-8
 Content-Length: 21
 Date: Sat, 07 Nov 2009 00:32:46 GMT
```

Listing 2. HTTP response header[9]

What happens if 1000 clients are simultaneously polling every second? The network throughput will then be $(871*1000) = 871000$ bytes$=$ 6968000 bits per second. That is 6.6 Mbps. This is only the headerdata that is send and received. There is no data in it.

3.3.2 Comet

Comet uses long-polling[15] to enable bidirectional communication. This means that the overhead counts when sending a request from the client to the server and when the server sends data to the client. Other than Ajax Comet is not polling for example every 2 seconds, but waits for the server to send data over the long-lived request. The responsetime of the server depends on the application logic. If the server for example responds every 4 seconds then the average overhead with 1000 clients is: $(871/4)*1000 = 217750$ bytes $= 1742000$ bits per second. That is 1.6 Mbps.

3.3.3 Websockets

Websockets do not use polling to enable bidirectional communication. Websockets use pushing [13] to enable bidirectional communication. According to the websocket protocol rfc [3] the opening and closing handshake looks like listing 3 and listing 4

```
GET /chat HTTP/1.1
Host: server.example.com
Upgrade: websocket
Connection: Upgrade
Sec-WebSocket-Key: dGhlIHNhbXBsZSBub25jZQ==
Origin: http:/example.com
Sec-WebSocket-Protocol: chat, superchat
Sec-WebSocket-Version: 13
```

Listing 3. Websocket client handshake [3]

```
HTTP/1.1 101 Switching Protocols
Upgrade: websocket
Connection: Upgrade
Sec-WebSocket-Accept: s3pPLMBiTxaQ9kYGzzhZRbK+xOo=
Sec-WebSocket-Protocol: chat
```

Listing 4. Websocket server handshake [3]

Websockets keep the connection alive by sending a keep-alive message every ten seconds, this message is four bytes in size. [3] Every time data is send to the server or client the data is wrapped in a so called websocket frame that has just two bytes of overhead. What happens if 1000 clients are simultaneously polling every second? The network throughput will then be $(2*1000) = 2000$ bytes $= 16000$ bits per second. That is 0.015 Mbps.

4 IMPLEMENTATION

Now we will discuss and give examples on how the different technologies are implemented on client side. Writing regular AJAX or Comet code can be a lot of work, because different browsers have different syntax for AJAX implementation. This means that you will have to write extra code to test for different browsers. However, the jQuery team [6] has taken care of this, so that we can write AJAX and Comet functionality with only a few lines of code. All our client-side technologies are implemented using the jQuery library to simply the code.

4.1 Ajax

Ajax in combination with jQuery is the most simplest of the three technologies to implement, in this example we do a new AJAX request every two second using the standard Javascipt function setInterval(function,interval).

```
var getData = function() {
   $.ajax({
      url: 'ajax.php',
      dataType: 'json',
      cache: false,
      type: "POST"
   }).done(function( data ) {
      if(text)
         $('#chat').append(data.text);
   });
};

setInterval(getData, 2000);
```

Listing 5. Basic Ajax example with a 1 second interval

4.2 Comet

The Comet front-end code is almost identical to the AJAX example only here there is no need for a setInterval time function. Whenever a AJAX request is complete or failed a new Comet(AJAX) request is made.

```
var listen = function() {
   $.ajax({
      url: 'ajax.php',
      cache: false,
      dataType: 'json',
      type: "POST"
   })
   .done(function( data ) {
      if(text)
         $('#chat').append(data.text);
      listen();
   })
   .fail(function( text ) {
      listen();
   })
   .always(function() {
      listen();
   });
};
listen();
```

Listing 6. Basic Comet example

4.3 Websockets

In this simple websocket example we open up a socket and wait for incomming message's. We used the jQuery plugin 'jquery-websocket'[7] to futher reduce and to simplify the code in the example.

```
var ws = $.websocket("ws://127.0.0.1:8080/", {
   events: {
      message: function(e) {
         $('#chat').append(e.data);
      }
    }
});
ws.send('message', 'hi');
```

Listing 7. Basic WebSocket example

5 CONCLUSION

It is hard to say what technique is the best to use when creating a website that requires Pushing to a mobile phone browser. When only one message is expected it is best technique to use Ajax since other

techniques will be overkill for performing simple tasks.
For applications where a lot of messages are expected for example a chatbox client websocket is the best method to use. The amount of data that is send with websockets over the mobile network is very small in comparison with the other two techniques. However websockets are not widely supported by all mobile phone browsers. It could therefore be an obstacle for web developers to use websockets.

Now we have given three techniques which can be used to display information on a mobile phone website without needing to refresh the entire web page, it is up to the web developer which technique best fits the requirements of the mobile phone website. With this paper we hope to help web developers to make the choise for a technique a little bit easier.

REFERENCES

[1] S. Awamleh. Polling. http://whatis.techtarget.com/definition/polling, April 2005.
[2] Chromium. Blink: A rendering engine for the chromium project. http://blog.chromium.org/2013/04/blink-rendering-engine-for-chromium.html, April 2013.
[3] I. E. T. Force. The websocket protocol. http://tools.ietf.org/html/rfc6455, December 2011.
[4] @Fyrd. Web sockets can i use it. http://caniuse.com/websockets, March 2013.
[5] Google. Google talk. http://www.google.com/talk/, Januari 2011.
[6] jQuery. jquery framework. http://www.jquery.com, April 2013.
[7] jQuery. jquery websocket framework. https://code.google.com/p/jquery-websocket/, April 2013.
[8] kissmetrics. How loading time affects your bottom line. http://blog.kissmetrics.com/loading-time, April 2011.
[9] P. Lubbers and F. Greco. Html5 web sockets: A quantum leap in scalability for the web. http://www.websocket.org/quantum.html, November 2009.
[10] magomimmo. Tutorial 10 - introducing ajax. https://github.com/magomimmo/modern-cljs/blob/master/doc/tutorial-10.md, March 2013.
[11] Opera. Opera gears up at 300 million users. http://business.opera.com/press/releases/general/opera-gears-up-at-300-million-users, February 2013.
[12] O'Reilly. What is web2.0. http://oreilly.com/pub/a/web2/archive/what-is-web-20.html, September 2005.
[13] M. Rouse. Push (or server-push). http://whatis.techtarget.com/definition/push-or-server-push, March 2011.
[14] Seneler, Basoglu, and Daim. Interface feature prioritization for web services: Case of online flight reservations. *Computers in Human Behavior*, 25(4):862–877, July 2009.
[15] Wikipedia. Push technology. http://en.wikipedia.org/wiki/Push_technology#Long_polling.

Verification of SAX assumption: time series values are distributed normally

Harm de Vries, Herbert Kruitbosch

Abstract—In this paper we verify an assumption made in a recently proposed representation for time series: Symbolic Aggregate ApproXimation (SAX). We outline the importance of representation of time series and give a short survey on time series representations. Opposed to other representations, SAX is a *symbolic* representation that also allows for *tight* lower bounding of a similarity measure on a reduced representation with respect to the original raw representation. We point out that this latter property is important for indexing of time series. Furthermore, we show that a tight lower bound in SAX correspond to well chosen breakpoints. Namely, breakpoints according to the distribution of the time series values. When SAX was proposed, the authors defined these breakpoints according to the standard normal distribution, since they claimed that all normalized time series are Gaussian. We verify this claim on the UCR time series database for two specific mining tasks: indexing and anomaly detection. Finally, we show that this claim holds for indexing of time series, but *not necessarily* for anomaly detection.

Index Terms—time series, representations, anomaly detection, indexing, mining tasks, SAX, breakpoints, normal distribution, Gaussian distribution.

1 Introduction

A time series is a sequence of data points measured at successive points in time. To date a tremendous amount of time series data is generated by various applications e.g. heart beat monitoring, daily stock prices, waterheights readings from sensors, etc. As a consequence, there has been a growing interest in the analysis of time series, i.e. extracting useful information from the data. Examples of such analysis tasks are classification, clustering, querying and detection of anomalies in time series.

A challenging aspect in all these tasks is that a time series is often very large. Since each point in time is often considered as a feature of the time series, this *raw* representation, which stores all these points, results in a high-dimensional vector. Researchers avoid working with a raw representations of time series for three main reasons. Firstly, the features of a time series are typically highly correlated and may contain high levels of noise [4]. Therefore the exact values of the data points in a time series are by far less important than global features, such as trends, patterns and shapes. Secondly, the raw representation of the large data set often does not fit into main memory. Consequently, most of the execution time of the mining algorithm will be spent on moving data from the disk into main memory and vice versa. This is highly inefficient and a *reduced* representation that does fit into main memory can easily speed up the algorithm. Thirdly, beside computational and memory issues associated with high dimensional vectors, the concept of "most similar to" vanishes in high-dimensional spaces. The reasons is that as dimensionality increases, the similarity between feature vectors become so large that we can not discern between the closest and furthest distance anymore. This effect is known as the curse of dimensionality[7]. However, the *intrinsic* dimensionality of a time series is often much lower, and the similarity between two time series based on a reduced representation can allow for a reasonable comparison that does not suffer from the curse of dimensionality. In this paper we focus on the second problem of a raw representation.

- *Harm de Vries is a Msc. student at the university of Groningen, E-mail: mail@harmdevries.com.*
- *Herbert Kruitbosch is a Msc. student at the university of Groningen, E-mail: herbertkruitbosch@gmail.com.*

Manuscript received 31 March 2008; accepted 1 August 2008; posted online 19 October 2008; mailed on 13 October 2008.
For information on obtaining reprints of this article, please send e-mailto:tvcg@computer.org.

Due to the importance of a good representation in this process, much literature has focused on representations for time series[11, 16, 10, 4]. In this paper we verify an assumption made in a recently proposed symbolic representation, called Symbolic Aggregate approXimation (SAX) [15], which has gained much attention over the last years. The SAX representation assumes that a time series is distributed Gaussian. We will test this assumption for both indexing, a task posed on many time series, and anomaly detection, a task posed on a single time series.

This paper has been organised in the following way. In section 2 we give a brief overview of all representations, and show that SAX distinguishes itself by its symbolic representation. Furthermore we outline the importance of *lower bounding* for representations in indexing of time series, and show that SAX is suitable representation for indexing and anomaly detection. In section 3 we show that the assumption made by SAX holds for indexing but *not* for anomaly detection. Section 4 concludes the paper and gives possible future directions.

2 Background and Related Work

Although the previous section gave an introduction to time series and some of the problems they face regarding working with a raw representation, we only shortly mentioned mining tasks for which these representations are important. This section will first provide background regarding such mining tasks which use time series as input. Then we discuss the difference between continuous and symbolic representations, discuss some important continuous representations and review a recently proposed symbolic representation. Finally we show how dissimilarities in this symbolic representation are related to dissimilarities in a raw representation.

2.1 Mining Tasks

We consider four of many mining tasks which are, according to the data mining community, interesting [12]:

Indexing Given a query time series Q, and some dissimilarity measure $D(Q,C)$, find the most similar time serie S in database DB.

Clustering Find natural groupings of the time series in database DB under some dissimilarity measure $D(Q,C)$.

Classification Given an unlabeled time series Q, assign it to one of the predefined classes according to dissimilarity measure D.

Anomaly detection Given a time series Q, find the subsequence C which is most dissimilar, according to dissimilarity measure D, to the closest other subsequence.

All these mining tasks on time series have in common that they use a dissimilarity measure, often called distance measure, to define how alike two time series are. It is important to note that this paper focus on the effects of different representations on dissimilarities and will not discuss any specific similarity measure. However, the representations we discuss all support the most commonly used similarity measures as *euclidean distance* or *dynamic time warping*[5]. Furthermore, we verify assumptions made in SAX with respect to two mining tasks: indexing and anomaly detection.

2.2 Continuous and symbolic representations

As mentioned before, although most classical representations are continuous, the recently introduced representation is symbolic. A continuous representation is allowed to attain any (continuous) real value or a value in a continuous part R of all real values, whereas a symbolic representation is only allowed to attain a countable and finite amount of *symbols*. Typically we can define a time series in a given representation as the vector $(t_1, t_2, \dots, t_n)$, then $t_i \subseteq R \subseteq \mathbb{R}$ for a continuous representation and $t_i \in \{s_1, s_2, \dots, s_m\}$ for a symbolic representation for $1 \leq i \leq n$. This set of symbols $\{s_1, s_2, \dots, s_m\}$ is called the alphabet and is finite. So, the essence is that t_i is either continuous or discrete for continuous and symbolic representations respectively.

2.3 Continuous representations

We first focus on continuous representations before we review the symbolic representation SAX. We follow the classification made by Ding et al.[5] and show an overview of the major techniques in figure 1. The main categorization is based on the division in *data adaptive* and *non-data adaptive* methods. Data adaptive methods choose a common representation of the data such that the global reconstruction error is minimized. In contrast, non-data adaptive consider local properties and construct a approximation accordingly. As an example, in figure 2 one can see that the building blocks of the non-data adaptive methods as Discrete Wavelet Transform (DWT), Discrete Fourier Transform (DFT) and Piecewise Aggregate approximation (PAA) are always the same, no matter what the nature of the input data is. On the other hand, data adaptive methods such as PLA change their building blocks to the data. For example the width and slope of the linear segments of Piecewise Linear Approximation (PLA) depend on the data as shown in figure 2.

2.3.1 Discrete Fourier Transform

The first technique suggested for dimensionality reduction of time series was the Discrete Fourier Transform (DFT) [2]. The basic idea of spectral decomposition is that we can represent any signal by the superposition of a finite number of sine and cosine waves of different frequencies. With each wave we associate a single complex number known as the Fourier coefficient. In this way we can transform a signal from the time domain to the so-called frequency domain without loss of information. However, many of the Fourier coefficients are often very small and therefore do not contribute much to the signal. We can discard these coefficients to reduce dimensionality of the signal and thus save storage space, yet preserving most of the information in the data. An example of the DFT representation is shown in figure 2.

2.3.2 Discrete Wavelet Transform

Wavelets are mathematical functions that represent data in terms of the averages and differences of a prototype function, called the analyzing or mother wavelet [1]. In this sense, the wavelet transform is closely related to the Fourier transform. However, a fundamental difference is that wavelets are *localized in time*. In other words, all fourier coefficients represent global information about the signal, whereas some wavelets coefficients portray information about very small, local subsegments of the time series. This property of wavelets allows for

- **Data adaptive**
  - Piecewise Linear Approximation (PLA) †
  - Adaptive Piecewise Constant Approximation (APCA) †
  - Singular Value Decomposition (SVD) †
  - Symbolic
    - Non-lowerbounding strings
    - SAX †
    - Clipping †
  - Trees
- **Non-Data adaptive**
  - Discrete Wavelet Transform (DWT) †
  - Random mapping
  - Spectral
    - Discrete Fourier Transform (DFT) †
    - Discrete Cosine Transform (DCT) †
    - Chebyshev Polynomials †
  - Piecewise Aggregate Approximation (PAA) †

Fig. 1. Hierarchical overview of representations, where † denotes the support for lower bounding.

multi-resolution analysis of data. The first coefficients represent a very coarse view on the data, while latter coefficients can be thought of as zooming in to a specific region of the signal. For wavelets we can reduce dimensionality in similar way as for the Fourier transform by discarding coefficients with low values. We show an example in figure 2 based on the Haar mother wavelet.

2.3.3 Piecewise Aggregate Approximation

The symbolic representation which will be discussed in section 2.4 uses Piecewise Aggregate Approximation (PAA) as a basis. PAA reduces the data by partitioning the time series into N disjoint frames of equal size[16]. The representation is obtained by taking the mean of all data points falling in each frame. This representation has two trivial reductions. First, when the number of frames is equal to the length of the time series, we keep the original time series. Second, when $N = 1$, we obtain the mean of the signal. One of the main advantages is that a PAA representation is very fast to compute.

2.3.4 Piecewise Linear Approximation

In Piecewise Linear Approximation we represent the time serie by K linear segments[11]. Since K is typically much smaller than the number of data points N, we reduce the dimensionality of the signal. Since an "optimal"K can be hard to determine, some implementations work with a user specified maximum linearization error ε. In this way, a datapoint is at most ε away from the linear segment.

2.4 Symbolic Aggregate approXimation

All representations we considered so far are continuous. This limits the set of algorithms, data structures and definitions available for them. For example, in anomaly detection we cannot meaningfully define the probability of observing any particular set of wavelet coefficients, since the probability of observing any real number is zero. Such limitations have lead researchers to consider using a symbolic representation of time series. There have been many symbolic representations proposed over the years e.g. SDA[3], IMPACTS[9]. We focus here on SAX which is, to the best of our knowledge, the only symbolic representation that allows for lower bounding[]. We discuss the importance of this latter property in section 2.5.

An essential preprocessing step for SAX is that a time series is

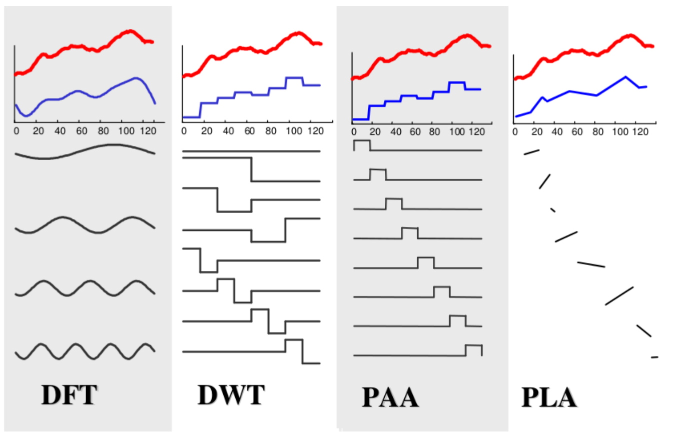


Fig. 2. The red curve is an example of a time series, the blue curve represents the same curve using a high-level representation: DFT, DWT, PAA and PLA. The black curves identify the building blocks of the high-level representation. Therefore the blue curve is a superposition of the black curves.

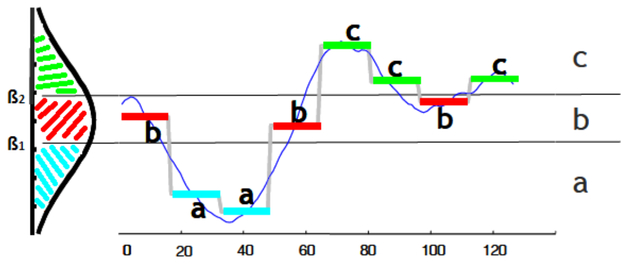


Fig. 3. The dark blue curve is a time series and is assumed to be distribution with respect to the Gaussian distribution shown along the y-axis. The breakpoints partitions this distribution in equi-probable regions shown in green, red and light blue. The PAA representation of this time series is shown as a grey curve, although each window is coloured cyan, red or green, depending on the value of the PAA coefficient. The SAX representation then assigns **a**, **b** and **c** to the cyan, red and green windows, respectively. Image taken from [15].

normalized to zero mean and standard deviation of one, since it is not meaningful to compare two time series with different mean and/or amplitude. As an intermediate step the preprocessed time series is transformed into a Piecewise Aggregate Approximation representation by dividing the time series into w equal sized frames, and assigning to each frame the mean value of all data points in that frame. Finally we create a symbolic representation by mapping real-valued PAA values into discrete symbols. Information loss is inherent to this mapping, since it is impossible to determine the PAA real-value from a SAX symbol. However, from the many mappings possible, the information loss is minimal when symbols occur with equal probability. This argument comes from the field of information theory which states that the uniform distribution maximizes the entropy for discrete distributions[8]. For example, consider the extreme case in which all PAA values are mapped to the same out of five symbols. In this way we lose all information contained in the original data, since all time series are mapped to the same sequence of symbols. Therefore any shape, trend or pattern visible in the PAA representation will be lost in the SAX representation. In other words, each time series representation will be exactly the same, and hence no meaningful dissimilarity measure exist. In general, shapes and trends become less visible if the distribution becomes less uniform.

As a consequence we need to know the underlying distribution of the data to be able to split it into equal partitions. Lin et al. claim that a set of multiple normalized time series have a highly Gaussian distribution[15]. Therefore they determine breakpoints that will produce equal-sized areas under a Gaussian curve. We properly define both the alphabet and these breakpoints by the following definitions.

Definition 2.1. *The ordered set* $\Sigma = (\alpha_0, ..., \alpha_{a-1})$ *is called the* ***alphabet of cardinality*** $|\Sigma| = a$.

Definition 2.2. *The sorted numbers* $B = \beta_1, ..., \beta_{a-1}$ *are called* ***normalized Gaussian breakpoints*** *for an alphabet of cardinality a if the area under Gaussian curve* $\mathcal{N}(0,1)$ *from* β_i *to* $\beta_{i+1} = \frac{1}{a}$. *As a consequence of this definition,* $\beta_0 = -\infty$ *and* $\beta_a = \infty$.

The exact values of these breakpoints depends on the user-specified cardinality of the alphabet a, i.e. the number of symbols. We can simply find these values in a statistical table. For example, for $a = 3$ we have the following breakpoints: $\beta_0 = -\infty, \beta_1 \approx -0.43, \beta_2 \approx 0.43, \beta_3 = \infty$. In figure 3 we show a Gaussian curve and the corresponding breakpoints β_1 and β_2. These breakpoints divide the curve into a cyan, red and green area which are of equal size.

Given these breakpoints we construct the symbolic representations in the following way. All PAA coefficients smaller than or equal to the breakpoint β_1 are mapped to symbol **a**. All *other* PAA coefficients smaller than or equal to breakpoint β_2, which are larger than β_1, are mapped to symbol **b**, etc. If we recapture this formally, a PAA symbol $\hat{c}$ is mapped to a symbol α_j if and only if $\beta_j < \hat{c} \leq \beta_{j+1}$. In figure 3 we show an example for an alphabet size of 3. In general we map a set of PAA coefficients into a word in the following way.

Definition 2.3. *Given a sequence of PAA coefficients* $\bar{C} = \bar{c}_1, \ldots, \bar{c}_n$*, and a set of breakpoints* $B = \beta_1, ..., \beta_{a-1}$*. Furthermore, let* α_j *denote the jth element of the alphabet with cardinality a, i.e.* $\alpha_1 =$ **a**, $\alpha_2 =$ **b**, *etc. Then the set of PAA coefficients* $\bar{C}$ *is mapped into a* ***word*** $\hat{C} = \hat{c}_1, \ldots, \hat{c}_n$ *by*

$$\hat{c}_i = \alpha_j \quad \mathit{iff} \quad \beta_j < \bar{c}_i \leq \beta_{j+1}.$$

In figure 3 we also show an example of the transformation of a PAA representation into a SAX word **baabccbc**. This figure also demonstrates that the PAA representation can take any real value, in particular the light blue windows have different real values. Yet, the SAX representation only attains values **a**, **b** and **c**.

Finally Lin et al. specify a dissimilarity measure on SAX words that lower bounds the similarity on the raw representation. We omit further details and refer the interested reader to [15], but conclude that this property allows for fast indexing of time series as discussed in the next section.

2.5 Indexing and lowerbound

In the previous section we pointed out that SAX is the only known symbolic representation that allows for lower bounding. In this section we explain the importance of this property for indexing of time series.

Faloutsos et al. showed in [6] that a necessary property for fast similarity search is lower bounding of the distance on the representation. This means that for all raw timeseries Q and C, and representations Q' and C' respectively, we have that

$$D_{LB}(Q', C') \leq D(Q, C). \tag{1}$$

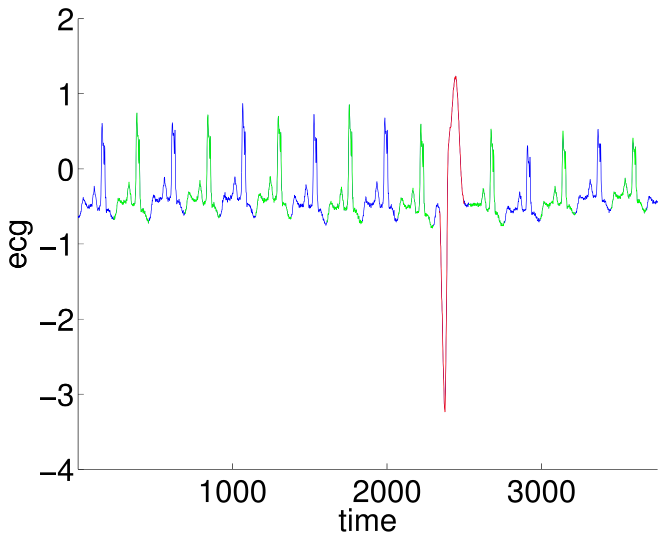


Fig. 4. Plot of the an ECG which contains an anomaly (marked red), taken from the UCR time series data set. All periods in the non-anomaly part are marked in alternating colors.

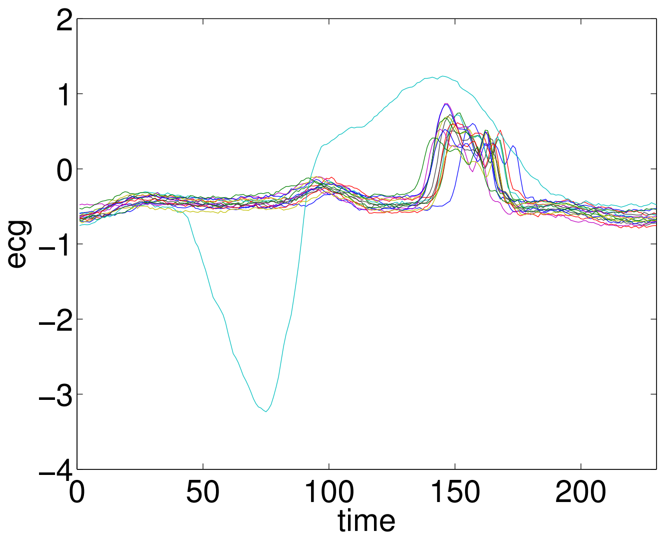


Fig. 5. Plot of ECG periods shown in figure 4. The curve that is clearly different from the others is the anomaly.

In other words, we can guarantee that the dissimilarity between the representations of the time series $D_{LB}(Q',C')$ is *less or equal* than the dissimilarity in the raw time series $D(Q,C)$.

To see that this can indeed give a speed up for indexing time series, recall that the whole time series database is too large to fit into main memory. Hence, the naive approach by simply comparing two time series would result in slow processing since we have to move time series between disk and main memory many times. So, a better approach would be to have two versions of the data. A reduced representation that does fit into main memory and a raw time series that is saved on the disk. Then, to find the most similar time series, we iterate over the representations in main memory. Now, for each time series representation we can safely discard the time series when the dissimilarity over the reduced representation is greater than the dissimilarity to the most similar time series found so far. This way, we hope to make as few disk calls as possible, but we can still guarantee that we find the same solution as when we would iterate over the original time series. We show pseudocode in Algorithm 1. The astute reader might have already noticed that we could always obtain a trivial lower bound on the representation by setting the dissimilarity between every time series representation to zero. However, this would not speed up algorithm 1, since we still have to make a disk call for every time series in the loop. Hence, a useful measure for the performance of the representation is the *tightness* of the lower bound TLB, defined by

$$TLB = \frac{D_{LB}(Q',C')}{D(Q,C)} \tag{2}$$

In words, it is the ratio between the distance of the representations and the distance of the time series in the original space. By the lower bounding property of equation 1, we know that the tightness of the lowerbound is at most one. In this case the distance on the representation and the raw time series are equal, and the algorithm would not have to make any disk call.

This important result have highly influenced the choice of representations over the years. In figure 1 we denote the representations that support lowerbounding with a †, and point out that these representations became popular because they support lower bounding.

Algorithm 1 *Fast* similarity search using lowerbounding

```
Require: timeserie $Q$
    time series database $DB$
Ensure: Time serie $S$ such that $D(Q,S)$ is minimal

    Representation $Q'$ of $Q$
    Representation $S'$ of all $S \in DB$ in main memory

    bestSoFar = $\infty$
    bestS = $NaN$
    for $S' \in DB$ do
        if $D_{LB}(S',Q') < bestSoFar$ then
            Retrieve original S from disk
            if $D(S,Q) <$ bestSoFar then
                bestSoFar = $D(S,Q)$
                bestS = $S$
            end if
        end if
    end for
```

A natural question that comes to mind is which representation is most suitable for indexing. In Ding et al. [5], an extensive comparison of all lower bounding representations is performed based on the tightness of lowerbound. The results are quite suprising. Over all datasets there is not much difference between representations in terms of pruning power. However for data sets with specific characteristics there are some differences. For example SAX performs slightly worse on periodic data than the spectral representations as DFT. We show in this paper that we can improve SAX for periodic data by assuming another distribution on which the breakpoints are based.

2.6 Anomaly detection

In anomaly detection we typically extract all subsequences of a user-specified length in the time series. These subsequences may and in many cases will overlap. The anomaly is defined as the subsequence which is most dissimilar to the closest subsequence. However, we must be careful, since matches to a subsequence tend to be located few, i.e. one or two, points to the left or the right of the subsequence in question[13]. We exclude these matches to meaningfully define an

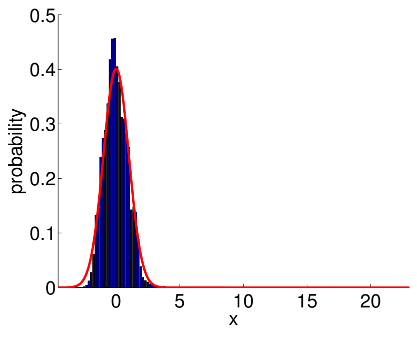


Fig. 6. The blue bars form a histogram of all training data in the UCR, each time series in this data set was z-normalized with respect to the mean and variance of that particular instance of a time series. Ultimately, the histogram of all these values was scaled to have an area of 1. The red curve is a Gaussian approximation $\mathcal{N}(0, 0.995)$ of this histogram.

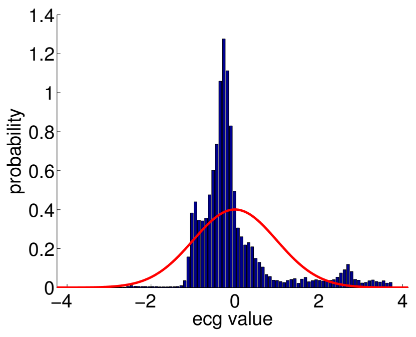


Fig. 7. A histogram of all ECG windows of length 230 in the UCR data set, the red curve is a normal approximation $\mathcal{N}(0, 0.995)$ of this histogram. The histogram is scaled to have an area of 1. This histogram is generated with respect to algorithm 3.

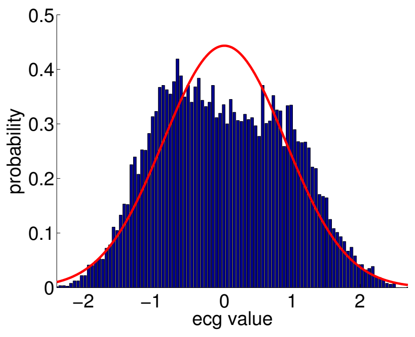


Fig. 8. Distribution of the ecg-values in all z-normalized windows of length 10 in the ECG time series from the UCR data set (blue bars) and a Gaussian approximation $\mathcal{N}(0, 0.9)$ (red curve). The histogram area is scaled to have an area of 1. This histogram is generated with respect to algorithm 3.

anomaly. More formally we first define a subsequence and a non-self match:

Definition 2.4. *Given a time series $T = (t_1, \dots, t_m)$ of length m, a* ***subsequence*** *C* ***of*** *T is a sampling of length n with $1 \leq n \leq m$ of continuous position from T, that is, $C = t_p, \dots, t_{p+n-1}$ for $1 \leq p \leq m - n + 1$.*

Definition 2.5. *Given a time series T, containing a subsequence C of length n beginning at position p and a subsequence M beginning at q, we say that M* ***is a non-self match to*** *C at dissimilarity of $D(M, C)$ if $|p - q| \geq n$.*

The brute force algorithm requires a loop over all subsequences, and for each of these subsequence we have to loop over all non-self matching subsequences to determine the most similar subsequence. Then the anomalous subsequence is the subsequence in the outer loop with the largest dissimilarity. So, an anomalous subsequence is the subsequence where the minimal dissimilarity regarding all non-self-matching subsequences is maximal. We show pseudocode for this process in algorithm 2. The astute reader might already have noticed numerous speed ups in algorithm 2, e.g. by cleverly ordering of the subsequences in the inner and outer for-loop. In HOT SAX[13] the authors proposed a heuristic ordering in the outer- and inner-loop based on the SAX representation of subsequences and a suffix tree. However, this algorithm is outside the scope of this paper and we refer the interested reader to [13]. We emphasize that the success of HOT SAX is highly depended on tightness of the lower bound i.e. the determined breakpoints in the underlying distribution of the time series.

We conclude this section with a typical example of an anomaly marked as the red subsequence in the electroencephalographic (ECG) plot in figure 4. Figure 5 shows how equal these periods are.

Algorithm 2 Anomaly detection

```
Require: Timeserie $T = (t_1, \dots, t_m)$
    anomalous subsequence length $n$
Ensure: Most anomalous subsequence $C$

  bestSoFarDist = 0
  bestSoFarLoc = 0
  for $p = 1 \to |T| - n + 1$ do
    NNDist = $\infty$                          ▷ Nearest Neighbour distance
    for $q = 1 \to |T| - n + 1$ do
      if $|p - q| \geq n$ then                         ▷ Non-self match
        if $D((t_p, \dots, t_{p+n-1}), (t_q, \dots, t_{q+n-1})) <$ NNDist then
          NNDist = $D((t_p, \dots, t_{p+n-1}), (t_q, \dots, t_{q+n-1})$
        end if
      end if
    end for
    if NearestNeighbourDist > bestSoFarDist then
      bestSoFarDist = NearestNeighbourDist
      bestSoFarLoc = p
    end if
  end for
```

3 Improving SAX for anomaly detection

We have seen in the previous section that an important property of a representation for indexing time series is lower bounding. Furthermore, we have shown that SAX is the only known symbolic representations that allows for tight lower bounding, and thus is a suitable representation for indexing. In this section we verify Lin et al.'s claim that breakpoints for SAX can be best determined by a Gaussian distribution for *all* mining tasks, including anomaly detection.

3.1 Indexing

We first focus on indexing of time series where we are looking for the most similar time series in a large database of heterogeneous time series. In order to verify the underlying distribution of most time series data, we have to select a database of time series. We choose to use the UCR time series database [14] since it is considered to be the largest publicly available test data set for time series. The data set consist of 42 diverse data sets, including heart beat monitoring and diatom recognition, grouped into two parts. For computational reasons we selected only the first part of twenty datasets. All datasets are divided into a training set and test set, since the data sets are mainly used for classification and clustering. However, we argue that we can also use the data set to verify the underlying distribution for indexing because the *only* difference is that indexing datasets are much larger. Furthermore, we assume that a larger data set more likely cause a Gaussian distribution of the time series values.

We verified the Gaussian distribution hypothesis of SAX by selecting all normalized time series from the training sets, and plotting the distribution of all data points. We show the result in figure 6 in which we also plot an almost standard normal distribution. We can clearly see that the distribution of normalized time series resembles this normal distribution. This result is not surprising, since the data set consist of diverse sets of time series. Therefore each time series on its own will have a highly non-Gaussian distribution, but on average it will tend to a Gaussian distribution. This strong result is known as the principle of maximum entropy [8] which states that if we do not know the underlying distribution of the data we should choose the distribution with the largest entropy. It is well known that for continuous distributions with known mean and variance this is the Gaussian distribution [8]. Hence we conclude that a Gaussian distribution is appropriate to select the breakpoints for SAX in indexing of time series, since indexing considers a set of time series.

3.2 Breakpoints for anomaly detection

In the previous section we have shown that the underlying density is highly Gaussian for indexing of time series. However, the key insight was that we could expect any type of time series. Hence, the best we can do is to choose the distribution that contains the most information i.e. the Gaussian distribution. This is not the case for anomaly detection, in the sense that we only have one specific time series. For one such time series, intrinsic knowledge and empirical measurements may suggest a different distribution.

We verify the underlying distribution of the time series data on the electroencephalographic (ECG) data shown in figure 4. This time series is periodic, shown right in the figure, but contains one anomaly. Algorithm 3 determines the empirical distribution of all subsequences of a certain length. Although we can not make any statistical claims based on the distribution of only one particular time series, we emphasize that this is an illustrative example which may generalize well to most other periodic time series.

To verify the Gaussian distribution hypothesis in anomaly detection we selected all subsequences of various lengths in the ECG time series and plotted the empirical distribution. We visually verified whether the underlying distributions resembled a Gaussian distribution. For small window sizes up to length 40 we noticed that the distribution is still close to a Gaussian distribution. An example for window size of 10 is shown in Figure 8. However, for larger window sizes that are close to the period of the time series the distribution is clearly not Gaussian anymore. We can see this in figure 7 where we show the distribution of normalized subsequences of length 230.

Algorithm 3 Determine the empirical distribution of all values in all windows in a time series

```
Require: Time series $T = (t_1, \dots, t_m)$
    Subsequence size $n$
Ensure: Empirical distribution (or histogram) $E$

  $C$ = empty list, which will grow to size $n \cdot (|T| - n + 1)$
  for $i = 1 \to |T| - n + 1$ do                 ▷ Iterate over all subsequences
    $Sub = (t_i, \dots, t_{i+n-1}$                ▷ One subsequence, starting at $i$
    $\mu = \overline{Sub}$                          ▷ Mean of $Sub$
    $\sigma^2 = \sum_i (Sub_i - \overline{Sub})^2$  ▷ Standard deviation of $Sub$
    for $Sub_i \in Sub$ do
      $t_i = \frac{Sub_i - \mu}{\sigma}$            ▷ $Z$-normalize $t_i$
      Append $t_i$ to $C$
    end for
  end for
  $E$ = histogram of all values in $C$.
```

4 Conclusion and future work

In this work we have investigated the symbolic representation SAX. The lower bound property of this representation turns out to be very important for the efficiency of indexing of time series. SAX obtains a symbolic representation based on the underlying distribution of a (set of) time series. In this paper we verified the claim that this distribution resembles a standard normal distribution. We have both shown empirically on the UCR time series database and theoretically that for indexing of time series a Gaussian distribution is a logical choice. This result and the lower bound property make SAX a very effective representation for indexing of time series.

For anomaly detection, however, we have shown that the distribution of time series values are *not* Gaussian distributed. We provided an ECG time series taken from the UCR data set for which the assumption that it is distributed Gaussian is invalid. Hence using Gaussian breakpoints in a SAX representation may be suboptimal for anomaly detection. Future research may conclude that breakpoints based on the empirical distribution of the time series may better capture the information contained in the time series. These studies may incorporate testing this hypothesis on a larger data set suited for anomaly detection by comparing results where Gaussian and non-Gaussian breakpoints are used. Moreover, such research may also evaluate if these non-Gaussian breakpoints also correspond to tighter lower bounds. Finally, forthcoming studies may assess if our finding that Gaussian breakpoints are not necessarily valid can be extended to other mining tasks, like motif discovery, which only consider one single time series.

References

[1] *Front Matter*, chapter 0, pages i–xix.

[2] R. Agrawal, C. Faloutsos, and A. Swami. Efficient similarity search in sequence databases. pages 69–84. Springer Verlag, 1993.

[3] H. Andre-Jonsson and D. Z. Badal. Using signature files for querying time-series data. In *First European Symposium on Principles of Data Mining and Knowledge Discovery*, pages 211–220, 1997.

[4] K. Chakrabarti, E. Keogh, S. Mehrotra, and M. Pazzani. Locally adaptive dimensionality reduction for indexing large time series databases. *ACM Trans. Database Syst.*, 27(2):188–228, June 2002.

[5] H. Ding, G. Trajcevski, P. Scheuermann, X. Wang, and E. Keogh. Querying and mining of time series data: experimental comparison of representations and distance measures. *Proc. VLDB Endow.*, 1(2):1542–1552, Aug. 2008.

[6] C. Faloutsos, M. Ranganathan, and Y. Manolopoulos. Fast subsequence matching in time-series databases, 1994.

[7] J. Friedman. On bias, variance, 0/1loss, and the curse-of-dimensionality. *Data Mining and Knowledge Discovery*, 1:55–77, 1997.

[8] R. M. Gray. *Entropy and information theory*. Springer, 2011.

[9] Y.-W. Huang and P. S. Yu. Adaptive query processing for time-series data. In *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '99, pages 282–286, New York, NY, USA, 1999. ACM.

[10] K. Kawagoe and T. Ueda. A similarity search method of time series data with combination of fourier and wavelet transforms. In *Temporal Representation and Reasoning, 2002. TIME 2002. Proceedings.Ninth International Symposium on*, pages 86–92.

[11] E. Keogh, S. Chu, D. Hart, and M. Pazzani. Segmenting time series: A survey and novel approach. In *In an Edited Volume, Data mining in Time Series Databases. Published by World Scientific*, pages 1–22. Publishing Company, 1993.

[12] E. Keogh and S. Kasetty. On the need for time series data mining benchmarks: A survey and empirical demonstration. In *SIGKDD'02*, pages 102–111, 2002.

[13] E. Keogh, J. Lin, and A. Fu. Hot sax: efficiently finding the most unusual time series subsequence. In *Data Mining, Fifth IEEE International Conference on*, page 8 pp., nov. 2005.

[14] E. Keogh, X. Xi, L. Wei, and C. . Ratanamahatana. The UCR Time Series Classification/Clustering Homepage, 2006.

[15] J. Lin, E. Keogh, S. Lonardi, and B. Chiu. A symbolic representation of time series, with implications for streaming algorithms. In *Proceedings of the 8th ACM SIGMOD workshop on Research issues in data mining and knowledge discovery*, DMKD '03, pages 2–11, New York, NY, USA, 2003. ACM.

[16] B.-K. Yi and C. Faloutsos. Fast time sequence indexing for arbitrary lp norms. In *Proceedings of the 26th International Conference on Very Large Data Bases*, VLDB '00, pages 385–394, San Francisco, CA, USA, 2000. Morgan Kaufmann Publishers Inc.