University of Groningen

4th SC@RUG 2007 proceedings

Smedinga, Rein

IMPORTANT NOTE: You are advised to consult the publisher's version (publisher's PDF) if you wish to cite from it. Please check the document version below.

Document Version
Publisher's PDF, also known as Version of record

Publication date:
2007

Citation for published version (APA):
Smedinga, R. (Ed.) (2007). *4th SC@RUG 2007 proceedings: Student Colloquium 2006-2007.* Rijksuniversiteit Groningen. Universiteitsbibliotheek.

SC@RUG 2007 proceedings

Rein Smedinga
editor

2007
Groningen

Contents

Contents

About SC@RUG

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 2006-2007 SC@RUG was organized for the fourth time as a conference. Students wrote a paper, participated in the review process, gave a presentation and were session chair during the conference.

The organizer Rein Smedinga 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. He would also like to thank Femke Kramer from the Faculty of Arts for her help in organizing this course and Janneke Geertsema for her workshops on presentation techniques and speech therapy.

In these proceedings all accepted papers are published.

Organizational matters SC@RUG 2007 was organized as follows. Students were expected to work in teams, consisting of two persons. 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 three papers about the same subject (within Computing Science). Soms sets of papers contained conflicting meanings. 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 own conclusions about the subject.
Some teams proposed their own subject.

After submission of the papers individual students were assigned one paper to review using a standard review form (see Appendix A of the first StudColl2004 proceedings). The colleagues who had provided the set of papers were 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.

All students were asked to present their paper at the conference and act as a chair and discussion leader during one of the other presentations. Half of the participants were asked to organize one of the conference days (i.e., to make the time tables, invite people etc.) The audience graded both the presentation and the chairing and leading the discussion.

Femke Kramer of the Faculty of Arts gave an introductory lecture about general aspects of presentation techniques to help the students with their presentation. She also did a workshop on writing scientific papers. Janneke Geertsema gave workshops on presentation techniques and speech therapy that was very well appreciated by the participants.

Students were graded both on all three aspects: the writing process, the review process and the presentation. Writing and rewriting counted for 50% (here we used the grades given by the reviewers and the re-reviewers), the review process itself for 15% and the presentation for 35% (including 5% for the grading of being a chair or discussion leader during the conference). For the grading of the presentations we used the judgements from the audience and calculated the average of these.

On January 22nd and 23th, the actual conference took place. Each paper was presented by both authors. Both days, we had ten presentations, each consisting of a total of 30 minutes for the presentation and 10 minutes for discussion. As mentioned before, each presenter also had to act as a chair and discussion leader for another presentation during that day. The audience was asked to fill in a questionnaire and grade the presentations, the chairing and leading the discussion. Participants not selected as chair were asked to organize both days. They did an excellent job and even provided coffee and tea and a lunch on both days.

Thanks We could not have achieved the ambitious goal of this course without the invaluable help of the following expert reviewers:
– Marco Aiello
– Jos Roerdink
– Nicolai Petkov
– Ahmad Waqas Kamal
– Anton Jansen
– Paris Avgeriou
– Michel Westenberg
– Ronald van der Berg
– Gerard Renardel

Also, the organizer would like to thank the *School for Computing Science* for making it possible to publish these proceedings.

Rein Smedinga

Transaction Management
A comparison of database and Web Service Transaction Management

G. Biemolt

S1510266

G.A.Biemolt@rug.nl

H. Groefsema

S1514059

H.Groefsema@student.rug.nl

Abstract. Web Services are self-contained units of work, ranging from simple functions to complicated business processes. These Web Services communicate with other Web Services to do business. Such a loosely coupled system is regarded as being a Service Oriented Architecture (SOA). To operate and do business in a consistent way Web Services make use of Transaction Management (TM) to ensure a save and agreed upon outcome of the individual units of work.

The TM used in the Web Services (WS) stack is divided in WS-Atomic Transaction (WS-AT) and WS-Business Activity (WS-BA). WS-AT is inherited from the database TM that is built upon the ACID properties: Atomicity, Consistency, Isolation, and Durability. WS-BA is commonly implemented in an asynchronous way and is different from the more traditional WS-AT. WS TM thus makes use of non-ACID properties due to its loosely coupled architecture.

This paper looks into the four ACID properties inherited from database TM and investigates how they can be implemented in WS TM. WS-AT and WS-BA are to be compared to the ACID properties and this ultimately leads to an explanation for the differences between these standards.

Keywords: Transaction Management, Web Services, Databases, WS-AT, WS-BA, Service Oriented Architecture

1. Introduction

The original ACID standard of database Transaction Management (TM) has been adapted to work in the area of Web Services, due to the absence of TM in this field. This paper focuses on the differences between these types of TM, and why changes had to be made to make it function as needed.

In order to understand the concept of TM it is required to define what a transaction exactly is in this context. 'A transaction reflects the idea that the activities of a particular user are isolated from all concurrent activities, but restricts the degree of isolation and the length of the transaction' [Hear83]. Where an object can be read as a record in a conventional database. Transaction management is needed to keep these objects consistent to prevent other users or services from working with inconsistent or outdated data.

There have been several suggestions for WS TM standards, but this paper focuses on the ones contained within the WS stack [Figure 1]. One of these two specifications is WS-Atomic Transaction (WS-AT), which is based upon the common ACID (described in Section 3) transactions. The other one is WS-Business Activity (WS-BA) that supports loosely coupled asynchronous non-ACID transactions.

The main difference between database transactions and WS transactions is the domain in which they operate. Database transactions occur in a single controlled domain and are synchronous, whereas WS transactions can spread several business domains and are thus loosely coupled and not necessarily synchronous. WS-BA is an example of a specification that fulfils the need for such non-atomic transactions.

This paper starts explaining database TM and what Web Services are in Section 2 and 3. Based on those facts we give the explanation for the need of changing and elaborating on the traditional ACID database transactions in Section 4, which also

holds the description of the WS TM standards, WS-AT and WS-BA. While we finally will give an overview of the major advantages and disadvantages of the elaborated standards and the shortcomings that they still have.

2. Web Services

Web Services (WS) are self-contained units of work. These can range from large business processes to simple functions. The Web Service stack is a technology based upon the concepts of Service Oriented Architecture (SOA), which defines a loosely coupled architecture consisting of services that can call upon each other independent of implementation. They form the building blocks for creating distributed applications[Papa06].
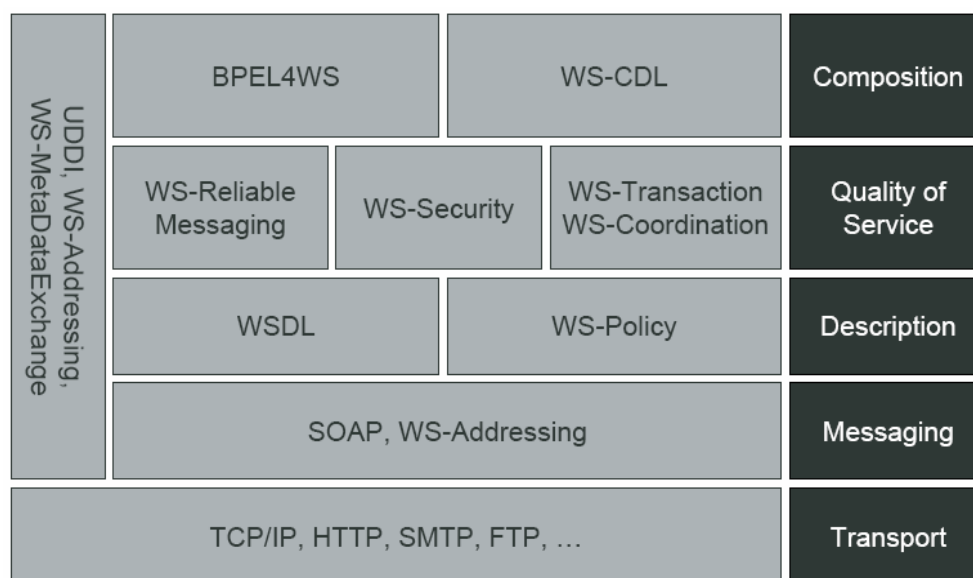
Figure 1: The Web Service stack.

The Web Service stack (Figure 1 [Dust06]) consists of a multitude of standards. Each standard fills a separate, unique and important role in the WS stack and can reside in one or more layers of the Web Service stack. The bottom layer is the Transport layer and consists of well-known transport standards such as TCP/IP, HTTP, SMTP, FTP and more. On top of the Transport layer resides the Messaging layer, this layer consists of the Simple Object Access Protocol (SOAP) and WS-Addressing standards, which allow for transmission. The Web Service Description Language (WSDL) and WS-Policy standards reside one layer up, in the Description layer. These standards provide for means of describing the Web Service and its functions. The Quality of Service layer consists of WS-Reliable Messaging, WS-Security, and WS-Transaction. The latter shall be the focus of Section 4 and provides the means for getting an agreed upon and consistent outcome of a transaction within the Web Services stack. The last layer, the Composition layer, consists of the Business Process Execution Language (BPEL) and the WS-Choreography Description Language (CDL) standards. These form the choreography specification and implementation for the web services [Biegr06]. The standards on the left reside on all the top four layers and consist of the Universal Description, Discovery, and Integration (UDDI) and WS-Addressing standards, which provide means for finding Web Services.

When a Web Service first starts, it needs to be found in some way by the programs that wish to make use of its services. In order to do this, the Web Service is published in an UDDI supported repository. Other programs then search the UDDI to find suitable services for their needs, and use the information in the UDDI to contact

the Web Service. A connection is then made with the Web Service by means of a XML based SOAP connection to overcome possible compatibility problems between the two systems. SOAP again relies on protocols such as HTTP and SMTP for the actual message transportation. SOAP however would require information about the Web Service its interface i.e., its calls, their parameters, and their returns. The WSDL provides this functionality. Each Web Service defines a WSDL document that describes how to invoke a service and provides information on the data being exchanged, the sequence of messages for an operation, protocol bindings, and the location of the service [Papa06]. Through such a connection Transactions can be made between different Web Services.

3. Database Transaction Management

Database transactions are based on the ACID properties. Using these properties will guarantee that transactions do not lock records for a long time, and that the outcome is consistent and expectable. If for whatever reason a problem occurs during the transaction then a recovery action has to be performed to force the transaction to a consistent outcome.

The acronym ACID stands for:
- **A**tomicity: Either all of the changes happen exactly once or none of the changes happen.
- **C**onsistency: State changes of objects in a transaction should be consistent before and after the transaction.
- **I**solation: During a transaction the effects of an object are not effected by other concurrent operations on other objects and intermediate states are not externally visible.
- **D**urability: The result of a transaction should persist and not be undone.

These requirements ensure that transactions can be executed in parallel and have an expected consistent outcome. A recovery action should be taken if a transaction meets a failure. Such an action has to make sure that the outcome remains consistent and atomic. Thus the actions that have been executed before the failure are undone and the transaction rolls back to the initial state.

There are three type of failures [Hear83]. A *transaction failure* occurs when a transaction does not commit due various reasons related to the data in the system, e.g. aborting the transaction or exceeding timing constraints. *System failure* occurs when there is fault in the code of the database, a system fault or hardware problem. If there is a problem writing to the disk(s) or fatal crashes in the hardware it is called a *media failure*. There are several ways of anticipating these failures [Hear83]. But these are outside the scope of this paper.

The most well known transaction protocol in the world of database transactions is the Two Phase Commit (2PC) Protocol. 2PC can be used for both single and composite transactions. It is based on the ACID properties [Gran99] and works in two phases. At first it determines if the **change** can be executed. If it is a single transaction, it can be committed or aborted directly, which makes the transaction end in a consistent state. In a composite transaction, all the dependant changes must be executed or not, to enforce atomicity. Rollback on the other changes has to be performed if not all changes are able to commit. This results in the initial state in which none of the changes of the transaction have happened. The disadvantage of the 2PC protocol is that it locks the resources while it uses them. Thus the length of the transaction directly relates to the usability of the resource. Other concurrent transactions are more likely to exceed their timing constraints while waiting on the availability of the resource.

4. Transaction Management with Web Services

In the Web Services stack (Figure 1) there are several standards which relate to transactions, but the most important ones are WS-Coordination and WS-Transaction. WS-Coordination defines a framework for coordinating the actions of distributed applications via context sharing [Papa03], where WS-Transaction defines the actual protocols used in the transaction.

WS-Coordination offers a framework for coordinating the transaction. In order to do this it offers the following three services:

- An Activation Service, which defines a CoordinationContext which in turn is used to set up the communication between the Web Services.
- A Registration Service that allows Web Services to register at the Coordinator.
- And a Coordination Protocol Services, which holds one of the transaction protocols described in WS-Transaction.

WS-Transaction consists of two transaction protocols: WS-Atomic Transaction (WS-AT) and WS-Business Activity (WS-BA). The first is directly related to the classic Database TM as it is a fully ACID protocol. The second however is only ACID *based* and uses several other mechanisms in order to satisfy the needs of the loosely coupled environment of Web Services.

4.1 WS-Atomic Transaction

WS-Atomic Transaction is a fully ACID transaction protocol. To achieve this it uses the 2 Phase Commit (2PC) protocol as discussed in Section 3. The statechart of the WS-AT 2PC protocol is shown in Figure 2. Due to the constant re-affirming of the protocol it ensures that all participants are in a mutually agreed upon state. This in turn ensures that the protocol is atomic and predictable.
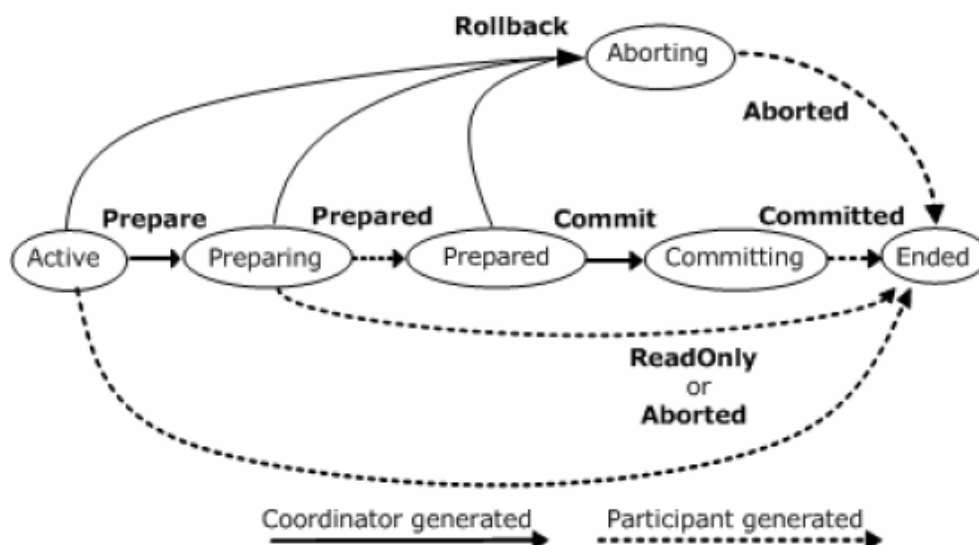
Figure 2: WS-AT State Chart

However, due to the nature of the protocol, this means that such a transaction can only be a short lived (synchronous) transaction. Since it requires all participants to lock the resources required for the transaction, using it for a long running transaction is simply unadvisable and expensive. The locking of resources however is required in this protocol because else the participants can not guarantee a mutually committed outcome after entering the commit state. I.e., the resources could have simply been taken by other concurrent transactions. Since aborting is not possible at this point, a different solution is required for such transactions.

4.2 WS-Business Activity

WS-Business Activity supplies the WS stack with a means of a more flexible transaction protocol that is capable of handling longer-running transactions over different trust domains. This transaction protocol is based upon the ACID properties, but is not fully atomic. This non-atomicity is the result of the fact that it allows selective confirmation (commit) or cancellation (rollback) of participants (even if they are capable of committing) [Papa03]. This again, implies that such a transaction consists of several smaller transactions, which each can be committed or cancelled. Such a transaction can be regarded as a nested transaction.
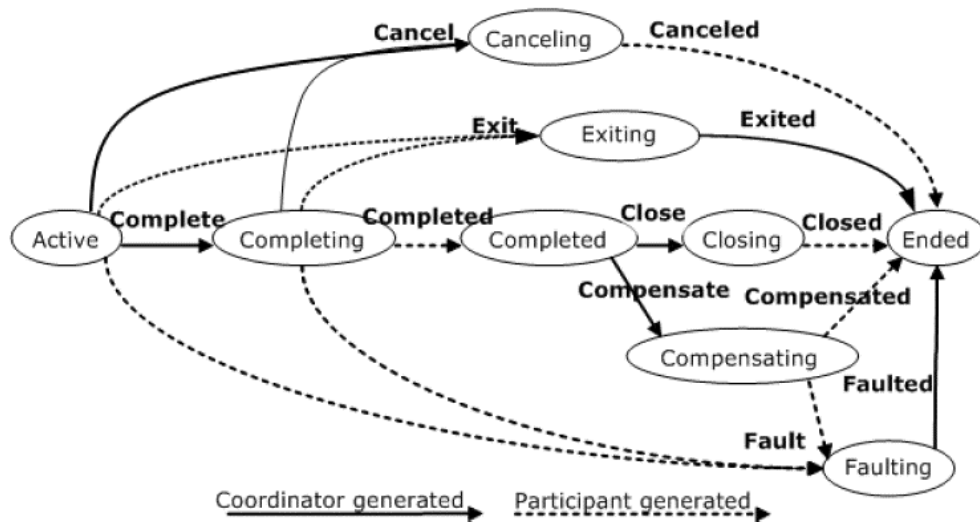
Figure 3: WS-BA State Chart

The transaction protocol of WS-BA is shown in Figure 3. In contrast with WS-AT it uses a compensation state instead of a rollback. This compensate state change allows a transaction to return a participant to its original, or an almost equal to the original, state from before the transaction. This is different from a rollback because a rollback is initiated when the transaction *has not yet* committed its resources, and when a compensating action is initiated the transaction *has* committed at least some of the resources. The compensating action will try to undo the changes made as well as it can.

As mentioned, WS-BA follows some unconventional atomicity properties for business transactions. These atomicity properties can be classified into three categories; *system-level atomicity*, *operational-level atomicity* and *business interaction level atomicity* [Papa03]:
1. System-level atomicity.
   - *Service request atomicity*.
     A single operation occurs completely or not at all.
2. Business interaction level atomicity.
   - *Non-repudiation atomicity*.
     A participant can not deny that the transaction has occurred. This is achieved by signing the content at application level.
   - *Conversation atomicity*.
     This is a non-repudiation atomic primitive that allows a pair of collaborating services to correlate sequences of requests within a logical unit of work [Papa03].
   - *Contract atomicity*.
     If the transaction is successful it is regarded as legally binding.

3. Operational-level atomicity.
   - *Payment atomicity*.
     This atomicity covers the transfer of funds between the participants.
   - *Goods atomicity.*
     Goods atomicity protocols are payment-atomic, and also affect an exact transfer of goods for money. Goods atomicity implies that the (tangible or non-tangible) goods will be received only if payment has been made [Papa03].
   - *Certified delivery atomicity.*
     Goods atomicity guarantees delivery of goods, but not if the right goods were delivered. Certified delivery atomicity allows the participants to prove which goods were delivered.

These types of atomicity are related to different levels of a business transaction. Business transactions can be separated in the above categories for better understanding of the atomicity level in the transaction.

5. Conclusion

The paper shows that the original database transaction management works fine for the simple short lived synchronous transactions in the Web Service environment. The 2PC protocol that is usually used in such transactions, that first prepares a transaction and then finally commits the transaction, however is not useful for long lived (asynchronous) transactions. Due to long term locking of the resources, WS-BA starts immediately with asynchronously completing the transaction and closes it afterwards. This means that the connection is not blocked the entire time. The disadvantage of intermediate states that are visible to the outside world throughout the entire transaction is not a problem, because the transaction can be seen as a composition of two transactions that do not have intermediate states.

Thus WS-BA is based on ACID DB transactions but with adapted atomicity and without isolation. WS-BA supports asynchronous messages, loose coupling and long lived transaction which are not traditionally atomic. Also intermediate states are not allowed by isolation.

The next step in this research could be to investigate how to use transaction management for web services in the real world by implementing a working example or how other transaction protocols and standards relate to the WS-T. Such a protocol could be the Business Transaction Protocol.

References

[Biegr06]   G Biemolt, H Groefsema. Web Service Transactions. 2006.
[Dust06]   S Dustdar. Lecture notes Information Systems. 2006.
[Hear83]   T Hearder, A Reuter. Principles of Transaction-Oriented Database Recovery. In Computing Surveys, Vol.15, No. 4, December 1983.
[Gran99]   M Grand. Transaction Patterns. In PLoP conference 1999.
[Papa03]   M Papazoglou. Web Services and Business Transactions. In World Wide Web: Internet and Web Information Systems, 6, 49-91, 2003.
[Papa06]   M Papazoglou. Web Services Technologies and Standards. In Computing Surveys, 2006.

Real-Time Atmospheric Rendering From Any Viewpoint

Emil Loer and Thomas ten Cate

Rijksuniversiteit Groningen

Abstract We will describe an algorithm that uses modern graphics hardware to render a realistically looking approximation of the atmosphere at interactive frame rates. We will base our method on an algorithm by Dobashi, Yamamoto and Nishita (2002) and adapt it to work equally well from any viewpoint.

Rendering the atmosphere in a realistic way is essential in virtually any rendering of outdoor scenes, from landscapes to pictures of the solar system. The atmosphere in landscapes provides the typical colour gradations in the sky, whereas displaying the atmosphere as seen from outer space adds to the sense of realism. Atmospheric rendering is a difficult problem because of the scattering of light that travels through air or another gaseous medium.

A great amount of work in this field has been done by T. Nishita in cooperation with several other authors, building on earlier work by R. Victor Klassen, amongst others. The focus can either be on physically correct modelling or on creating a visually appealing picture without strict requirements of accuracy, where only the latter is feasible to do in real-time on current hardware. Some authors focus only on rendering the atmosphere from a certain point of view, e.g., only from the ground. Assuming only a limited range of viewpoints is insufficient if one wants to display, for example, the launch of a space rocket. These problems are tackled by the described algorithm.

1 Introduction

Realistic image synthesis is a very important research topic in the field of computer graphics. One of the greatest challenges in this subject is rendering the earth and its surrounding environment. When rendering planets and their landscapes one of the key factors to creating a realistically looking image is the presence of an accurate rendition of the atmosphere.

A phenomenon playing a large role in the lighting of a planet is atmospheric scattering. This is caused by light rays being reflected off dust and other particles that are present in the air. Scattering gives typical blue hues in the sky and reddening at sunset, when viewing from the Earth surface, and blueish tones from outer space. It is also responsible for a small amount of surface illumination.

Rendering this atmosphere can be done using simple approximative techniques such as a flat colour, a simple gradient or a texture created from a photograph. However, in order to archieve a realistically looking atmosphere from

both the surface and from outer space, these techniques will not suffice. On the other hand, when rendered using high quality ray tracing techniques one can not reach interactive frame rates necessary for applications like games and (space) flight simulators. This means a compromise has to be made.

Using the mathematics behind the scattering process we will give an explanation of a proposed real time scattering algorithm by Dobashi et al. [1]. The described algorithm is capable of rendering planetary atmospheres from any point of view, maintaining interactive frame rates at all times.

2   Theory

The nature of light scattering in the air is described in detail by R. Victor Klassen [2]. We summarize the parts of his work that are relevant to this paper, drawing some more recent results from Nishita et al. [3]. Some of the equations do not match theirs exactly; moreover, we added as parameters to functions every symbol that is not constant for the entire atmosphere. This will prove useful later when lookup textures are introduced.

The intensity $I_v(\lambda)$ of the light of wavelength λ that reaches the eye from a certain direction is the sum of two components:

– the intensity $I_r(\lambda)$ of the light reflected by the object we're looking at, attenuated accordingly (if there is no object and we look at infinity, e.g., up into the sky, this intensity will of course be zero), and
– the intensity $I_s(\lambda)$ of the light scattered 'into' the view ray at each point, attenuated accordingly and then integrated over the view ray.

Therefore,

$$I_v(\lambda) = I_r(\lambda) + I_s(\lambda).$$

This is illustrated in the figure below, where the rays have been drawn a little apart for clarity.
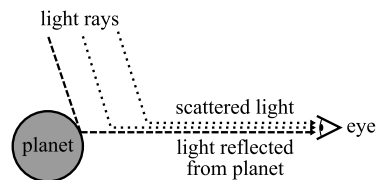
The computation of I_r and I_s will be treated in sections 2.2 and 2.4. First we will discuss the details of light attenuation, needed in both these sections.

2.1   Attenuation

Part of the light that travels through the atmosphere from point A to point B is scattered. This results in *attenuation* of the original light. The fraction g of

light that remains, $g \in [0, 1]$ can be computed using

$$g(AB, \lambda) = e^{-\tau(AB,\lambda)},$$

$$\tau(AB, \lambda) = \int_A^B \beta_R(\lambda)\rho_R(h(t)) + \beta_M(\lambda)\rho_M(h(t))\mathrm{d}t. \qquad (1)$$

The function τ is called the *optical depth*. $\beta_R(\lambda)$ and $\beta_M(\lambda)$ are the extinction coefficients for Rayleigh and Mie scattering respectively; see section 2.3. The integral $\int_A^B$ means we integrate over a straight line segment from A to B.

In the case of a planetary atmosphere, the densities ρ_R and ρ_M are exponential functions of the height h:

$$\rho_R(h) = e^{\frac{-h}{H_R}}$$

$$\rho_M(h) = e^{\frac{-h}{H_M}}$$

H_R and H_M are scaling constants depending on the atmosphere. Note that there is no boundary to the atmosphere; in practice, we will apply a sufficiently large cutoff height above which no air exists.

Using g, we can compute the intensity $I_B(\lambda)$ at the end from the intensity $I_A(\lambda)$ at the beginning from

$$I_B(\lambda) = g(AB, \lambda)I_A(\lambda)$$

2.2 Reflected Light I_r

We now compute how much light is reflected in the viewing direction from the planet we look at. Suppose P_g is the point where the view ray intersects the ground and E is the position of the eye:
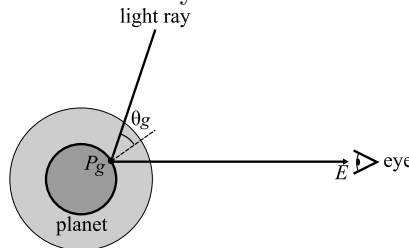
The amount of light $I_{in}(P_g)$ reaching P_g is the light intensity of the sun I_{sun}, attenuated by a factor g_l:

$$I_{in}(P_g, \lambda) = g_l(P_g, \lambda)I_{sun}(\lambda) \qquad (2)$$

For any point P in the atmosphere, we can rotate and translate our coordinate system in such a way that the planet's centre is at the origin, P is on the positive y axis and the sun is in the xy plane. From this transformation it can be observed that g_l only depends on the height $h(P)$ of this point and the angle $\theta_{sun}(P)$ between the light and the vertical (which becomes the y axis after said

transformation) at P. We will therefore write $g_l = g_l(h, \theta_{sun}, \lambda)$. In this case, $h = 0$ so g_l only depends on the angle θ_g between the light and the vertical on the ground.

The light reflected diffusely by the planet is computed by Lambertian shading (but any other shading model could be used) using the angle θ_g and the reflectivity of the object $r(P_g, \lambda)$:

$$
\begin{aligned}
I_{out}(P_g, \lambda) &= \cos(\theta_g) r(P_g, \lambda) I_{in}(P_g, \lambda) \\
&= \cos(\theta_g) r(P_g, \lambda) g_l(0, \theta_{sun}, \lambda) I_{sun}(\lambda).
\end{aligned}
$$

Finally, this light is again attenuated while travelling from P_g to the eye at E. We call this attenuation factor g_v. This gives us the final result for I_r:

$$
\begin{aligned}
I_r(P_g, E, \lambda) &= g_v(P_g E, \lambda) I_{out}(P_g, \lambda) \\
&= g_v(P_g E, \lambda) \cos(\theta_g) r(P_g, \lambda) g_l(0, \theta_g, \lambda) I_{sun}(\lambda). \quad (3)
\end{aligned}
$$

2.3 Scattering

Imagine a light beam travelling through the atmosphere. At any point of the beam, some of the light rays it consists of will be reflected on particles in the air. One simplifying assumption is made here: no beam of light is scattered more than once (no 'multiple scattering' occurs).[1] Because of this, we can assume that all light scattered into the view ray comes from a single direction only: the sun. This greatly simplifies computation, as we do not need to integrate over all possible directions. (For multiple light sources, each one can be treated separately.) Also, because the sun is very far away, incoming light rays are considered parallel.

How much of the light is scattered in a particular direction depends on the angle φ between the original and the scattered light ray:
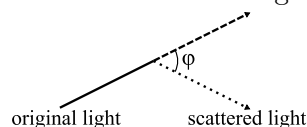
We call this dependency the 'angular scattering function'.

Define $I_{in}(P, \lambda)$ to be the intensity of the light coming into point P (called the *scattering centre*) for a given wavelength λ. Define $I_{out}(P, \varphi, \lambda)$ to be the intensity of the light scattered at an angle φ at P. Two types of scattering occur:

– *Rayleigh scattering* is scattering by air molecules. This shows a strong dependence on wavelength. The intensity $I_{out,R}(P, \varphi, \lambda)$ of the light scattered

---

[1] The effect of multiple scattering can be approximated [1] by introducing an appropriate ambient term.

in the direction φ is given by

$$I_{out,R}(P, \varphi, \lambda) = K_R(\lambda)\rho_R(h(P))F_R(\varphi)I_{in}(P, \lambda),$$

$$K_R(\lambda) = \frac{K}{\lambda^4},$$

$$F_R(\varphi) = \frac{3}{4}(1 + \cos^2 \varphi)$$

where K is a constant depending on certain properties of the atmosphere. F_R is called the *phase function* for Rayleigh scattering. The amount of Rayleigh scattering depends on the wavelength λ of the light. For example, blue light is scattered more than the other colours, which causes the sky to look blue. Moreover, the amount of scattering obviously depends on the density of particles in the atmosphere.

– *Mie scattering* is scattering due to larger particles in the atmosphere (such as dust and water droplets), so-called *aerosols*. The dependence on wavelength for this type of scattering can and will be neglected. The scattered light intensity $I_{out,M}$ is given by

$$I_{out,M}(P, \varphi, \lambda) = K_M \rho_M(h(P))F_M(\varphi)I_{in}(P, \lambda),$$

$$F_M(\varphi) = \frac{3(1 - g^2)(1 + \cos^2 \varphi)}{2(2 + g^2)(1 + g^2 - 2g \cos \varphi)^{\frac{3}{2}}}$$

where K_M is an atmosphere-dependent constant and g is given for Earth conditions by [4,3]:

$$g = \frac{5}{9}u - (\frac{4}{3} - \frac{25}{81}u^2)x^{-1/3} + x^{1/3},$$

$$x = \frac{5}{9}u + \frac{125}{729}u^3 + \left(\frac{64}{27} - \frac{325}{243}u^2 + \frac{1250}{2187}u^4\right)^{1/2}.$$

(The authors are glad that physicists had figured this out already.) Here, u is a constant dependent on atmospheric conditions and ranges between 0.7 and 0.85.

The total scattering of light in a particular direction is the sum of the Rayleigh and Mie scattering. We define R as the sum of the scattering factors, resulting in

$$\begin{aligned} I_{out}(P, \varphi, \lambda) &= I_{out,R}(P, \varphi, \lambda) + I_{out,M}(P, \varphi, \lambda) \\ &= R(h(P), \varphi, \lambda)I_{in}(P, \lambda), \\ R(h, \varphi, \lambda) &= K_R(\lambda)\rho_R(h)F_R(\varphi) + K_M \rho_M(h)F_M(\varphi) \end{aligned} \tag{4}$$

2.4 Scattered Light I_s

Besides light reaching the eye from the object, for every point on the view ray there is also a certain amount of light scattered from the light source towards the eye. Consider a point P along the view ray where a scattering event occurs:
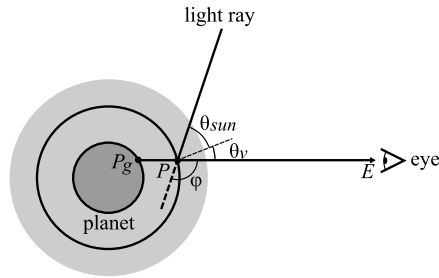
The intensity of the light reaching P is, as before in (2),

$$I_{in}(P, \lambda) = g_l(h(P), \theta_{sun}(P), \lambda)I_{sun}(\lambda).$$

Part of this light is scattered in the view direction as in (4):

$$I_{out}(P, \lambda) = R(h(P), \varphi, \lambda)g_l(h(P), \theta_{sun}(P), \lambda)I_{sun}(\lambda).$$

Finally, this scattered light is attenuated by g_v in the same way as (3):

$$\begin{aligned} I_s(P, \lambda) &= g_v(PE, \lambda)I_{out}(P, \lambda) \\ &= g_v(PE, \lambda)R(h(P), \varphi, \lambda)g_l(h(P), \theta_{sun}(P), \lambda)I_{sun}(\lambda). \end{aligned}$$

This gives us only the amount of light scattered into the view ray at a single point P. To find the total intensity of the scattered light for each point along the view ray, we need to integrate over the ray:

$$\begin{aligned} I_s(\lambda) &= \int_0^T I_s(P(t), \lambda)\mathrm{d}t \\ &= I_{sun}(\lambda) \int_0^T g_v(P(t)E, \lambda)R(h(P(t)), \varphi, \lambda)g_l(h(P(t)), \theta_{sun}(P(t)), \lambda)\mathrm{d}t. \end{aligned}$$

$$(5)$$

3   Optimization

One way of rendering a scene involving lighting based on the principles of atmospheric scattering is by using a ray tracing method. The results of ray tracing are very realistic. However, because of the lack of speed of ray tracing techniques, it is not suitable for real time purposes.

Dobashi et al. [1] have proposed an efficient algorithm that is able to make a good approximation of the ray traced model using the capabilities of modern graphics hardware. This algorithm applies a technique often used for rendering volumetrically lit scenes, e.g., beams of light in a foggy room, using multiple planes.

3.1   Sampling Spheres

The general idea of the algorithm is to speed things up by using a layered sampling approach. This approach uses n concentric sampling spheres numbered

$1\ldots n$, each representing a spherical 'shell' of air of a certain thickness. Sampling sphere k is at height h_k above the earth surface. Sphere 0 coincides with the planet surface and is not actually rendered.
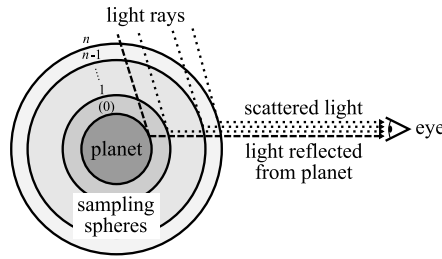
Using these sampling spheres we can split into segments the part of our view ray that passes through the atmosphere. Let us call the intersections with the sampling spheres, ordered from back to front, $P_0, \ldots, P_m$:
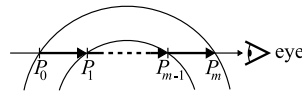
This subdivision can be used to compute the reflected and scattered light I_r and I_s.

3.2 Reflected Light I_r

Using the subdivision of the view ray, we can split up our attenuation function g_v into separate attenuations Δg_v for each segment:

$$g_v(P_g E, \lambda) = \prod_{i=0}^{m-1} \Delta g_v(P_i P_{i+1}, \lambda) \tag{6}$$

This neglects the attenuation on the first segment of the view ray (from E to P_m), but with a sufficient number of sampling spheres this part will be relatively short or, if E is outside the atmosphere, not contribute at all.

Like $g_l(h, \theta_{sun}, \lambda)$ before, $\Delta g_v(P_i P_{i+1}, \lambda)$ depends only on the height h_k of the corresponding sampling sphere and the angle θ_v between the view ray and the vertical, and can therefore be stored in one precomputed one-dimensional texture per sampling sphere.

Now (3) for the reflected light can be rewritten as:

$$I_r(P_g, E, \lambda) = \left(\prod_{i=0}^{m-1} \Delta g_v(P_i P_{i+1}, \lambda) \right) \cos(\theta_g) r(P_g, \lambda) g_l(0, \theta_g, \lambda) I_{sun}(\lambda). \tag{7}$$

3.3 Scattered Light I_s

In a similar fashion, we split the integral from (5) into a sum, again neglecting the first segment of the view ray:

$$I_s(\lambda) = I_{sun}(\lambda) \sum_{i=0}^{m-1}$$
$$\int_{P_i}^{P_{i+1}} g_v(P(t)E, \lambda) R(h(P(t)), \varphi, \lambda) g_l(h(P(t)), \theta_{sun}(P(t)), \lambda) \mathrm{d}t.$$

If we assume that g_l is nearly constant over a segment (which will be true when using sufficiently many layers) we can take it out of the integral:

$$I_s(\lambda) = I_{sun}(\lambda) \sum_{i=0}^{m-1} g_l(h(P_{i+1}), \theta_{sun}(P_{i+1}), \lambda)$$
$$\int_{P_i}^{P_{i+1}} g_v(P(t)E, \lambda) R(h(P(t)), \varphi, \lambda) \mathrm{d}t.$$

We can also take out a large part of the attenuation g_v:

$$I_s(\lambda) = I_{sun}(\lambda) \sum_{i=0}^{m-1} g_v(P_{i+1}E, \lambda) g_l(h(P_{i+1}), \theta_{sun}(P_{i+1}), \lambda)$$
$$\int_{P_i}^{P_{i+1}} g_v(P(t)P_{i+1}, \lambda) R(h(P(t)), \varphi, \lambda) \mathrm{d}t.$$

Now we can split up g_v like in (6):

$$I_s(\lambda) = I_{sun}(\lambda) \sum_{i=0}^{m-1} \left(\prod_{j=i+1}^{m-1} \Delta g_v(P_j P_{j+1}, \lambda) \right) g_l(h(P_{i+1}), \theta_{sun}(P_{i+1}), \lambda)$$
$$\int_{P_i}^{P_{i+1}} g_v(P(t)P_{i+1}, \lambda) R(h(P(t)), \varphi, \lambda) \mathrm{d}t.$$

Finally, the integral can be rewritten:

$$I_s(\lambda) = I_{sun}(\lambda) \sum_{i=0}^{m-1} \left(\prod_{j=i+1}^{m-1} \Delta g_v(P_j P_{j+1}, \lambda) \right) g_l(h(P_{i+1}), \theta_{sun}(P_{i+1}), \lambda) \tag{8}$$
$$(F_R(\varphi) \Delta I_R(h_k, \theta_v, \lambda) + F_M(\varphi) \Delta I_M(h_k, \theta_v, \lambda)).$$

Dobashi et al. [1] provide the details of this step. The essence is that ΔI_R and ΔI_M, like g_l and Δg_v before, only depend on two variables and can therefore be stored in lookup textures.

We now have equations for I_r and I_s that no longer use integrals, and only require additions, multiplications, trigonometry functions and lookups in two-dimensional tables.

4 Implementation

4.1 Precalculation

In order to achieve interactive frame rates we have to precalculate as much data as possible. This data will be stored in look-up tables which will, during the initialization of the graphics hardware, be uploaded to a number of textures. We will calculate each function three times using appropriate λ values for red, green and blue wavelengths. These three resulting tables can be fit into the individual color components of the texture. Using the fixed values for λ, we can generate look-up tables for $g_l(h, \theta_{sun}, \lambda)$, $\Delta I_r(h, \theta_v, \lambda)$, $\Delta I_m(h, \theta_v, \lambda)$ and $\Delta g_v(h, \theta_v, \lambda)$ that only cover up to two dimensions.

4.2 Rendering the Frames

The actual rendering of the individual frames is done in two passes. The first step is to render the scene geometry as usual. This geometry includes the earth, sun, stars and other heavenly bodies. No attenuation of any objects (including the earth) need to be taken into account here: this will be resolved in the next pass. Concretely, we ignore the product term in (7).

The second pass renders the atmosphere. This is done with depth testing enabled, to avoid drawing atmosphere behind objects, but with depth writing disabled to prevent depth buffer fighting between atmospheric layers.

Each layer is rendered twice. First we apply multiplicative blending to attenuate the light from the objects and layers behind it. Second we apply additive blending to add the scattered light to the light intensity that is already there. Because this scattered light will be attenuated again by drawing later layers in front of it, we ignore the product term in (8).

As attenuation is done by drawing additional layers, we need to make sure that the layers are rendered in back-to-front order. For this the spherical layers are split into two caps in a viewpoint-dependent way. The rear caps are rendered in descending order, followed by the front caps in ascending order. The rendering order for 3 layers and a viewpoint outside the atmosphere would look like this:
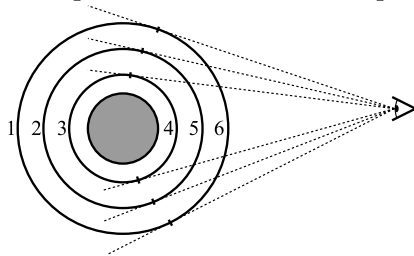
5 Results

When implemented properly the proposed algorithm can produce a realistically looking rendition of a planetary atmosphere while still archieving interactive

frame rates. Our proof of concept application produced a framerate that was always in the 20-30 frames per second area. These rates were archieved on an NVIDIA GeForce 7800 GS graphics card using settings matching the conditions of the earth. The atmosphere was rendered in 32 layers. Some screen captures of our implementation are supplied below.
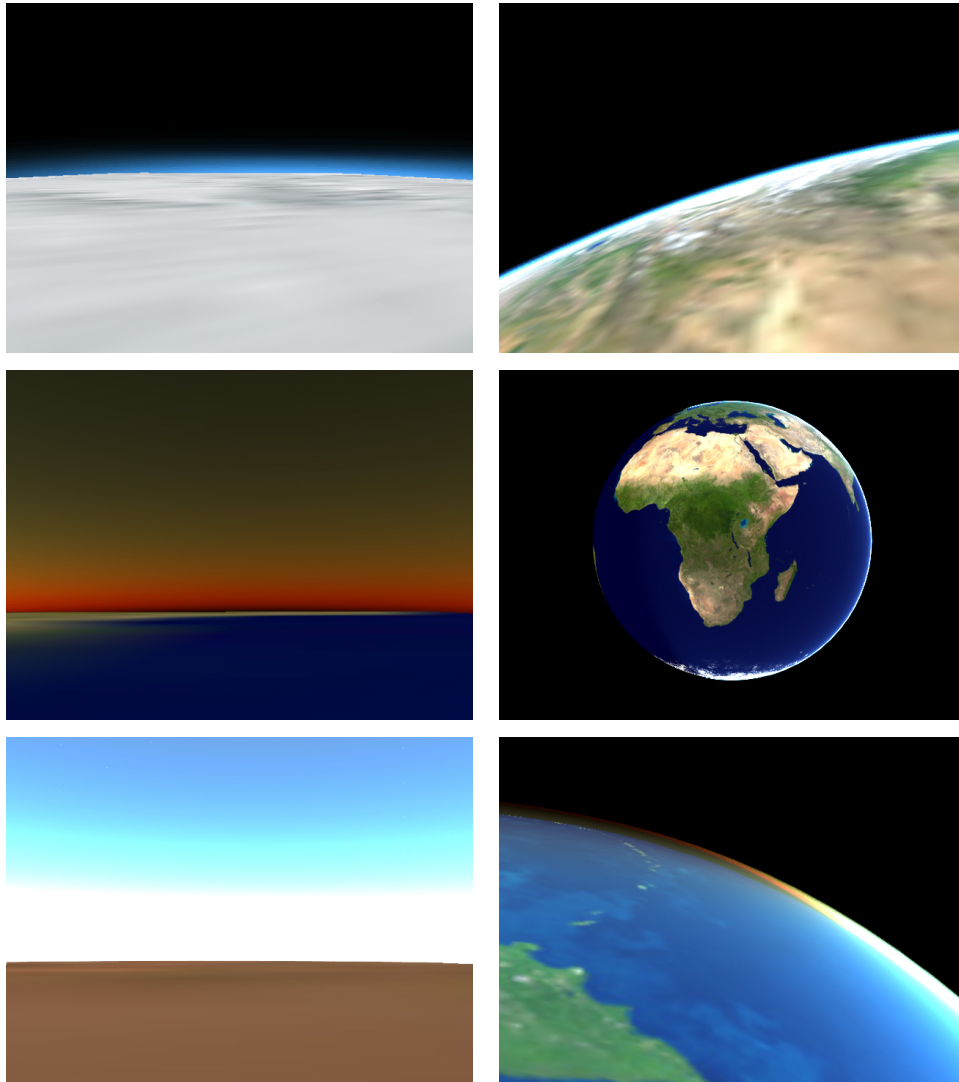
6 Conclusion and Future Work

We have explained the different formulae behind the atmospheric scattering concept. Using these mathematical constructions we have examined the characteristics of the scattering algorithm proposed by Dobashi et al. This algorithm has

proven to be suitable for the real time rendering of atmospheric environments on current graphics hardware.

In the future, taking into account the continuous improvements in graphics hardware architectures, this algorithm can become increasingly useful for larger simulations, such as entire solar systems. Because the algorithm is sufficiently fast for interactive frame rates it can be used for demanding purposes such as astronaut training simulators.

Future work on this subject could include implementing multiple scattering. The computing capabilities and programmability of graphics hardware is continuously improving, meaning that development of an efficient algorithm for multiple scattering might become more feasible. The addition of high dynamic range imaging technologies to recent graphics cards also presents a very promising possibility for adapting this algorithm to create even more realistic images.

References

1. Dobashi, Y., Yamamoto, T., Nishita, T.: Interactive rendering of atmospheric scattering effects using graphics hardware. In: HWWS '02: Proceedings of the ACM SIGGRAPH/EUROGRAPHICS conference on graphics hardware, Aire-la-Ville, Switzerland, Eurographics Association (2002) 99–107
2. Klassen, R.V.: Modeling the effect of the atmosphere on light. ACM Transactions on Graphics **6**(3) (1987) 215–237
3. Nishita, T., Sirai, T., Tadamura, K., Nakamae, E.: Display of the earth taking into account atmospheric scattering. In: SIGGRAPH '93: Proceedings of the 20th annual conference on computer graphics and interactive techniques, New York, NY, USA, ACM Press (1993) 175–182
4. Cornette, W., Shanks, J.: Physical reasonable analytic expression for the single-scattering phase function. Applied optics **31**(16) (1992) 3152–

The Continuation of Visualization

Writser Cleveringa, s1346032

Liewe Kwakman, s1351214

Abstract.

Visualization is the field in computer science dealing with the problem of representing and exploring masses of data effectively. The visualization discipline is currently reaching its maturity. Common interest in the research of visualization is decreasing, mainly because for most general problems standard commercial solutions are available.

Some feel that the field of visualization has drifted too far away from its practical uses. Visualization is, after all, a tool to help other people with analyzing data. These scientists feel that they should collaborate with medical specialists and businessmen to design software that is more practical to end-users. Others feel that they should work together with the field of Computer Vision, trying to automatically detect features in datasets.

There is a lot of discussion about what research should be focused on now. In this article we talk about the future of visualization. We based our research upon three different papers. First we tell something about the history of visualization. How and why the discipline did come to existence. We consider a few viewpoints on how the discipline should continue in the near future. What should happen to renew interest for the discipline? How can visualization be brought to the next level?

Background

We could say 1987 is the year in which visualization was born. Before, the computer graphics field already existed, but mainly focused on the entertainment industry. Scientific and biomedical communities needed ways to visualize the huge data masses that they were generating and felt that appropriate research was lacking. The visualization field was established to meet the requirements of these communities.

Visualization was born to satisfy a need to explore masses of scientific data. Acquisition devices like MRI-scanners, large-scale simulations on supercomputers, but also stock exchanges produce very large amounts of data. Visualization methods allow researchers, analysts, engineers and the general public to obtain insight in data in an efficient and effective way, using the human visual system. This system has unique capabilities to detect interesting features and patterns effectively.

Problem statement

Nowadays a lot of methods for visualization have been developed and some of them are routinely used in practice. However, most methods developed recently are considered incremental and conferences are rarely visited by customers and users anymore.

There is a consensus that the visualization field needs to be reconsidered. In the way the visualization community is now, there is little interest from potential customers. Most things customers are interested in have become common knowledge or are at least commercially available.

Another reason for reconsideration is the fact that the field is getting more specialized and critical. In the beginning there were a lot of things that had to be found out, nowadays it is harder to find relevant subjects and most work is incremental. The big problems are mostly solved already. What can we do about this?

In this document we review three articles. First we give a brief overview of the background of visualization. After that we discuss how the discipline should go on. We do this by looking at the viewpoints of each article and point out how they complement each other.

The future of visualization

Now that the field has reached its maturity one could ask: how should we go on? Lorensen

1) states that if nothing changes, the visualization community is going to pass away. According to him, the community has drifted to far away from its customers. In his paper he gives some solutions for this problem. Suggestions made in the other two articles could also help increasing the chance of continuation of the community. Van Wijk 2) tried to develop a method to asses the value of visualization methods. This way we could understand what is going wrong nowadays and what we can improve in the near future. Duke, Brodlie and Duce 3) suggest that the field of scientific visualization should be formalized to improve communication and standardization.

Embrace the customer

In his paper "On the death of visualization" Lorensen 1) states that Scientific Visualization has lost its customers. At one side of the spectrum customers have disappeared because they can buy solid applications that do the tasks they are expected to do.

Not much spectacular breakthroughs have been achieved in the last few years and customers are satisfied with the software they can currently buy on the market.

More important is that at the other side of the spectrum customers have disappeared because researchers are not responsive to their wishes. This is only the surface of the problem though. The main issue is that scientists do not 'understand' the problems of their customers and vice versa. The scientist thinks about efficient volume rendering using ray casting, good transfer function design and hardware acceleration. Granted, these might be very important points while implementing an algorithm, but for the customer this is not important at all. He or she is not interested in the visualization itself.

For example, a surgeon might want to scan for tumors in a medical image. He does not care about the way the visualization is created, he only wants to spot tumors as fast and easy as possible.

A meteorologist working at the NASA is not interested in tumors, but in displaying cloud movements. He is not looking for details, but wants to see the global movements of clouds.

The surgeon and the weather forecaster use scientific visualization for completely different goals. But they both see visualization as a tool, not as a goal in itself. We cannot expect them to be both happy with the same program, since they want to use it for very different purposes.

Researchers in the field of Scientific Visualization should realize this. Instead of going their own way they should actively collaborate with the customers that use (or could use) their technology. Find out what problems customers are facing, instead of searching for problems yourself. Ask customers where they see innovation playing a role. Organize congresses where both customers and researchers meet and can discuss problems and possible solutions. The author of the article suggests that this way Scientific Visualization will become more useful again.

We have some doubts about this though. Of course communication with the customer is very important. But should we really do everything the customer wants? What about fundamental research? For our customers money usually is the driving factor. We believe that should not be the case for scientists.

Form alliances with other fields

Lorensen also suggests that researchers in the field of Scientific Visualization should form alliances with researchers in other fields.

Let's go back to the example of the surgeon again. Wouldn't it be great if the visualization algorithm can detect and show possible tumors itself? This would spare a lot of time and could prevent human errors (like a tired surgeon overlooking a small tumor). A researcher in the field of Scientific Visualization has usually not enough knowledge about the human body to do this. On the other side, a medical image analyst could possibly help with detecting a tumor but cannot present a compelling visualization to the surgeon. That is why the two of them should work together. Again, Scientific Visualization is a tool, not a goal. Working together with researchers in other fields can increase the value of this tool.

The example with the surgeon is just that, an example. But there are countless other

possibilities. Think about a meteorologist that enhances a hardware-based visualization algorithm to also do weather forecasting. The possibilities are endless.

On the other hand, we think that methods like volume rendering are fundamental and require research of their own. To us, working together with scientists from other fields sounds like a good idea. But we will have to watch out not to become a lapdog of other researchers. Our field is interesting and important enough to warrant research itself.

Grand Challenges

One other way to revitalize the field of Scientific Visualization would be to define one or more *Grand Challenges*. These are long-term projects for researchers across multiple fields. A classical example of a grand challenge might be the race to the moon. Physicians, programmers, researchers and other specialists combined all their forces to tackle a huge problem. A similar approach could be used to produce interesting results in the field of Scientific Visualization, though probably on a smaller scale. In his paper Lorensen poses several ideas:

- The Digital Human. Produce a complete working, viewable model of the human body with working simulations of all major systems within. Include systems on all scales, from organ level to cell level to molecular level.
- The Digital Medical Illustrator. Produce patient-specific illustrations of a (partial) body that match images drawn by an expert medical illustrator (see Figure 1).
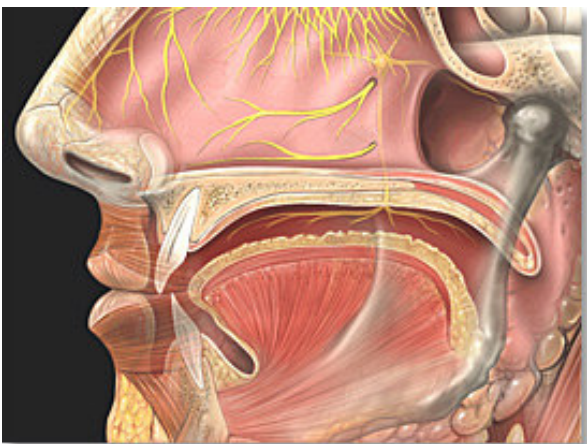
Fig. 1. Will computers ever be able to produce such drawings?

The value of visualization

In his paper van Wijk 2) promotes good assessment of visualization for an application beforehand. A model is presented to contribute to this. In short it comes down to this; the full model is presented in 2).

To assess the value of visualization, one could look at the knowledge gained by using visualization. For this you can visualize three entities:

- The data to assess.
- The specification on how to visualize.
- The knowledge of the user.

The data is transformed recording to the specifications into an image (or other form of visualization); this image is perceived by a user and increases the user's knowledge. The user could change the specification in order to explore the data further. How knowledge is gained depends on the image, the knowledge of the user, the perception and cognition of the user.

When one wants to know if a visualization method is worthwhile, you can look at the initial development costs of the visualization, the initial costs per user, the initial costs per session and the perception and exploration costs. If the total costs are lower than the value of the gained knowledge, the visualization method is interesting.

However, it is not easy to assess the value of the gained knowledge. The traditional aim of visualization is insight. With insight users are enabled to see things they were not aware of and it helps them to define new questions, hypotheses and models of their data. The problem with assessing the value of gained insight is that it is unknown how much insight there is to be gained from a certain amount of data, so it is unknown how successful it is.

Another way to approach the value of visualization is to measure whether it influences decision-making. For example: does a visualization method sometimes help a surgeon

to decide his course of action? If it never does, there is a good possibility that the visualization in question is not very valuable. If the decisions to be taken are known, one can assess the value of the method by measuring whether it influences the decision-making process. Of course the importance of this influence depends on the importance of the decision itself.

A visualization method could be seen as having a high value, but that does not directly guarantee that it will be used in practice. If there are other visualization methods with a higher value, or there are already existing methods, it might be more profitable to use these. It is also possible that there is a non-visual method that can extract the relevant information automated.

New visualization techniques are often not used in practice. First of all it is not always obvious to realize visualization could help one to understand his data. When the realization is made, the choice is often to use inferior visualization features of commercial simulation tools. Of course when these are sufficient these are highly cost-effective. When not available or not sufficient enough, one has to study research papers or get in contact with an expert, because, although there books with basic techniques, there are no books that present and compare the latest novel visualization techniques. After that the development of the visualization is often very costly and takes lots of time and effort while it is often unclear if it will solve the problem.

Visualization is not very objective. The increase in knowledge is very dependant on the skill and current knowledge of the user and the specifications used. The results will differ very much between different users. Visualization should not be used to verify the final truth but rather to inspire to new hypothesis.

When developing a visualization tool one has to realize more interaction is not always better. It can lead to lots of time spend on resetting parameters and re-rendering. Also it is possible to tune specifications to get a desired outcome thus compromising subjectivity.

Visualization can be used for presentation or exploration. Most researchers see exploration as the main reason to use visualization and presentation as something extra. Presentation however is just as important. It is very useful to use to show to other people how and why

something is scientifically sound. Lots of people can gain knowledge with it, so the value of visualization used this way could be high.

Formalization

Duke, Brodlie and Duce 3) believe that the time has come to develop a rigorous foundation for the field of Scientific Visualization. Their suggestion is to introduce an ontology to formalize aspects of the visualization process. Ontology means, literally, the study of existence. The authors want to create formal definitions of data sets, interfaces and theories. To clarify: they suggest developing a standard to define Scientific Visualization concepts, based on XML. This could help in sharing scientific problems and solutions. Also, the standard could be used to create Visualization web services and would make it much easier to create, for example, distributed visualization grids. Collaboration between researchers themselves and between researchers and the public will receive a huge boost.

Discussion

The authors of this article feel the three articles are a complement of each other. The main issue is how to go on in the future.

Although 2) and 3) aren't claiming their suggestions are contributing to the continuation of the community, they do help to achieve the goals mentioned in 1). In 1) is stated the community can't go on like it does, it needs to embrace customers, communicate with them. In 2) is suggested before beginning with visualization an assessment of the value of visualization in a particular application is useful. Embracing of the customer and assessing the value of visualization will go better with improved communication. According to 3), to improve communication, formalization is necessary. An ontology should be developed which is generally used so everybody talks about the same thing when using a term.

Conclusion

In the field of Scientific Visualization a critical point has been reached. The field is

getting mature, and we have to determine what to do in the future. The consensus is that Scientific Visualization is part of a larger process. Visualization in itself is not a goal but a tool to reach a bigger goal. That is why researchers must works together with other researchers, experts from other fields and customers. Collaboration needs improvement and in this article we have given several possible ways to achieve this.

- Work more closely with customers and experts from other fields to make our research more useful.
- Define grand challenges to stimulate collaboration between researchers.
- Assess the value of using a visualization method for a given application more elaborate.
- Formalize Visualization concepts in an ontology to create a standard way in which people and machines can communicate about aspects of visualization.

For every method we have given advantages and disadvantages. But the question still remains: what to do in the future? To be honest, we do not know. In this paper we treated several possible solutions. It is now up to the field to determine what to do. The field is too interesting to let it pass away

Bibliography

1) B. Lorensen: "On the Death of Visualization. Can It Survive Without Customers?" Position Papers NIH/NSF Proc. Fall 2004 Workshop Visualization Research Challenges, 2004. http://visual.nlm.nih.gov/evc/meetings/vrc2004/position_papers/lorensen.pdf

2) J.J. van Wijk: "The Value of Visualization". In: C. Silva, E. Groeller, H. Rushmeier (eds.), Proc. IEEE Visualization 2005, p. 79-86, 2005. Best paper award. http://www.win.tue.nl/~vanwijk/vov.pdf

3) D.J. Duke, K.W. Brodlie, D.A. Duce, I. Herman. "Do You See What I Mean?". IEEE Computer Graphics & Application, 25(3), May/June, pp. 6-9, 2005. http://www.cwi.nl/~ivan/AboutMe/Publications/IEEEOntology.pdf

Contour detection by suppression of texture edges

Pjotr Svetachov and Arjan Somers

Rijksuniversiteit Groningen

Abstract. Classical contour detection algorithms, such as the Canny edge detector, are receptive to edges in the texture of the objects in an image, thus giving poor result when trying to process natural or noisy images, such as a photograph of a cluttered scene. We show a contour detection algorithm that was proposed by N.Petkov and M.A.Westenberg which uses a biologically motivated (by mimicing the human visual system) approach to suppress edges that are part of a texture. This method is called nonclassical receptive field (non-CRF) inhibition, or surround suppression. The method is based on mathematical models of the workings of the visual cortex. The algorithm is actually an extra computational step, which can be used to extend traditional edge detection algorithms, such as the Canny operator. We also show a biologically motivated operator, called the Gabor energy operator. This results in an algorithm that extracts isolated edges but doesn't extract edges that are part of a texture. To give an understanding how the method performs we will use natural images with associated desired output contour images. When using surround inhibition better approximation of the desired contour images are generated compared to algorithms without surround inhibition. So the proposed method can extract contours better than traditional methods making it very usable in practice.

1 Introduction

Edge detection is a fundamental operation in image processing and computer vision, in which a lot research has been done. While a large number of edge detection algorithms are proposed, there is still much progress made today. Popular current edge detection algorithms, such as the Canny edge detector [1], define edges as a local change in luminance of a certain strength, for which a gradient can be defined, without looking at it's context. This means there is no distinction is made between object contour edges, and texture edges. For most computer vision applications only contour edges are needed, and texture edges can often be considered noise in these applications. These edge detectors are known as non-contextual edge detectors [2]. There are also contextual edge detection algorithms, that take in account additional information such as local image statistics, image topology, perceptual differences in local cues, edge continuity and density. These contextual edge detectors use this extra information to make a distinction between object contour edges, and texture edges. There is
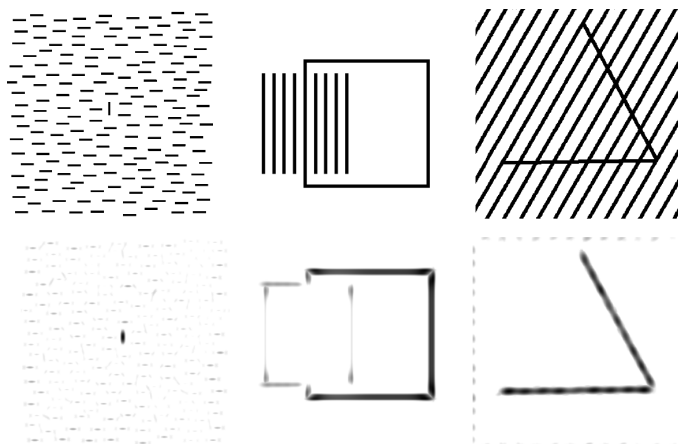
Fig. 1. (Upper-row) Stimuli. (Lower-row) Surround inhibited responses

evidence that the human visual system also makes this distinction in it's early stages of visual information processing.

The human visual system has orientation selective neurons which respond to lines or edges at a certain position. Initially two types of orientation selective neurons where found. One type of neurons which was sensitive to the contrast polarity of edges and lines, called simple cells, and a type which was not, called complex cells [3], [4]. These cells only respond to edges and lines in their receptive areas, also called the classical receptive field (CRF). As opposed to non-CRF which would include the surrounding area.

As said before, the human visual system makes an early distinction between isolated edges, such as object contours and region boundaries, and edges in a group, such as edges in a texture. Psychophysical studies show that the perception of stimuli, such as line segments can be influenced by the presence of other stimuli in its neighborhood. There is a reduced response of orientation selective neurons to line segments in their receptive fields when there are other line segments with the same orientation in it's neighborhood. The response depends on the contrast in orientation of the different line segments. This is called the pop-out effect [5]. These effects are shown in figure 1. Here different images are shown with their surround inhibited responses. These responses are similar to the response the human visual system has to these images. In the first image the line with the different orientation pops out. In the second image we humans see two squares just like the surround imhibited response. And there is a similar effect int the thirdh image set. So all the surround inhibited responses are like the mental image humans get when looking at an image, where certain features get a high response, while other parts are supressed. This reduction in response is referred to as non classical receptive field (non-CRF) inhibition, or surround suppression. Using surround suppression there is large response to edges which are part of a contour, while edges that are part of the objects texture are suppressed.

The reviewed papers [6], [7], and [8] all use non-CRF inhibition. In [6] the perception of lines with varying stroke widths and sizes with band spectrum noise using non-CRF is discussed. [7] continues with a more general discussion of non-CRF inhibition. It proposes a biologically motivated combination of a Gabor energy edge detector and non-CRF inhibition is proposed, creating a computational model of the human visual system. In this paper a performance measure is defined, and performance is discussed. [8] continues the discussion of [7], using Canny instead of Gabor, and proposes a better binary map computation method. We discuss the methods and results of these papers. Section 2.1 discusses how non-CRF can be combined with the Canny edge detector and the biologically motivated Gabor energy edge detector which models the simple and complex cells. In section 2.2 non-CRF inhibition is discussed. Two types of inhibition are discussed, anisotropic inhibition, where only responses to lines and edges in the with the same orientation as the CRF contribute to the suppressions and secondly anisotropic inhibition, where all responses outside the CRF contribute to the suppression. In section 4 the performance is discussed. To do so a performance measure is defined and experimental results are compared to the results of other edge detection algorithms.

2 Methods

In this section non-CRF inhibition will be presented. Non-CRF inhibition requires a edge magnitude and orientation map of an image. Therefore in section 2.1 two methods which calculate those maps will be described. In section 2.2 those maps will be used with non-CRF inhibition.

2.1 Contour detection operators

This section presents two edge detection algorithms that can produce edge magnitude and orientation maps. First we will discuss the well known Canny edge detection operator. Then we will discuss the biologically motivated Gabor energy operator.

Canny The Canny edge detector works by computing the gradient of an image. Areas with high gradient usually contain an edge. This approach doesn't give good results when there is noise in the image so the image is first smoothed by a convolution with the following Gaussian function:

$$g_\sigma(x,y) = \frac{1}{2\pi(\sigma)^2} \exp\left(-\frac{x^2+y^2}{2(\sigma)^2}\right) \tag{1}$$

After the image is smoothed the gradient is calculated. This can be done by calculating the gradient of $g_\sigma(x,y)$ first and then convolving the image by the result.

$$\nabla f_\sigma(x,y) = (f * \nabla g_\sigma(x,y)) \tag{2}$$

Now let $\nabla_x f_\sigma(x,y)$ and $\nabla_y f_\sigma(x,y)$ be the x and y components of Eq. (2)

$$\nabla_x f_\sigma(x,y) = (f * \frac{\partial g_\sigma}{\partial x})(x,y) \tag{3}$$

$$\nabla_y f_\sigma(x,y) = (f * \frac{\partial g_\sigma}{\partial y})(x,y) \tag{4}$$

Now we can define the gradient magnitude $M_\sigma(x,y)$ and orientation of the gradient $\Theta_\sigma(x,y)$ as

$$M_\sigma(x,y) = \sqrt{(\nabla_x f_\sigma(x,y))^2 + (\nabla_y f_\sigma(x,y))^2} \tag{5}$$

$$\Theta_\sigma(x,y) = \qquad atan\left(\frac{\nabla_x f_\sigma(x,y)}{\nabla_y f_\sigma(x,y)}\right) \tag{6}$$

The higher the magnitude the more likely that an edge is present. We will use the orientation later for anisotropic non-CRF inhibition.

Gabor The Gabor operator is a biologically motivated edge detection operator. This operator was used both in [6] and [7]. In [8] this operator was dropped and it was shown that non-CRF inhibition can also yield good results with standard gradient-based edge detectors like the above mentioned Canny operator.

The Gabor operator uses a modified Gabor function to take into account restrictions found in experimental data. Two types of cells are modeled using the modified Gabor functions, simple cells and complex cells.

Simple cells The impulse response of a simple cell can be modeled by the following function:

$$g_{\lambda,\sigma,\theta,\varphi}(x,y) = e^{-\frac{X^2 + \gamma^2 Y^2}{2\sigma^2}} \cos\left(2\pi \frac{X}{\lambda} + \varphi\right)$$
$$X = x\cos\theta + y\sin\theta$$
$$Y = -x\sin\theta + y\cos\theta$$

Here $\gamma = 0.5$ is a constant that controls the ellipticity of the input response. σ determines the size. θ determines the orientation and φ the phase offset. λ is here the wavelength and the ratio σ/λ determines the how many striped zones the input response has, see figure 2 for an example. In [8] the ratio σ/λ was set to $\sigma/\lambda = 0.56$. And like with Canny we just convolve the image with our operator so our response $r_{\lambda,\sigma,\theta,\varphi}(x,y)$ is

$$r_{\lambda,\sigma,\theta,\varphi}(x,y) = (f * g_{\lambda,\sigma,\theta,\varphi})(x,y)$$

Complex cells Complex cells are modelled using two simple cells with a phase difference of $(\pi/2)$. This results in the operator $E_{\lambda,\sigma,\theta}(x,y)$

$$E_{\lambda,\sigma,\theta}(x,y) = \sqrt{r^2_{\lambda,\sigma,\theta,0}(x,y) + r^2_{\lambda,\sigma,\theta,-\frac{\pi}{2}}(x,y)} \tag{7}$$

this operator is called the Gabor energy operator. No orientation function is presented because the operator can be evaluated for different orientations.
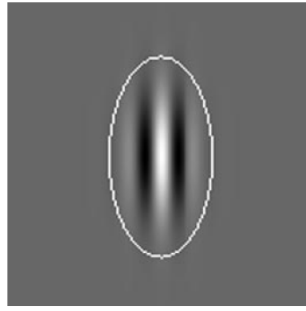
Fig. 2. A Gabor filter. The ellipse specifies the boundary of the receptive field outside the boundary the function takes negligibly small values.

2.2 Non-CRF inhibition

One of the above filters can now be extended by adding a term that takes the surroundings of a given point into account. This term is a weighting function

$$w_\sigma(x,y) = \frac{H(DoG_\sigma(x,y))}{||H(DoG_\sigma)||_1}$$

where

$$H(z) = \begin{cases} 0 & z < 0 \\ z & z \geq 0, \end{cases} \tag{8}$$

and $||.||_1$ is the L_1 norm.

The function $DoG_\sigma(x,y)$ is the difference of two Gaussian functions:

$$DoG_\sigma(x,y) = \frac{1}{2\pi(4\sigma)^2} \exp\left(-\frac{x^2+y^2}{2(4\sigma)^2}\right)$$
$$-\frac{1}{2\pi(\sigma)^2} \exp\left(-\frac{x^2+y^2}{2(\sigma)^2}\right)$$

To summarize $w_\sigma(x,y)$ is just the difference of two Gaussian functions, capped at the bottom so $z > 0$ and normalized so the integral of $w_\sigma(x,y)$ is 1. See figure 3 for the plot.

Anisotropic non-CRF inhibition Anisotropic inhibition takes into account the difference in the directions of the gradient in the central point and the surrounding points. This is done using the function

$$\Delta_{\Theta,\sigma}(x,y,u,v) = |\cos(\Theta_\sigma(x,y) - \Theta_\sigma(u,v))|$$
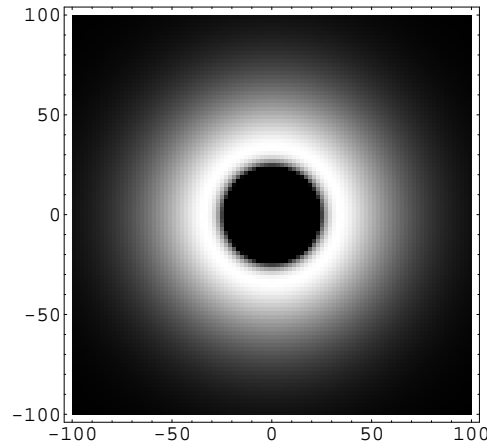
Fig. 3. Impulse response of $w_\sigma(x, y)$. The brighter the point the more we consider that point.

Here $\Theta_\sigma(x, y)$ is the gradient orientation in point (x, y). So if the gradient orientations are identical $\Delta_{\Theta,\sigma}$ will be (at it's maximum of) 1 and this function decreases to 0 when the gradient orientations of point (x, y) and (u, v) are orthogonal. Lets take one of the filters described in section 2.1 and it's outputted gradient magnitude $M_\sigma(x, y)$ (equation (5) and (7)). Now for every point (x, y) the anisotropic term $t_\sigma^A(x, y)$ can be defined by

$$
\begin{aligned}
t_\sigma^A(x, y) = \int\int & M_\sigma(x - u, y - v) w_\sigma(u, v) \\
& \times |\cos(\Theta_\sigma(x, y) - \Theta_\sigma(x - u, y - v))| \\
& du dv
\end{aligned}
$$

where the integral is over the domain of the picture. Now we can introduce the operator $C_\sigma^A(x, y)$ as

$$
C_\sigma^A(x, y) = H(M_\sigma(x, y) - \alpha t_\sigma^A(x, y))
$$

with $H(z)$ defined as in (8) and where α controls how much effect the suppression term has. So when there are a lot of nearby edges that have the same orientation as the point (x, y) then the anisotropic suppression term $t_\sigma^A(x, y)$ will be strong and this may cancel out the gradient magnitude $M_\sigma(x, y)$ of the point (x, y). And if there are less nearby edges that have the same orientation as the point (x, y), the anisotropic suppression term will be low and we will find an edge at that point. So the operator will still respond strong to isolated edges and lines but the operator will not respond to edges that have nearby edges with the same orientation.

Isotropic Non-CRF Inhibition Isotropic inhibition works almost the same as anisotropic inhibition except that isotropic inhibition does not take into account the orientation of nearby edges when calculating the suppression for a point (x, y). So now the term $t_\sigma^I(x, y)$ is defined as

$$t_\sigma^I(x, y) = \int \int M_\sigma(x - u, y - v) w_\sigma(u, v) du dv$$

And the operator $C_\sigma^I(x, y)$ is defined as

$$C_\sigma^I(x, y) = H(M_\sigma(x, y) - \alpha t_\sigma^I(x, y))$$

So again α is used to control the strength of the suppression. So the operator will still respond strong to isolated edges and lines but the operator will not respond to edges that have nearby edges invariant of the orientation.

3   Binary map computation

The described algorithm outputs grey scale images. To create images like the canny algorithm, where only lines are outputted, using a binary image, the images have to be converted. During this step, the points on the outputted grey scale images which are likely to belong to an edge have to be connected to neighboring points, in order to create lines which represent the edges. This is done using a algorithm described in [8], which connects points depending on the orientation of edges. This step is needed to be able to compare the algorithm to the Canny algorithm.

4   Results

4.1   Performance measure

To measure the performance of this algorithm natural images (e.g photographs) are used with predefined desired operator outputs. The desired operator output is manually created and therefore subjective. The desired output in this case are the contours of the input image. A pixel belongs to a contour in the desired output if it's part of an occluding contour of an object or it belongs to a contour in the interior of an object or if it makes part of a boundary between two (textured) regions (e.g. sky and grass or water and sky).

The following performance measure function is used:

$$P = \frac{N}{N + FP + FN}$$

Here N is the number of correcly detected pixels. FP is the number of false positives, that is the number of pixels the edge detectors detects as an edge while they belong to the background in the desired output. FN is the false negatives, that is the number of pixels the edge detector missed and thus didn't detect as the an edge.
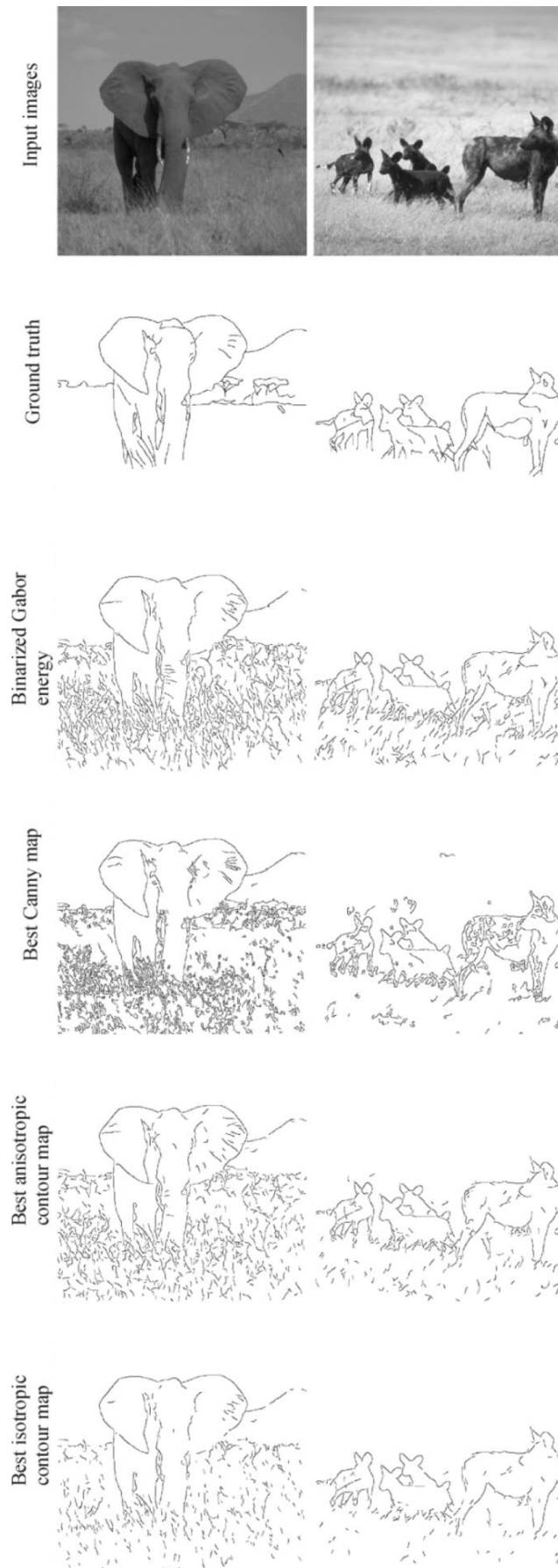
Fig. 4. Input image, desired output, and the best output of the Canny algorithm and the anisotropic and isotropic non-CRF inhibited Gabor results

4.2 Performance

The before mentioned performance measure is used to compare the results of the outputs of the different algorithms. Two test images, the desired outputs of those images, and the best outputs from different algorithms are shown in figure 4. Fig. 5 shows comparative statistical box-and-whisker plots for five test images. The plots reveal a consistent better performance of the contour operators using surround inhibition. In all cases, the best performance (the top end of a whisker) is higher in comparison to the best performance of the Canny edge detector and the Gabor energy operator without surround inhibition, and the plots show that in most of the cases, the isotropic inhibition gives the most effective results. In the first boxplot this is not the case, here anisotropic inhibition scores better.
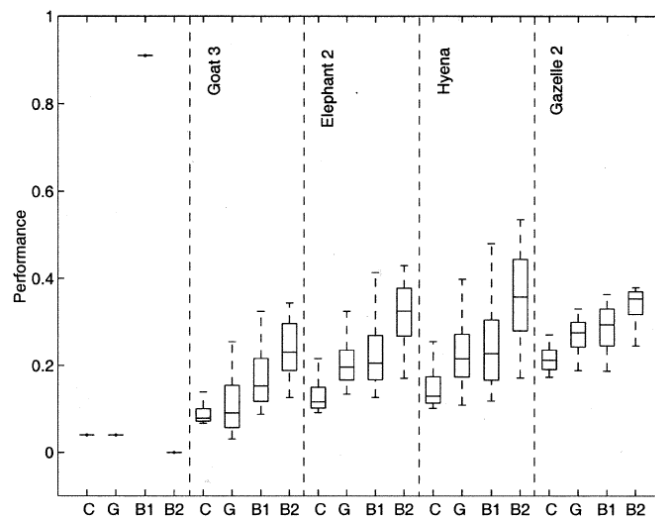
Fig. 5. Box-and-whisker plots of the performance of the Canny edge detector (denoted by C), the Gabor energy operator (denoted by G), the contour operator with anisotropic inhibition (denoted by B1) and isotropic inhibition (denoted by B2) for five of the test images. The first boxplots are from the triangle seen in figure 1, the thirdh and fourth boxplots are the results for the images shown in 4 and the second and fifth boxplots belong to test images not shown in this paper.

5 Conclusion

It is shown that the biologically motivated step of surround inhibition can be used in conjunction with gradient based edge detection algorithms, and a better performance is achieved using this extra step on all the images of the test set. In [6] and [7] the Gabor operator was used. But later in [8] the Canny operator was used because it required less computation (you don't need to evaluate the operator for more orientations) while giving pretty good results. Also a better

binary map computation model was given in [8]. So while it's less biologically motivated it is more practical to use. And probably in the future this method will be extended to for special cases of edge detection becoming less biologically motivated but more practical to use.

References

1. Jan-Mark Geusebroek, Arnold W. M. Smeulders, and J. van de Weijer. Fast anisotropic gauss filtering. In *ECCV (1)*, pages 99–112, 2002.
2. D. Ziou and S. Tabbone. Edge detection techniques - an overview. *International Journal of Pattern Recognition and Image Analysis*, 8:537–559, 1998.
3. D. H. Hubel and T. N. Wiesel. Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *Journal of Physiology-London*, 160(1):106, 1962.
4. David H. Hubel and Torsten N. Wiesel. Sequence regularity and geometry of orientation columns in the monkey striate cortex. *The Journal of Comparative Neurology*, 158:267–294, 1974.
5. H.C. Nothdurft. Texture segmentation and pop-out from orientation contrast. *Vision Research*, 31(6):1073–1078, 1991.
6. N. Petkov and M. A. Westenberg. Suppression of contour perception by band-limited noise and its relation to non-classical receptive field inhibition. *Biological Cybernetics*, 88(10):236–246, 2003.
7. C. Grigorescu, N. Petkov, and M. A. Westenberg. Contour detection based on nonclassical receptive field inhibition. *IEEE Transactions on Image Processing*, 12(7):729–739, 2003.
8. C. Grigorescu, N. Petkov, and M. A. Westenberg. Contour and boundary detection improved by surround suppression of texture edges. *Image and Vision Computing*, 22(8):609–622, 2004.

Contour detection improved by surround suppression

Piet den Dulk (1470639), Roelof Anne Schoenmaker (1603078)

Computer Science Department, University of Groningen

Abstract. Our review paper discusses the part of the visual cortex (part of brain) that only has to do with detection of contours. This part does some low-level processing which can be expressed in terms of a system. The paper attributes also to NON-Classical Receptive Field (NON-CRF) suppression in the visual area called V1. This visual area V1 is suggested as the possible origin of various perceptual effects.

Research has recently been done with humans and macaque monkey's. The results have delivered points of interest for today's contour detection algorithms. We will review three papers and construct from the Gabor Energy filter two new operators, one with an Anisotropic suppression term and one with an Isotropic suppression term as proposed in these papers. Also we will review results from the performance of these two new operators and compare them with performance results of the SUSAN and Canny operators.

1 Introduction

The human brain can do various tasks. Some tasks resides in a part of the brain. When a person does some activity like watching TV, then a part of the brain gets active and processes information that belongs to that certain human activity. The part of brain that maps to human vision is called the "visual cortex". When one watches TV the visual cortex gets active and processes information in order to fulfill a certain task. The whole brain exists of certain cells which we call neurons. Neurons are connected to each other and work together to manifest intelligence. When a certain action is performed like watching TV, then the neurons that work together light up and process information in order to do the task "watching TV". You can consider the neurons like a group of ants that work together to do a certain task like bringing food to their nest.

Our review paper is about contour detection, thus everything we discuss in this article implies to the "visual cortex" of the brain. The visual cortex exist of smaller sub-parts, where each of them are responsible to a collection of tasks. Those parts do various processing steps like detecting colors or detecting contours for example. Contour detection is our topic of interest, it resides as a low-level processing step and is done in the the visual area called "V1". V1 exists (for our review) of two types of cells: simple cells and complex cells, which will be explained in section 2.

We will review in this paper the reports [1], [2] and [3]. They all discuss contour detection with surround suppression terms. We first explain the two most important features from the three papers (mainly based on [1] & [2]), so that we have (at the end of section 3) a clear view over the two proposed suppression terms.

To give an outline of the structure of this paper: In section 2 we discuss "Classical Receptive Field" (CRF) with Gabor filters. In section 3 we discuss "Non-Classical Receptive Field" (NON-CRF). Here we will extend the knowledge from section 2 with the proposed suppression terms: Isotropic and Anisotropic. Furthermore we will review the results from all three papers in section 4, and in section 5 we will briefly discuss the results we have found with this review.

2 Classical Receptive Field

In this section we will model the simple and complex cells. We will model the simple cells through the Gabor filter and model the complex cells through the Gabor Energy filter. The simple cells will finally give the base for modeling the complex cells. Where the complex cells on their turn will give base for further modeling in section 3.

2.1 Simple cells

A simple cell can be represented by a classical receptive field (CRF) function and this function can be mathematically expressed by a Gabor Filter. The Gabor Filter is composed by a Gaussian function and a harmonic function, which are multiplied and form the following formula:

$$g_{\lambda,\sigma,\theta,\wp}(x,y) = e^{\frac{\left(-\tilde{x}^2 + \tilde{y}^2 \gamma^2\right)}{\left(2\sigma^2\right)}} \cos\left(2\pi\frac{\tilde{x}}{\lambda} + \wp\right)$$

$$\tilde{x} = x\cos(\theta) + y\sin(\theta)$$
$$\tilde{y} = -x\sin(\theta) + y\cos(\theta) \tag{1}$$

The formula consists of the following five parameters: γ is the ellipticity of the neuron, the size of the neuron is stated by σ and is the standard deviation of the Gaussian part of the formula. The wavelength is represented by λ, the phase offset by φ and the preferred angular orientation of the neuron by θ.

The Gabor Function models the simple cell, but excitement comes when the neurons receives an input signal. So eyes receive an input signal and convert it as a signal to the brain. Where the neurons, in V1 of the visual cortex, responds and thus process the image.

To model this response we use the next formula:

$$r_{\lambda,\sigma,\theta,\wp}(x,y) = \left(f * g_{\lambda,\sigma,\theta,\wp}\right)(x,y) \tag{2}$$

The response of the neuron is a convolution between the input image with luminance distribution f(x,y) and the Gabor Function.

2.2 Complex cells

The other type of V1 cells are the complex cells. A complex cell consist of a pair of simple cells. A pair of simple cells only forms a complex cell when they are in phase difference to each other. The function that belongs to complex cells is called the Gabor Energy Filter and is expressed as:

$$E_{\lambda,\sigma,\theta}(x,y) = \sqrt{r^2_{\lambda,\sigma,\theta,0}(x,y) + r^2_{\lambda,\sigma,\theta,-\pi/2}(x,y)} \tag{3}$$

The formula takes the response function of each of the two neurons in phase difference. The response of a complex cell is a Gabor Energy function.

3 NON-CRF inhibition models

Now we extend the complex cell model with a suppression term. This reproduces the influence of suppression that surround texture could have in a qualitative way. It actually means that we qualitative reproduce the NON-CRF suppression behaviour of almost all orientation-selective cells. This suppression term is calculated by the balanced summation of the other neurons responses in a ring-formed area surrounding the involved neuron (CRF), centered at the concerned point in the image (see fig. 1 from [1]).
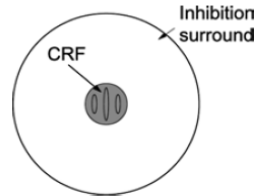
Fig. 1. Non-CRF suppression is caused by the surrounding neurons of the CRF, which is defined by the weighting function.[1]

We are using the normalized weighting function defined as:

$$w_\sigma(x,y) = \frac{H(DoG_\sigma(x,y))}{\|H(DoG_\sigma)\|_1}, \quad H(z) = \begin{cases} 0, & z < 0 \\ z, & z \geq 0 \end{cases} \quad (4)$$

The construction $\|.\|_1$ denotes the L1 norm and gives the normalizing of the operator. The L1 norm is also called the Manhattan or city block distance. The H(z) function is called the halve-wave rectification and it ensures that the operator has only positive responses.

For completeness; the Difference of Gaussian function or DoG_σ is computed by:

$$DoG_\sigma(x,y) = \frac{1}{2\pi(4\sigma)^2} e^{-\frac{x^2+y^2}{2(4\sigma)^2}} - \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \quad (5)$$

Two types of suppression will be considered in this paper: a) Isotropic, here all responses of the orientation-selective cells outside the CRF (concerned neuron) contribute, independently from the preferred orientation of the cells, in an equal way to the suppression. b) Anisotropic, here only the responses from the same oriented cells as the concerned neuron, are contributing to the suppression. We first consider the Anisotropic suppression and then the Isotropic suppression.

3.1 Anisotropic inhibition

The Anisotropic inhibition will be computed by the suppression term:

$$t^A_{\lambda,\sigma,\theta_i}(x,y) = (E_{\lambda,\sigma,\theta_i} * w_\sigma)(x,y) \quad (6)[1]$$

This term is a convolution of the weighting function $w_\sigma(x,y)$ and the Gabor energy $E_{\lambda,\sigma,\theta_i}(x,y)$ for each orientation θ_i.

---

1 *A in formula = Anisotropic! It will also be used for other formula to give a difference between Anisotropic and Isotropic.*

To make the Anisotropic inhibition model complete, a new operator is introduced:

$$\tilde{b}^{A,\alpha}_{\lambda,\sigma,\theta_i}(x,y) \;=\; H\!\left(E_{\lambda,\sigma,\theta_i}(x,y) - \alpha\, t^{A}_{\lambda,\sigma,\theta_i}(x,y)\right) \tag{7}$$

This operator is using the H(z) as defined in (4). Also it uses α as a factor that controls the strength of the suppression of the surroundings on the Gabor Energy operator. Another important feature of α is: when taken negative values for α, it can model enhancement.
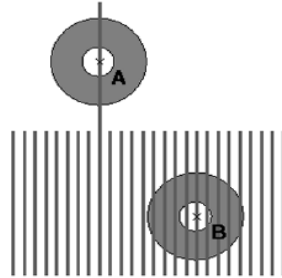
Fig. 2. In contour point A, the suppression term is small. In texture point B, this term could become very strong. [2]

The function of this new operator can be interpreted as follow. Suppose there is no texture in the surrounding of a given point (see fig. 2 point A from [2]), then this operator will respond equally to the Gabor energy operator (because there is no suppression). On the other end of the spectrum, when there are other textures (for example edges in case of point B in fig. 2 from [2]), then the suppression term could become so huge that it will completely cancel the Gabor energy operator and the response of our new operator will be zero.

From this new operator we will construct the contour operator:

$$b^{A,\alpha}_{\lambda,\sigma}(x,y) \;=\; max\{\tilde{b}^{A,\alpha}_{\lambda,\sigma,\theta_i}(x,y)|i=1,\ldots,N_\theta\} \tag{8}$$

This contour operator will give maximum response values over all orientations. Because the operator is defined this way, we could say that it responds to isolated textures (such as edges and bars) of any orientation and that it not responds to groups of <u>same</u> oriented textures (see fig. 3c from [3]).
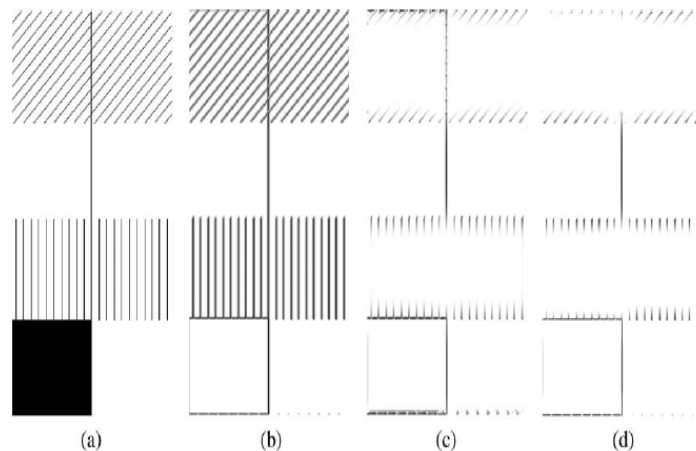
Fig. 3. (a) The input image (b) The complex cell detects all edges (c) The complex cell with Anisotropic surround suppression (d) The complex cell with Isotropic surround suppression[3]

3.2 Isotropic inhibition

To construct the Isotropic suppression term we first need to construct an energy map:

$$\widehat{E}_{\lambda,\sigma}(x,y) \;=\; max\{E_{\lambda,\sigma,\theta_i}(x,y)|i=1,\ldots,N_\theta\} \qquad (9)$$

The energy map gives maximum Gabor Energy response values, with this energy map we construct the Isotropic suppression term:

$$t_{\lambda,\sigma}^{I}(x,y) \;=\; (\widehat{E}_{\lambda,\sigma}*w_\sigma)(x,y) \qquad (10)^2$$

As before by the Anisotropic suppression model, we have a convolution with the weighting function. Also we construct a contour operator for the Isotropic suppression model:

$$b_{\lambda,\sigma}^{I,\alpha}(x,y) \;=\; H(\widehat{E}_{\lambda,\sigma}(x,y)-\alpha t_{\lambda,\sigma}^{I}(x,y)) \qquad (11)$$

Again is H(z) defined as in (4) and factor α controls the strength of the suppression of the surroundings on the maximum energy map. This contour operator will respond in the same way as the contour operator with Isotropic suppression to isolated textures (such as edges and bars). It actually differs from the Isotropic suppression contour operator, because it gives a reduced or no response at all to groups of <u>any</u> oriented textures (see fig. 3d from [3] on the previous page).

4 Results

Our results will be based on the results from the papers [1], [2] and [3]. First we will look how we can interpret the different results from the three papers and after that we will try to compare them.

4.1 Interpretation of the three papers

When we are looking at the three papers, then we conclude that [1] and [2] are based on simple cells (with Gabor filter) and complex cells (with Gabor Energy filter). As a matter of fact the authors continued with two suppression terms based on the complex cells. In [3] they propose two gradient-based contour detection operators that incorporate surround suppression. They use a similar technique as Canny [3] has proposed.

We could say that [3] is more different from the other two, then [1] and [2] from each other. Still we would like to compare the results of them. We discovered that it is possible to compare [2] and [3] on the basis of the performance, because they used the same performance measure formula. Nevertheless it should be said that under the bonnet they still have a different approach. In contrast; [1] has no explicit performance measure formula for the results.

So we could compare [2] and [3] on the performance measure (formula), but for [1] we don't have it. The papers [1] and [2] have more or less the same base models and suppression terms. Therefore we could make a good assumption of the differences in the results, to only compare the performance measures from [2] and [3].

---

2 *I in formula = Isotropic! It will also be used for other formula to give a difference between Anisotropic and Isotropic.*

4.2 Comparison of the three papers

The performance measure in [2] and [3] are done by having man-made and natural images. From these images is indicated what the desired output should be. After this, the performance is evaluated for the different operators. In [2] these operators are: a Canny operator, a Gabor Energy filter without suppression term, one with the Anisotropic and one with the Isotropic suppression term. In [3] these operators are: a SUSAN operator, a Canny operator without suppression term, one with the Anisotropic and one with the Isotropic suppression term.

The just mentioned performance measure formula is defined as:

$$P = \frac{card(E)}{card(E) + card(E_{FP}) + card(E_{FN})} \tag{12}$$

Card(Y) will give the number of elements of set Y. E is defined as the set of correctly detected contour pixels, E_{FP} is defined as the set of false positives (operator indicates it as pixel of contour, but it belongs to the background of the desired output) and E_{FN} is defined as the set of false negatives (desired output contour pixel missed by operator). When the performance measure P=1, then all the desired output contour pixels are detected and no false positive or false negatives are detected by the operator. Because the P-value will be on the interval [0,1], then the outcome will mean that the performance is less optimal when P is lower then 1.

When we look at the results mentioned in [2] and [3], then the general result is that man-made images have the best Performance measure with Anisotropic suppression (see P for the Triangle of table 1 from [2]). In contrast with this; when having tested several different natural images, then at first instance (supported by table 1 from [2]) the best Performance measure is taken by the operator with Isotropic suppression.

Taking a closer look to the results of [3], then they learn us that the scale of the images is influencing the result of the operator. But when there is no information for picking the best set of parameters (for instance one of the parameters could be the scale) for one of the operators, then there is a higher probability that the surround suppression operators are delivering a better result then the Gabor Energy filter, Canny and SUSAN operators (supported by [2] and [3]).

5 Discussion

This review has delivered the conclusion that the two proposed suppression terms (Isotropic and Anisotropic) are usable for Gabor filter, Gabor Energy filter and Gradient based operators. The Isotropic suppression term delivers us an equal suppression for the surrounding of the concerned neuron, no matter which orientation the surrounding neurons have. In contrast the Anisotropic suppression term gives us only suppression for the same oriented neurons, surrounding the concerned neuron.

The three reviewed papers ([1], [2] and [3]) have delivered us the result that the surround suppression operators proposed in these reports are generally better, then the Canny, SUSAN, Gabor filter and Gabor Energy filter operators.

References

1. N. Petkov, M. A. Westenberg, "Suppression of contour perception by band-limited noise and its relation to Non-Classical Receptive Field inhibition", 28 February 2003
2. C. Grigorescu, Student Member, IEEE, N. Petkov, and M. A. Westenberg, "Contour Detection Based on Non-Classical Receptive Field Inhibition", 7 July 2003
3. C. Grigorescu, N. Petkov, M. A. Westenberg, "Contour and boundary detection improved by surround suppression of texture edges", 16 December 2003.

Web Service Composition Review

Ilkka Harmanen[1] and Moses Matovu[2]

[1] ilkka.harmanen@nic.fi - Student number: 1666800
[2] M.Matovu@student.rug.nl - Student number: 1655655
Univertity of Groningen, The Netherlands

Abstract. *The Web services model is used to develop and implement network distributed business processes using standard interfaces and prorocols. The aim of this article is to give a review of four articles about Web services and Web Service Composition. Various approaches to Web Service Composition have been proposed and also, several protocols and languages are being used to combine Web services, but currently there is none that completely solves the underlying issues of automatic Web Service Composition.*

1 Introduction

Internet is widely considered as a large database with several interfaces, however with the recently developed state-of-the-art Web interface methods and technologies, the Web has the potential to be transformed into a complex distributed network of Web based services. Web services are autonomous network accessible software that can enable the provision of useful services to easily achieve a simple task. This is however much more convenient if individual services (however simple they may be) are combined to handle more complex tasks and it would even be a more interesting aspect to have in place automatic service composition based on users' needs.

The development and execution of business processes distributed over networks and accessed using standard interfaces and communication protocols are based on Web services model. Web services application is an emerging area of interest, not only in Service-Oriented Computing, but also in Service-Oriented Architectures. Web Services Co-ordination and Transaction support require frameworks and standardized protocols. Transactions in Service-Oriented Computing are more complex and involve many roles. Web Services Transactions require coordination behavior to control the operations and results of applications. They

also need the capability and flexibility to handle the coordination of processing outcomes from various services.

Web services are self-contained and self describing module based applications composed of several components that allow distribured commuincation using Simple Object Access Protocol (SOAP). This means that the componentised view of the Web applications is becoming an emerging platform for distributed computing and the individual components interact over XML messaging protocols and interoperate with each other using SOAP.

It is widely believed that the next generation of mainstream applications will be based on autonomous Web services and the implications of this autonomy are central to the architecture. If individual Web services were combined to handle complex tasks, it would be more convenient and above all, it would be also crucial and would need automatic composition of Web services based on user requirements.

The success of the internet has made it possible for many companies to conduct part of (or all) their businesses on the Internet. However, as providing such services is getting more complex, so is the customers need for different and more extensive services.

This paper is organised as follows: in Section 2, we give a brief description of the reviewed articles, Section 3 provides a description of the basic concepts behind Web services and Web Services Composition including the analysis of some of the technolgies (protocols and languages) used. In Section 4, we discuss the different approaches used in all the reviewed artcles and finally give our conclusions in Section 5 and a summary of the envisaged possible future directions of this particular field of study.

2   Reviewed Articles

Web Service Composition is an area of ongoing research and is now being studied extensively. The aim of this article is to review four papers about Web Services Composition by Fensel and Bussler [1]; Dustdar [2]; Pistore ,Traverso, Bertoli and Marconi [3] and Srivastava and Koehler [4].

There is an increasing need for dynamic Business-To-Business interactions on the Web which calls for the implementation and application of certain Internet interface standards and/or communication protocols all over the globe. Different

Web development languages and standards are being used and others are being proposed to develop Composite Web Sservices. Business Processing Extraction Language for Web Services (BPEL4WS) is one of these standards; it describes the behavior of services and how components are represented as stateful processes that publish interaction protocols with external Web services [3].

Article [3] gives an overview of automatic Web Service Composition, how to find the cheapest and most fitting Web service to execute a required task. The paper also states that current technologies provide only limited support in automatic service recognition, configuration and combination. The authors of the paper consider semantic Web technologies to be one of the solutions for this problem. They introduce a framework titled Web Service Modeling Framework (WSMF), that is centered around two principles "Strong de-coupling of the various components that realize an e-commerce application" and "Strong mediation service that enables anybody to speak with everybody in a scalable manner".

Article [2] considers Web services and their composition as newly emerging areas of interest; it discusses the automatic composition of Web services and proposes the need for Web services frameworks and protocols. The author states that the existing protocols do not cater for quality-awareness and long-running transactions, but in general the transaction model with Web services should be more relaxed

Article [3] describes how to generate automatic executable abstract BPEL4WS. The authors go on to state that automatically created BPEL4WS descriptions are essentially as good as hand-written programs, although slightly longer. There is a unique feature in this article, which concerns using the EAGLE - language for expressing the composition requirements.

3   Web Services

The Web has registered a phenomenal and enormous success as far as human-computer interaction over the Internet is concerned. The use of protocols like HyperText Transfer Protocol (HTTP) and languages like HyperText Markup Language (HTML) on Web browsers nowadays has turned out to be cost-effective in projecting user interfaces over a wide range of devices.

Web services communicate using a set of standards, that share a common architecture like the World Wide Web does. Web services (like the Web), require

an infrastructure that provides a mechanism for clients to find the Web services (UDDI), also to define collections of network endpoints or ports (WSDL) that provide a standardized way of describing the Web services strucrure. In this structure, SOAP provides support for information binding. In a typical Web services scenario, a business application uses the SOAP protocol over HTTP to send a request for a service at a Unified Resource Locator (URL). The service provider receives the request, processes it, and returns a response. Web services architecture is designed for highly dynamic program to program interactions and can support the implementation of several types of distributed systems including asynchronous and synchronous messaging systems, distributed computational clusters, mobile- networked systems, grid systems and peer-to-peer environments.

The World Wide Web Consortium (W3C) Web services architecture gives two aspects of a full description of a Web service and the first is the syntactic functional description as represented by WSDL whereas the second is described as the semantics of the service and is not covered by any specification.

Web services and their consumers are typically business oriented, making Web services predominantly business-to-business (B2B) transactions. An enterprise can be a Web service provider and at the same time, a consumer of other Web services. This explains why Web services are used to implement business solutions.

Web Services Composition supports the reuse of published services to reduce the development time and the efforts involved in developing a new application(s). Composite Web services operate by providing interaction between components of the Web services. The life cycle of Web services involves development, deployment and usage phases; and in particular, it may involve models, languages and interfaces for description, implementation, publishing, discovery and binding, invocation and execution. Web services end-points are described using WSDL.

Web services implementations need to be published in a registry in order for users to be able to search and find them, and thereafter be able to access and utilize them. Registries are normally hosted by private companies or third parties.

3.1   Web Services Composition

Article [1] describes several ways how Web services can be composed; one of the methods is called combined logic, which is when a Service requester calls several services from single provider; another method is called complex Web service, which is when a Service provider combines several of its services, but has problems if the Service requester needs to know the status of sub-services during execution. Also, all the input data must be available at the time upon starting the service; a Service requester can also call a complex service which in turn calls several services from other Service providers and provide this as a service. The last possibility the authors of [1] describe is defining the constraints and invocation sequences for several services, which could be seen by the requester as another Web service.

BPEL4WS is a business protocol specification language that operationally describes the stateful behavior of Web services on top of the service interfaces provided by the specifications in WSDL. An abstract BPEL4WS description identifies those parts of a Web service, its internal variables and operations [3] that can be accessed when a service in needed. BPEL4WS processes are encoded as state transition systems that describe dynamic systems.

Over the web, communication protocols and message formats are standardized and this is important in describing communications in a structured and standard way. WSDL addresses this need by defining an XML syntax for describing network services as collections of communication endpoints capable of exchanging messages.

WSDL is an XML format for describing network services as a set of endpoints operating on messages containing either document-oriented or procedure-oriented information. WSDL as an XML messaging protocol has collections of message-enabled endpoints or ports. The association of protocols and data format specifications is used to implement the Web services on the web. The operations and messages are described abstractly, and then bound to a concrete network protocol and message format to define an endpoint. Related concrete endpoints are combined into abstract endpoints (services). WSDL is extensible in that it allows the description of endpoints and their messages regardless of what message formats or network protocols are used to communicate.

Article [1] describes UDDI (Universal Description, Discovery and Integration of business for the Web), a standard that provides mechanism for clients to find

Web services. Fensel and Bussler [1] describe UDDI-registry as a phonebook containing white pages (general information about the company such as address, short description, contact information, etc), yellow pages (service descriptions and list of categories that describe the service more accurately, such as purchasing, shipping, etc). Finally, within the business service, there are the so-called green pages that provide technical information about the service.

3.2 Semantic Web Solutions

The semantic Web can be used as method to tackle the Web service composition problems. In practice semantic Web technologies enable adding machine-processable meta-information on data. With this kind of information, it is possible for computers to actually understand the context of the document and thus make semantic connections around the internet; enabling new way of linking services and webpages. As of now, even though the required technologies are becoming more common, there is still not enough meta-information online for forming decent semantic-network [1].

Semantic Web enabled Web services have a huge potential in public process description and advertisement, discovery of services, selection of services, composition of services, delivery, and lastly monitoring and contract negotiation [1].

4 Comparisons

A direct comparison between these papers is not possible since each focuses mostly on different parts of Web services life-cycle. There is no standard transport mechanism that is used by Web service requestors and providers. Both have to agree on a mechanism to be used when service requests are executed. For each available mecahnism, a layout of the message has to be agreed upon and how the document shall be presented in the message. As for exchange sequence definition, communication over networks is unreliable. Service requesters and providers should use protocols to ensure that messages are transmitted exactly once. UDDI, WDL and SOAP are important steps in the Web services composition, however, they only address part of the overall requirements needed to have expressive Web services.

Web Service Modelling Framework (WSMF) [1] was developed to provide a rich conceptual model for the development and description of Web services to bring the technology to its full potential. This describes Web services as interfaces accessible via a network - the external aspects of a Web service - unlike WSDL and other Web description approaches discussed in other papers that do not make an explicit distinction between the internal description of a Web service and its external visible description. In articles [2] and [3], WSDLs define elementary Web services as simple input and/or output boxes whereas complex Web services as those that breakdown the entire process into sub-tasks that may call other Web services. This distincton may lead to misconceptualisation because it is not the complexity of a Web service and/or its simplicity that makes a crucial difference but rather the complexity of its interface. This is to say that, complex exernal visible description makes the relevant difference as it is the case in WSMF. WSMF language does not define concrete syntax or semantics for WSMF but it is an extension of WSDL.

Srivastava and Koehler [4] investigated how Web services in WSDL and Web services in the semantic Web differ with respect to modelling, verification and deployment of services and their respective inference methods and runtime support that they assume. The authors state that Web services description can be solved in various ways; and they give a typical problem and provide its industrial solution using WSDL and BPEL4WS, and at the same time give a semantic Web solution using RDF/DAML-S and Goleg/Planning. The specification of composite services whether given in BPEL4WS or DAML-S, encodes process information that can be bound to different protocols. BPEL4WS puts much emphasis on error handling and message collection however both BPEL4WS and DAML-S allow customer satisfaction of plan execution at runtime.

5 Conclusions

The purpose of this paper wass to reviewing the Web Services Composition approaches contained in four different research papers mentioned in section 2. Web services description, composition and standards have been discussed in particular among many other asects discussed and the findings are briefly summarised below.

The use of prospective coordination middleware efficiently manages and monitors large-scale Web services workflows and also deals with short-lived and long-lived types of transaction.

Executable BPEL4WS processes compose Web services which can be generated automatically from composite requirements and abstract BPEL4WS descriptions of components. Automated synthesis is feasible within reasonable time and much faster than the manual development of BPEL4WS composite process however, the quality of both automatic and manual BPEL4WS processes is not clearly discussed and elaborated. The Semantic Web community tries to solve the problems related to automated synthesis by use of Semantic Web to describe services such as Ontology Web Language service (OWL-S) descriptions of the input and/or output's pre-/post-conditions.

Furthermore, several approaches are proposed and many are used to support various forms of Web services compositions, but none addresses fully the issue of automatic Web services composition. Automatic synthesis of composite BPEL4WS only considers a small part of the entire problem at hand since it only focuses on generating a new composite Web service that interacts with the existing Web services rather than the issue of Web Services Composition in general.

As for the foundations for BPEL4WS coordination middleware and related coordination and transaction frameworks, Web service workflows must cope with not only short- but also long-living transactions, which is currently beyond the state-of-the-art solutions. Some of the current services assume tight coupling between the services, however the very idea behind Web services is based on the possibility of loose-coupling.

WSMF, is a modelling framework whose main elements are ontologies, goal descriptions, elementary and complex Web services and mediators. Its goal is to enable fully flexible and scalable e-commerce based on Web services. Presently, most work on Web services is focused on Web services description, flow and execution.

In typical Web Service Composition scenarios like plane ticket, hotel room and/or car rental bookings; it works fairly well. However, since Web Service composition has the possibility to automate the entire B2B processes, this could mean that every single order could be automatically and transparently com-

pleted. However, it is not only price, but also quality that need to be included in the descriptions.

6 Future Research Interests

The future interests of the research should address the problem of associating finite ranges to various data types in the generation of the state transition systems from the BPEL4WS Web Services focussing in particular the techniques to discover the right ranges.

Also future work should focus on Service oriented transactions to cater for multiple views on services, their conversation, aggregation, integration, coordination, enactment, and transactional properties.

Finally, one of the other most interesting aspects of Web services is Enterprise Application Integration (EAI). As a future area of interest, adapters should be put in place for older web-based applications, to make them (older web-based applications) work as loosely-coupled software so as to to make the cost and difficulties of integrating older systems cheaper.

References

1. D. Fensel and C. Bussler: *The Web Service Modeling Framework WSMF, 2002, Electronic Commerce: Research and Applications, 1(2): 113-137, 2002*
2. Sachahram Dustdar: *Web Services Workflows - Composition, Co-ordination and Transactions in service-oriented computing, 2004, Concurrent Engineering, Sep 2004; 12: 237 - 245*
3. M. Pistore & P. Traverso, P. Bertoli, A. Marconi: *Automated Synthesis of Composite BPEL4WB Web Services, 2005, 3rd IEEE International Conference on Web Services 2005*
4. Biplav Srivastava & Jana Koehler: *Web Service Composition - Current solutions and open problems, 2003, ICAPS 2003*

Linking the customer to the software development process

The Looping Framework

R. Krooman (1650114), M. de Jong (1650181)

rkrooman@xs4all.nl, michel@hostname.nl
Department of Computer Science, University of Groningen
Published: 21 January 2007

Abstract. This article introduces a software architectural framework (called The Looping Framework) that primarily focuses on successfully incorporating the customer into the software developing process. The entire framework is based on a series of states which allows separate addressing of concerns of the involved stakeholders. Achieving maximal involvement of the customer through constant repetition over the states during the development process. *Keywords: software architecture, framework, software development, design, customer.*

1 Introduction

The development of software remains a difficult task and can be considered an architectural art. The word 'software' should be read as 'good software', meaning that it was completed to the satisfaction of the customer. Numerous completely different factors play a role in successfully constructing a software program.

One has to consider the costs, time, and availability of employees and so on. Yet, without probably realizing it, one of the largest considerations is the customer. Their ideas play the most prominent role in building the software that complies with the customer's wishes. Therefore we can certainly state that one of the biggest concerns in software development is correctly understanding the customer. In order to properly translate these wishes into code we turn to software architectures.

Software architectures describe how a system is decomposed into components, how these are interconnected, and how they communicate and interact with each other. When poorly understood, these aspects of design are major sources of errors [4].

2 Earlier scientific research

The research area of software architecture is an emerging one, with little agreement over the definition of architecture. The only consensus seems to be that architecture is related to the structure of a system and the interactions among its components [2, 3]. In general, the literature describing a certain software architecture mostly relies on informal examples and anecdotes, and most of the work has not been proven to be successful on large projects. Scientific research addresses these problems in profound detail, focusing on distinguishing different types of architectures that support different types of projects [4]. Also great effort has been made to construct a way to quantify quality attribute requirements [1][5].

Another paper provides a view model to capture the gist of an architecture through the usage of five views [6]. In all of the these scientific papers involvement of the customer is implicitly addressed. We propose to explicitly study the participation of the customer in the developing process. An author of a software book once said: „Software architecture is the set of design decisions which, if made incorrectly, may cause your project to be canceled."[1]. Considering this paradigm and with referral to the study of the essential role of a customer in the developing process, the focus of this paper lies on successfully involving the customer with the development process. This leads to the following research question:

"How can the customer be actively linked into the developing process, and keep this involvement as undemanding as possible?"

---

[1] Software quality attributes are the benchmarks that describe system's intended behavior within the environment for which it was built. The quality attributes provide the means for measuring the fitness and suitability of a product.

3 Roadblocks and bulldozers

In order to connect the customer to the entire developing process we should first look at the different roadblocks that cause a project to become unsatisfactory. The biggest obstacle is where to involve the customer in the development process. Smaller blockades are: who will communicate with the customer, engineers, software management, et cetera…?

To pave the road to a successful project we have developed a framework. In this framework every aspect of the development of a software program is taken into account. This is accomplished by taking a different point of view for every aspect. When translated into a framework, this reflects as four different states. Each of the states has its own specific responsibility.

- Customer state

This state describes and specifies a design in an informal way. The customer brainstorms about an idea and informs software management about this idea.

- Functional state

In this state the design is translated into a software diagram which divides the design into several smaller designs.

- Technical state

In the third state the translation to a more technical diagram is made. Also early problems from either the functional or technical design will be noted here.

- Evaluation state

The last state in the looping process is communicating with the customer about the possible problems and questions. This will involve answering and solving the questions and problems. Finally leading to a revision of the original design. The revised design is called a system. In case of approval of the system The Looping Framework delves into a subsystem.

4 The Looping Framework

Why use a framework that continues to loop over a series of different states? To answer this question we should first take a look at the problem that arises when using a single run framework or design method. Suppose using a framework or design method that passes every state once. When facing problems in one of these states it is often hard to solve the problem without backtracking to the former state. When backtracking to an earlier state the customer is often left out. Not to mention the impact it will have on the entire design: time, costs and scheduling of the project.

The Looping Framework deals with very high levels of abstraction. It does not try to separate design concerns or be applicable to a certain software architecture [4].
Nor does it try to solve the lack of specificity of quality attributes when developing software[6]. Its main point is creating a high level of organizational abstraction in the development of software. By doing so, it separates itself from the earlier scientific research made with regard to software architectures.
It does however use ideas from the '4+1' view model [5] approach. The concept of decomposition of the system into several subsystems makes for a solid base. When using several views (or states), addressing the concerns of a software system with various stakeholders becomes a much less complicated and problematic task.

4.1 How does it work

In The Looping Framework this problem is solved through continuously looping over the system and its subsystems by traversing four states. In every state the subsystem is scrutinized from a different point of view. By viewing the subsystem from different angles, problems will easily be recognized. These problems shall be summarized in a problem log[1].
In the last state (evaluation state) this problem log is transformed into an easy readable report which will be passed on to the customer. The customer is given the chance to give feedback, and after this the changes will be incorporated into the design. This chain of events will continue during the entire project. This way the customer is completely aware of any faults or problems taking place during the course of to the project.
Next to problem solving, the customer is given the chance to brainstorm about specific implementation related issues. Besides setting and developing the global outlines of a project, most customers desire to be part of developing subsystems or even smaller parts of the system. The Looping Framework allows customers to take hold of that prominent role they ought to have within a project. In figure-1 the design of The Looping Framework design can be found.

---

[1] A problem log is a manually created collection of problem
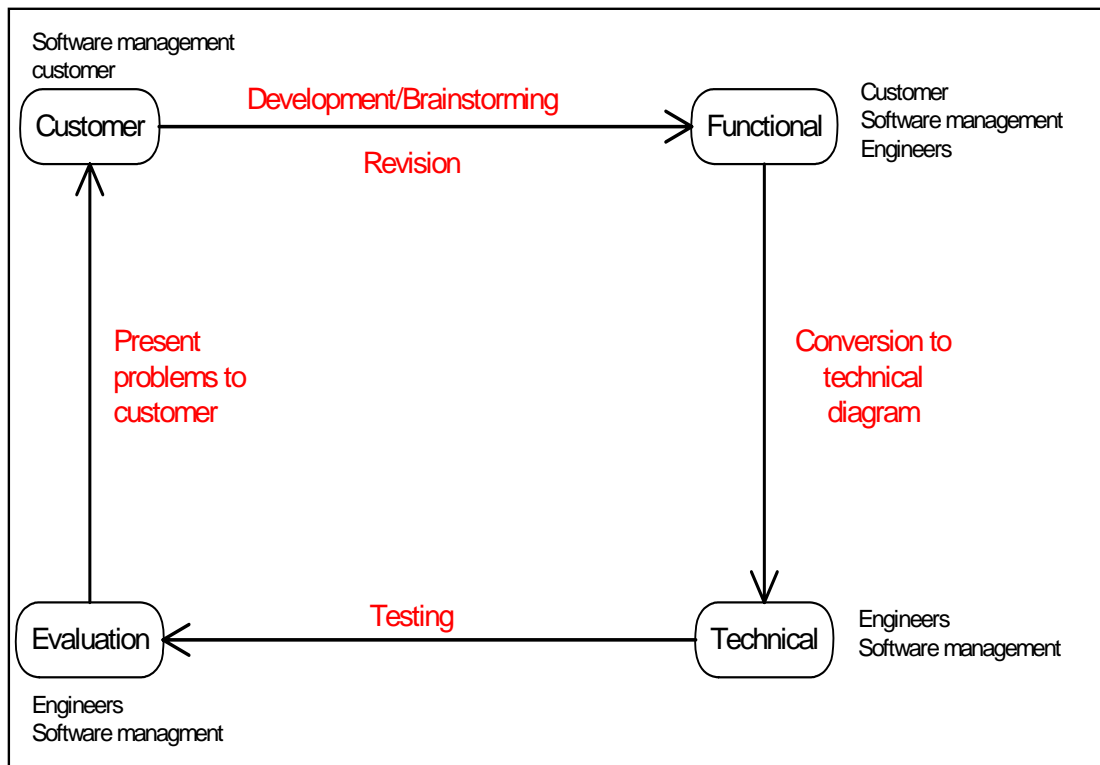
Fig. 1 *The Looping Framework*

5 Customer state

A customer comes up with a great idea or needs to automate something. What they want, needs to be translated into a program that complies with their initial idea. Unfortunately what the customer actually wants, tends to be a little vague. So the information really needs to be extracted from the customer. The responsibility for getting this information lies with the developer, not the customer.

Because the customer is not aware of all the implications of the required functions -let alone know every required function- this information can not be retrieved in a single talk or meeting.
When retrieving the required information for designing the software, the customer could also decide to change certain aspects of the project during its course. Also the customer may disagree on certain solutions for a problem or find that their idea was not correctly interpreted by the developers.

This state provides the customer and software management with all the required details to make a solid (informal) design. In the first run the focus lies on extracting all possible functions required by the program. The following runs provide the customer with possible problems and questions that rose while constructing a solid design. In every run the customer should respond to these questions in order to help solving these problems. This continuous looping over the customer state enables the construction of a clear and correct design. In turn of the engineers this will provide an understandable interface to build the software.

6 Functional state

In cooperation with software management, the customer created an informal blueprint of what they want. For the engineers to make a good technical representation of this blueprint, it is imperative to have a well-structured functional design.
A functional design is a level of the design process in which subtasks are specified and the relationships among them are defined, so that the total collection of subsystems performs the entire task of the system.
Software management and the engineers will use the informal blueprint to create a functional design using **U**nified **M**odelling **L**anguage (UML).

UML shows the dependencies between subsystems and how they relate to each other. This is a very important issue, because if there is an error anywhere in the relationship, it is mostly very inefficient to repair this in a later stage of the project. Often these errors lead to different problems for the actual writing of the software and will inevitably lead to the wrong solution.
UML is the standard of gradually creating software in a way of dividing a software program into multiple subsystems. These subsystems can be libraries, packages, other (sub)programs or chunks of program code (often divided into classes). What defines a subsystem depends on aspects such as the reuse, portability and security. The system also becomes easier to modify as more subsystems are created. The (final) UML results in the (final) product and is considered to be the most important aspect of designing a new system.

7 Technical state

In this state the product of the functional diagram is used to create a technical design. In the technical design concurrency plays a part as well as class dependency, which are both less important in the functional design. Usually a lot of questions arise when an architect develops a technical UML-analysis. This state becomes more and more important as the project progresses.
In the early stages of a project making a correctly detailed technical UML-analysis is often impossible. However, concurrency possibilities can be easily spotted.

7.1 Concurrency

Concurrency is the possibility of running multiple tasks at the same time. In a lot of problems adding concurrency leads to problems. The trick of fast programs (especially internet applications) is to gain the most concurrency. In general the concurrency will not lead to any problems. Often asking the customer about their experiences or predictions about the problems, can solve concurrency issues, which will result in faster development.

In some cases concurrency plays a more essential part. In banking, concurrency is of the highest priority, while in other programs it may give a boost in performance, but it is no necessity.

7.2 Process

In the first couple of loops the main focus will be on concurrency and which problems can arise. In later loops the UML grows as the functions are described more accurately. The better the functions are described, the more detail is possible. This results in questions about the relationship between subsystems. The customer should define how these relationships need to be implemented. This leads to better understanding of the customer's wishes.

In one of the last loops the actual code will be written. There will be a few more loops to improve on performance (if needed). As soon as the actual writing of the code starts, questions towards the customer should no longer be asked.

8 Evaluation state

The evaluation state changes as the project progresses and exists of two different elements. The first is to summarize problems that occur during previous states. The second is testing of the design and implementation; this is done by looking at the specifications given by the customer. Eventually, the testing stage will be running given pre sets of input and evaluating the output of the program with the output the customer wants.

8.1 Summarize problems

In each loop the questions of the previous states (of the same loop) will be reformulated in a way that is understandable for the customer. This step is often not done until the software architect is face to face with the customer. The reason for putting this step into our framework is because this is a source of miscommunication. Often words used by the software architect have no meaning, or a different meaning to the customer than for the architect.

8.2 Testing

Tests are done in order to evaluate the program as far as possible. Testing can be done in different ways. In the first loops the testing is done by comparing which functions are distributed in which way in the UML (comparing the technical state with the functional state). At this point, modifiability becomes important. If a function changes, the UML must also reflect this. If it is unclear whether there's a need for modifiability, this will lead to new questions.

In later loops the testing will be done by writing input files (with an output created by the customer) for the developed program. If the program and the customer come to the same conclusion, the program is correct to that point. If there is any difference in the output, the programmer of the functions will be requested to reprogram the functions. Again this will also lead to a question for the customer about their ideas of the correctness of the implementation.

Results that can be shown, should also be put into a presentation. The results shall be discussed with the customer. Results contain:

Response measurements: Time needed in order to complete a given input.
Evaluation of test sets: Where do the program and the customer agree and more
                            importantly on which
                            points to they disagree.
Possible questions: This depends on the project.
Contradictions: Two problems both have a same path but one fails in the
                   evaluation test set and one is correct, this always indicates
                   missing constraint.

9 Discussion

This paper poses a structurally different aspect than the researched theses. The level of abstraction of The Looping Framework is extremely high, whereas the other papers discuss more specific design structures.

On the other the hand, when looking at paper [5] The Looping Framework shows significant similarities. Paper [5] served as a inspiration for constructing a customer oriented framework.

Besides this, the quality attributes requirements discussed in [6] can easily be embedded in the design of The Looping Framework. For instance Business Qualities [7] like 'Cost and Schedule'[1] can easily be adapted to guard the financial aspects of a software project. Finally the different architectures of [4] can all be applied to The Looping Framework. Trivially said The Framework defines the outlining and high level responsibilities of a project. Therefore software architectures and quality attributes can be easily used in coherence with The Looping Framework.

10 Future work and conclusions

The Looping Framework is still in the stage of development. In spite of this similar frameworks have been successfully applied to several large projects. Maximizing involvement of the customer has proven to be of great advantage during these projects.

Keeping in close contact with the customer during these projects was as beneficial for the customer as for the developers. The customer got exactly what they wanted. Developers had more time to increase modifiability of the software, maintaining a low threshold when working on the project.

During the next couple of months The Looping Framework will be tested more intensively. It will be applied to several fundamentally different projects. The diversity of these projects lies in the way of how actively the customer takes part of the developing process.

---

[1] The cost of the system with respect to time to market, expected project lifetime, and utilization of legacy systems.

References

1. Rozanski, N., E. Woods, 'Software Systems: Working with Stakeholders Using Viewpoints and Perspectives', Addison Wesley Professional (2005)
2. Garlan, D., and M. Shaw. An introduction to software architecture, in Advances in Software Engineering and Knowledge Engineering. V. Amriola and G. Tortora (Editors), Volume 1, World Scientific Publishing Company, New Jersey, 1993.
3. Harel, D,. Statecharts: A visual formalism for complex systems, Science of computer programming, Volume 8 pages 231 – 274, 1987.
4. Soni, D., R. Nord and C. Hofmeister. Software Architectures in Industrial Applications. Siemens Corporate Research inc.
5. Kruchten, P. Architectural Blueprints The '4+1' View Model of Software Architecture. Rational Software Corp.
6. Bachmann, F., L. Bass, M. Klein and C. Shelton. Designing software architectures to achieve quality attribute requirements.
7. International Standard ISO 9126-1: 'Information technology: software product evaluation – quality characteristics and guidelines for their use'. International Organization for Standardization, Geneva, 2000.

Approaches for Integrating Architecture Knowledge in Architectures

Wouter-Tim Burgler[1] and Marnix Kok[2]

[1] `w.t.burgler@gmail.com`, s1526669
[2] `marnix@linux-box.nl`, s1526677
University of Groningen, The Netherlands

Abstract. To keep development costs for software systems low, software architectures (SAs) are used as input for many business processes. SAs are sets of documented design decisions that together make up the design of a system. However, documenting decisions without architectural knowledge (AK), *e.g.* rationale and alternative solutions, decreases the added value of using SAs [1]. Therefore, we present three [2–4] approaches for integrating AK in SAs and compare them on the complexity of documenting and retrieving this AK. The Archium approach [4] integrates SA and its AK in the system its source code. This coupling between AK and source code increases the maintainability of the system best, thus we propose it is the best approach for integrating AK in SAs.

1 Introduction

Throughout the years, complexity of software systems has increased. In a marketplace where being the first to ship a product usually, means a bigger profit-margin, time to market plays an important role as well. These factors put additional pressure on a development team to create products quickly and correctly - software architectures help achieve this goal.

A software architecture (SA) is the result of a series of decisions that are taken during the design process. By documenting these decisions, different parties can develop important insights in the structure and behaviour of the software system.

Unfortunately, it is common practice to only document the decision and not the process that preceded it. The missing information is called architectural knowledge (AK) - Babar [2] describes it as information that explains the context, reasoning about why this solution is better, tradeoffs, criteria and how the decision to use the current solution was reached.

Failing to explicitly document AK leads to evolution problems and increased design complexity. Because the AK only exists in the minds of the developers, this tacit knowledge dissipates in time as employees leave the company or simply forget important details. When alternative solutions and the reasons for their rejection are not documented, they may be chosen later, when the objections are forgotten. Also, useless legacy code may remain in systems because its purpose is unknown. Therefore, it is essential to the successful development of the software system to make this knowledge explicit.

Documenting AK has the advantage that project maintainability increases. During the evolution phase of the software system developers will have a complete overview of the decisions made. Each decision also has an abundance of associated information which can be used to make an informed decision on the path to take. Besides that, a good architecture contributes to the flexibility, integrity and comprehensibility of the software system. These notions emphasise the importance of a properly documented SA and its AK.

Because of the advantages of documenting AK and the disadvantages of not doing so, we propose that AK should be integrated in SAs.

Our contribution to the research of software architectures is the presentation (section 2) and comparison of three approaches to intergrating architectural knowledge in software architectures. The comparison (section 3) specifically looks at the complexity of documenting and retrieving AK in the approaches. It leads to selecting the approach that handles the integration of AK in the most convenient way (section 4).

2 Approaches

This section discusses the approaches to solving the integration problems proposed by the reviewed research.

2.1 Ontology

Kruchten *et al.* [3] propose an ontology of architectural design decisions and their relationships. An ontology is a hierarchial structuring of knowledge about things, by subcategorising them according to the qualities, that define them. This architectural ontology can be used to retrieve the architectural knowledge needed to perform some task, such as: reviewing changes made since a specific date or evaluating the impact of a design decision.

The proposed ontology allows four kinds of design decisions being: existence, non-existence, property, and executive. Existence decisions state that some element/artifact will exist in the system's design or implementation, non-existence decisions are the opposites of these. Enduring traits or qualities of a system are stated in property decisions. Executive decisions do not relate directly to design elements or their qualities, but are driven by political, personal, cultural, financial, and technical constraints. Six attributes are deemed essential for describing any design decision. Obviously, the first one is a short textual statement of the decision itself. Second is the intrinsic rationale stating why a decision was made. The third attribute is scope, which can be universal, system, time, or organization. The fourth attribute contains history information, *i.e.* who made the decision and when, changes to the decision are also recorded here. The state attribute contains the state of the decision, possible values are: idea, obsolete, rejected, tentative, challenged, decided, and approved. Finally, a design decision may belong to one or more categories which allow them to be easily grouped.

Eleven types of relations between design decisions are supported by the ontology. Relations may express conflicts, overrides, alternatives, and constraints between decisions. The extrinsic part of rationale, *e.g.* why a decision was made with respect to other decision, is also contained in these relations. Relations between decisions and other artifacts like a requirements document or the source code are also possible.

The ontology can be created and edited using any ontology tool, it can be visualized using the Aduna Cluster Map Viewer which shows the ontology as a cluster map where edges represent decisions and vertices the relations between them. It allows the various stakeholders to query for information, making this tool very useful for tasks like reviewing for specific concerns and finding critical decisions. However, it is not very suitable for tasks where grouping in categories of entities is insufficient because their relation in hierarchy or time is needed.

This approach integrates AK as textual statements in an ontology that describes the architecture. Both intrinsic and extrinsic rationale are documented explicitly and can be queried by all stakeholders at any time.

2.2 Framework

M.A. Babar *et al.* [2] have specified a framework which can contain a software architecture description. Within this framework the definition of software architecture differs from that used in the other papers [2, 4]. Babar has chosen architectural patterns as the base element for software architectures.

Architectural patterns are similar to design patterns as used in software development. They offer a standard solution to problems, at different levels of abstraction, that often occur when creating a software architecture. Implementing such a pattern in the software architecture has consequences for the quality attributes of the system. Each pattern influences different types of quality attributes, for example implementing pattern A may make the system slower. This means the responsiveness quality attribute changes.

The software architecture description not only contains pattern descriptions, scenarios also are part of the description. A scenario is a description of a situation the software system is used in. Not only does it describe how the system should handle input and output, it also contains a list of quality attributes that are of importance to that particular scenario. Combining the quality attributes of the patterns and those of the scenarios, one is able to assess how much the architecture is in alignment with the requirements.

The framework that implements the architectural entities mentioned above, consists of three major components. They complement each other in capturing, structuring and maintaining architectural knowledge. The three components provide means, procedures and a model. Means to capture architecture knowledge from a myriad of sources, both human and electronic. Procedures for capturing architecture knowledge. This knowledge is then associated with patterns, scenarios and quality attributes. A model that allows the architect to create an overview of the main architectural constructs and their relationships with other entities.

There are several ways to capture architecture knowledge. In some cases designers themselves are obligated to document the rationale of decisions they take. Often, the rationale quality depends on the sense of benefit the designer gets out of it. It is therefore vital that the importance is stressed, and demonstrated at a later date.

It is also common practice that a person is appointed the task of extracting design knowledge from meetings, emails, memos and other type of useful documentation. It has been shown that this approach is very rewarding and has been proposed to become part of any software development process.

It is important that these patterns are extracted from the available architectural resources, human or electronic. Unfortunately, since pattern mining is a manual process it is highly dependent on the experience of the miner. Therefore procedures containing a process model, guidelines and templates to identify, capture and document have been created.

2.3 Archium

Van der Ven *et al.* [4] propose to model software architectures as sets of design decisions. These decisions explicitly contain rationale and the considered alternative solutions.

Design decisions are expressed in Archium its design decision model. Archium is an extension of Java, consisting of a compiler and run-time platform. Its model uses an issue-based approach, the issues are problems that are (partially) solved by the solutions of architectural design decisions.

Rationale, motivation, cause, and choice are all elements of a design decision in Archium. Rationale is described in natural text within the scope of a design decisions. In the rationale text, explicit references to design elements can be made to create a close relationship between rationale and the elements it describes.

All the elements of design decisions are modelled in Archium by expressing them in source code; the AK elements are described in natural text. Modifications to the architecture, which are the result of a chosen solution, are modelled in Archium or Java code that alters the existing design implementation. Because this code is written per design decision, changes to the design (and thereby its history) are recorded.

By combining SA and source code, Archium solves the major problem of having inconsistencies between these two. These inconsistencies can arise when artifacts are updated independently, *i.e.* the source code is changed but the SA is not altered accordingly or vice versa. They practically make the SA useless as it makes no sense to reason about the system when it no longer accurately describes the system.

3 Comparison

This section compares the approaches that are reviewed in section 2. The comparison is made with regard to the complexity of documenting and of retrieving

AK in the architecture. Of course, both documenting and retrieving AK should be very easy to use, in order to make these processes accessible for all stakeholders.

3.1 Ontology

Compared to the other approaches, this approach contains a large categorisation of design decisions and their relations. An abundance of information associated with design decisions is not a bad thing. However, it is easy to get lost in the vast amounts of information if no proper tooling is available.

Because of the differences and subtleties of this categorisation, extensive training is necessary to effectively use the ontology. Tooling support is available but is fairly complex for both documenting and retrieving AK, and does not allow all tasks to be performed.

3.2 Framework

Unlike the Ontology and Archium approaches, this approach chooses a different set of definitions by which to categorize the contents of an architecture (section 2.2). This difference is not necessarily a bad thing as it creates a perspective on the architecture that includes quality attributes.

However, the methods suggested to keep the framework up-to-date are quite time-consuming, paper [2] even suggests hiring new personnel to keep the SA documents up-to-date because of this. This has a negative influence on the architecture integrity because developers have to communicate their findings, possibly forgetting to share all the details.

3.3 Archium

The last of the approaches is perhaps the most interesting as Archium takes a rigorous approach to integrate the AK and software architecture.

Archium models the architectural entities such as design decisions in a superset of the Java programming language, thereby integrating the source code and architectural documents into one. This increases component reusability, maintainability and removes the need for developers to constantly update documents to reflect the current state of the software system.

Because Archium is able to integrate the architecture and source code, not just the relation between the decision and its AK is documented but also the relation with the source code. We regard this as a major advantage as it allows to find the code that goes with some decision but, maybe even more interesting, it allows the decision that goes with some code to be retrieved. Because of this, useless legacy code can be easily removed when the decision that it belongs to is withdrawn and useful code can be reused when a decision is made again in another system.

There are some problems however in the use of Archium. Firstly, the integration of architectural entities with Java causes a paradigm shift. The development

team needs to be trained and has to build experience in utilizing Archium efficiently. Also, because AK is integrated in the source code it will mostly be documented by software engineers which is not always the best solution. Finally, the use of Archium is the first design decision for the SA of a system which currently implies that Java will be used as the programming language of choice.

However, when Archium will support other programming language its most important restriction will disappear. Also, with proper tooling and visualization AK may be documented and retrieved by any stakeholder. Finally, because of the advantages of Archium we think that the investment in training the development team is worth it.

4 Conclusion

We compared three different state-of-the-art approaches for integrating AK in SAs on their complexity of documenting and retrieving this AK. The Archium approach was chosen as the approach that integrates AK in a way that it can be easily documented and retrieved. Although the use of Archium requires a shift of paradigm, we propose it is the most appropriate approach for integrating AK in SAs because (a) it allows documenting AK such that tacit knowledge is preserved and (b) it integrates the SA and its AK in the source code, making inconsistencies between SA and source code impossible.

Using Archium, the maintainability of a software system is increased because AK is never lost thus it can always be used to make informed decisions. Also, SAs documented in Archium allow stakeholders to reason about the design of a system easily which will increase its quality. Finally, Archium supports reusability of architectural knowledge because implementations of design decisions may be used again in other systems.

Conclusively, Archium provides all of SA and AK its advantages and even adds some because of the integration in source code. This allows high quality software systems to be created relatively quick while being highly maintainable, giving companies using the Archium approach an advantage over others.

References

1. Tyree, J., Akerman, A.: Architecture decisions: Demystifying architecture. IEEE Software **22**(2) (2005) 19–27
2. Ali Babar, M., G.I.B.K.: A framework for supporting architecture knowledge and rationale management. Rationale Management in Software Engineering (2006)
3. P. Kruchten, P.L., v. Vliet, H.: Building up and reasoning about architectural knowledge. The Second International Conference on the Quality of Software Architectures (QoSA) (2006)
4. van der Ven, J.S., Jansen, A.G.J., Nijhuis, J.A.G., Bosch, J.: Design decisions: The bridge between rationale and architecture. In Dutoit, A.H., McCall, R., Mistrik, I., Paech, B., eds.: Rationale Management in Software Engineering. Springer-Verlag (2006) 329–348

Solutions and motivations for preserving architectural knowledge

Adam Loorbach, Erik Staal

A.Loorbach@student.rug.nl, H.B.Staal@student.rug.nl

Abstract. A lot of existing software architectures suffer from architectural erosion, as a result of lost architectural knowledge. This paper will start with a quick overview of the problems with documenting design decisions in the most commonly used approaches to software architecture. To solve these problems, the solutions of various authors have been reviewed. These proposed solutions will be compared to determine the major similarities and differences between them in terms of impact and implementation.

1. Introduction

The design of software architectures should focus on two questions: 'How?' and 'Why?' The 'How' part states how the architecture should be implemented and is usually straight forward, it is primarily used during the implementation phase. Less straight forward is the 'Why' part; this involves the design decisions and is primarily used before and after the implementation phase. It contains the reasons why a specific approach was chosen in favor of the alternatives. In the design phase the design decisions and their rationales are mainly used to convince stakeholders that the right decision has been made, by informing them of the alternatives and their consequences.

After the implementation phase, the design decisions are used to deal with changes in the environment, which require the system to be adjusted to regain environmental 'fit'. When a system is changed without the engineer knowing the reasons behind the design decisions, the changes may inadvertently cause undesired side effects. These side effects on their turn require new changes to be made to the system; this is known as architectural erosion. However, when the reason for a decision is exactly known, the decision can be re-evaluated. This way you can evaluate all the consequences and guarantee a sound architecture.

An architecture reflects many major and minor decisions, either to guide or constrain the technical implementation. According to Tyree[3], design decisions are based on the stakeholders' concerns and are architecturally significant if they affect one or more system qualities like performance or security.

Our research will show that currently used methods to describe software architectures have significant shortcomings with respect to documenting design decisions. Architectural knowledge is lost over time, which leads to erosion of the architecture. Solutions found in the literature range from additional documentation to whole new approaches to software architecture. However, every solution has a cost associated with it. We will show how the solutions differ from each other in terms of implementation, cost, possible benefits and ease of introduction.

2.   Problems tracking decisions in traditional approaches

The overall problem with the traditional approaches is caused by the lack of a clear view on why the architecture looks the way it does. Design decisions are embedded in the architecture and not explicitly documented. Knowledge about these decisions is lost over time and this leads to a number of problems, as identified by Jansen and Bosch[4]:

- Design decisions are cross cutting and intertwined

Decisions that affect multiple parts of the design simultaneously, lead to design decision information being fragmented across various parts of the design, making it difficult to find and change decisions.

- Design rules and constraints are violated

During system evolution, previously taken decisions may be violated, which will lead to architectural drift.

- Obsolete design decisions are not removed

The system has a tendency to erode more rapidly when obsolete design decisions are not removed, yet removing decisions takes a lot of effort with no immediate benefits. It may also result in unexpected effects on the system.

These problems result in systems which are expensive to change and individual artifacts that are difficult to reuse, because the design knowledge has vaporized.

3.   Proposed solutions in literature

This section will review four papers from the literature. Not all of the papers provide a complete solution for the problem, but rather discuss different aspects of the problem. Kruchten[1] focuses primarily on defining design decisions, an overview of research is provided. Babar[2] has done research on how architectural information can be captured and Tyree[3] focuses strongly on how to document all the design decisions. Bosch[4] finally proposes a new approach towards software architecture. All these papers are discussed in more detail below.

Solution 1: Kruchten

Kruchten defines 4 types of design decisions:
- Existence decisions ("ontocrises"), these decisions state that some element/artefact will show up, i.e. will exist in the system's design or implementation.
- Bans or non-existence decisions ("anticrises"), this is the opposite of an existence decision, and it states that some element or artifact will not appear in the design and implementation.
- Property decisions ("diacrises"). A property decision states an enduring, overarching trait or quality of the system.
- Executive decisions ("pericrises"), these are the decisions that do not relate to the design or quality but are more driven by for example the business environment.

For a decision Kruchten lists the essential attributes:
- Epitome, a short textual statement of the decision.
- Rationale, a textual justification of the decision.
- Scope, for example time scope, when is the first customer release?
- Author, Time-stamp, History.
- State, decisions evolve in a a manner that may be described by a state machine or statechart.

And the relations between the decisions:
- Constrains
- Forbids (Excludes)
- Enables
- Subsumes
- Conflicts with
- Overrides
- Comprises
- Is an Alternative to
- Is bound to (strong)
- Is related to (weak)
- Dependencies

The paper does not give a clear solution how the design decisions should be captured. There is an overview of actors who may use the architectural knowledge from the repository, but not how the repository is build. Visualizing the decisions and their interconnections is marked to be very important. A tool called Aduna is able to visualize ontologies that describe a domain through a set of classes and their hierarchical relationships.

Solution 2: Babar

This approach uses a three component framework for managing architectural design knowledge. The components are created with information from different sources, like architects, artifacts and patterns. The three components are derived the following way:

- Capturing knowledge underlying decisions from architects

A manager can be appointed to the task of capturing design knowledge, it's also a possibility to let the architects do this themselves, but their knowledge can become a bottleneck.

- Capturing architecture knowledge and associated rationale from patterns

Using a template to document architectural design knowledge extracted from patterns. Because design patterns are reused in different projects, the design knowledge can be reused.

- Characterizing the main architectural constructs and their relationships

The "DAta Model for Software Architecture Knowledge" (DAMSAK) is a customizable model to characterize the data required to capture architecture knowledge and rationale. The model constructs a repository of reusable architectural design knowledge.

Solution 3: Tyree

This solution uses a decision hierarchy for connections between design decisions. It first states that architects should make as few decisions as possible and leave the rest till later in the lifecycle. The only decisions the architect really needs to make are those that identify the system's key structural elements, their relationships and their externally visible properties. Once an important design decision is found, a set of alternatives is developed. A simple comparison of pros and cons is used to assign viability to the alternatives. For listing the essential information provided for the best alternative Tyree uses a table derived from the REMAP and DRL meta-models with two additional fields, Related Principles and Notes.

As the result of a design decision a number of implications can arise, these implications can lead to new design decisions. Repeating this recursively this can deliver a complete decision hierarchy. The decision hierarchy depends highly on the main decision and changes in the main decision will have a ripple effect across the whole hierarchy. That is why an "Architecture Decisions Model" is made to represent the hierarchy. All decisions and their tables and models will be described in a simple separate deliverable.

Solution 4: Bosch

The proposed solution is a whole new approach to software architecture. It aims to define software architectures as a set of architectural design decisions. Aside from solving the problems determined in the previous chapter, it will also help the architect with:

- Guarding the conceptual integrity
- Explicit design space exploration
- Analysis of the architecture
- Improved traceability

The Archium approach
The Archium approach tries to define the relationship between general design decision models and software architectures. Archium is fundamentally different from most other design methods, as it does not promote design for or with change, but rather designing using change.

The Archium meta-model consists of three sub-models:
- Architectural model

Defines software architecture using ADL concepts
- Design decision model

Model design decisions with their rationale
- Composition

Model elements to unite previous sub-models

These models together make Archium capable of describing a software architecture as a set of design decisions.

4.   Similarities and differences between proposed solutions

The authors of the above mentioned solutions all agree that preserving architectural knowledge is of high importance. They differ in the way in which they propose to actually capture the knowledge. Tyree[2] proposes to add an additional document to the architecture that explicitly documents the decisions and the reasons behind them. It does not propose any other alterations to the architecture design process. In terms of implementation this solution is relatively cheap: the method is easy to learn and understand and although it requires some effort to document the decisions, it will not cause any serious delays in the architecture design process.

Kruchten[1] does not propose a complete solution, but rather raises questions as to what information should be contained within an architectural knowledge repository, and how this information should be used. The paper does not reach decisive recommendations, but it is made clear that the ultimate implementation will entail a lot more than a simple documentation of each individual design decision. A lot of effort is invested in visualizing the decisions and their interconnections. It is therefore logical to assume that this will require some fundamental changes to the way software architects work. For that reason it will be more costly to implement than the proposal of Tyree.

Babar[2] focuses on obtaining the actual architectural knowledge whereas the other authors suggest this knowledge is readily available during the design phase. Babar raises questions as to who is responsible for capturing the knowledge and what the used framework should look like. Similar to Kruchten, a complete solution is not proposed, but the main idea is a knowledge repository, not a new approach towards software architecture.

Bosch[4] proposes the most radical change in architectural design. The current ways of software architecture are abandoned and replaced with a method that is centered around architectural decisions themselves. As this requires a radical shift in practices, it is very costly to implement.

5. Conclusion

The authors of the reviewed papers all agree that preserving architectural knowledge is of key importance to prevent architectural erosion. They also agree that this concept is new and emerging practices are still in their infancy. The complete new approach Bosch[4] proposes has to deal with the major disadvantages for a new approach:
- Conservative designers are most likely unwilling to change their approach
- Retraining architects is expensive
- Software engineers need to be schooled to understand the architecture

This means the method should be really convincing. The other methods can be introduced in an easier way because the process itself does not have to change in a fundamental way; it just produces more or larger deliverables. The authors also conclude that not only the main attributes of a design decision have to be documented but also the relations between the individual decisions and the impact a certain change will have on other decisions. Capturing the underlying knowledge of decisions made by architects may also be a good idea; experienced designers can see certain anti-patterns or mistakes in the reasoning of new designers. Using design decision patterns is, like using patterns in general, a good idea but the traps should be avoided.

With architectures in general becoming more and more complex, combined with increasing demands for software reuse, preserving architectural knowledge becomes increasingly important. This development will increasingly improve the trade-off in favor of the approaches that best serve these new demands, even if they are more costly to implement.

References

[1] "Building up and Reasoning about Architectural Knowledge"
- Philippe Kruchten, Patricia Lago, Hans van Vliet

[2] "A Framework for Supporting Architecture Knowledge and Rationale Management"
- M.A. Babar, I. Gorton, B. Kitchenham

[3] "Architecture Decisions: Demystifying Architecture"
- Jeff Tyree and Art Akerman

[4] "Design Decisions: The Bridge between Rationale and Architecture"
Jan S. van der Ven, Anton G. J. Jansen, Jos A. G. Nijhuis, Jan Bosch

A different approach on comparing ADLs
Draft submitted for the SC@RUG 2007 conference

M.R. Fremouw (1526642) and H. Lenting (1526685)

University of Groningen, The Netherlands.

Abstract. Software Architectures are used to organize complex software. There are different methods to describe a software architecture. Such languages are called Architectural Description Languages (ADLs). An ADL is used to describe an architecture in a formal way. There are several ADLs available today, for example: C2, Wright, ACME and Rapide. All these ADLs have their own approach on how to describe a software architecture in a standardized way. This paper gives an answer on two research questions. The first one is how useful are ADLs exactly and the second one is how to select an ADL. . . .

1   Introduction

An architecture represents a critical link between the design of a system and the requirements engineering processes. The goal of the architectural design processes is to setup a basic structural framework. A basic structural framework contains all major components of a system and identifies the communication between these components. An architecture that is explicitly designed and documented has several advantages:

1. Improve communication. Because the architecture is an abstract view of the system it can improve the communication with the stakeholders. Mostly because it is a high-level view and all the small disturbing details are not visible.
2. Analysis in an early stage. When the architects build the architecture, they are forced to analyze the system in an early stage. In an early stage it is clear if the system can meet the requirements such as performance, reliability and maintainability.
3. Improve reuse. A system architecture describes a system in components and their communications. This system architecture is often the same for other systems with the same or almost the same requirements. So it is possible to interchange components between (sub)systems. This can result in large-scale software reuse.

Software architecture researchers make clear that software quality can improve by modeling important architectural aspects. Especially early in the development life cycle. Also many researchers believe that a software architecture must provide with its own body of specifications languages and analysis techniques. These languages provide abstraction that are needed to build complex and large systems, but also give some room to insert special, low detail, properties. In the past ten years many ADLs have been proposed. Looking at different ADLs it is hard to find the characteristics of an ADL. The idea is clear; structure the software architecture through a language but how this should be done is not always obvious.

The two main research questions for this paper are 1) "How useful are ADLs" and 2) "If you want to use one, how to choose between them?". The first question about how useful ADLs are is relative easy to answer however the second question about how to choose is relatively complex to answer. The reason for this is that comparing the available ADLs is hard because there is no uniform way to compare them. In this paper an applicability framework is introduced to compare the different ADLs. With the help of this framework the reader can improve his knowledge on different ADLs and make a better decision which ADL to use for a particular situation.

2 About ADL

While architectural designs are very important for software processes, the architectural designs are not always fully understood by other architects and stakeholders. At least not the way an architecture designer had in mind. Most architecture designs are specified in a non standard way, because of a lack of good design rules and tools.

The choice of an architecture can determine if a system satisfies all requirements or not. Several academia and also the industry proposed an "Architectural Description Language" as a formal way of specifying architectural designs. ADLs commonly provide a conceptual framework and concrete well-defined semantics. Some languages are not primary designed to be an ADL, especially UML. But they turn out to be suitable for describing and also analyzing an architectural design in a formal manner.

3 How useful ADLs are

How useful are ADLs actually? What is the drive for a company to invest in using an ADL? In the past architectures where mostly visualized by box-and-line drawings. While this is quite easily for an architect to accomplish, for third parties like stakeholders it was not as easy to understand the thoughts of the architect. With the help of an ADL the following points are dealt with:

Mutual communication – By using an ADL it is easier for third parties to understand how the architecture created by the architect is defined. This gives the possibility to have more communication between stakeholder and architect. Eventually this leads to a system which fits the needs of the stakeholders in a better way.

Embodiment of early design decisions – With an ADL it is possible to give body to design decisions in a very early stage. This gives the architect also the possibility to early detect possible problems in the architecture, which at that point can more easily be discussed with the stakeholders.

Transferable abstraction of a system – In a large company with several projects, some of these projects will have some similarities. With the help of an ADL it would be possible to share a specific part with another project.

There are different reasons why you should use an ADL. A few of them are mentioned below.

– To obtain the benefits of an explicit architectural focus, an ADL can help with introducing its own specification languages and all kind of different analyze techniques.
– ADLs are useful for defining and analyzing properties of a system early in the development cycle.
– An ADL provides abstraction that is needed for designing a (large) system.
– With the help of an ADL, architects use a more standardized way for communication, for example with stakeholders.

Using an ADL also has some disadvantages. The disadvantages are now summed up.

– Because an ADL is (in most cases) very strict the danger is that the focus lies to much on the rules for designing an architecture. Which results that the architectures attention is drawn away from the requirements.
– Most ADLs are not developed with commercial goals in mind. This results that not all ADLs are suited to be used in a commercial environment.
– The ADLs discussed in this paper all have their own specific optimization towards a particular goal.
– Not all available ADLs are supported by commonly used commercial tools.

The authors believe that an ADL is very useful for visualizing, specifying, constructing and documenting a system. Mostly because box-and-line drawings are not the best way to describe an architecture.

4 Applicability framework

The authors defined a few comparison criteria for the framework to be able to compare each ADL. Below are the chosen framework criteria explained in more detail.

4.1 Semantics

Every ADL has its own way to specify semantics. Two types of semantics are defined, component and connector semantics. Component semantics resemble the definition of modeling components behavior. While connector semantics say something about the interaction between components.

4.2 Scope

The ADLs selected for comparison have different scopes. The scope shows a clear view of what type of architectures an architect can create. For example some ADLs are more suitable for architecting architectures for embedded systems while other ADLs do a better job on creating an architecture for a distributed system.

4.3 Features

The authors have defined a selection of mandatory features. The authors think these features are absolutely necessary to be able to use an ADL in practice. Although the last point does not really look like a feature, it is defined as one. The authors see it as a special addition, which an ADL has or has not.

- An environment to work in.
- Analysis to early detect errors and possible problems.
- Some kind of community to discuss problems.

4.4 Tool support

To be able to really use an ADL for defining an architecture a set of decent tools is needed. This is one of the criteria the authors use for the comparison between ADLs. Obviously not every ADL has the same amount of tools available. The criteria for toolsupport are:

Active specification – A few of the compared ADLs have support for specification of an architecture. There are two types, proactive or reactive. Proactive tools limit the design possibilities on the current design. Reactive tools detect errors and then inform the architect.

View – Does it support a graphical or textual view of an architecture or maybe even both.

Analysis – What type of analysis tools are available, this can be a parser, compiler or a type checker.

Code generation – Some ADLs have tools for generating code from an architecture. Does the ADL have this, and if so which languages are supported.

5 Comparing ADLs

The second main research question is how to choose an ADL. In this paper not all different ADLs are discussed because there are too many. The authors selected a few based on a variety of reasons. The most important reason is to get a good mixture of all kind of ADLs based on the their scope. In this section we compare the chosen ADLs and we will also give a short description of each ADL.

5.1 Comparison

With help of the earlier defined criteria for the applicability framework, it is possible to create a real comparison between the selected ADLs. The results of this comparison are put in table **??**. The idea behind this table is to give the reader an indication on what ADLs are best for a specific project. The table uses the same points for every ADL, this makes the comparison more easy to understand. The points mentioned give a clear view on what is possible with the specific ADL. With the desired architecture in mind it is also more easily to see which ADL has the most similarities.

ADL	Semantics	Scope	Features	Tool support
C2	Support for modeling components, invariants and operation pre- and postconditions. For modeling connectors, has partial by using message filters.	Architecting of distributed, evolveable and dynamic systems.	A lot of tools available. But it is outdated and there already is an successor, xADL.	Has graphical and textual view, both proactive and reactive. Code generation for C/C++, Ada and JAVA. Has parser and type checker for analysis.
Darwin	π-calculus for modeling components, none for modeling connectors.	Architectures of distributed systems.	Has several tools available, but it lacks a community. There is not much information available.	Has graphical, textual and even a hierarchical view. Generation of C++ code. Has parser and compiler.
UML	None. UML does not distinguish components and connectors.	Architectures of a variety of systems.	There are a lot of (open source) tools available. It is not very constrained. But there is a lot of information available about UML and using UML as ADL.	Graphical view. No support for code generation, or analysis.
Rapide	Partially ordered event sets for both modeling components and connectors.	Creating dynamic architectures.	Has several tools and there is a lot of information available on the internet.	Textual and graphical views. Has its own sub language for executable simulation. Has parser and compiler.
Wright	Only through CSP because this is not the focus of Wright.	Architectures of concurrent systems.	There is a tutorial available but the tools are not yet finished.	In this paper the authors tried to give an answer to the two questions: "How useful are ADLs" and if there is need for an ADL "Which ADL fits best".
ACME	Only through other ADLs.	Interchange between architecture.	ACME's main focus is architectural interchange, features of other ADLs can be used.	Supports only a textual view.

Table 1. Comparison between ADLs using the applicability framework.

5.2   Short description each ADL

With the help of the applicability framework it is easy to find the global characteristics of an ADL. This way you can choose every suitable ADL. After this step it is also necessary to know some more background information and that is the reason for a short description for each ADL. These descriptions can be found below.

C2   C2 is based on a component and message-based architectural style. An architectural design contains a set of components and connectors. Each component is connected to one top connector above and one bottom connector below it. This way components can only communicate through connectors. This model is strictly layered, because of this components can only communicate (with the use of a connector) with the layers immediately above or below it.

The strength of the C2 style is a strictly layered approach. Because of this C2 can support substrate independence and component substitutability. For example, applications with a layered nature can be build with an architectural style like C2. C2 is focused on highly distributed systems. C2 is one of the few ADLs which has support for almost every kind of tools needed by system development. A code generator is available, which is able to generate C/C++, Ada and Java code with help of DRADEL. It has a graphical- and text-based view of the development processes. There is also a tool for creating to-do lists and storing design critics. For the analysis part there is a parser, type checker and style rule checker available. One of the advantages of C2 is that there are a lot of tools available. Another advantage is that there is code generation support for multiple languages. The C2 style does not allow communication through shared memory, which might be required by particular high demanding systems.

Darwin   Darwin can be characterized through his simple and elegant grammar, a good concept of components and the introduction of controlled dynamism in the specification of software architectures. The creators of Darwin summarized it: "*The overall objective is to provide a soundly based notation for specifying and construct distributed software architectures.*".

The Darwin style works with components and ports. Ports are used for communication between the different components. They behave like services, it enables other components to interact with the component but also setup communication to other components by itself. A port is associated with a type: the interface of the service it provides or requires. The focus of Darwin is, just like C2, architectures of highly-distributed systems. But Darwin is har more focus on dynamism. Darwin has just like C2 also support for almost all tools needed in the design process of a system. Like many other ADLs Darwin has a parser and compiler for analyzing. Darwin differs from other ADLs with support for "what if" scenarios, the other ADLs discussed in this article do not support this

feature. With Darwin it is only possible to generate C++ code. Another nice feature of Darwin is that it has support for constrained dynamic change of the architecture at run time and compilation. One of the main features which Darwin distinguishes itself from the other ADLs is that it is able to create distributed dynamic systems. Another feature is the operation model, which is described in π-calculus. Darwin is only suitable for highly-distributed architectures. A disadvantage is the current development status. There is not much activity anymore around the development of Darwin, it uses outdated software and it is still in an early development stage.

UML The industry mostly uses UML instead of ADLs. Today many companies are using UML for architectural descriptions. The section "About ADL" give the reader some information about the fact that most ADLs are not primary designed to be a real ADL. UML is a good example of this. Because companies used UML like an ADL, UML evolve towards a real ADL. This resulted that UML has become suitable for describing and analyzing an architecture.

There are architects that say that UML is an ADL but there are also architecture that say that UML is a modeling language and not a process or method. The authors believe that UML is a sort of ADL but UML is less expressive than other ADLs. For instance, UML does not distinguish components and connectors. Beside this UML had no built-in notation and does not have a built-in notion of architecture style constraints. Because of this it is better to combine UML with an other ADL to get more strictness in the way an architecture describe a architecture. When we look to UML alone it is clear that UML can be used for a large range of software systems. Besides this UML is also supported by many tools.

Rapide The Rapide architecture works with modules and connections. Connections are used to give information about the flow of events between the different modules (or components). The Rapide architecture style can be described as an event-based architecture description language. Each module has an interface and the definition of these module interfaces is part the of Rapide language. The architectural definition language is used to build such modules. The behavioral specification of interfaces is handled by the constraints language.

The focus of Rapide is modeling and simulation of the dynamic behavior described by an architecture. Rapide contains a few build-in tools. There is a tool available that can give background information of execution behavior by animating simulations. This information is published in a graphical- and textual view available. Of course there is a parser and compiler available, it is also possible to compile sub languages. Dynamic changes in the architecture, just as with Darwin, at run time and compilation level is possible. Rapide has several distinguishing features like event-based architectures specified using partially-ordered sets of events. It also has all kinds of simulation tools to check interactions of

event-based components. There is no support for active specification in Rapide. Active specification can limit design decisions (proactive) or detect errors (reactive).

Wright With the Wright language it is possible to organize the structural and behavior aspects of software systems. Because of the precise semantics for its notation this solution offers analysis and verification techniques of the properties. Wright works with basic abstractions of components, ports and connectors. The notation of Wright has many similarities with communicating sequential processing (CSP).

Wright is mostly focused on modeling and analysis of the dynamic behavior of concurrent systems. Analysis has its primary focus on deadlock analysis. Unlike all other discussed ADLs Wright does not have support for a graphical view of the system, only a textual view is available. It does have a parser, model checker for conformance of ports and roles and analysis of connectors for deadlocks. The most important feature of Wright is that connectors are explicit with checkable formal semantics. For Wright not a lot of tools are available, while other ADLs have tools for almost all design cases, Wright has no active specification, code generation or dynamism support.

ACME ACME is a type of ADL but ACME has more goals then developers can expect from a more conventional ADL. ACME is developed as a compromise by the software architecture research community. The idea behind this ADL is to give a more common representation for a wide variety of architectural tools. The people behind ACME believe that here is sufficient commonality in the capabilities of ADLs to share ADL independent information.

ACME has its focus on architectural interchange, especially at the structural level. ACME does currently not have as many tools available as other ADLs, but there are still new tools created for ACME. Currently it features an Eclipse plug-in to create the system. Besides the plug-in there is also a parser and some performance analysis tool available. The main feature of ACME is the ability to share architectural descriptions. It does this with help of a structural framework for characterizing architectures. One of the disadvantages of ACME is the lack of supporting tools. All other ADLs have some kind of support for dynamic configuration changes, ACME does not support this.

6 Conclusion

In this paper the authors tried to give an answer to the two questions: "How useful are ADLs" and if there is need for an ADL "Which ADL fits best".

This paper should make clear to all readers that an ADL is very useful. As stated earlier ADLs make architectures, especially for third parties, more readable and gives the possibility for reusing earlier designs. There is no reason for an architect to still use the old box-and-line-drawing style.

It is hard to give a clear answer to the question: "Which ADL fits best", because it is not easy to get an overview of the possibilities and domain of every ADL. However with help of the comparison table it has become a lot easier to make the choice for which ADL is best to use. The table gives a clear answer to what ADL is best for what specific domain and what tools are available. After making a rough selection with help of the table, a more precise selection can be made by reading the detailed sections of the selected ADLs.

ADLs have been around for a while now and are becoming more and more popular by architects. It is safe to say that ADLs are becoming a mandatory tool for every architect.

Software Architecture Description Options: UML or ADLs

Ahmad Waqas Kamal, Callo Trosky

Department of Mathematics and Computer Science
University of Groningen, Postbox 69042, The Netherlands
a.w.kamal@rug.nl, trosky.callo@esi.nl

Abstract. Research and development in software engineering involves modeling, analysis and description of software architectures. In contrast to modeling and analysis, there is not a defined consensus about how and what is done in software architecture description. Two main approaches are ongoing options for architectural description. One is the set of architectural description languages (ADLs), where each of them is mainly designed for specific architectural domain and specialization. The second stream is the Unified Modeling Language (UML), with a standard set of predefined constructs for multiple modeling purposes including some for software domain. In this paper we review the new relevant features for software architectural description in UML 2.0, and update a previous approach to describe software architecture with UML 1.5. Finally, we compare UML against main features of ADLs in order to measure the current maturity of UML for architectural description.

1 Introduction

A sound architecture design is crucial to the success of a software system. Current research trends are focused towards a coherent approach for architecture based development understandable by different stakeholders [4]. However, the practice of software architecture design is still ad hoc, informal, isolated and poorly understood by stakeholders. It has become a challenging task to analyze software architecture for consistency, completeness, evolution and extensibility [3]. Architecture description languages are used as a solution to such problems. Numerous architectural description languages like Aesop, Artek, C2, Darwin, Lileana, Metah, Rapide, SADL, Unicon, Weaves and Wright provide architecture solutions for designing, describing and implementing software projects [4]. These architecture description languages mostly work in isolation and fulfill domain specific needs.

UML and ACME are distinctive for their presentation as general standardized languages. However, UML bears the responsibility to cover a wider domain of industries which makes it a weak option for software architecture community. UML gets the advantage of being a famous notational language understood by wider community, but it is a semiformal language that does not cope with detailed granularity and variability

of software architecture design. Medvidovic et. al. has studied number of existing architecture description languages and uses this knowledge to test the power of UML on modeling software architecture [1]. His work is focused on UML 1.5, though UML 2.0 has introduced new features, still his work carries a significant contribution as evaluated in [1][2]. UML has the advantage of close association with development languages but it provides poor support for detail analysis at architecture level. Other description languages provide reasonably good support for variability but they differ extensively in their semantic and design notations. ACME is used as a language to provide a common platform but in itself it does not provide full feature support to architecture description languages. Thus industry still lacks the presence of a widely accepted architecture description language understandable by a wider community of software architects.

Typically each architecture language takes in to consideration components, connectors and their configurations. With specific ADLs, these components and connectors demand use of certain rules, selected style specification and topological constraints that when designed needs to be taken care of. Similarly ADLs itself inhibit modeling notations or facilitate incorporation of new notations for components and connectors to design software architectures. However, few of these architecture description languages inherently support style based architecture design.

In this paper both UML and ACME are not differentiated from other architecture description languages. By considering them as architecture description languages we compare their features with other architecture description languages. Further, the current specialization of UML 2.0 to be used as an ADL is focused. Variability support provided by description languages, modeling issues and tool support of different ADLs is also discussed and limited scope, restrictions and implementation issues of these languages are highlighted. Another motivation of this work is to highlight features offered by UML 2.0 in context with existing description languages.

2   Architecture Description Languages

Architecture description languages are used to model, configure and provide tool support for architecture based development [4]. ADLs have become an intense area of research in software engineering community. Numerous description languages have been proposed or are under improvement to coup with demanding needs of software development. Many of these description languages are specific to their domain but some are designed for general purpose use as well. However, there is no strictly defined consensus in research community about what exactly an ADL is, what aspects should necessarily be modeled in ADL and what should be interchanged in interchange language [4]. These description languages contain appealing features on individual level such that if they were combined much better results could be produced. For example, transfer from one view in a description language to another one in a

different description language might be more beneficial. But in the absence of a common, formalized notation this goal is far from success.

Components and connectors are used as architecture building blocks. Components encompass functionality while connectors work as glue between components and work to transfer data and control. When used within the boundaries of ADLs they need to be designed and formulated to fulfill the specific needs of selected ADL. Most of the ADLs provide built-in features that can be used for effective architecture design. These components and connectors need to be modeled explicitly to the selected architecture description language to build and maintain their topology. An example is, a component can be a client, server, or filter and a connector can be a pipe or data passage source.

Currently in industry, adopting a specific ADL tool require a lot of investment in training, installation and understanding the semantics of the tool [3], which makes it harder for the engineers and researchers to share knowledge among different tools. One way to solve this problem is to use architecture interchange language like ACME or a generalized language such as UML that can be understandable by a wider community. But these description languages themselves have certain barriers such as lack of common vocabulary, lack of support for capturing and exploiting architectural concerns and modeling issues [1] [3] [4].

3   UML 2.0 as an ADL

Practitioners and researchers have been exploring the power of UML for architectural description since early versions of UML. This initiative has been mainly supported by the fact that UML has been largely accepted within the software industry, especially for large and complex applications. The needs for improvement in design time, implementation, componentization and source code reutilization are aspect within the software development lifecycle that have been covered by UML, and supported by the set of method and tools implemented around it [9], [10], [13]. This large acceptance of UML and the availability of tools and methods make UML appear to be as an option for architectural description of software systems.

Several approaches have formally presented and described how UML can be used for describing and documenting software architectures [1], [6], [11]. Although, these are different approach with different perspectives, they commonly showed that the standard set of constructs and semiformal defined notations of early UML specifications could not explicitly capture architectural concepts like components and connectors as specialized ADLs. Although the set of notations in UML 1.x includes constructs for modeling software components (their interface and deployment) these are considered as executable entities. Executable entities are not able to decompose systems as conceptual artifacts decompose the system states and behavior. Specifically, UML com-

ponents structure is not able to satisfy rules or constraints for architectural styles, neither provides specific or primitive artifacts to model architectural connectors.

We are interested in observe UML features that can improve the direct usage of pure UML for modeling software architecture, and let UML be able to explicitly model architectural elements with the less effort possible as specialized ADLs do. Previous approaches [1] and [11] describe software architecture with early specifications of UML. Rather than the approach in [11] that is specialized for Real-Time systems, the first strategy from the approach by Medvidovic in [1] is more generic and suitable to be reviewed and observe the current capability of UML as an ADL. This strategy, Use UML "as is" is an immediate step to take UML for architecture description. Although this strategy was not conceived for UML 2.0, its pure UML usage to represent architectural components and connectors still represents a valid approach for UML 2.0. The study case presented for this strategy is a system with a C2 architectural style. Although the strategy uses pure UML classes, it is necessary two different set of classes to model interfaces and connectors, plus another set of classes for components. This situation reflects the limitations of UML 1.x to represent components and connectors, and its weak semantic for interfaces, or the aspect that UML classes were not able to represent or content interfaces and connectors.

The effort in the UML 2.0 specification also reflects the continuously growing and adoption of UML for software systems design. New aspects and improvements within UML 2.0 come too from the need to use UML for architecting systems and capture architectural concepts like components and connectors. Interfaces and Ports are new improvements in UML 2.0 for the benefit of architectural description. An approach reported in [6] shows how to represent architectural components using classes too, but with UML 2.0. The main difference with the first strategy in [1] is interfaces representation. The approach in [6] exploits new concepts of UML 2.0. Two main aspects are relevant as main featured in pro of UML for architectural description. The first are interfaces, these can be associated to class elements using ports. Two kinds of interfaces are available in UML 2.0. *Provided interfaces* represent the services that a class implements. And *required interfaces* are the services that other classes must provide for the class to properly operate in a specific environment. The second are ports, Interfaces and classes are associated by means of ports. A port provide a particular interaction point for the container class and its environment, by means of classifying the kind of signals that can be send or receive through its interfaces. These interface classification and port implementation provide more expression power or clear semantic to UML class artifact for component and connectors representation.

With the new features of UML 2.0 for ports and interfaces, we revise the first strategy in [1], and try to observe how UML 2.0 has improved for components and connectors representation for this strategy. As a reference, Figure 1, shows the original set of class diagrams that describe the C2 architecture for the Meeting Scheduler Application in [1]. Three set of class diagrams describe components, interfaces, and connectors. Our first assessment with these diagrams is that they show inheritance, relation-

ships, and associations between class elements, but besides the stereotype for interfaces, there is not explicit distinction of architectural elements and interaction.
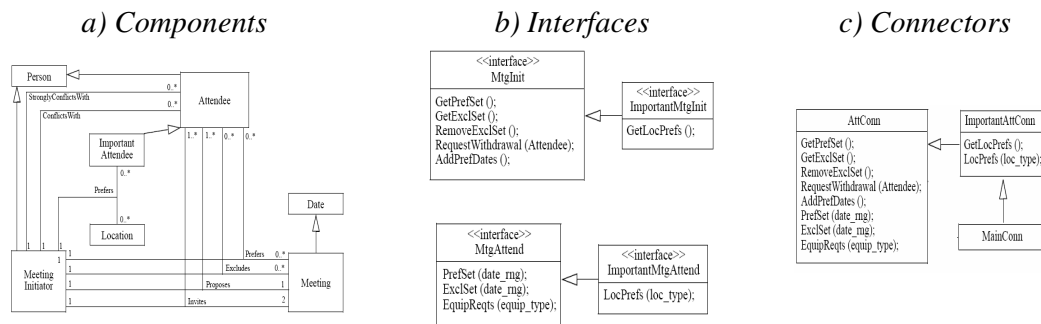

Fig. 1. UML 1.5 class diagrams for the meeting scheduler application (take from [1])

Interfaces in UML 2.0 are already primitive artifacts, and it is not necessary stereotyped classes for its representation. The port at which an interface is attached determines the interfaces implementation or behavior. Thus, rather than only stereotyping classes as interfaces to represent component interfaces, we model new component representation in Figure 2b and 2c, using UML 2.0 interfaces and ports according to the original C2 ADL textual specification in Figure 2a from [1].
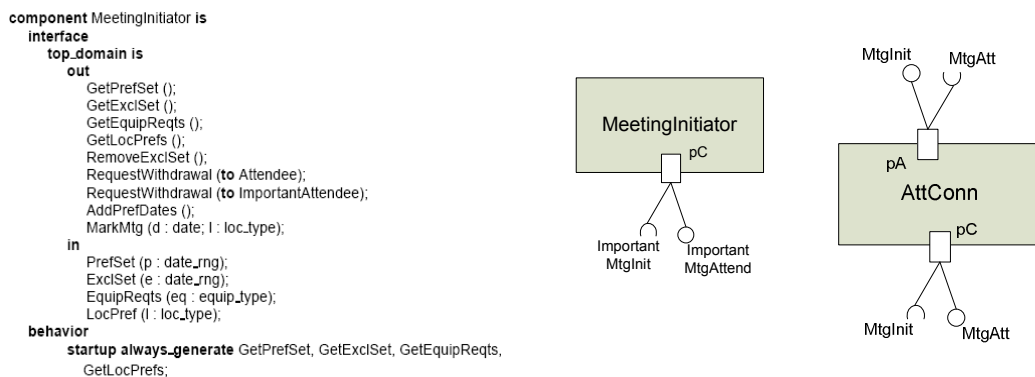
Fig. 2. From textual ADL representation to UML 2.0 representation

Figure 2b shows the MeetingInitiator component with the attached interfaces. These interfaces represent those in Figure 1b, with the same methods and attributes. Figure 2b shows that the ImportantMtgAttend interface is a provided interface, and its methods are provided by the MeetingInitiator component. The gain value with this representation is that the provided interface methods list matches exactly to the methods under the "In" clause of the textual ADL representation. Similar situation results for

the required interface ImportantMtgInit, where all its methods belong to the "Out" clause.

A more complex representation is in Figure 2c, shows how UML ports define interaction points and behavior. The AttConn component shows two different ports with similar interfaces, but with opposite functionality. At port pA, MtgInit is a provided interface; in contrast at pC MtgInit is a required interface. This is due to pA is intended to be connected to an Attendee component, and pC to the MainConn component, both with complementary functionalities. This initial approach shows that UML 2.0 has not only increased the capabilities of UML to represent architectural components and connectors, but also can capture some ADL semantics, in this case C2.

4   ADLs Feature Comparison

In current paradigm both UML and ADLs work in isolation. ACME [3] is an effort to bring different architecture description languages closer to each other. However this effort still lacks a standardized tool like UML that can be understandable by main community of software architects. To make a brief analysis of architecture description languages we provide a widely accepted definition for architecture description languages. *"ADLs provide features for modeling software system's conceptual architecture, distinguished from system's implementation. ADLs provide both a concrete syntax, and a conceptual framework for characterizing architectures"* [3]. Many architecture description languages support component and connector specific rules that make such description languages a specialized domain specific tool. We define high level behavior of these components and connectors as semantics of the associated description language. Issues faced in modeling these components and connectors in different description languages are also addressed.

4.1 Semantics

It is important that architecture description language support variability in a way that it is able to handle large scale software development [4]. All architecture description languages support specification of component semantics to a varying degree like component property lists (UniCon), models of dynamic component behavior(Rapide and Wright), invariants and operation pre and post conditions (C2) and models of interaction and composition properties (Darwin) [4]. Connectors specified in UML have fixed interfaces for components which do not fully facilitate generic connections. C2 supports generic connectors that can accommodate any type of C2 components [1]. To fulfill this gap UML uses extension mechanism to solve the problem. This representation of semantics is exhibited in different ways in different languages. Languages that implement connectors as first class objects like Rapide model connectors as semantics while languages that model connectors explicitly like ACME do not always provide means for defining semantics [4]. One greater advantage of UML 2.0 is to integrate activities with their actions which allow executable models and expressing actions

with procedural semantics [6]. As UML specifies no notation for its actions, so tool vendors specify their own syntax which in itself is considered as a barrier to generalize the language. Thus it provides a flexible description of semantics that can be tightened in one area or specified in some other [6].

Further the behavior of components and connectors is largely dependent on the use of associated styles like blackboard, layered, client-server etc. and properties associated with components and connectors is defined by selected ADL. Like properties of roles, ports, binding issues etc. Such behaviors include adding or removing ports, associating roles to ports and adding new components and connectors.

4.2 Modeling Issues

Languages that are flexible to model components and connectors face less modeling issues while languages that are less flexible provide more resistance to generalized architecture design. Basic ideology of ADLs and UML is to support model based views of the system that can facilitate stakeholders to assess requirements of the system. Current trend in modern programming languages is to have interfaces as a means to describe services available of specific elements [6]. We can also list the order in which services offered by these components are invoked by using different views [6]. However, languages like UML in itself can not be used as a replica of programming languages. Further improvement in UML 2.0 is its use as an executable model which provides benefit that the model becomes independent of the platform and target language [6]. Still UML faces challenge to express designs at the level of granularity that can compete with isolated ADLs. In particular UML lacks in through support for modeling and exploiting architectural styles, explicit software connectors and local and global architectural constraints [1]. Nenand [1] and Garlan [3] propose that benefits of standardization should work in a way that it does not lose the power afforded by specialized notations. Wright provides a strong support to model flexible connectors. In fact the concept of provided and required interfaces introduced in UML 2.0 and ACME is used in Wright with the names of 'providing' and 'using' services. However, Wright specification provides much rich facilities to define behavior of these services as it includes a description of sequencing and choice [12]. Despite the strong representation of components and connectors; Wright is still a weak paradigm for the use of styles. It doesn't provide guidance rules to use styles effectively as does Aesop.

On one side UML bears the responsibility to address needs of a large user base but on the other side each ADL work in isolation. Although the work of ACME [3] is a significant contribution towards generalization of architecture design still every concept of every ADL can not be combined at one place. Languages like C2 and Wright provide more flexible ways to design components and connectors and provide associated toolset support but their function is still considered as domain specific in comparison to ACME and UML.

4.3 Tool Support

The variety of ADLs has carried too the variety in toolset for architectural design. Each ADL provides concrete syntax for characterizing software architecture, thus every ADL include particular graphical tools for visualizing and manipulating architectural structures. Although there is a great benefit for architecture design field from ADL's effort in exploring and developing toolset, it doesn't contribute to their interoperability, share architectural descriptions, and it is recognized by the community that there is a redundancy and waste of resources in developing toolsets for ADLs [3]. An extensive description and comparison of available toolset for ADLs is presented in [4] point out that currently toolset are mainly advocate to facilitate manipulation and visualization of architecture design, ant the relation ADL and Toolset is almost one to one.

A small subset of tools is oriented to consistency and analysis. Some like Darwin, Rapide and UniCon provide powerful architecture modeling environments. Only C2 and Darwin provide tools for all manipulation, visualization, analysis and design. Recently development of the AcmeStudio[5] tool features has not been included in [4]. The AcmeStudio tool is an architectural design environment that supports the ACME ADL. In contrast to other revised tools, AcmeStudio allow us to draw architectures in different styles, as well as manipulate and analyze those designs. Although the set of ADLs and the architectural styles supported by ACME is wider than the rest of toolsets, the modeling views and traceability in the development cycle are limited to the specific domain.

 In contrast to specialized ADLs, the available toolset for UML is way wider, for instance UML toolsets range from high expensive development suits, to mature open source options [10, 13]. And choosing the right UML modeling tool is not an immediate decision, it has been also matter of discussion according to the needs particular features of the organization [9]. With the current extension of UML 2.0, most of these tools have also improve their features for architecture modeling, and spam it along their previous traceability in the supported development cycle.

5   Related Work

Modeling Software Architectures in the Unified Modeling Language [1] presents two strategies for supporting architectural concerns within UML. We have looked at the first strategy with current UML 2.0 as is described in section UML as an ADL. The second strategy incorporates useful features of existing ADL as UML extensions. They conclude by the time of their work, that UML lacks support for capturing and exploiting certain architectural concerns, in particular support for modeling and exploiting architectural styles and software connectors. A more recent publication in using UML 2.0 for architecting systems is [6]. They present how UML improvements can significantly increase development's efficiency. UML 2.0 improvements high-

lighted by [6] are those for modeling structure, system decomposition, and system behavior modeling.

The diversity and little consensus about ADLs are extensively covered in [4]. Their contribution is a framework to classify and compare existing ADLs. Relevant aspects for assessment include what aspects of architecture should be modeled in an ADL, which ADLs are best suited for particular domain problem, etc. The framework improves the process to identify key properties of ADLs. The comparison highlights areas where existing ADLs provide extensive support and those in which they are deficient, suggesting a research agenda for the future.

This paper review is mainly based on related papers that cover ADLs [4] [3] and UML [6] [1] for architecture modeling description. In the work by Nenand [1] authors explore strategies to model software architecture with UML. This strategy is oriented in constrain the UML Meta model using UML's built-in mechanisms, such as stereotyping and Object Constraint Language OCL [7]. But overall our work is more close to Nenand [4] who uses a framework to compare different architecture description languages. However, his work does not address the features of UML. We have worked to put an edge to his work by discussing latest features offered by UML 2.0 in comparison to other description languages.

6   Conclusion

Although the current UML 2.0 doesn't provide first-class artifacts for architectural components and connectors, there is a good improvement to exploit and model architectural styles. The new specification about required and provided interfaces for classes can capture and exploit significantly to describe services. On the other side ACME provides a common platform for isolated architecture description languages to share architectural descriptions. A comparison of these description languages and UML can lead to a consensus towards a common standardized language to design software architecture.
We conclude that specialization is a common aspect among software architecture design with ADLs and UML. At the first place, ADLs still keep the domain specific orientation. Although, the effort with ACME aims to improve the interoperability of specialized ADLs, this does not cover generalization of ADLs. ADLs may continue keeping and improving the domain specific, but ACME may help to increase the number of common practitioners among specific domains. On the second place, it is the UML specialization represented by the set of improvements in UML 2.0 for software architecture and design. This UML specialization is not domain oriented as in ADLs, but the specialized subset of UML constructs and artifacts for software architecture and design, already contribute to the UML specialization.

7   References

[1] Nenad Medvidovic, David S. Rosenblum, David F. Redmiles and Jason E. Robins, 'Modeling Software Architectures in the Unified Modeling Language', ACM Transactions on Software Engineering, Volume 11, January 2002

[2] Jacobson I, Booch, G., and Rumbaugh, J. 1999. The Unified Software Development Process, Addison-Wesley,Reading,MA.

[3] David Garlan, Robet Mornroe and David Wile, 'ACME: An Architecture Description Interchange Language', Computer Science Department, Carnegie Mellon

[4] Nenand Medvidovic and Richard N. Taylor, 'A Classification and Comparison Framework for Software Architecture Description Languages', IEEE Transactions on Software Engineering, Volume 26, January 2000

[5]   Acme   Lab: Getting   to   know   AcmeStudio, http://www.cs.cmu.edu/~acme/AcmeStudio/index.html

[6] Morgan Bjorkander and Cris Kobryn, 'Architecting Systems with UML 2.0', IEEE computer society, Copyright 2003

[7] The Object Management Group, http://www.omg.org/

[8] Unified Modeling Language: Superstructure version 2.0, http://www.omg.org/docs/ad/03-04-01.pdf

[9]   Choosing   a   UML   Modeling   Tool, http://www.objectsbydesign.com/tools/modeling_tools.html

[10] Architecture and Design: Unified Modeling Language (UML), http://www.cetus-links.org/oo_uml.html

[11] Selic, B. & Rumbaugh, J. "Using UML for Modeling Complex Real-Time Systems," http://www-106.ibm.com/developerworks/rational/library/content/03July/1000/1155/1155_umlmodeling.pdf (March 1988)

[12] Robert Allen and David Garlan, 'A Formal Basis for Architectural Connection', ACM Transactions on Software Engineering and Methodology, Vol. 6, No. 3, July 1997

[13] Unified Modeling Language (UML) Tools, http://www.jeckle.de/umltools.htm

Multi-dimensional Transfer Function Design

Automation for Volume Rendering

Cherian Mathew, João Mimoso

Abstract - Transfer function design plays a crucial role in direct volume rendering . It is an integral part of the classification step, wherein it provides a mapping between the voxel value and the color/opacity, which is eventually used to render the volume data. This survey paper attempts to compare existing methods of designing transfer functions. The focus of the paper is on studying multi-dimensional adaptive methods, which provide for more effective and intuitive transfer functions. Two techniques are chosen for comparison, the first of which generates multi-dimensional transfer functions by adding spatial information to the histogram of a volume. This information can then be used to classify the histogram and derive a transfer function by assigning unique colors to each class of the histogram. The second technique is an iterative adaptive process, which couples machine learning and a painting metaphor to allow more sophisticated classification in an intuitive manner. The aim of the paper is to provide a comparative evaluation of the techniques mentioned, while mentioning the conditions which allow a certain technique to be superior to the rest. A new design method as a combination of the two is also proposed.

1 Introduction

Direct volume rendering techniques are a powerful and flexible method of visualization. These techniques are widely used for displaying medical volume data generated by computed tomography or magnetic resonance imaging.

One visualization method is the maximum intensity projection which consists of projecting in the visualization plane the voxels with maximum intensity that fall in the way of parallel rays traced from the viewpoint to the plane of projection. This implies that two MIP renderings from opposite viewpoints are symmetrical images. Recent visualization methods use opacity and color to render volume. Color mappings are used to map data values to meaningful colors. The opacity mapping is used to expose the part of the volume most interesting to the user and to make transparent the uninteresting parts.

Visualization of volume data derives from two key steps: classification and rendering. Classification is achieved by defining a mapping between data values and their corresponding colors and opacities. This mapping is impemented using transfer functions. If we think of a CT scan, a simple transfer function to map density values would assign one color to the bones and another to the skin, normally trying to make the colors look like reality, for a more intuitive visualization. Then the bones would have high values of opacity, leaving the skin more transparent.

The need of a suitable transfer function to efficiently explore the capabilities of these techniques has driven many investigators to propose new and automated methods for its design. Although the 2D slice-by-slice traditional viewing is still the standard visualization method in clinical practice, as the size of the datasets increase, the 3D visualization methods become a needed resource. Since the correct visualization of the objects is the most important thing, the acceptance of this technique is connected to its ability of representing and isolating specific features from the dataset. One common problem with direct volume rendering is that some of those features may occlude each

other. To overcome that problem, a suitable transfer function must be designed. Manually designing the transfer function is a hard long and discouraging process. There has been several approaches to overcame that problem, one is the use of several preset transfer functions in witch the user picks the best one within the set of available functions. The downside of this method is that if there is no suitable preset, the transfer function as to be tuned manually. Different methods make use of the opacity function to automatically make uninteresting regions more transparent than the ones with high information content.

The main problem is conceiving a good automated transfer function design to facilitate data exploration and make 3D visualization more appealing.

2 Traditional Transfer Function Design Drawbacks

Traditional techniques of transfer function design have been found to have limited effectiveness in performing the actual classification of data. The main case against these techniques is the requirement for the user to work in the derived transfer space, which, in most cases, can be unintuitive.

2.1 One-dimensional Transfer Functions

A fundamental drawback of the traditional 1-D function based methods is the fact that the mapping from the domain of the volume to the domain of the transfer function is injective. Therefore the separation of features with the same statistical properties is impossible. For example, tissue can be separated from bone in CT scans, but a single bone cannot be separated from other bones.

2.2 Two-dimensional Transfer Functions

An improvement on the 1-D form has been the 2-D transfer functions which depend on the scalar value and the gradient magnitude. Due to the inclusion of the gradient magnitude, material boundaries can be visualized with 2-D opacity maps. The material boundaries correspond to arcs in the two dimensional histogram or scatter plot. In practice, however, these arcs could be more or less pronounced and not well defined. Such ambiguous arcs are particularly seen in the case of CT scans with reconstruction artifacts and density distribution of the scanned object. In the case of MRI scans, as the scanner cannot distinguish between air and bone, the resulting histogram usually shows only one large unpronounced arc between the footprints of water and non-water bearing material. This implies that the possiblity to distinguish a feature in the volume domain depends on how separable its statistical footprint is in transfer function space. It may be possible to overcome this problem by applying segmentation algorithms in the volume domain, but this solution has been found to have slow processing times for the purpose of user interaction.

3 Multi-Dimensional Transfer Functions

The flexibility of transfer functions has been improved by the introduction of multi-dimensional transfer functions. Higher-dimensional classification can lead to better results since it uses more properties for each voxel, such as gradient information, its location, and local texture. Here we describe two design methods to generate such kind of transfer functions . The methods are intuitive , adaptive and allow for user interaction to dynamically improve the performance of the method. The first method uses spatial information of the input data along with the scatter plot to separate as many features as possible. The second method employs a machine learning process by accepting user-painted volume regions as training data to "learn" a transfer function.

3.1 Spatialized Transfer Functions

Roettger et al[1], have considered spatial information to be important, because a feature by definition is a spatially connected region in the volume domain with a unique position and certain statistical properties. This could be done simply by including the volume coordinates with the scalar value and gradient magnitude to generate a 5D transfer function, but such a system would be difficult to set up. Instead, a derived form of the standard 2D transfer function is generated, where the volume coordinates correspond to the RGB color channels of the function. As shown in *Figure 1*, this method uses spatial information to derive the color, while the statistical information from the histogram is used to derive the opacity. The following section describes the process in detail.

Description

The first computation is that of opacity. Consider the global opacity as c_α where $c_\alpha > 0$. The opacity of the entry $F(s,t)$ of the transfer function with s $\in$ [0, 1] being the normalized scalar value and t $\in$ [0, 1] being the normalized gradient magnitude is $F_\alpha(s,t) = tc_\alpha$. This technique of gradient weighted rendering , returns high opacity values for material boundary points. The next phase of the process attempts to find a mapping which transforms unique features in space to unique colors in the transfer function. This mapping assumes that the same color be assigned to data points which map to almost the same position in the volume domain, which holds true in most cases. The pre-requisite information includes :

- $p_i(s,t), i = 1...n$ representing the normalized positions of the n contributing voxels of the entry $H(s,t) = n$ of the histogram,

- $b(s,t) = \frac{1}{n}\Sigma_{i=1}^{n} p_i$ the barycenter of the voxels,

- $v(s,t) = \frac{1}{n}\Sigma_{i=1}^{n}\|p_i(s,t) - b_i(s,t)\|$ the spatial variance of the voxels and

- r the maximum radius of the features to be detected.

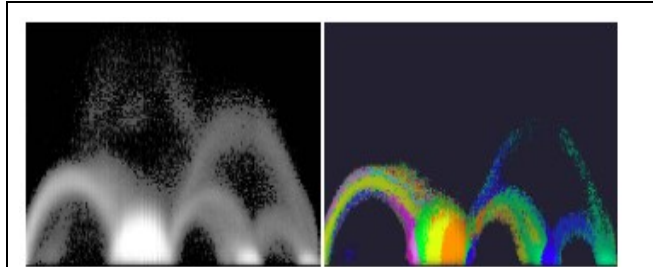The barycenters and the variance are computed once for each volume in a preprocessing step.

Figure 1: 2-D Histograms based on scalar values on the x-axis. The y-axis represents the gradient for the left histogram and the spatialized transfer function (r = 0.1) for the right histogram.

Assuming that the variance $v(T)$, where the tuple $T = (s,t)$, is sufficiently small, then for a given reference tuple T_0 and radius r, all tuples for which $\|b(T) - b(T_0)\| < r$ belong to the same feature. As the variance increases (especially when $v(T) \geq r$), the separation of features becomes more difficult. To overcome this problem it is required to choose a reliable measure of spatial correspondence i.e. whether or not a tuple T belongs to the same feature as

the reference tuple T_0. The distance norm given by,

$$N(T, T_0) = \|b(T) - b(T_0)\| + |v(T) - v(T_0)|$$

is selected as the measure. Consequently, each group entry T is given the emmision,

$$F_\lambda = CRGB\, c_\lambda \text{ for all } T : N(T, T_0) < r$$

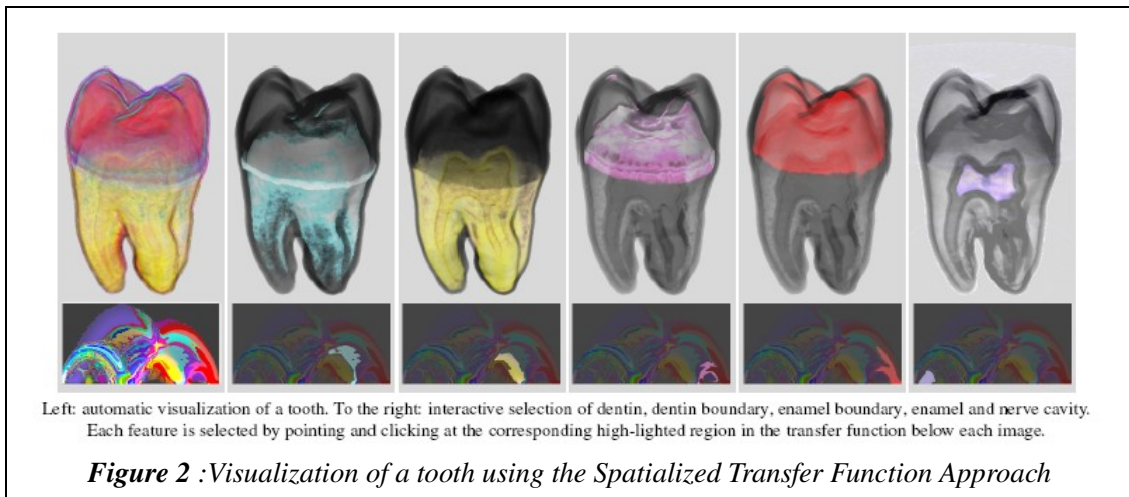where c_λ is a constant which defines the global emission and $CRGB$ is a RGB color triple with random hue value. The initial reference tuple T_0 is the tuple which has the highest histogram count. The process is then iterated for all un-assigned tuples until every tuple is assigned to a

specific feature group. The only parameter to be choosen manually is the maximum feature radius r (and once for each dataset the global opacity c_α and the global emission c_λ). For this method a radius between 0.05 and 0.25 seems to provide good results.

Once the automatic setup is complete , the user can click on the histogram (*Figure 2*) to select a specific class and the corresponding feature is displayed by simply setting the emmission of the remaining classes to zero.

The separation of features can be furthur enhanced by applying techniques like pseudo-shading and reduction of noise.

The primary advantage of this transfer function design method is that it is intuitive, natural and provides a flexible interface which allows for user-defined separation of features. This makes it ideal for applications in volume domain segmentation, aneurysm visualization, improved MRI representation and many others.


Left: automatic visualization of a tooth. To the right: interactive selection of dentin, dentin boundary, enamel boundary, enamel and nerve cavity. Each feature is selected by pointing and clicking at the corresponding high-lighted region in the transfer function below each image.

***Figure 2** :Visualization of a tooth using the Spatialized Transfer Function Approach*

3.2 Intelligent System Transfer Functions

Another approach to the problem, presented by *Tzeng et al [2]*, is the development of an intelligent system to generate transfer functions by relying on user interaction to separate regions of interest from the rest . The separation is performed by an interface which allows the user to classify features by applying one color of paint to parts of the volume they are interested in and another color to regions they want to exclude. During this interaction the system uses machine learning to build transfer functions based on the painted regions. The implementation of this approach can be done using a number of classification-based training models like neural networks, support vector machines, Bayesian networks or hidden Markov models.

Description

Figure 3 below describes the visualization process. The painting interface provides direct access to the volume data, allowing the user to indicate regions that are part of the material class. These regions constitute sample points, which act as training data for the machine learning process.

***Figure 3** :Intelligent System Process*
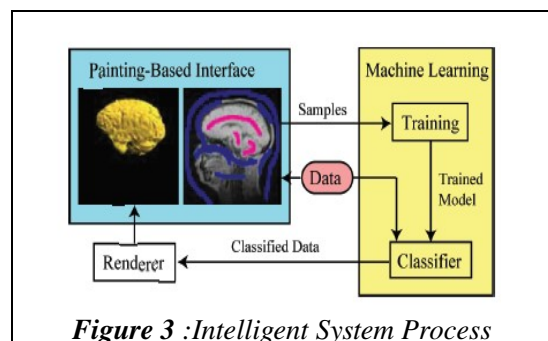
When using a machine learning classifier there are two steps invloved. First, the classifier is trained with input data which include the scalar value (or for color data, the R,G,B value) of the voxel, gradient magnitude, position of the voxel, neighbouring voxel values and an output value which indicates membership into a material class.

Training is an iterative process wherein the intelligent system provides real-time visual feedback which can be used to furthur revise the classification.

After training, a classifier is generated, to provide an output value between zero and one, which indicates the likelihood that the voxel belongs to a particular material class. This data is then used to classify the selected regions and perform volume rendering to display intermediate results.

The painting-based intelligent user interface (seen in *Figure 4*) is a set of familiar painting tools along with x,y and z-aligned slicers of the volume for the user to view and paint on the volume. At each iterative step, the user isolates specific features by applying one color for the interested region and another for the regions that are not part of the material class. The number of painted slices required for a desirable result depends on the data set and the type of information used by the training model.
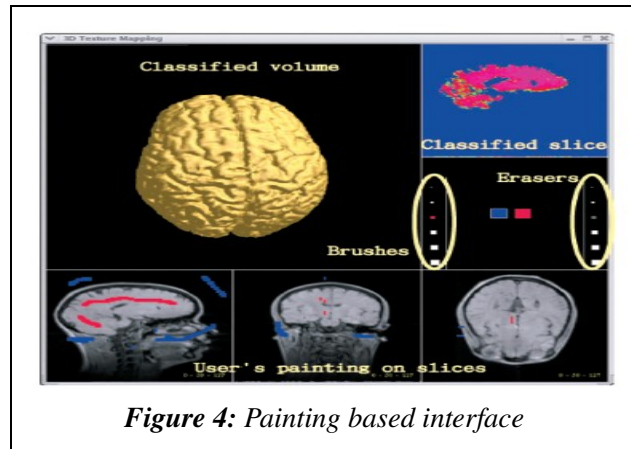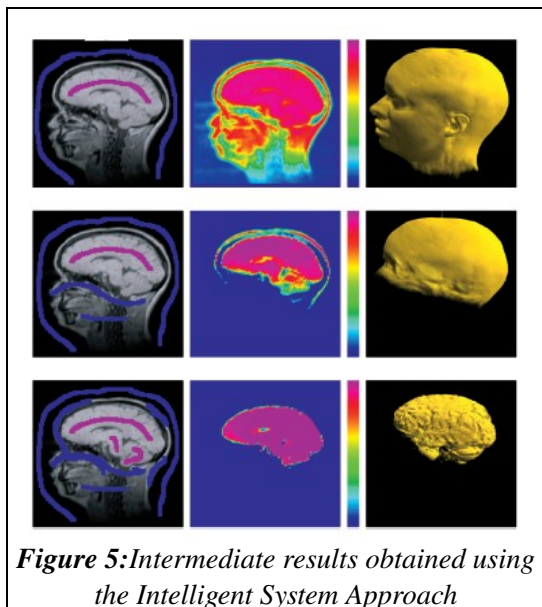

Figure 4: Painting based interface

Figure 5:Intermediate results obtained using the Intelligent System Approach

The progression of a session of a user employing this technique is shown in *Figure 5*. The left row shows a slice painted with pink representing the area the user wants to see and blue representing the area the user does not want to see. The middle row shows the result of the color-coded classification by a machine learning model. As indicated by the color bar, if a pixel's color is closer to pink, the pixel is more likely to be part of the material of interest. The right row is the volume rendering of the intermediate classification. If the user is not staisfied with the classification, the results can be reverted or furthur painting can be applied to achieve detailed results.

As already mentioned, a number of machine learning classifiers can be used in this process, all of which have the same goal – to minimize errors in classification of separate data into different classes.

Most machine learning methods (such as neural networks) reduce the probability of misclassification within the training data, while others (like support vector machines) reduce the probability of misclassifying unseen data. As the intelligent system approach performs volume display repeatedly, the classifiers that can be easily implemented on graphics hardware are preferred , providing acceleration for volume rendering. Thus, the various machine learning algorithms that can be used for classification have different trade-offs in accuracy, complexity and performance.

The main advantage of this kind of transfer function design is that the user is not required to work in the transfer function domain and has direct control over the classification process, due to the ability to view results in real-time.

4 Results

This section provides a comparison of the spatialized design technique and the intelligent system approach . A new design method based on the combination of the two methods is also proposed.

4.1 Performance

With respect to performance the spatialized transfer function is a better choice as the segmentation of features is done in the transfer domain itself , instead of the volume domain. The intelligent system approach involves volume rendering and display at each iteration implying more processing time and effort.

4.2 User Interface

Although the user interface provided by the spatialized method is intuitive, the user has to still work in the derived transfer function domain, which can be restrictive. This aspect is improved on by the intelligent system where the classified volume is rendered in real-time, which simplifies the task of the user.

4.3 Multiple Features

The intelligent system technique is essentially a single feature classifier .Classification of multiple features in the intelligent system is possible but implies the creation of a new classifier for each feature to be rendered, which would degrade performance. As spatialized transfer functions are based on separation of multiple features this factor is inherently implemented.

4.4 Information Reuse

Information reuse refers to the ability of a transfer function technique to be reused for other data sets. Almost all existing methods require that acquired data sets (e.g. MRI / CT scans), have a certain number of common characteristics. As the spatialized transfer function depends entirely on the position of the sample points, it is essential that all successive data sets on which the generated transfer function is applied contain points with similar spatial characteristics which can be difficult to attain. The intelligent system does not depend greatly on the positions of the sample points, so a classifier generated using this method is found to be more reusable. For example, the classifier can be reused to monitor a patient's condition over a period of time.

4.5 Proposed transfer function design

After having discussed the pros and cons of the spatialized and intelligent system transfer approaches, we propose a new transfer function design which combines the two. This combination can be obtained by integrating the two interfaces to allow the intelligent system rendering in the volume domain as well as specification of spatialized transfer function parameters - $r , c_{\alpha} , c_{\lambda}$. The process begins with the use of the intelligent system to isolate a specific region in the data set – for example , the brain region in a MRI scan. Once this is done the spatial information can be used to separate the features in the selected region, using the spatialized transfer function method.

The performance of this combined system will be lower than each of the described methods, but the trade off with respect to accuracy makes it acceptable. The intelligent system is used only to isolate a global region, which avoids a large number of iterations and the spatialzed technique then separates the features in a single pass.

The combined user interface is based on the intelligent system interface and hence retains all of its user friendly attributes. The spatial element of the interface can be made more user friendly by using a color chart instead of a 2D color histogram, to view the different features.

The multiple feature aspect in the combined design is provided by the spatialized transfer function and the intelligent system classifier can be utilized for information reuse.

5 Conclusion

Transfer function design remains one of the most difficult aspects of the volume rendering process. Since the types of structures of interest vary widely depending on the user of a system, transfer function design techniques must be powerful enough to provide high-quality classification, yet intuitive enough to be accessible to a wide range of people. This combination of performance and user-friendliness of the two techniques described in this survey paper make them excellent candidates for various types of volume rendering applications. The positive aspects of the two methods when combined, could provide even better results. This hybrid approach blends together the benefits of performance and multiple feature extraction of the spatialized approach and the user-friendliness and information reuse capability of the intelligent system method.

References

[1] S. Roettger, M. Bauer and M. Stamminger, "Spatialized Transfer Function", IEEE VGTC Symposium on Visualization, 2005

[2] F. Tzeng, E. B. Lum and K. Ma, "An Intelligent System Approach to Higher-Dimensional Classification of Volume Data", IEEE Transactions on Visualization and Computer Graphics, VOL. 11, NO. 3, May/June 2005, pp. 273-284.

Three Methods for Classifying Volume Data

Tiemen Rozeboom, Jordy Oldenkamp

1. Abstract

In this paper we compare three different techniques proposed for volume data classification. Volume classifiers are widely used in the medical field (for instance in classifying parts of the brain in MRI or CT-scans). Incidentally the persons using these classifiers are not generally skilled in the, often quite complex, field of volume data classification. As a result there has been a search for user-friendly yet effective ways to classify volume data. In this paper we will make a comparison of several semi-automated methods, while keeping in mind the human aspect as well as the classifier's performance and flexibility.

2. Introduction

As a contribution to the field of visualization this paper compares different ways to classify volume data. We will try to find an answer to the following question: What are the different methods available for volume data classification and how is their performance outlined to their usability and flexibility? There exists many different methods, but the three methods we consider are based on the articles [1], [2] and [3]. These three have some very different characteristics, and in this paper we will compare some of the advantages and disadvantages of each of these classifiers. We will start by giving an introduction in the field of volume data classification.

Volume data consists of voxels; these are similar to pixels in the 2D plain only in this case the voxels contain certain scalar values (for instance density) and are in 3D.

Because the volume-data is generally rendered in three dimensions so-called transfer functions (TF) are needed to assign specific colours and opacities to the values so that when rendering the data, the user is presented with a clear view of the areas of interest. In particular, material boundaries (which have high gradient values) get high opacities in order to emphasize these and blend out the uninteresting (low-gradient) regions.

Because of this, TFs are often in the space of scalar value and gradient magnitude, in this 2D space boundaries show up as arcs. As an example consider the left and middle pictures in Figure 1, where a part of the brain has been emphasized by mapping the colour and opacity of the rendering using a 1D and 2D transfer functions (the transfer function is shown in the bottom). In these images the user selects a part of the brain by pointing and clicking a class in the TF. This part is then emphasized by setting the emission of all the other classes to zero.

One problem with using 2D TFs is that material boundaries often overlap in this space. Therefore in order to obtain better results with regard to the classification higher dimensional TFs can be used. Using more dimensions mean using more properties per voxel allowing for an easier separation of boundaries. An example of using a 10-dimensional transfer function is also shown in Figure 1 from [1].
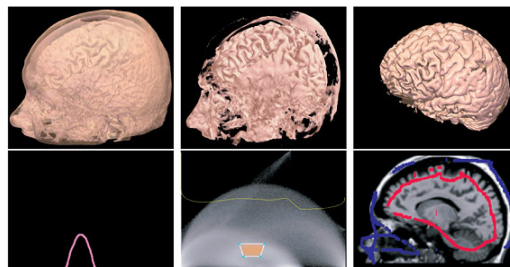

Figure 1: rendering with 1D, 2D and 10D transfer function

Several ways have been proposed to automatically set-up such transfer functions without having to manually tune it (as this can be quite complex). A semi-automatic setup can for instance be realized by generating several preset TFs with corresponding volume rendering, allowing the user to select the one which result is closest to the user's expectation. However, this approach is not very flexible as, when there is no suitable TF available, it still has to be manually set.

The methods we will discuss in this paper all have different approaches to this problem and we will try to give a comparison of these.

In the next section we will give an overview of the three techniques; we will start with the Hierarchical Clustering method in 3.1 and then proceed with Spatialized Transfer Functions in 3.2 and the Intelligent Systems approach in 3.3. We will then proceed to section 4 where we compare the techniques and finally give a conclusion in section 5.

3. Techniques

3.1 Automating Transfer Function Using Hierarchical Clustering

In order to counter the inflexibility without compromising user friendliness Šereda, et al. [3] propose an automated hierarchical clustering solution.

Part of the solution involves using the so-called LH-space, opposed to the normal scalar value / gradient magnitude space. The LH-space takes the origin and destination of an arc (in the scalar/gradient space) and uses these two values as coordinates. As a result, in LH space, boundaries appear as blobs instead of arches, reducing boundary overlap and making it easier to separate them.

As tuning of a clustering technique is usually difficult; Šereda, et al. propose a hierarchical clustering method using a framework where the user interacts with this hierarchy of clusters.

3.1.1 Hierarchical clustering

We will now give a short overview of how this hierarchical clustering is achieved. Having a set of n elements, the hierarchical clustering describes the order in which the elements join into clusters. To start, the initialization step assigns each element to a cluster; therefore the hierarchy has as many levels as there are elements. At the first level there are n clusters, after which the two most similar clusters join at each step. The difficulty is to find the initial set of elements and to define a function to find the two most similar clusters.

3.1.2 Similarity and initial set

Šereda, et al. proposes two different similarity measures. Namely a solution that groups similar boundaries in LH space and one that looks at the spatial connectivity of clusters.

To group similar elements in LH space one looks at the positions, sizes and shapes of the initial elements in LH space. Šereda, et al. group these elements based on the following criteria:

- Distance (close elements have similar L and H values)
- Separation (a deep valley between two peaks)
- Direction of elongation (join elements in the direction they elongate)

These criteria can be combined by using Bayesian decision theory as proposed by Šereda, et al. [3].

The second method groups boundaries based on spatial connectivity of clusters. Spatial similarity is evaluated by means of the number of direct neighbours between the clusters. For each voxel in an element, the method looks at its 26 neighbours and counts how many of these belong to another element and counts these, forming the direct neighbours. To group clusters that belong to the same boundary a weighting method is used [3]. The actual clustering is done similar to method one, it groups the two most similar clusters.

Šereda, et al. state that for both similarity measures, the time-complexity of joining two clusters is $O(K)$, where K is the number of clusters remaining in the hierarchy [3].

Initial clustering is done based on the blob like characteristics of clusters in the LH space. Every local maximum in the 2D LH histogram is labelled and all bins that belong to this peak are grouped in the same element. To get rid of the small clusters, because too many clusters lowers performance, a 2D Gaussian kernel of a small standard deviation equal to the bin size is used to blur the histogram [3]. After that, clusters that have a voxel count below a certain threshold are joined with their direct neighbours.

3.1.3 The framework and results

Based on the methods described above Šereda, et al. created a framework which enables real-time interaction with the clustering process. This is done because selecting thresholds and restarting the clustering algorithm is a tedious process. Another problem to which the framework offers a solution is deciding when two clusters are similar. This framework offers different similarity measures, so different parts of the hierarchy can be generated by a different similarity measure, making it more flexible. An illustration, provided by Šereda, et al. [3], of this (where **s** is the similarity measure used to cluster the elements) is shown in Figure 2.
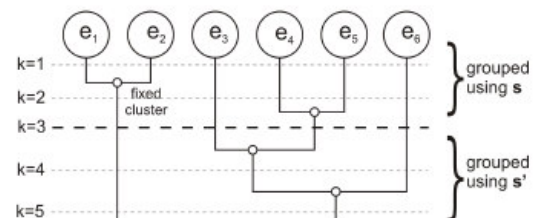

Figure 2: hierarchical clustering at level k using similarity measure s

It is also possible to fix clusters; these clusters will be taken out of the hierarchy. Likewise, clusters can also be selected to be explicitly joined together.

3.2 Spatialized Transfer Functions

The next technique is proposed by Roettger, et al. [2]. This method is presented as a way to automatically setup multi-dimensional TFs. Instead of trying to identify a single feature, it tries to separate as many features as possible using what Roettger et al. call spatialized transfer functions (STF). Then, after separating different features, a specific part of the volume can be selected by removing irrelevant features.

3.2.1 The Basic Principle

The paper suggests separating the features by using scalar/gradient values to determine the opacity and using spatial information to derive separate colours.

As explained in the introduction, using multidimensional TFs improves performance as there are more features per voxel, allowing for a better separation of different features, i.e. creating less overlap. Problem is these multidimensional TFs are even harder to setup. The paper claims that different features can be separated by using the gradient values (as these represent borders) as well as their position in space. However, there is no need for a 5D TF but only 2D or 1D TFs with the spatial information captured in the colours.

3.2.2 Automatic Transfer Function Setup

Roettger, et al. [2] continue to describe an automatic setup for the transfer function.

The first step in the setup is using the gradient values to blend out uninteresting parts and emphasize the interesting parts (i.e. borders). Although a simple method is proposed to achieve this, other, more advanced gradient weighed rendering methods can also be used.

The next step consists of colouring features with similar scalar characteristics that are in close vicinity of each other. Basically when voxels have the same scalar and gradient values, they should be painted in the same colour as long as the voxels are within a certain maximum radius of the features that are to be detected. To give an idea of a STF as, an example of a STF as a result of a (smoothed) 2D histogram (with scalar values on x-axis and gradient values on the y-axis) is provided from [2] and shown in Figure 3.
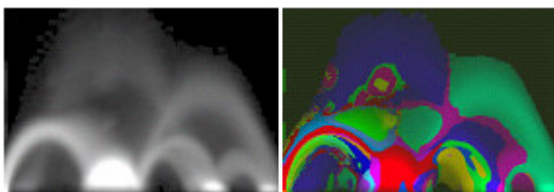

Figure 3: 2D scalar (x) vs. gradient (y) histogram with corresponding STF after smoothing

Furthermore the writers suggest starting off with a large radius and decreasing it until the desired features are properly separated.

3.2.3 Exploring the Volume

After the generation of the STF the user can further explore the data by selecting an area (cluster) from the histogram. The corresponding feature is then emphasized by making the other features transparent. Now, because the STF performs segmentation in the transfer function domain, the feature radius can be changed interactively without having to reprocess all voxels

and only a reclassification in the transfer function domain is needed. This leads to quick redrawing.

3.2.4 Shading

Another enhancement proposed is the use of pseudo-shading to further distinguish separate features. This is achieved by decreasing the emission where the scalar values of an object are low, this results in a dark rendering of the objects borders.

3.3 Intelligent System Approach

A different approach is proposed by Tzeng, et al. [1]. Instead of trying to devise a transfer function to improve classification, they offer a user interface in which the user paints directly in the volume data. These painted voxels are then used for training in an iterative process.

Tzeng, et al. offers an intelligent system, using a painter interface, where the user paints the regions of interest in the volume data. Then a machine learner is applied using the painted data as training data. A graphical representation of this is shown in Figure 4 (picture taken from [1]).
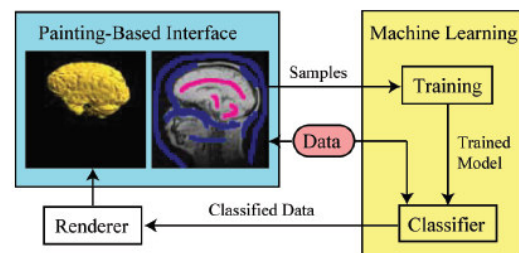

Figure 4: the proposed classification framework

When looking at practical use, like MRI scans, the solution provided can be helpful, since users can focus on their expert knowledge instead of the details of creating a transfer function.

In their research, Tzeng, et al. found that their painting-based interface indeed provides a simple and intuitive means of specifying regions of interest. As in traditional TF design combined with new datasets, a lot of time goes into experimenting with TFs. The interface by Tzeng, et al. frees the user from this experimenting.

3.3.1 The painting interface

The goal of the painting interface is to partition the data set into different classes of material. These partitions will then be passed on to the classifier to train. To achieve this the user paints on slicing planes, using two different colours, one to indicate the material belonging to a certain class, the other to paint the materials that do not belong to the class. These slices presented to the user are slices of the x, y and z plane. Another feature of the interface is real-time visual feedback from the classifier such that the user can identify errors

made by the classifier. An example of the interface is shown in Figure 5 (taken from [1]).
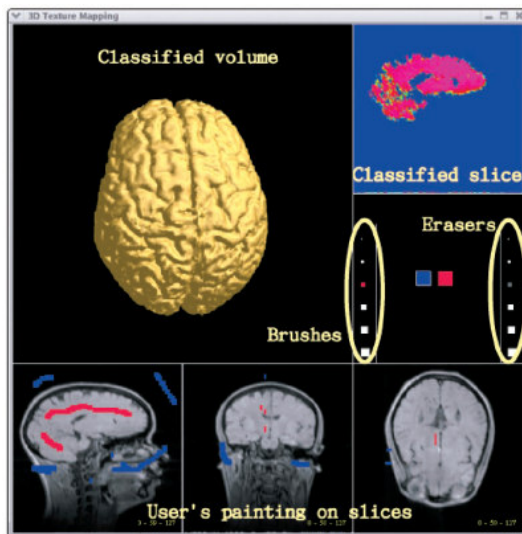

Figure 5: the painting user interface

3.3.2 Classification

The second part of the solution is the classifier, which is hidden from the end user. Tzeng et al. state that any method of supervised machine learning can work with their interface as long as it is able to classify the data from the samples provided by the user. Although any classifier would work, only results for Neural Networks (NN) and Support Vector Machines (SVM) are provided. Tzeng et al. researched the difference between these NNs and SVMs on their interface. The main difference between NN and SVM is the training. Where NN's train repeatedly, SVM's train only once with the entire training set and then provide the optimal solution when the training is done. NNs train repeatedly; therefore they can get stuck at local minima, where SVM's are guaranteed to reach a global minimal solution. Tzeng, et al. experimented with both classifiers and came up with the results shown in Table 1 and Table 2.

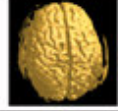**Table 1: using the Training Set Obtained Based on an NN Classifier**

Machine Learning Algorithm	NNs	SVMs
Number of Training Samples	7988	7988
User's Input Time	2-3 mins	-
Training Time	1.3 secs	0.74 secs
Classifying the Volume	6.28 secs	55.47 secs
Rendered Classified Volume		

Table 2: using the Training Set Obtained Based on an SVMs Classifier

Machine Learning Algorithm	NNs	SVMs
Number of Training Samples	818	818
User's Input Time	-	< 30 secs
Training Time	1.41 secs	0.06 secs
Classifying the Volume	6.47 secs	48.78 secs
Rendered Classified Volume		

The main difference in the results of these experiments is that the NN classification needs more training data and takes longer time for training, but outperforms the SVM when the actual classification is done.

4. Comparison

For comparing the three methods of classification we constructed Table 3, comments on this table are given in this chapter.

Table 3: table comparing the three methods.

	Hierarchical clustering	Spatial clustering	Intelligent systems approach
Flexibility	High	Low	Very high
Ease of use	High	Normal	Very high
Performance	Depending on similarity measure	High	Depending on learner

- Flexibility for hierarchical clustering is high due to its possibility to insert virtually any comparison function and change them on separate levels in the hierarchy (although this reduces usability). The spatial clustering framework is limited to the TF domain, therefore it scores low on flexibility. For the intelligent system approach a very high score is given for its ability to easily use multidimensional transfer functions. These are then simply taken as the input for a machine learner. Also, the possibility to use virtually any machine learner allows for great flexibility.

- As far as ease of use is concerned, hierarchical clustering gets a high score as the hierarchical interface somewhat shields the user from manually setting parameters. The spatial clustering method gets a normal assessment because the radius still has to be manually set. In the intelligent system approach the user is completely shielded from any transfer functions and can simply draw on the different data slices themselves, allowing expert

knowledge of the volume domain itself to be put to good use.

- The performance is hard to judge, except for the spatial clustering, because the other methods depend on similarity measures and machine learners used. On the other hand, the spatial clustering method seems a very optimal way to classify, using only spatial similarity and scalar/gradient data. Although a thorough comparison between qualities is not really possible, since the software solutions cannot be tested.

Another advantage of the intelligent system approach is its ability to 'spot features', for instance a neural network can be trained and (re-)used to spot tumors or similar tasks, this preserves the expertise used to train the neural network. Although on the other hand we imagine settings for any other method can also be stored and reused for different tasks.

5. Conclusion

When looking at the comparison chart we can see that the hierarchical and intelligent systems methods have great flexibility. However, the downside of this is that when using a different similarity measure or machine learner, these have to be set up. Setting this up seems a fairly complex procedure; especially in the intelligent system method this might require quite some expert knowledge.

While the spatial clustering method is an optimized way for classification using only spatial data, this might not outweigh the flexibility of the other methods. Also the hierarchical clustering method provides a method using spatial data, overlapping the spatialized clustering method. Where the spatialized clustering method is already maximally optimized, the intelligent systems approach can benefit from future optimizations in the machine learning field. Therefore we think the intelligent systems approach, based on its intuitive interface and flexibility will be the way to go in the future.

References

[1]. **Tzeng, F., Lum, E. B. and Ma, K.** An intelligent System Approach to Higher-Dimensional Classification of Volume Data. *IEEE Transactions on Visualization and Computer Graphics.* 2005, pp. 273-284.

[2]. **Roettger, S., Bauer, M. and Stamminger, M.** Spatialized Transfer Functions. *IEEE/EuroGraphics Symposium on Visualization.* 2005, pp. 271-278.

[3]. **Šereda, P., Vilanova, A. and Gerritsen, F. A.** Automating Transfer Function Design for Volume Rendering Using Hierarchical Clustering of Material Boundaries. *EuroVis - Eurographics/IEEE-VGTC Symposium on Visualization.* 2006, pp. 187-194.

Modelling the search for salient locations in images

Jan-Jaap Bakker, Hessel Hoogendorp

Rijks*universiteit* Groningen

Abstract - By modelling the early Human Visual System it is possible to create a fast system for determining salient locations in still images. An established, context independent model for this purpose is described. In addition, an established model for predicting the efficiency of search for motion pop-out phenomena is discussed and an attempt is made to apply it to the former model to predict the efficiency of search for a salient location in a series of greyscale still images.

Keywords - Pop-out phenomena; HVS; saliency; visual attention; feature extraction; target detection; visual search

---

1 Introduction

Humans effortlessly notice salient objects. We automatically pay attention to objects that require it. While driving, we pay attention to traffic signs, other vehicles (especially vehicles that move differently from the rest) and Gatso cameras. It is assumed that within the human brain, the Human Visual System (HVS) consists of two parts [1]. One is a top down, knowledge driven system, for example searching for your own car in a parking lot. The other, which is the focus of this paper, is a bottom up, saliency based system. Together they form the ability to see and classify our surroundings. For instance, if, while reading this paper, a spider crawls down the top left corner of the readers view, it is the bottom-up system that tells the top-down system to pay attention to that region of your view. How much attention is spent on the spider, depends on how much of an arachnophobe the reader is. The bottom up system is a necessity since the top down system alone would not be able to handle all the stimuli coming from our retinas [2]. In this paper it is shown how the bottom up part of the HVS can be modelled for still images [3]. The top-down part is not handled, so the model merely returns the most salient locations of the input images. The authors also show how an established model [4], for predicting how easy humans can identify moving objects in various conditions, can be used for still images.

2 Modelling the Human Visual System

The early human visual system consists of many layers of neurons. In the first layers some basic, but effective, processing is done. Some neurons are organised to capture and enhance differences in orientation, while others do this for intensity, colour, etc. Lateral inhibition is one example of the strategies used (it is used in all of our senses, and explains many optical illusions). It are these basic processes, that are going to be modelled. In the model, maps for colour, intensity and orientation are created. These are treated separately and transformed into conspicuity maps, which only contain information about the saliency of each location in the image, independent of visual features like colour, orientation and intensity. In the absence of top-down supervision, the maps are normalized (this

process is explained later) and combined into a single saliency map. Then a winner-takes-all principle is used to find the most salient location. This simply means that the most salient location in the image is declared the winner, without caring about the runners-up. To find the next most salient location, the previous one is simply inhibited in the saliency map.

3 A model of saliency-based visual attention for rapid scene analysis
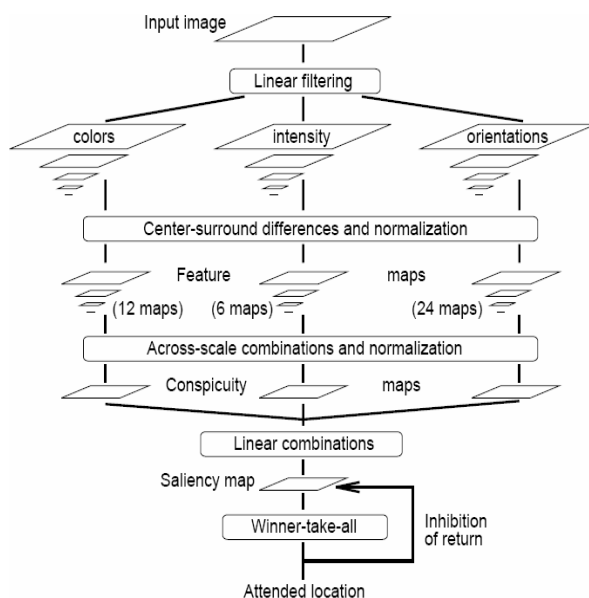

Figure 1: Overview of the model (Figure taken from [3])

In [3], Itti et al. propose a model of saliency-based visual attention. This basic model for saliency based visual attention is depicted in Figure 1. The steps that are taken in this model are, firstly, the extraction of three early visual features from the input image, into maps. These early visual features are contrast intensity, colour and orientation. Secondly, from these early visual feature maps, nine spatial scales are created using dyadic

Gaussian pyramids [1][5]. By now, each visual feature is represented by a number of feature maps. As a third step, these feature maps will be used to create three conspicuity maps. Lastly, a final saliency map is created from the conspicuity maps. It is this final saliency map that is then used to determine the most salient location in the image using a winner-takes-all strategy. In the next sections, these steps are described in more detail.

3.1 Extraction of early visual features into feature maps

3.1.1 Colour

The input image consists of three colour channels, r, g and b. For each of these colours, its value relative to the average value of the other two colours is calculated, as follows:

```
R   = r - (g + b) / 2
G   = g - (r + b) / 2
B   = b - (r + g) / 2
```

Also, a colour value for yellow is calculated from the original colour values:

```
Y = (r + g) / 2 - |r - g| / 2 - b
```

Next, four Gaussian pyramids $R(\sigma)$, $G(\sigma)$, $B(\sigma)$ and $Y(\sigma)$, with scale $\sigma \in [0...8]$, are created from these relative colour values.

As a last step, centre-surround difference is used to model lateral inhibition. The centre-surround difference between two maps, denoted by 'Θ', is obtained by interpolation of the coarser scale to the finer scale, followed by point-by-point subtraction. Several scales are used both

---

[1] Gaussian pyramids involve scaling down images using Gaussian blur. Then, when applied multiple times, the smaller images stack on top of the larger, creating a pyramid. The blur is needed to prevent the typical 'pixel' edges that occur when resizing an image.

for the centre and the surround, yielding true multi-scale feature extraction.

In the centre of their receptive field, neurons are excited by one colour and inhibited by another, while the converse is true in their surrounding [6]. Accordingly, two sets of six colour feature maps are computed as follows:

```
RG(c, s) =
  |(R(c) - G(c)) Θ (G(s) - R(s))|,

BY(c, s) =
  |(B(c) - Y(c)) Θ (Y(s) - B(s))|
```

Here $c \in$ `{2, 3, 4}` and $s = c + \delta$, $\delta \in$ `{3, 4}`.

3.1.2 Intensity

As said before, the input image consists of the r, g and b colour channels. From these colour channels an intensity image I is computed:

```
I = (r + g + b) / 3
```

Similar to the case of colour, a Gaussian pyramid `I (σ)`, with $\sigma \in$ `[0...8]`, is created from the computed intensity image. To separate hue from intensity, the r, g and b colour channels are normalized by `I`. Also, locations in the image which have an intensity value less than one tenth of the global maximum intensity are set to zero. This is done because variations in hue are not perceivable by humans at such low intensities. To obtain the six final intensity feature maps the centre-surround differences are computed for $c \in$ `{2, 3, 4}` and $s = c + \delta$, $\delta \in$ `{3, 4}`:

```
I(c, s) = |I(c) Θ I(s)|
```

3.1.3 Orientation

The previously calculated `I` are used to create a set `O(σ, θ)` of oriented Gabor[2] pyramids (Figure 2 shows an example of the output of a Gabor filter). Here, $\sigma \in$ `[0...8]` is the scale, and $\theta \in$ `{0°, 45°, 90°, 135°}` is the orientation. Using the centre-surround differences like before, 24 (six for each angle) orientation feature maps are obtained:

```
O(c, s, θ) = |O(c, θ) Θ O(s, θ)|
```


Figure 2: Combined result of applying four Gabor filters (one for each orientation) on an example image.

3.2 Construction of the conspicuity maps and the saliency map

Now that all feature maps have been calculated, the three corresponding conspicuity maps need to be computed, from which the final saliency map can be constructed. Before the feature maps can be combined into a single saliency map, the feature maps need to be normalized such that their values fall within the same range. Furthermore, maps containing a small number of feature peaks tend to be masked out by maps containing large numbers of feature peaks. To compensate for this, the former maps are globally promoted, while the latter maps are globally demoted.

To realize this, the map normalization operator `N(.)` is introduced. Firstly, this

---

[2] A Gabor filter is a linear filter whose impulse response is defined by a harmonic filter multiplied by a Gaussian filter.

operator normalizes the values of the map to a particular range and, secondly, globally promotes maps with a small number of feature peaks and globally demotes maps with a large number of feature peaks. This last property is implemented by multiplying the map by *(M − m)²*, where *M* is the map's global maximum and *m* is the average of the map's all other local maxima. This squared difference effectively measures how different the maximum feature peak is from all other feature peaks. If this difference is large, the map is globally promoted. Similarly, if this difference is small, it is globally demoted. An example is illustrated in Figure 3.
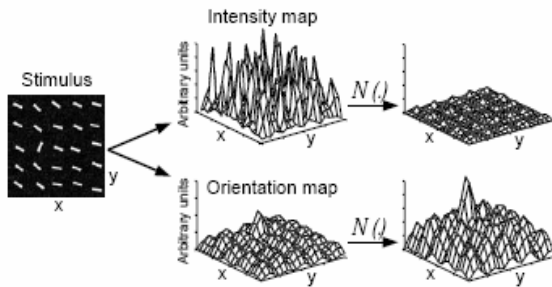

Figure 3: Effect of the normalization operator (Figure taken from [3])

The conspicuity maps for each of the three features are obtained by point-by-point addition ($\oplus$) of the values from all the feature maps of that feature. For colour and intensity this yields, respectively:

$$\overline{C} = \mathop{\oplus}\limits_{c=2}^{4} \mathop{\oplus}\limits_{s=c+3}^{c+4} [N(RG(c,s) + N(BY(c,s)))],$$

$$\overline{I} = \mathop{\oplus}\limits_{c=2}^{4} \mathop{\oplus}\limits_{s=c+3}^{c+4} N(I(c,s))$$

For orientation this trick needs to be repeated, and their results added, for every orientation, yielding:

$$\overline{O} = \sum_{\theta} N(\mathop{\oplus}\limits_{c=2}^{4} \mathop{\oplus}\limits_{s=c+3}^{c+4} N(O(c,s,\theta)))$$

where $\theta \in \{0°, 45°, 90°, 135°\}$

The saliency map is now constructed by the simple addition of the normalized conspicuity maps:

$$S = \frac{1}{3}(N(\overline{C}) + N(\overline{I}) + N(\overline{O}))$$

Notice that the normalization takes place on both the feature map and the conspicuity map level.

4 Quantifying efficiency of search for motion pop-out phenomena

In [4], Rosenholtz suggests a model for the quantification of the efficiency of search for motion pop-out phenomena. Efficiency of search is defined as the reaction time per number of display elements. One of the elements is the target, the rest are distracters.

The first step of this model is to consider the velocity of each display element as a point (v_x, v_y) in velocity space and to calculate the mean μ and covariance Σ of the distracters. It is suggested in [4] that the inclusion or exclusion of the target element's velocity in this calculation has a negligible effect on the resulting μ and Σ for sufficiently large data sets.

Next, target saliency is defined as the Mahalanobis distance Δ between the target velocity v and the mean distracter velocity:

$$\Delta^2 = (v - \mu)^T \Sigma^{-1} (v - \mu)$$

In other words, target saliency is defined as the number of standard deviations between the target velocity and the mean distracter velocity. Rosenholtz states that the larger the target saliency, the more efficient the search for that target can be performed.
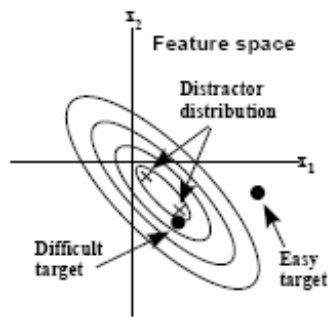
Figure 4: Graphical depiction of the saliency model. Ellipses represent points of equal saliency. Outer ellipses correspond to greater saliency and thus easier search (Figure taken from [7]).

5 Investigating efficiency of search for greyscale still images

Rosenholtz [4] suggests that, although experiments have not yet been performed, the model can also be applied to the luminance domain. To investigate this, the authors created a simple test.

The authors used a computer to generate images consisting of a hundred, somewhat noisy, circles. These circles all have the same diameter and are uniformly distributed over the plain. Their luminance is Normal distributed (μ, σ).

After the placement and colouring of these circles, another circle is added, also at a random location. However, its luminance is chosen in the range $[0, \mu - 3\sigma]$, making the last circle the most salient object. The model predicts that the larger the deviation from μ, the easier it is to select it as the most salient object (exactly how much easier, e.g. linear or quadratic, the model does not state).

It is intuitively clear that for humans this prediction will hold true. The authors are interested in whether it holds for an implementation of a bottom up visual attention model. In other words, a test is performed with a model as the (non

learning) test subject. The used general purpose model contains no specific knowledge about the test to be performed. Figure 5 shows which location the model selected as being the most salient.
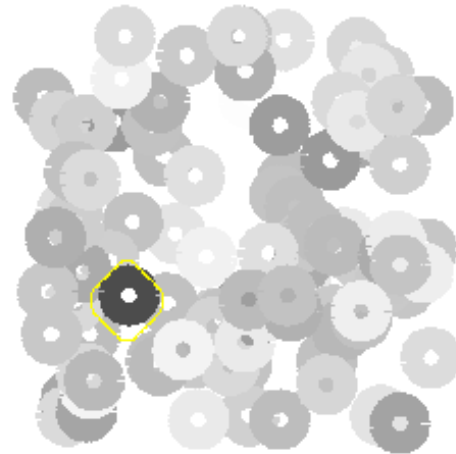
Figure 5: An example test image. The last circle is marked with a yellow line.

For a number of intensities selected from the range stated above, the average simulated time it took the model to find the most salient location, is measured. The results are depicted in Figure 6. As can be seen, the model indeed produces results faster when the deviation of the luminance of the target from the average luminance of the circles increases.
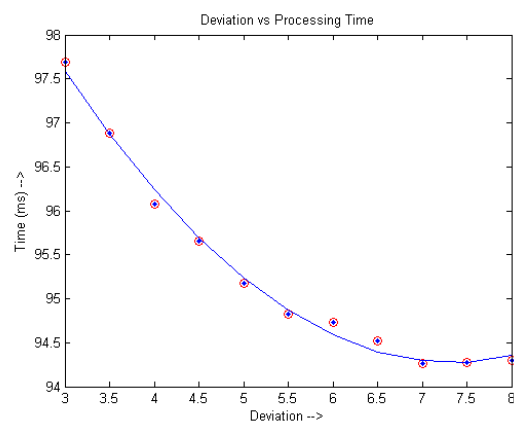
Figure 6: The deviation from μ against the reaction time.

6 Discussion

The model by Itti et al. was extensively tested on both artificial and natural images. Its results show great resemblance to those of human test subjects. It outperforms spatial frequency map models, especially in the presence of noise.

Due to the splitting up and separate treating of the feature maps, the model is suited for parallelization, enabling real time execution. For some purposes the salient locations can be used directly, for instance for automatic thumb-nailing. But in most cases the resulting saliency map will be passed on to other processes. This way, it is possible to execute more complex and time consuming algorithms only on the salient/interesting parts of the image.

The experiment performed is not meant as proof of the usability of Rosenholtz model for still images. It is, however, an illustration of what such a test could look like. To fully test the model, one should use multi-dimensional features, with different distributions and, of course, human test subjects.

7 References

[1]   J.K. Tsotos, S.M. Culhane, W.Y.K. Wai, Y.H. Lai, N. Davis, F. Nuflo, "Modeling visual attention via selective tuning," *Artificial Intelligence,* vol. 78, no 1-2, pp. 507-545, Oct. 1995

[2] E. Niebur and C. Koch, "Computational architectures for attention", R. Parasuraman, (Ed.), *The attentive Brain*, Cambridge, MA:MIT Press, pp. 163-186, 1998.

[3] Laurent Itti, Christof Koch, Ernst Niebur, "A Model of Saliency-based Visual Attention for Rapid Scene Analysis", *Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, issue, 11, pp. 1254-1259, Nov. 1998

[4] Ruth Rosenholtz, "A simple saliency model predicts a number of motion popout phenomena", *Vision Research*, vol. 39, pp. 3157-3163, Sep. 1999.

[5] H. Greenspan, S. Belongie, R. Goodman, P. Perona, S. Rakshit, C.H. Anderson, "Overcomplete steerable pyramid filters and rotation invariance," Proc. IEEE Computer Vision and Pattern Recognition (CVPR), Seatle Washington, pp. 222-228, Jun 1994

[6]   S. Engel, X. Zhang and B. Wandell. "Colour tuning in human visual cortex measured with functional magnetic resonance imaging," Nature, vol. 388, no. 6637, pp. 68-71, Jul. 1997.

[7] R. Rosenholtz, Yuanzhen Li, Jonathan Mansfield and Zhenlan Jin. "Feature Congestion: A Measure of Display Clutter" , April 2005

Visual Salience: A Method for Rapid Scene Analysis

Gerard van der Lei

Abstract:

In this paper several models for determining the visual saliency of a scene will be discussed. First two models that work with static colour images. The first model will be discussed in some detail. The second is there to provide some perspective on the subject and won't be as detailed as the first.

Motion is an important feature for determining the saliency of a scene. Therefor a third model will be presented. This model has a focus on motion, but can also be applied to the saliency based on colour. How this model can be applied to saliency based on motion will only be discussed. This model also gives an estimation of the difficulty of determining visual saliency and why this is relevant.

In the end an example of the results of the first model will be shown, to illustrate the end result of an model that predicts visual saliency.

Introduction:

The human visual system (HVS) is believed to be divided in two types of processing, attentive processing and preattentive processing. Attentive processing is a top-down process, where a scene is analysed based on an objective. Preattentive processing is the processing of a complex scene in a bottom-up manner. This is an involuntary process which bring attention to feature in a scene that contain a higher saliency.

This preattentive processing assists humans in processing the huge amount of information which is presented at every waking moment. Instead of having to process every part of a scene in detail, this preattentive processing turns our attention to parts of the scenes which are called focus of attention (FOA). Each of these areas can then be processed in detail, for example by an attentive process, when the goal would be to find a certain object in a scene.

There are certain number of elementary features in a scene which humans seem particular good at distinguishing, features like hue, saturation, orientation and motion. Especially when there are large differences in these features our attention is drawn to them, although this is not always true for motion, where attention is more drawn towards fast moving object among slow moving then vice versa, but differences do draw our attention.

In this paper several methods for determining the visual salience of a scene will be discussed. All these techniques work on the same basic method of determining a contrast in one or more features. They try to determine where there is a sudden change in brightness, or colour, or find areas were the directions of edges or motion is different than in the surrounding parts.

First model:

Laurent Itti, Christof Koch and Ernst Niebur have written a paper[1] in which they describe a method of determining visual saliency inspired by the behaviour and the neuronal architecture of the early primate visual system. In this model they combine multiscale image features into a single topographical saliency map. A dynamical neural network then selects attended location in order of decreasing saliency.

Input is provided in the form of a static image, therefor it cannot determine saliency based on motion. However the three remaining features are taken into consideration when determining saliency. Important for this technique is the fact that it works with multiscale images. Therefor the input image will first be downscaled to nine spatial scales using Gaussian pyramids, which progressively lowpass filter and subsample the input image, resulting in horizontal and vertical image reduction factor ranging from 1:1 to 1:256.

The scene is broken up in the different components. The first component of interest is the saturation or intensity. This is calculated by adding up the R, G and B components of the image and then normalising the answer. If this is below a certain threshold the value is set to zero. This is done because hue variations are not perceivable at very low saturation levels and are therefor not salient.

Then the colour information is transformed into four colour channels. $R=r-(g+b)/2$ for red, $G=g-(r+b)/2$ for green, $B=b-(r+g)/2$ for blue and $Y=(r+g)/2-|r-g|/2-b$ for yellow (negative values are set to

zero). These four colour channels are needed because the HVS works with the opponency of the red and green colours and blue and yellow colours. Based on these four channels, two new channels are calculated; *RG* which is the contrast between the *R* and *G* channel and *BY* which is the contrast between the *B* and *Y* channels.

The last of the feature that applies to still images is the orientation. The image is run through several filters which determine the stimuli at each area for different orientations. The orientations that are being calculated are at 0, 45, 90 and 135 degrees. The result of this is, that for each pixel, for each tested orientation a stimuli is calculated. This orientation can be calculated purely based on intensity channel using gabor pyramids.

All these feature are extracted in the nine spatial scales. This means that at this point there are nine different intensity maps, 18 different colour maps (nine scales for both colour channels, RG and BY) and a total of 36 orientation maps(nine scales for all four different orientation).

Based on these maps the contrast maps can be computed based on a set of "center-surround" operations. Each pixel in each channel is compared with its surrounding. This can be done at a number of different levels, increasing the size of the surrounding at each level. This is where the different scales are used. The downsampled images are interpolated to higher scales, then point-by-point subtraction is applied. The result of this, is that for each pixel at the higher scale, the difference with its surrounding is calculated.

Before we can add these different channels together, resulting in our final saliency map, we have to correct for the fact that at certain areas in one channel a feature can be very salient, but when adding the maps together it will be suppressed by noise that is present in other channels. Therefor we normalise channels in such a way that values that lie close to the mean will be suppressed and values far from the mean of a channel will be amplified.

After the normalisation all the different channels can be added together resulting in the final saliency map.

To find the maximum of the saliency map, it would be possible to just find the

Figure 1: General Architecture

maximums in the map and return those, but because in this implementation the aim was to find a biological plausible mechanism, the saliency map is modelled as a 2D layer of leaky integrate-and-fire neurons. These model neurons consist of a single capacitance which integrates the charge delivered by synaptic input of a leakage conductance and of a voltage threshold. This feeds into a biologically-plausible 2D winner-takes-all (WTA) neural network.

Each saliency map neuron gets its input from S, which causes the neurons at more salient location to increase faster. These saliency map neuron excites its corresponding WTA neuron, which evolve independently of each other until one reaches the threshold.

The Focus of Attention (FOA) is shifted to this location, all WTA neurons are reset to zero and the saliency map neurons in the FOA area are reset. This resetting of the saliency map neurons is done so that the next most salient location will be the next winner. To create a slight bias to location close to the current FOA the saliency map neurons close the the FOA are slightly excited.
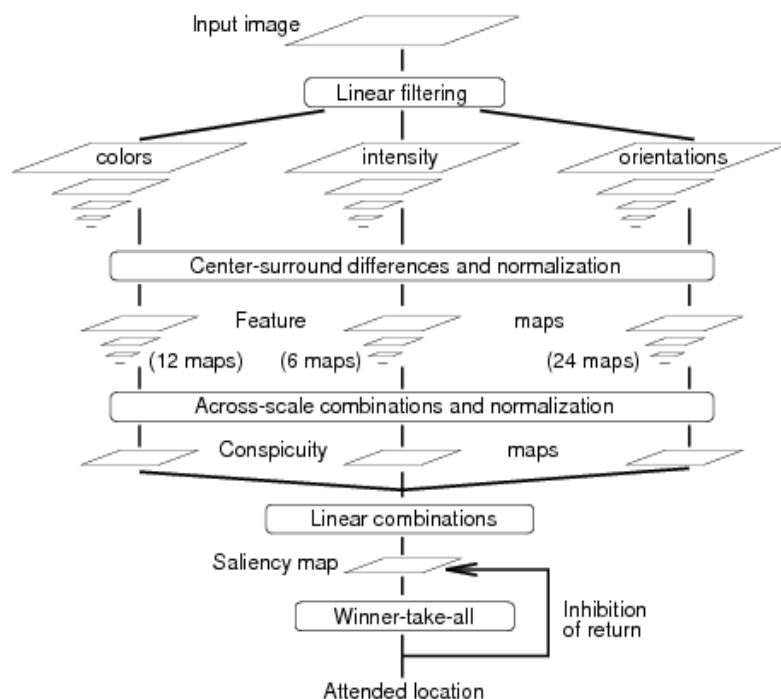
Because there is no top-down model modelled we do not know how large and of which shape the FOA is, for this reason the FOA is a simply disc with one sixth the size of the smaller of the input image width or height.

Alternative model

Itti et al's approach is by no means the only approach to finding the most salient area. O. Le Meur, P. Le Callet, D. Barba, D. Thoreau and E. Franois have written a paper [2] which takes even more inspiration from the human visual system to model their approach to finding areas of high saliency. This is a complicated model, whose details are beyond the scope of this paper. However, a general impression can be given to set Itti et al's model in perspective.
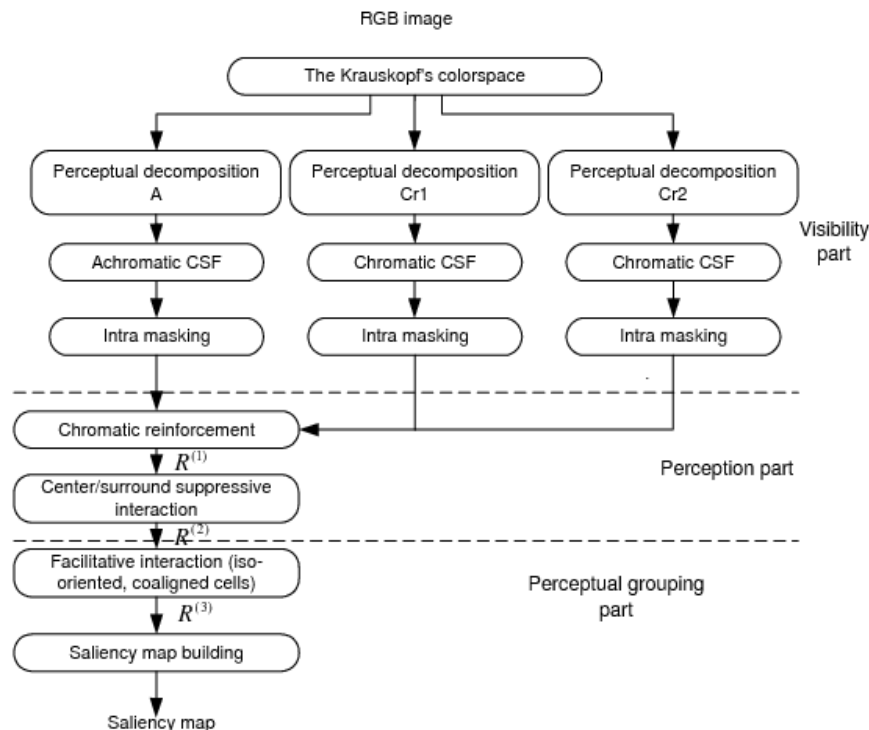
Figure 2: General synoptic of Le Meur et al's model

This model also takes an RGB image as input. It decomposes this image into the Krauskopf's opponent-colours space composed by the cardinal directions A, Cr1 and Cr2. It is believed that the brain uses three different pathways to encode the information, the luminance signal (A), the red and green opponent component (CR1) and the blue and yellow opponent component (Cr2).

To these three components a perceptual subband decomposition is applied. This decomposition is based on different psychophysics experiments. The result of this decomposition is the carving up of the 2D spatial frequency domain both in spatial radial frequency and orientation. Resulting in several subbands with different angular selectivity.

Biological evidence suggest that visual cells respond to stimuli above a certain contrast, this contrast value is called the visibility threshold. A contrast sensitivity function expresses the sensitivity of the human eyes which is the inverse of the contrast threshold. To each of the subbands this contrast sensitivity functions is applied, these are multivariate functions mainly depending on the spatial frequency and the orientation.

After the CSF is applied, masking is incorporated as a weighting of the outputs of the CSF functions. Masking describes interactions between stimuli, mainly between stimuli located in the same perceptual channel or in the same subband.

This concludes the visibility part of the model. The perception part consists out of the reinforcement of the achromatic subbands(A) based on the colour subbands(Cr1, Cr2) and modelling the center-surround suppressive interactions by a two-dimensional Difference-of-Gaussians.

The last stage is the perceptual grouping part. In this part center-surround facilitative interactions

are modelled, this facilitative interaction is usually termed contour enhancement or contour integration and improves the saliency of isolated straight lines. The last part of the perceptual grouping part is adding up the different subbands, resulting in the saliency map.

Incorporating motion

The models described above do not take into consideration the effect of motion. Even though in the human visual system the visual saliency based on motion is very important. Consider for instance a object, like a ball, moving fast towards your head. Even if you did not give this object any prior attention, it would be in your best interest to move your head out of the path of this object.

Ruth Rosenholtz wrote an paper[3], which gives an approach for determining the saliency of moving object. This paper further concentrates on prediction the difficulty of determining the visual saliency of an object.

In this paper it is hypothesised that the more an object draws our attention, the more efficient the search for that object would be, opposed to an object that does not draw our attention. This gives way to a naïve hypothesis that search efficiency could possibly be determined by the distance in feature space between the feature value for the target and for the most similar distractors.

For motion search this means that one might expect the search efficiency to be determined by the difference between the target velocity and the velocity of the distractors that moved most like the target. However, for motion search a number of search asymmetries show that the reality is not as simple as this.

Research[4, 5, 6] shows that search for a moving target among stationary distractors is easier than the search for a stationary target among moving distractors. The search for a fast target moving among slow distractors is more efficient than search for a slow target among fast distractors [7] and adding variability in speed when searching for a unique motion direction has little effect, while adding variability in direction when searching for a unique speed makes the search task more difficult [8].

The paper[3] of Rosenholtz presents a model for the bottom-up mechanism behind the motion popout that explains all of these motion search results. In this model the motion of all display elements are represented as a point in velocity space (v_x, v_y) From the distribution of the motions present in the display the mean and covariance of the distractor motions, μ and Σ, are calculated. In practise, instead of only calculating the mean and covariance of the distractor motions, the motion of the target is also incorporated.

The saliency of a display element is then defined as the Mahalanobis distance, Δ, between the target velocity and the mean of the distractor distribution, where

$$\Delta^2 = (v - \mu)^T \Sigma^{-1} (v - \mu)$$

To verify the results they are plotted. A 1σ covariance ellipse is plotted centered at the mean distractor velocity. The target is also plotted. If the target falls inside this ellipse, it is considered to be hard to search for and therefor has a low saliency. If instead it lies outside this ellipse, the search for the target is considered to be easy and therefor the target has a high saliency.
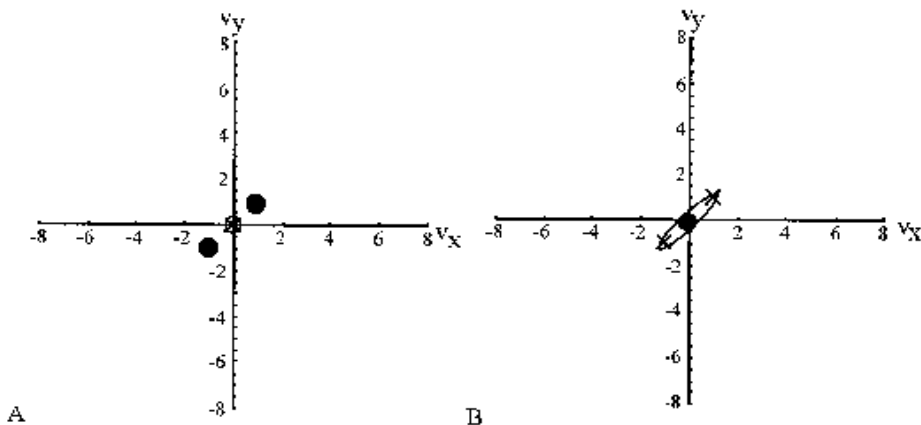

Figure 3A: The search for a moving target among stationary distractors   **Figure 3B: The search for a slow moving target among fast moving distractors**

Figure 3A show the results for the search for a moving target among stationary distractors. It can be seen that the target is easily outside the ellipse, which is very small, because all the distractors have the same speed and direction. Because the target is far outside the ellipse this model predicts it will be easy to search for and has a high saliency.

Figure 3B shows the search for a slow moving target among fast moving distractors. These distractors all move symmetrically. The target lies in the middle of the ellipse, which means that the search for this target is predicted to be very difficult and thus has a low saliency.

Symmetrically moving distractors are used, because if the motion would be randomly distributed, the mean motion would have large chance of being close $(0,0)$, which is equal to the motion of a stationary target. If instead of this, symmetrically moving distractors are used, the mean won't be close to $(0,0)$, but the search is still predicted to be difficult. This suggest that the search efficiency is not determined by the distance in feature space between the feature value for the target and for the most similar distractors, like naïvely hypothesised earlier.

A

B
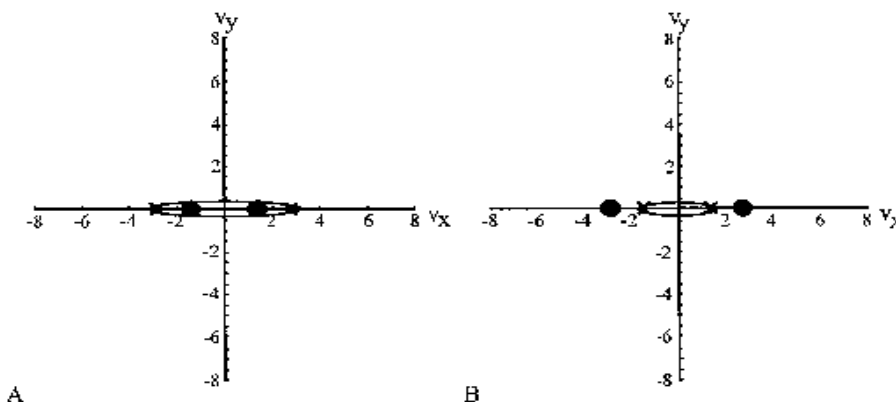
Figure 4A: The search for a slow target among fast distractors

Figure 4B: The search for a fast target among slow distractors

Figure 4 show that this model also correctly predicts the search difficulty of a slow moving target among fast distractors versus the search for a fast moving target among slow distractors. The slow moving target lies inside the ellipse and is predicted to be hard to search for, in contrary the fast moving target lies outside the ellipse and therefor is predicted to be easy to search for. Which corresponds with a high saliency.

Results:

All these models give different results, but to give an impression a result of the Itti et al's model is shown in figure 5. It is an example of determining the saliency of a natural image. At the top the input image is shown, below that are the three intermediate saliency maps. The first for the colour, in the middle is saliency of the intensity and the right picture is for the orientation.

When they are combined you get the final saliency map, from which the first four FOA areas are extracted. Which can be seen in figure 6. It can be seen that in the saliency map the location FOA is nullified (made black). After each FOA is found the surrounding area is made a little bit brighter.
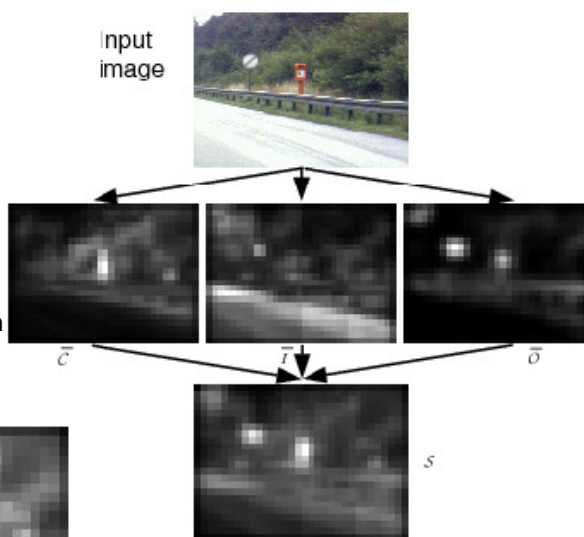
(Above)Figure 5: Example of saliency map

(Left)Figure 6: The first 4 FOA

Conclusion

In this article we have seen three techniques for determining visual saliency of a scene, two techniques used static images and a third that worked mainly on motion. One technique for static images(Itti et al's model) has been discussed in some detail, including showing a little of the results of this model. The second model(Le Meur et al's model), although not discussed in detail, does give an impression of the depth of this field of research.

In the model for determining the saliency of model some attention has been given to determining the difficulty of determining the saliency of an object. This technique has been discussed here only applied to motion, but in the original article[3] it is also extended to working with colour.

After the visual saliency of a scene has been analysed and the most salient areas are determined a next stage of processing should be done on these areas. First to determine the size of the actual object which has drawn attention. After the size of the object has been determined, an top-down process could start processing the image and determine if it is actually worth the attention. For instance when looking for a certain object it could determine if this is that object or not.

In many ways determining the visual saliency is only the fist step in processing visual information. Most computer visual system get their input within strict prior known limitations, but when computer vision systems will get more and more freedom of movement in the world determining the saliency of the input will become very important.

Bibliography

[1] L. Itti, C. Koch, and E. Niebur, "A model of saliency-based visual attention for rapid scene analysis," IEEE Trans Patt Anal Mach Intell) 20(11), pp. 1254--9, 1998

[2] O. Le Meur, P. Le Callet, D. Barba, D. Thoreau and E. Franois, "From low level perception to high level perception, a coherent approach for visual attention modelling", in proc. SPIE Human Vision and Electronic Imaging, San Jose, CA, 2004

[3] R. Rosenholtz, "A simple saliency model predicts a number of motion popout phenomena", in Vision research (Vis. res.) ISSN 0042-6989 1999, vol. 39, no19, pp. 3157-3163

[4] M. Dick, P. McLeod, Z.Dienes, "Parallel and Serial Processes in Motion Detection" in Science vol. 237, 400, 1987

[5] M. Dick. Thesis, Weizmann Institute (1989)

[6] N.L. Klempen, E. Shulman, C. Royden, J.M. Wolfe, "Visual search asymmetries in. motion and orientation" in Investigative Ophthalmology and Visual Science (Supplement) vol 39, 165, 1998

[7] R. Ivry, A. Cohen, "Asymmetry in visual search for target defined by differences in movement speed" in Journal of Experimental Psychology: Human Preception and Performance vol. 18, pp. 1045, 1992

[8] J. Driver, P. McLeod, Z. Dienes "Are speed and direction coded independently by the visual system?" in Spatial Vision vol 6 pp 133, 1992

Visualization: at a crossroad

Menno Nijboer and Ceesjan Luiten
E-Mail: `colloq@qtea.nl`

Major Computer Science Students at the State university of Groningen (RuG)

May 23, 2007

Abstract

The information visualization world is at a critical point right now. Leading scientists are saying that customer interest is dropping, and so far the field has not been very successful in establishing a scientific formalization of the knowledge and evaluation models. In our paper we will discuss the opinions and remarks of three of those scientists.

1   Introduction

Around 1987 the visualization field saw its first light. The first steps were done by — to name a few — Harvy Klein, Bill Lorensen and Pixar[1]. Since then, the field has grown substantially. Nowadays, many different specializations exist within the field. For example, one of these is the visualization of 3D models, which we are all quite familiar with. Another is the one that will be discussed in this paper: the visualization of complex datasets. This field has mostly revolved around incremental improvement regarding iso-surfaces and volume rendering. Lately, however, there has been a stagnation in interesting topics and papers. We will discuss three views of leading researchers about the current state of the field and their ideas of what needs to change.

2   Gap

We start with discussing the view of Bill Lorensen [1]. According to him, there is a visible gap between the field of visualization and its customers. Most people from the field have no idea how customers from the real world use visualization in their work nor what they think of certain visualization methods. On the IEEE Visualization Conference all customers have disappeared. The organizations that are still visiting the Conference are sending their computer scientists, not the scientists that just use visualization.

The field is clearly disconnected from the rest of the world, which can be easily observed when looking at the commercial market. The company Vital Images [6] is selling a product

---

[1]A visual entertainment company, leading the animated movie industry.

called Vitrea. Part of this program is a Volume Rendering application, capable of visualizing extensive data sets. The foundation of this program lies within the visualization field and yet not that many people from visualization are familiar with this program. Because this kind of method is already available on the customers market, the area of Volume Rendering should try to shift their goals and steer themselves in a different direction.

A remedy for the gap is also given by Lorensen [1]: The field must more actively approach customers and get their perspective, their opinions, their visions on visualization. Basically, there needs to be more interaction. Another path[2] that can and should be taken is to solicit more application papers which are co-authored by domain specialists.

We think the perceived gap is not a real problem of the field. It is a scientific world, and in science the goal is not to produce working products for the market but to research new areas and gain knowledge.

3   New challenges

The field is also suffering from a lack of grand challenges. Like the search for the *unified theory* is a driving force within the physics field, the visualization field need great goals to strive for. Lorensen [1] suggested two things: The first being the production of patient-specific illustrations of anatomy that can rival with those of expert medical illustrators. The second suggestion done by Lorensen is a complete simulation of the human body including multi-scale information and simulations from the organ, the cell, and all the way up to the gene.

We feel that looking for grand challenges is a

---

[2]One that should be taken in parallel

good way to motivate people, but it might also lead to tunnel vision, neglecting other areas of the field.

4   Judging added value

In an unrelated but equally important paper by Van Wijk [2], a method is proposed for determining the gain of knowledge of a visualization system. His way of giving the field a new jolt of energy is by making the field more economically aware.

The field can benefit from a general model to assess the usefulness of visualization models in order to give the field more structure, to better judge whether a model can be successful. Such a method is currently lacking.

Van Wijk explains a simple, coarse system to give a model a certain qualification. It exists out of weighing attributes like the costs, a certain measurement for the gained amount of knowledge and the size of the user base. The costs are a summation of initial development costs, initial cost per user, initial costs per session and the investment it takes to modify parameters at runtime. His model shows that a great visualization method should be used by many people, who use it routinely to obtain highly valuable knowledge, without having to spend time and money on hardware, software and effort.

Measuring this gained knowledge is not a simple task. In the ideal case it should not matter who judges a model, and the analysis should be consistent over all observers. Unfortunately, this is not possible with examining the visualization methods: the knowledge gain depends heavily on the *a priori* knowledge and the perceptual skills of the observer and other factors like the type of hardware that is used. Therefore

the model as proposed by Van Wijk is far from complete, and the field should try to expand on it.

A major factor in the model is this gained insight in the data. As Van Wijk [2] states, the goal of visualization is to inspect complex data where there are no alternative methods to inspect this data. But it should be noted that visualization does not always provide a gain in knowledge. It is possible for a visualization to be misleading, giving rise to incorrect hypotheses and conclusions. This will be expressed as a negative term. Papers about the pittfalls of visualization are scarce and the field would benefit from discussing failures.

To us, Van Wijk's model seems too simple to be practical. The variables are too abstract be to determined consistently and it requires knowledge beyond what is normally available, but it is a start. The field should do more research in this area, where contact with other fields like psychology is needed to bring about better versions of the model.

4.1 An example

Let us try to use Van Wijk's model to evaluate a real-life visualization. Wills [3] has created a visualization for very big graphs. That is, graphs with so many nodes that they cannot be viewed all at once on a reasonable, modern day computer. A particular version of this visualization is used to visualize websites. Websites and the relations they have with other sites on a network can grow incredibly large. The program puts all the "nodes" in a circle, with the node they are linked to in the middle. These nodes exist out of links, images, pages, et cetera. Nodes can be connected through lines. Users can highlight certain node types and iteratively go deeper into this structure by selecting cer-

tain parts of the graph which can be zoomed in on. Looking at the website graph, we will have certainly gained some knowledge. But how do we express how much knowledge we've gained? Furthermore, not all the knowledge we gain is useful. Do we differentiate between useful and useless knowledge? Of course, and this is something Van Wijk himself says, this value is subjective and will differ from person to person. The webmaster of the site will certainly learn more by looking at the graph than a random person would. Even if we found some sort of measure, it would be hard to account for the knowledge we've gained subconsciously.

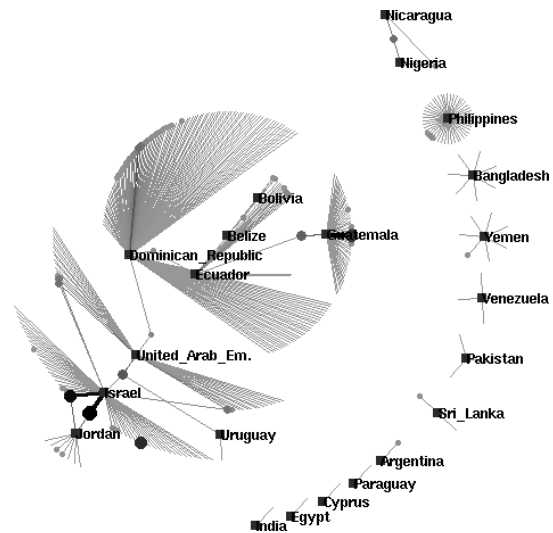Fig. 1: A graph produced by Wills

Part of his economic formula suffers far less from the quantification issue, and that is the cost formula. The costs in our example could be expressed in time for example which could then be converted into a measure of money. The initial cost per user for our example would be the time spent reading the manual and experimenting with the program until the user understands

all its features expressed in seconds or minutes. The profit function uses the acquired knowledge as a variable, thus suffering from quantification issues.

5 Communication

We now come to our third author, and with that our third subject. When two people talk about something, it is essential that they give the same meaning to words. This, of course, sounds obvious as a discussion is not possible without this basic assumption. In normal, everyday conversations this assumption does not pose any problems, as most of the words we use have a common agreed upon meaning that has been instilled since childhood.

This changes however, when considering the language used between people regarding a highly specialized subject. Many of the words used in such environments have been created specifically for that subject. Consider research groups working within the same field. If they wish to exchange data, the assumption of a common meaning is paramount for the correct interpretation of this data. In many fields this agreed meaning is firmly present in a formal way. Duke et al. [4] believe visualization might benefit from a more rigorous approach towards this in order to help progression in the field.

5.1 Communication levels

In increasing formality, the agreed meaning is usually organized in the following three groups or levels: terminology, taxonomy and ontology. Terminology can be seen as jargon. It is an informal way to describe and express concepts. The meaning of such things is properly defined

within the work that uses or introduces it, but not beyond that.

In taxonomy concepts are grouped or organized in some way. The concepts themselves are still defined in the informal way of the terminology level. This means that while there is a consensus within a certain taxonomy, it might still prove difficult to compare different taxonomies. The basis, or terminology, which these taxonomies are formed upon might not have the agreed meaning needed for comparison.

Finally there is ontology. Ontology[3] is a data model that represents a domain, in this case visualization. It is used to reason about objects within that domain and the relations between them. Thus it is a formal description with pre-agreed meanings. Duke et al. reason that the visualization field is currently mostly defined on the terminology level, with some work having been done on the taxonomy level.

5.2 Why ontology?

Duke et al. [4] also bring forth several arguments why the creation of an ontology is important for the visualization field. One of these is collaboration: communication between researchers would be made more efficient when a greater level of formalization is in place. Also, interaction between separate visualization systems would need such formalization. Sharable data sets is an example of such a need. Secondly, education is required in order to keep the field alive and attract new people. The books currently available on the subject are aimed at a very specific audience and are not suited for beginners. Also, because of a lack of agreed meaning, understanding a new resource requires studying its termi-

---

[3]A concept that has its origin in philosophy

nology because it might be different from your own interpretation.

There is also the matter of preservation. Obtained results should be repeatable by others. This can be made much easier through a more formal description of the process that led to the result.

These arguments seem rather logical. What Duke et al. are saying all boils down to the same thing: we need a method to unambiguously document methods and results.

5.3 Efforts made

A way to create a more formal way of talking about concepts is to restrict the language used to define them. For example, one attempt towards this is the E- and O-notations developed by Brodlie [5] which can be used to model structures of data fields and data representations. We will not discuss this method in depth as this is beyond the scope of this paper. In short, E-notations define the relationship between the data and the underlying field and O-notations formalizes the relationship between representation and data. Unfortunately, defining formalization requires informal text to establish its meaning. Also, while such a notation could formalize a small set of concepts, it would be difficult to link all these notations together into a cohesive whole.

Another thing that is aiding the establishment of a more formal approach are the tools created in the field. As certain tools become more popular within the community, the formal models which they use become more popular and widespread as well. They can even lead to standards. The World Wide Web Consortium [7] might also provide a possible solution in the form of their Resource Description Framework and Web Ontology Language. These

are models and languages which provide concepts and relationship "tools" to describe the meaning of entities within a domain. These tools are slowly gaining popularity. Something similar could be beneficial to the field.

5.4 Ontology acceptance

Duke et al. [4] believe that now is the right time to start work on an visualization ontology. They think that with the maturity and developments within the field an ontology would help with the creation and documentation of future projects. Their arguments seem sound and there is much to be said for the idea. But ultimately, the community itself will decide the future of such a large scale enterprise.

6 Discussion

We feel that the view presented by the researchers is a tad pessimistic. Concerning the gap between customers' and researchers, we feel the point is not valid. The scientific field is meant as a playground for new ideas and they should not have to concern themselves with specific practical applications. The sensed lack of interesting subject material leading to the decline of the field is — in our opinion — exaggerated. Many fields within computer science thrive on incremental advancement. This is not necessarily a bad thing.

Van Wijk's model for assessing the usefulness of visualization algorithms is overly simplistic, but the awareness he creates is a positive thing. We hope more researchers will follow him in this area.

Concerning the creation of an visualization ontology, we think it would be a blessing for the field. A more consistent approach would lead

to better literature and more collaboration with other fields. This might lead to new ideas and might bring more people to the field.

All in all, the field is nowhere near dead, and we see a bright future for visualization. Every year more raw data is being gathered than in previous years, and visualization can be a very useful tool in analyzing this endless stream of data.

References

[1] Bill Lorensen *On the Death of Visualization* Position Papers NIH/NSF Proc. Fall 2004 Workshop Visualization Research Challenges, 2004.

[2] Jarke J. van Wijk *The Value of Visualization* C. Silva, E. Groeller, H. Rushmeier (eds.), Proc. IEEE Visualization 2005, p. 79-86, 2005.

[3] Graham J. Wills *"NicheWorks Interactive Visualization of Very Large Graphs"* Journal of Computational and Graphical Statistics, Volume 8, Number 2, Pages 190-212

[4] Duke, D.J., Brodlie, K.W., Duce, D.A. and Herman, I. *Do you see what I mean?* IEEE Computer Graphics and Applications, 25 (3). pp. 6-9. ISSN 0272-1716

[5] K. W. Brodlie, *"A classification Scheme for Scientific Visualization"* , 1993

[6] Vital Images http://www.vitalimages.com/

[7] World Wide Web Consortium http://www.w3.org/

[8] Thomas Ball, Stephen G. Eick *Software Visualization in the Large* IEEE Computer, Vol. 29, No.4, April 1996. pp. 33-43.

Structural Similarity in Image Quality Assessment

Frans Delvigne

f.m.delvigne@student.rug.nl

Abstract. The Human Visual System is ideally suited to recognize structures in images and the quality of images that are distorted but retain their structural information is usually judged as acceptable by humans. Why is it then that current quality assessment algorithms do not take this into account? Quality is usually measured by calculating the mean squared error between the original image and the distorted image. This can be an accurate objective measure of quality, but has no real relevance when the image is intended to be viewed by humans.
In this paper a method will be presented which uses a structural similarity index to measure the quality of a (distorted) image.

Keywords: image quality assessment, full-reference, human visual system (HVS), sctructural similarity (SSIM).

Introduction

With the widespread use of digital images and the various ways in which they can be distorted: during acquisition, processing, compression, transmission and reproduction, it is important to be able to measure the quality of the image.

If there is an efficient way to objectively measure image quality, the quality can be monitored and adjusted in real-time. For instance, a network digital video server can examine the quality of transmitted video data to control and allocate streaming resources.

It can also be used to optimize algorithms and parameters of image processing systems. Lastly, it can be used to benchmark different image processing systems or algorithms.

When an image is intended to be viewed by humans the best way to assess the quality is by looking at the image and grading it subjectively. However to do this with all images would be costly and time-consuming. It is therefore necessary to come up with satisfactory ways in which to objectively quantify the quality of an image as it would be interpreted by the Human Visual System (HVS).

In image quality assessment there are three different categories based on whether or not the original distortion-free image is available for comparison.
-   Full-reference (FR) metrics are used when the complete original image is known at the point where the quality is measured.

- Reduced-reference (RR) metrics are used when only a feature set of the original image is available.
- No-reference (NR) (or blind) metrics work when the original image is not available.

Obviously full-reference metrics are preferable since the differences in the copy can be compared with the original undistorted image. However it is not always possible or practical to have the original image available at the evaluating point, for instance in broadcast TV or video on demand.

It would be useful to use no-reference methods, since it would then be possible to use this quality assessment in a wide variety of applications without having to find a way to send the original image along. However, to provide accurate quality assessment with no-reference is very difficult and so far a largely unexplored territory. At this point no-reference methods are only used in systems where the type of expected distortion is known beforehand.

The third option would be to send a set of representative feature data of the original image to the evaluation point. This feature data would provide just enough information to make a decent assessment possible without the added expense and complexity of sending the original image along.

From Error Visibility to Structural Similarity

The most used method of image quality assessment is based on mean squared error (MSE), which is computed by taking the average between the squared intensity differences of distorted and reference image pixels. This method is attractive as it is easy to calculate and has a clear physical meaning. However it is not very well-matched to the perceived visual quality by the Human Visual System.

In recent years there have been proposals to modify the mean squared error measure by giving weights to the errors based on their visibility. This error-sensitivity approach has a few limitations and difficulties, which will be discussed in briefly in this paper in section 2.1.

In section 2.1 a method for quality assessment will be presented that was proposed in the paper *Image Quality Assessment: From Error Visibility to Structural Similarity* by Z. Wang et al.[1].

The Human Visual System is ideally suited to recognize structural information in images. This approach therefore gives a good representation of how humans will perceive the quality of the image. By measuring the Structural SIMilarity (SSIM) and comparing local patterns of pixel intensities an assessment can be made of the quality that gives a better indication of the perceived image quality.
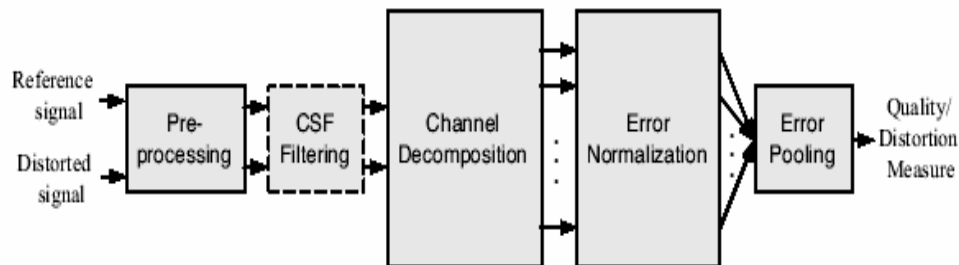
The Error Sensitivity Approach

Fig. 1. A prototypical quality assessment system based on error sensitivity. Note that the CSF feature can be implemented either as a separate stage or within Error Normalization.

When evaluating an image it can be regarded as the sum of an undistorted reference signal and an error signal, shown in figure 1.

- Pre-processing: This step performs several operations to eliminate known distortions from the images that are being compared. Things like scaling and aligning the images are done here.
- CSF-Filtering: The Contrast Sensitivity Function describes how sensitive the Human Visual System is to certain spatial and temporal frequencies. Weights can be added based on this sensitivity to increase the importance of those features in describing the perceived quality.
- Channel Decomposition: The images are separated into sub bands that are selective for spatial and temporal frequency as well as orientation.
- Error Normalization: This step calculates the difference between the decomposed reference and distorted images in each channel. Then this difference is normalized using a certain masking model.
- Error Pooling: This stage combines the calculated errors over the spatial extent of the image and all the channels into a single value. This is usually done using a Minkowski norm:

$$E(\{e_{l,k}\}) = \left(\sum_l \sum_k |e_{l,k}|^\beta\right)^{1/\beta} \tag{1}$$

Where $e_{l,k}$ is the normalized error of the k-th coefficient in the l-th channel and β is a constant, typically chosen to lie between 1 and 4.

Limitations of the Error Sensivity Approach

Most early models are based on linear or quasilinear operators, even though the Human Visual System is complex and highly nonlinear. In order to make use of these linear operators strong assumptions and generalizations need to be made. This leads to problems that impede quality assessment using these models.

- Quality definition problem: Some errors, while highly visible, are not considered by most humans as constituting a loss of quality, for instance multiplying the image intensities by a global scale factor.
- Natural image complexity problem: Most error sensitivity models are tested using experiments with simple geometric shapes such as spots and bars. But

can these models evaluate the interactions between the tens or hundreds of shapes that are in real-world images?

- Decorrelation problem: Most models make use of a Minkowski metric (1) for pooling errors. This assumes that errors at different locations are statistically independent. However this is not the case for linear channel decomposition methods such as the wavelet transform.

Cognitive interaction problem: Cognitive understanding and interactive visual processing (e.g. eye movements) influence the perceived image quality, but these things are difficult to quantify and therefore are left out of most models.

Structural Similarity (SSIM)

Since natural image signals are highly structured their pixels exhibit strong dependencies. These dependencies carry important information about the structure of the objects in the image.

Whereas the error-sensitivity approach estimates perceived *errors*, the new method in [1] looks at perceived changes in *structural information*. This can be seen in Figure 2. All distorted images have the same Mean Squared Error with respect to the original (2a) even though the differences in quality can easily be seen. This can be explained by the new method since in the first two copies the structural information is mostly preserved.

Underneath the images the Mean Squared Error (MSE) with respect to the original is show, as well as the Measure of Structural SIMilarity (MSSIM). well.

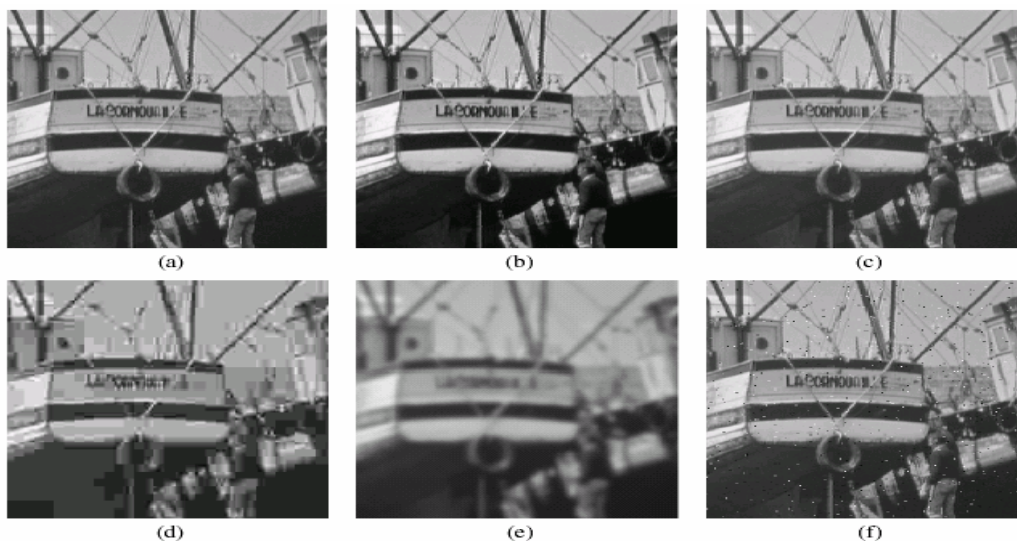**Fig. 2.** Comparison om images with MSE=210 (a) Original image (b) contrast stretched image, MSSIM=0.9168 (c) mean-shifted image, MSSIM=0.9900 (d) jpeg compressed image, MSSIM=0.6949 (e) blurred image, MSSIM=0.7052 (f) speckle noise, MSSIM=0.7748

These results show that, in these cases at least, the structural similarity index gives a better representation of perceived image quality than calculating the mean squared

error. In figures 2a and 2b the MSSIM index is close to 1, which means that the structural information from the original is almost perfectly preserved. This corresponds to what humans would perceive as good image quality. In the other images, the MSSIM index is much lower and this difference can easily be seen as

Calculating the Structural Similarity Index

This method makes use of a structural similarity index, this value is calculated according to the diagram show below. Here Signal x is the original undistorted image and Signal y is the image whose quality needs to be assessed.
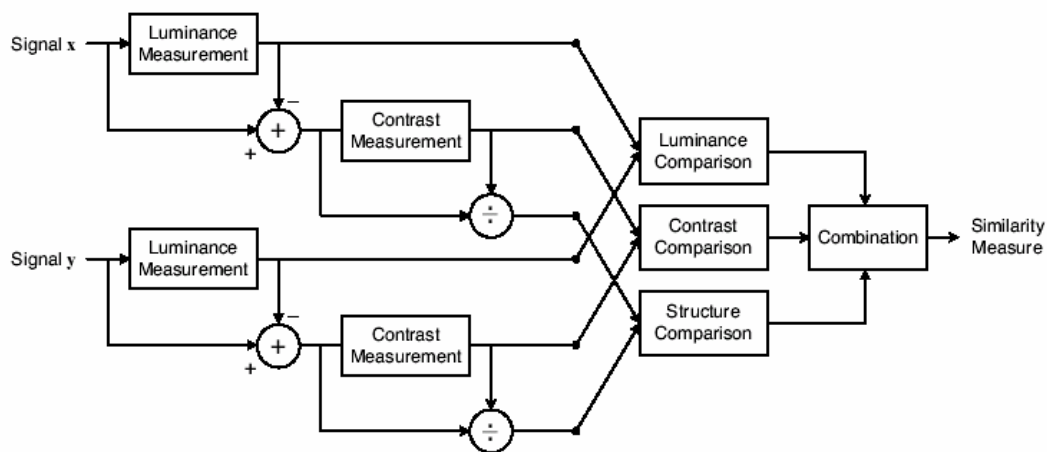
Fig. 3. Diagram of the Sctuctural Similarity measurement system

From the above diagram you can see that the structural similarity is calculated using separate calculations for the luminance, contrast and structure. These values are then multiplied to give us the final Structural Similarity Index, like so:

$$\text{SSIM}(x,y) = [l(x,y)]^{\alpha} \cdot [c(x,y)]^{\beta} \cdot [s(x,y)]^{\gamma}$$

The individual formulas look like this:
Luminance:

$$l(x,y) = 2\mu_x\mu_y + C_1 / \mu_x^2 + \mu_y^2 + C_1$$

Contrast:

$$c(x,y) = 2\sigma_x\sigma_y + C_2 / \sigma_x^2 + \sigma_y^2 + C_2$$

Structure :

$$s(x,y) = \sigma_{xy} + C_3 / \sigma_x\sigma_y + C_3$$

The variables α, β and γ are simply parameters to adjust the relative importance of the three components. To get the values calculated in [1], these parameters were set to 1.

Discussion of Structural Similarity for Image Quality Assessment

In this paper the traditional method of image quality assessment was explained briefly, including its limitations. In order to counteract these limitations a method using structural similarity was introduced and a few experimental results were shown using a Structural Similarity index. Early testing show that this method is well suited to measuring the quality of an image as it is perceived by the Human Visual System.

However, this method is still being further developed and some issues with the Structural Similarity index still need to be researched. Firstly the optimization of the Structural Similarity index for different image processing algorithms needs to be studied. It could be used for instance for rate-distortion optimization when designing image compression algorithms. This is not as easy as mean squared error, since the structural similarity function is mathematically more cumbersome. Secondly, since the structural similarity index is a symmetric measure it could be used to compare any two signals and not just images.

The Structural Similarity index is a technique that describes the perceived image quality more accurately than some other methods. However, it is just one possible approach in the field of structural similarity. With more research into structural similarity it is possible that other approaches may emerge that are very different from the indexing algorithm presented in [1].

References

[1] "*Image Quality Assessment: From Error Visibility to Structural Similarity*", Z. Wang, A.C. Bovik, H.R. Sheikh, E.P. Simoncelli

Assembling Protocols for Sharing Secrets

Jasper van de Gronde   Twan van Laarhoven

21st January 2007

Abstract

This paper deals with protocols for proving possession of secrets without giving them away. Although protocols for this problem come in all shapes and sizes, they often use similar methods. These similarities are unfortunately often ignored, which makes it hard to directly compare protocols. In contrast, we will break down the protocols to their basic building blocks. We present these building blocks as a base protocol and a number of transformations that can be applied to derive more complex protocols (with more desirable properties). An example is given of how the building blocks can be used to construct one of the protocols given by Teepe [1].

1. Introduction

Suppose there are two parties, lets call them Alice and Bob. They could be people, but they could also be computers on a network. The parties want to exchange knowledge of some secret, but they do not trust each other. For example Alice could ask Bob: "Do you know the combination of the safe?". Just answering yes or no is not enough, since Alice has no way of knowing whether Bob is truthful. On the other hand, Bob can not just prove his knowledge by giving the combination; Alice might not actually know the combination and could be fishing for information.

Alice wants to know whether Bob knows the combination of the safe, so we are interested in protocols that can determine whether a certain secret (the combination of the safe) is known to Bob. And it should work without letting Bob know the combination of the safe unless he already knew it.

There are several protocols to answer such questions, ranging from informal 'games' [2] to formal cryptographically sound protocols [1]. These protocols differ not only in how thoroughly they are described but also in the exact definition of the problem. In fact there is not a single unique problem being solved at all. In the next paragraph we will investigate these differences.

2. The problems

We will look at several variations of the basic problem described above. First of all, instead of wanting to know whether they both have the same value in mind, Alice and Bob might also want to perform more general comparisons without revealing their actual values. For example, Alice may want to know whether Bob is willing to sell his car for a price that is within her budget. Neither Alice nor Bob will want to reveal any exact prices however, as that might give them a disadvantage in the negotiations. In general we can say that Alice wants to know whether the two values of the secrets, if there are any, satisfy some relation R.

Usually you want to limit yourself to just a small subset of all the knowledge, as Alice and Bob will usually have much more knowledge than what is relevant to the situation. In many situations Alice will be able to simply tell Bob what she is interested in, but this is not always the case as it might give Bob some knowledge of the secret. For example, suppose Alice wants to know whether Bob knows about some fact or event, it can be quite tricky to ask anything about it without giving at least something away.

We can also distinguish between methods that guarantee that both parties will know the correct outcome and methods in which only one of the parties will know the correct outcome. In some cases the latter may be all that is needed, for example if Alice wants to talk about her secret with Bob, but Bob is not necessarily interested in what Alice knows.

Protocols can also be distinguished by the amount of certainty they give. Some protocols will give a solution with a very high probability, others will provide a definite answer. Protocols that give a solution with a very high probability are often easier or provide other advantages, but might not always be appropriate.

Finally, there are quite a lot of methods which use a more or less trusted third party to some extent. If a completely trustworthy third party is available the problem can be solved quite trivially. Both parties would simply tell that third party their secrets and be told the outcome of the comparison. But in general a third party can be used to prevent Alice or Bob from gaining knowledge they should not gain. For example the safe itself is a third party which can verify that person knows the combination.

3. A Model

We model the knowledge of the parties as sets of 'secrets'. A secret can be, for example, the combination of the safe or the price of a product. We call such a set of secrets a *knowledge base*, and we talk about the *knowledge* of a party. For example Alice's knowledge of the combination of the safe could be {1234}. Her entire knowledge of course contains much more, we model it as a set of tuples:

$$K_a = \{(\text{combination of the safe}, 1234), (\text{my name}, \text{Alice}), \ldots\}$$

Now the core of all the problems is whether one of Alice's secrets has a certain relation R with one of Bob's secrets. In other words, the outcome o of the comparison will be:

$$o = \exists s \in K_a \, \exists t \in K_b : R(s, t)$$

Usually we limit ourselves to just a small subset of all knowledge, for example only the combination of the safe.

4. Protocols

To be able to understand the different protocols and why they have their specific properties we want to break them down. We consider all protocols as a basic protocol to which some *transformations* are applied.

It turns out that the core of most of the protocols is the same, since all methods ultimately rely on comparing the knowledge. The difference therefore lies in the transformations.

To be able to reason about protocols we describe them by a *protocol signature*:
Protocol [x]
This signature refers to the class of protocols with properties x, as described in table 1. For example:
Protocol [compare: $=$; domain: $\mathbb{N}$; A learns: $\{o\}$]
refers to protocols where the knowledge consists of natural numbers, which are compared for equality. Only A learns the outcome o of the comparison.

Protocol transformations operate on protocol with a specific signature, and transform it into a new protocol with a different signature. We use the notation:
Protocol [x] $\rightarrow$ Protocol [y]
For example:
Protocol [x ; A learns: $\{o\}$] $\rightarrow$ Protocol [x ; B learns: $\{o\}$]
Transforms a protocol in which A learns the outcome into a protocol where instead B learns the outcome. All other properties x are unchanged.

compare: R	Knowledge is compared using the relation R.
domain: D	The values that are compared are in the domain D, relation R has the type $\mathcal{P}(D \times D)$.
complexity: $\mathcal{O}(x)$	The number of items that must be transferred is of the order $\mathcal{O}(x)$.
A learns: x	What does A learn? For example A learns: $\{o\}$ means that party A learns the outcome o of the comparison and nothing else.
extra: x	Any extra requirements of the protocol, for example that a third party is needed.

Table 1: Different properties with a shorthand notation and a description

4.1. Base protocol

Protocol [compare: any R ; domain: $\text{dom}(R)$; complexity: $\mathcal{O}(\#K_b)$; A learns: $\{K_b, o\}$; B learns: $\emptyset$]

The most basic method of comparison would be for Bob to simply tell Alice his secret(s):

1. B sends K_b to A.
2. A computes $o = \exists s \in K_a \, \exists t \in K_b : R(s, t)$

Obviously this protocol fails to fulfill even the basic requirement that A should not learn any of B's secrets if A did not know them already.

4.2. Security transformations

The base protocol does not satisfy the requirements of the problem at hand, A will learn secrets it is not supposed to know. It is therefore useful to look at a number of ways to modify the protocol so the security is improved.

Third party

Protocol [x ; A learns: y] $\rightarrow$ Protocol [x ; A learns: $\{o\}$; T learns: $y \cup \{K_b\}$; extra: 3$^{\text{rd}}$ party]

If a third party (T) is available that third party can be used to prevent A from learning too much as follows:

1. B sends K_b to T.
2. A and T perform the original protocol with T taking on the role of A and A the role of B, so at the end T will know the outcome o.
3. T sends the outcome to A.

Scrambling

Protocol [x ; X learns: $\{K_i\}$] $\rightarrow$ Protocol [x ; X learns: $\{f(K_i)\}$; A and B learn: $\{f\}$]

To prevent a party from learning any secrets of the other party the secrets can be scrambled first. The parties agree on a scrambling function f. This function is applied to all items in their knowledge base before sending it. Note that we use $f(K)$ to denote $\{f(k) \,|\, k \in K\}$. The new protocol becomes:

1. A agrees with B upon a scrambling function f.
2. A and B perform the original protocol with $f(K_a)$ and $f(K_b)$ instead of the normal knowledge bases.

f must have the following properties:

- f should be a one-to-one mapping. If it is not then it is possible that there is a false positive match. Alternatively it can be sufficient if the probability of collisions is very small.

- If the comparison relation R is used, then $R(f(s), f(t)) \Leftrightarrow R(s, t)$. In other words, f must preserve the comparison. For equality this is always the case, but when comparing using for instance $<$, f must be an increasing function.
- Since both A and B will learn f, either:
  - A must not learn K_b' and B must not learn K_A',
  - or knowing f and $f(s)$ must not lead to knowledge of s.

  The latter is possible by using a cryptographic hash function [3].
- Beforehand parties should not know or be able to compute $f(s)$ without also knowing s. This is often not the case, since $f(s)$ could be learned in a previous run of the protocol. f should therefore be unique, or be made unique by including for example the names of A and B or a random value.

4.3. Performance transformations

The base protocol requires that the entire knowledge base of B be sent to A. This is of course not practical. There are ways to drastically improve the performance.

Identifying question

Protocol [x ; domain: V] $\rightarrow$ Protocol [x ; domain: $N \times V$; complexity: $\mathcal{O}(1)$; B learns: $\{n\}$]

If the secret could somehow be identified first, then only the secret itself has to be transferred. To identify the secret, A could just name it, assuming that the name n would not tell B anything confidential about the secret. For example, when trying to confirm that Bob knows the combination of the safe it would usually do no harm to tell Bob that you are looking for the combination.

1. A sends n to T.
2. A and B perform the original protocol with $K_a' = \{v \mid (n, v) \in K_a\}$ and $K_b' = \{v \mid (n, v) \in K_b\}$.

Identifying predicate

Protocol [x ; complexity: $\mathcal{O}(c)$] $\rightarrow$ Protocol [x ; complexity: $\mathcal{O}(c/2^{\#P})$; B learns: $\{P\}$]

Another way to reduce the amount of information sent is for A to give B a predicate P, and say "The secret I am looking for satisfies P". The protocol now becomes:

1. A determines a predicate P. P must have the property that $\forall s \in K_a \, \forall t : R(s, t) \Rightarrow P(t)$. In other words, the secret A is looking for must indeed satisfy P.
2. A sends P to B.
3. A and B perform the original protocol with $K_a = \{k \mid k \in K_a \land P(k)\}$ and similarly for K_b.

The reduction of the number of items to be sent depends on the information content $\#P$ of P. So if $\#P$ is 1 then only half of the knowledge will satisfy it, the other half will not have to be sent.

Since the predicate P will be revealed to B, he should not be able to learn the secret from it. One way to ensure this is to use a cryptographic hash function [3]. When comparing secrets for equality, K_a will usually contain just one item we are actually interested in. Calling this item s, the predicate can be $P(t) := H(s) = H(t)$. Alternatively when the knowledge takes the form of tuples, a hash of the name n can be used instead of the name itself in *identifying question*.

One problem with hash functions is that there can be collisions, two different names which map to the same hash. To reduce that problem A can generate a random challenge C that further reduces the size of the knowledge base, and send it along with the hashed secret (see [1]).

Another problem is that knowing $H(s)$ is not completely useless to B, he can use it to determine if the same secret is asked for again. To prevent this a random value V can be added, so the predicate now becomes $P(t) := H(s + V) = H(t + V)$. Since the random values are different each time, the predicates can not be compared.

4.4. Symmetry transformations

In the base protocol only A learns the outcome. B might need to know the outcome too.

Broadcast

Protocol [x ; y = [extra: 3rd party ; T learns: $\{o\}$]] $\rightarrow$ Protocol [x ; y ; A and/or B learn: $\{o\}$]

In a protocol where a trusted third party learns the outcome it can simply broadcast it to A and/or B so they know the outcome as well:

1. Perform the original protocol.
2. T sends o to A and/or B.

Perform twice

Protocol [x ; A learns: $\{o\}$] $\rightarrow$ Protocol [x ; A and B learn: $\{o\}$]

If there is no third party it might still be desirable for B to learn the outcome of the comparison as well. The solution is simple:

1. Perform the original protocol.
2. Perform the protocol again, with the roles of A and B reversed.

Note that it is always possible for A to stop after the first step, or for the parties to lie in one of the steps. This could result in one of the parties (incorrectly) having $o = \mathsf{false}$.

5. Example

As an example we will reproduce the symmetric protocol from [1]. In this protocol A wants to verify that B possesses some secret s, both parties will learn the outcome.

1. We start with the base protocol. As we are only looking at the specific secret s, the protocol can use the following relation:
   $R(t, u) \ := \ t = s \wedge u = s$
   We will consider the items s, t and u to be strings, this means the basic protocol has the signature:
       Protocol [compare: R ; domain: Strings ;
              complexity: $\mathcal{O}(\#K_b)$; A learns: $\{K_b, o\}$; B learns: $\emptyset$]
   For brevity we will leave out the compare and domain part of the signature from now on.

2. Of course A should not learn K_b. We can prevent this using the *scramble* transform. Since A learns the scramble function as well we have to ensure that it is not invertible. A good approach is to use a cryptographic hash function, combined with the identity of B.
       Protocol [complexity: $\mathcal{O}(\#K_b)$; A learns: $\{f(K_b), f, o\}$; B learns: $\{f\}$]
   Since f is a known hash function, and it is infeasible to invert it, we can say that $f(K_b)$ and f are not useful information, so the signature simplifies to:
       Protocol [complexity: $\mathcal{O}(\#K_b)$; A learns: $\{o\}$; B learns: $\emptyset$]

3. Since we want B to know the outcome we use the *perform twice* transformation, giving:
       Protocol [complexity: $\mathcal{O}(1)$; A learns: $\{o\}$; B learns: $\{o\}$]

4. With this protocol B must send a value to A for each item in his entire knowledge base, this is a huge amount of information. We can greatly improve the efficiency by using an *identifying predicate*. Since we do not want to reveal anything we will use a hash function H together with a challenge C and random value V. Assuming the hash has length $\mathcal{O}(\log(\#K_b))$ (which is a reasonable assumption), this leads to a protocol with the signature:
       Protocol [complexity: $\mathcal{O}(1)$; A learns: $\{o\}$; B learns: $\{H(s + V), V, C\}$]
   Again, since it is infeasible to invert H all of the information B learns is useless to him:
       Protocol [complexity: $\mathcal{O}(1)$; A learns: $\{o\}$; B learns: $\emptyset$]

The protocol arrived at in step 4 is:

    I. A computes a random value V and challenge C.

   II. A sends the predicate P using $H(s + V)$, V and C to B.

  III. A and B perform the protocol of step 3 with the knowledge base filtered by P:

      (a) Perform the protocol of step 2, so the base protocol with $f(K_a)$ and $f(K_b)$:

         i. B sends the filtered $f(K_b)$ to A.

        ii. A computes the outcome o.

      (b) Perform the protocol of step 2 again, with the roles of A and B reversed:

         i. A sends the filtered $f(K_a)$ to B.

        ii. B computes the outcome o.

If you flatten this list, it corresponds directly to the protocol given by Teepe [1].

6.   Discussion/Future work

By combining different transformations many different protocols can be created. Not all of them are practical or even correct. In particular it is possible to create algorithms that give away the secret to the other party. Fortunately the protocol signatures give a reasonable indication when this is the case, since they will say: "B learns: *secrets*".

In this paper we give only one base protocol and six transformations. Using these we can already recreate the three protocols without encryption from [1]. However, the work by Fagin et al, [2], contains a wider variety of protocols. Among them are protocols based on different primitives, such as 'oblivious transfer' [4] and many physical objects. To describe these protocols in our framework at least a few new transformations would be needed and perhaps even a different base protocol.

Also note that we only give informal descriptions of the protocols and protocol transformations. In a future work it would be interesting to see whether these could be formalized further. The properties of the protocols could then be analyzed thoroughly. In particular it would be good to verify that the protocols do indeed work as intended and that the different transformations we described are not prone to any problematic interactions.

Having formalized the methods described in this paper it might even be possible to construct a framework to allow designing protocols semi-automatically. Such a framework would make it relatively easy to design protocols that are both correct and have the desired properties by combining different transformations. To ensure correctness this would of course first require a thorough analysis of the method.

An interesting open question is whether the overview given in this paper is in any way complete. It is possible to use the transformations to create all sorts of protocols with different combinations of properties. However, there are combinations of properties conceivable that are not possible to create with the transformations described in this paper. To what extent this is due only to missing transformations or protocols we do not know.

References

[1] W. Teepe. Proving possession of arbitrary secrets while not giving them away. *Synthese - Knowledge, Rationality and Action*, 149(2):409–443, 2006.

[2] Ronald Fagin, Moni Naor, and Peter Winkler. Comparing information without leaking it. *Communications of the ACM*, 39(5):77–85, 1996.

[3] Shahram Bakhtiari, Reihaneh Safavi-Naini, and Josef Pieprzyk. Cryptographic hash functions: A survey. Technical Report 95-09, Department of Computer Science, University of Wollongong, July 1995.

[4] S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proof systems. *SIAM J. Comput.*, 18(1):186–208, 1989.

Verifying knowledge without revealing it

Ando Emerencia& Eamon Nerbonne

Rijksuniversiteit Groningen
A.C.Emerencia@student.rug.nl, eamon.nerbonne@gmail.com
http://eamon.nerbonne.org/

Abstract. Dealing in secrets is a delicate business: how do people discuss confidential matters without inadvertently revealing information to each other? For instance, when two managers each receive the same complaint, they want to know whether it originates from only one person - but without revealing *their* source if the complaints came from different people. These managers are facing an instance of a wider class of problems known as *Comparing Information Without Leaking It* (CIWLI). Solutions to CIWLI are one way of comparing knowledge (such as the sources of complaints), without revealing this knowledge. This paper discusses two variants of CIWLI, "with reference" and "without reference"; the former assumes that both participants know which secret is being referred to, and the latter in which such an a priori agreement does not exist. The language of Quadratic Non-Residuosity (QNR), a superficially unrelated problem, contains only numbers that are not quadratic residues. The relation to CIWLI (and its protocols) is a two-person protocol, which lets one person (the verifier) verify whether a given term is a member of the language by asking the other (the prover) without revealing the term itself. This relation highlights the essence of the CIWLI problem: that of verifying knowledge without revealing it.

1 Introduction

Consider a person that wishes to find out if another person also knows some secret without either person revealing it if it is not shared. In a real life situation, such a scenario will result in a game of "hints": he asks the other to verify little details that don't give away the secret, thus building confidence that they really do share this secret before openly discussing it. Unfortunately, in doing so, these people *are* revealing parts of their secret in the process - small, less relevant details, but parts of the secret nonetheless. This problem is known as Comparing Information Without Leaking It: CIWLI in short.

We differentiate between two classes of CIWLI problems using the terminology introduced in [TEE06]. In CIWLI "with reference" it is clear to both persons which secret is to be compared, as there is some common reference such as a question for which both persons have a secret answer. In CIWLI "without reference" there is no common question, but rather a unilateral reference: one person would like to know if the other person knows the secret he does. In both cases, no knowledge may be leaked.

Is there a protocol to verify knowledge without revealing it? In search of such a protocol, we also examine the language of Quadratic Non-Residuosity (QNR), a seemingly unrelated issue. However, a protocol exists [GMR85] which allows a verifier to determine whether a term is a member of this language by communicating with a prover capable of computing such membership, while leaking neither the term nor its membership to the prover. This protocol solves the very CIWLI-like question "Do you know if the following term is a member of the language of QNR?" - without leaking the term.

2 CIWLI "with reference"

Two persons are thinking of a secret, and would like to know whether they are thinking of the same secret, without revealing these secrets. Several solutions to this problem are given in [FNW96], but the focus in that paper lies on solving the problem using simple, everyday methods, and (mostly) without complex programming. These simple solutions share many crucial features. We partitioned some of the known solutions by the type of communication used in the protocol: *Third person*, *Shared data* and *Direct messages*, and discuss them without going into implementation details.

2.1 Assumptions

Two persons A and B wish to compare secrets, and can only communicate by means of sending direct messages, looking into shared data, or by sending messages to a trusted third person C. A, B, and C are distinct. They are consistent and accurate in their computations. If a person consistently shares a value different from the "real" value of their secret, the protocols will work, but the result will reflect the false value, and not the real value. These protocols only compare one value though, namely the one claimed. If one party consistently claims a particular value, then the "real" value is irrelevant. Specifically, it should not be possible to change the value you are comparing mid-protocol and thus conclude facts about multiple secrets by trial and error without rerunning the protocol with cooperation from the other person.. Clearly, it should not be possible to cause the other person to leak *any* information to you by violating the protocol (or being inconsistent or inaccurate), but conversely, assuming you *are* truthful and consistent, you should never leak information, and when the other partner is equally honest, the protocol should always terminate with the correct answer.

2.2 Third person

The first solution to CIWLI "with reference" uses a trusted intermediary C. A tells C secret S_A. B tells C secret S_B. Finally, C checks if S_A and S_B are equal and notifies A and B about this result.

In this most basic form, C finds out everything - the secrets of both A and B (so whether they are equal or not), even when these secrets differ. To restrict the information C receives about the secrets of A and B, some kind of encoding is needed for these secrets.

Random Permutation An alternative approach that uses such an encoding is Random Permutation [FNW96]. Here, A and B first agree on two random numbers which are effectively used as encryption keys. So in this case, C only finds out whether the (encrypted) secrets of A and B are equal, but cannot derive any information about the secrets other than that they are equal or not.

Random Rotation A slightly weaker version of this concept is dubbed Random Rotation [FNW96], this solution does not require a predefined set of keys, but here C not only finds out whether the secrets are equal, but also the `mod` between indexes in the array of possible values for the secrets, in the case that the secrets are not equal.

Permutation Composition Permutation Composition is even stronger than Random Permutation in the sense that C does not find out whether the two secrets are equal. In this protocol, A and B first agree on a set of lists of permutations of some set to encode their secrets. C then performs some computation on these permutations and gives an answer that is encoded in such a way that only A and B can derive from it whether their secrets were equal.

Hashing Another alternative we mention here is that of Hashing. Like in Random Permutation, C only finds out whether the secrets are equal or not, but in this case, no predefined keys are needed assuming that the hash-functions used by A and B are equal and hard to reverse.

2.3 Shared data

Where shared data is an available method for communication and is secure in the sense that it can only be accessed by A and B, and that a person cannot read from the shared data when it is supposed to write in it, and vice versa. This replaces the need for a third person, but the computations performed on the shared data will instead have to be performed by either A or B, so caution must be taken that they can not directly infer information about each other's secrets if those secrets are not equal.

Hashing For instance the solution of Hashing can also be implemented using shared data and is informally described in [FNW96] as Cups. Here the shared data is an array of the size of possible secrets with all values set to 0. A and B each increment the index of their secret, the shared array is then randomly permuted and searched for the first index that contains a non-zero value. If this value is 2 then the secrets are equal, if this value is 1 then the secrets are not equal. Note that after the array has been randomly permuted, no information about the

secrets of A and B can be revealed other than the fact that they are equal or not.

Deck of Cards A weaker alternative named Deck of Cards [FNW96] presents the method of having a shared array of size 26, where A increments the alphabetical indices of the characters appearing in the secret in the array and B decrements these indices. The array is then randomly permuted and checked for non-zero values. This method is weaker than Hashing because the number of non-zero indices in the array gives a measure of equality between the secrets. This method is also more error prone since prior agreements will have to be made about spelling as some special characters are excluded and since no differences are detected between a secret and its permutations.

2.4 Direct messages

There exist solutions for situations where we do not have or want a third person or shared data, by only letting A and B directly message each other. To not reveal any information about their secrets other than the fact whether they are equal or not, some kind of encoding is needed in the messages conveyed. Here we cannot, for instance, agree on a common set of keys because A and B will then be able to decrypt each other's messages and find out about each other's secrets. One-way encryption methods such as hashing solve this problem, but introduce a small error chance that the encrypted messages are equal while the unencrypted messages are not.

Envelopes The solution Envelopes [FNW96] describes the solution where A and B each create a sequence of pairs of random values. These sequences have the length of the secret represented in bits. A steps through his sequence, uses each bit of his secret to choose one value from each pair (the first value should the bit be 0, and the second should it be 1), adds all these together and shares this computed value. B does the same with B's sequence. A and B then use oblivious transfer[1] to exchange their sequences,

---

[1] Oblivious Transfer [RAB81]: in the variant "1 out of 2 oblivious transfer", which we use here, the receiver receives only one of the two messages sent by the sender, and the message received remains unknown to the sender.

choosing one value from each pair as they did for their own sequence. They can then check whether they found the same value as shared by the other.

This method however has a small error probability that the randomly chosen numbers were equal for differing indices. Furthermore it relies on a complexity theoretic assumption and on the fact the computational power of A and B is limited to probabilistic polynomial time. We also note that this approach has the danger that one of the persons might "walk away" after receiving the other's information, and thus not giving the other person their information.

3 The language of QNR

The problem that occurs in many of the solutions listed above is that they still reveal some information about the secret. For instance, in many of the solutions that used a third person, this person would gain some information about the secrets of A and B. In this section, we discuss a protocol for a language in which a person A proves a fact about a secret using another person B, without B ever even getting to know this secret.

In [GMR85], a protocol is presented for proving the hypothesis $H_2 = a \in L_m$, that is, for proving a term a to be in the language L_m of quadratic non-residuosity. An element $b \in Z_m$ is a quadratic residue $\mod m$ if $b = x^2 \mod m$, where Z_m denotes the set of integers between 1 and m that are relatively prime with m. An element a is not a quadratic residue in Z_m, if for all x's in Z_m, a never equals $x^2 \mod m$. The language of quadratic non-residuosity L_m is thus defined as

$$L_m = \left\{ a \in Z_m \mid \neg \exists x \in Z_m \ a = x^2 \mod m \right\}.$$

This language is in fact found to be in NP as it has been shown that it can be reduced to the problem of factorizing m; this seems probable, since in a naive approach we would have to calculate all the quadratic residues from numbers in Z_m before being able to conclude that a term is not in this language.

A problem with the protocol from [GMR85] is that it only works for the language of quadratic non-residuosity and moreover only works with certain algebraic secrets. In the next section we discuss a zero-knowledge protocol solving CI-WLI "without reference" that can work with arbitrary secrets. Here however, we have to use

the extra assumption that the other person also knows the secret (note that this was not needed with QNR).

4 CIWLI "without reference"

The methods of comparing information without leaking it mentioned in section 2 make it possible to verify that indeed two persons are referring to identical pieces of information. Both persons need to know *which* piece of information they are comparing however, and verification can only meaningfully occur if they are indeed referring to the same information block (henceforth termed IB). [TEE06] calls the problem in which such an a priori reference does not exist CIWLI "without reference" as opposed to CIWLI "with reference". The aim of the protocol remains; neither person may learn anything new about the secret which (s)he didn't know beforehand. Compared to CIWLI with reference, the protocols to solve CIWLI "without reference" must surmount two additional difficulties: deciding which IB is to be compared, and uniquely referring to it without leaking information.

4.1 The protocol

The protocol, as proposed by [TEE06], differentiates the two persons involved into a prover and a verifier, and identifies the IB by means of a one-way function $H(x)$: a secure hash. The initiator (which may be either the prover or the verifier) chooses an IB, and identifies it to the other person by sending its hash. Thus agreed, the verifier sends a challenge to the prover; this challenge can only be correctly answered using the IB in question. The prover proves his knowledge then by answering the challenge. The challenge in question is simple, consisting of a block of information chosen by the verifier. The prover then sends $H(I + N + P + C)$ in return, where I is the IB in question, N is a *nonce* which is a mutually determined value intended to make eavesdropping impossible, P is the identity of the prover, which makes man-in-the-middle attacks[2] impossible, and C is the challenge issued by the verifier. The full algorithm including proof of correctness can be found in [TEE06].

---

[2] A man-in-the-middle attack is an attack in which a third party is able to intercept (and sometimes modify and generate) messages sent between the two main parties undetected.

4.2 Hash function

As stated by [TEE06], the above protocol depends on the existence of a one-way hash-function [DAM88] [GMW91], and it relies on an authenticated, modification-proof communication channel [DAM88] [TSU92] [BSP95] [SCH96]. The hash-function must satisfy two requirements. First, its output (the hash) must not reveal anything about its input. Second, given a hash $H(X)$, it must not be feasible to determine the hash of some variation of X without knowing X; that is, the only feasible means of determining the hash should be full recomputation [TEE06].

4.3 Protocol analysis

[FNW96] suggests a number of beneficial properties a CIWLI protocol should have. We analyze this protocol on those fronts which make sense in a computer implemented system.

Resolution The protocol should let the verifier determine whether the prover indeed knows what is claimed. This protocol succeeds at that; when the verifier receives the hash of the IB (with extra information), he knows that the prover has possession of the IB, as without possession of the IB, its hash cannot be determined.

Leakage The protocol should not leak information; Neither prover nor verifier should learn anything about the other's knowledge beyond whether they possess it. Assuming that the hash-function is indeed one-way, clearly no information of the IB itself was transferred, thus the algorithm is leakage free.

Privacy The protocol should not let third parties determine any information about the IB in question. Since only hashes are transferred, no third party can determine any information about the IB in question.

Simplicity The protocol should be easy both to implement and to understand. Given faith in one-way hash-functions, the rest of the protocol is surprisingly simple. However, verifying the properties of a one-way hash-function is not at all trivial. Furthermore, although the IB itself clearly is not revealed, the security of meta-information *about* it is a more difficult matter:

neither is it easy to see that the protocol provides security for more than just the information block itself, but also for meta-information concerning who knows what and avoiding third parties gleaning important revelations based on the communication. For instance, a *nonce* does nothing to avoid leakage of information of the IB itself, but does avoid third parties which already possess the IB from interpreting the protocol. More on this below.

4.4 Issues

A core tenet of CIWLI protocols is that they are zero-knowledge: This means that nothing of the secret is revealed by performing the protocol. Clearly, other information *is* transmitted - namely information about the information. CIWLI protocols are thus absolutely secure with respect to the data, but not necessarily with respect to the metadata. This suggests we look at the metadata which can be revealed. The CIWLI "with reference" problem and protocols are generally symmetric: that is, person A can only tell whether his secret is equal to B's if B can learn the same. It is generally not possible to cheat in such a way that only one person learns this information.

4.5 Asymmetry as metadata leak

CIWLI "without reference" is not symmetric however, and while it may be possible to develop a protocol which reveals its results only to both players simultaneously, the protocol in [TEE06] does not: the initiator that chooses the IB to be proved, and transmits this to the other person. At this point, if the other person has that IB, he has learned that the initiator possesses that IB (although the initiator may have simply copied the hash from someone else). Since the protocol terminates on any failure, a third party can, under the assumption that the two persons follow the protocol truthfully and faithfully, determine whether the prover was successful - even if that third party knows nothing of the secret. This problem might be mitigated by a multiple round variant of the protocol in which the amount of metadata knowledge leaked each round is kept small.

Of course, the very fact that two persons are communicating is a meta-information leak, so there is a limit to how much a well-designed protocol can achieve. Nevertheless, precisely because this issue is so difficult to control, attention and awareness of it is relevant.

4.6 Timing Attack

As the protocol depends crucially on one-way functions which need (costly) complete recomputation on any change, the players, and even a third party observer can draw conclusions about the maximum size of the IB in question by observing response times. This attack could for example allow a third party to determine whether the IB being verified is a short snippet of identity data (say, a database record), or a picture of a person (generally a much larger precompressed piece of data), and could be helpful in conjunction with other such leaks to reveal much about the aims of the persons involved.

4.7 Hash as a reference

The difference between CIWLI with and without reference which causes the metadata issue is one of symmetry: As CIWLI "without reference" needs to identify the secret to compare, it delegates that responsibility to one person who uses a hash to identify the secret. That hash acts as a reference which reduces the CIWLI "without reference" problem to that of CIWLI "with reference". Once the secret has been decided, any CIWLI "with reference" protocol can replace the rest of the protocol.

5 CIWLI "without reference" as a proof system

In [TEE06], a protocol is given for CIWLI "without reference", where the verifier V initiates the proof. In the following we shall use V to denote the verifier, P for the prover and x for the secret of which V wants to have proven that P knows it. Abstracting from the implementation we can model this in the spirit of [GMR85] as a proof-system trying to prove to V the hypothesis $H_1 =$ "P knows x" to V, where V needs to communicate with P to prove this, since P is assumed to have the knowledge and computational power to prove H_1. The protocol has zero knowledge complexity, which is possible because the data x that occurs in the hypothesis is assumed to be known by P a priori, and so it suffices to send some one-way encrypted description of x, so that only P can derive its actual value.

5.1 Problem setting of the language of quadratic non-residuosity

In [GMR85], a stronger yet more restrictive protocol is given for a language that also has knowledge complexity zero, the language of quadratic non-residuosity as discussed in section 3. In this protocol we again have the V to prove H_2 to and the P who has the knowledge or computational power to prove H_2. However, in this case, x is not known to P a priori and in fact P will never know x. The fact that P helps V with proving that x is in L_m without knowing x and with knowledge complexity zero is remarkable and we note that this holds only for certain algebraic languages such as L_m. Furthermore a small error probability is present and time complexity assumptions are needed.

5.2 Comparison

So in both protocols we have certain data x, V wants to prove a hypothesis about x, P has the information or computing power to prove the hypothesis, and the difficulty is proving this to V using P, while not disclosing information about x. In [TEE06], a protocol is given that has knowledge complexity zero assuming P knows x a priori, for any x; that is, no knowledge needs to be disclosed about x since P already knows x. In [GMR85], a protocol is given that has knowledge complexity zero while not disclosing to P any information about x such that it can deduce x, however this only works for very specific data $x \in L$, for specific algebraic languages L.

6 Conclusion

CIWLI aims to securely verify another's knowledge of a fact while maintaining maximal secrecy. The QNR protocol achieves extreme secrecy, allowing a prover to prove a fact while remaining ignorant of both the fact and whether it is true. Thus, viewed as an abstract proof system, both CIWLI and quadratic non-residuosity protocols achieve a similar goal: communicating a proof without transmitting other knowledge. CIWLI and its solutions are far more general, capable of proving possession of arbitrary secrets with knowledge complexity zero, but achieve that generality at the cost of an assumption: both persons must possess the secret in question in advance.

CIWLI "without reference" is asymmetric, and thus does not provide the same fairness guarantees that CIWLI "with reference" might. Whenever it is acceptable to potentially identify the secret in question, however, it enables a new level of flexibility by not requiring agreement upon a secret, achieving a secure means of verifying knowledge without revealing it.

References

[BSP95] S. Bakhtiari, R. Safavi-Naini, J. Pieprzyk: 1995, 'Cryptographic Hash Functions: A Survey', in *Technical Report 95-09*, Department of Computer Science, University of Wollongong.

[DAM88] L. Damgård: 1988, 'Collision Free Hash Functions and Public Key Signature Schemes', in D. Chaum and W. Price (eds.), *EUROCRYPT, Lecture Notes in Computer Science*, Volume 304, Springer Verlag, Berlin, pp. 203-216.

[FNW96] R. Fagin, M. Naor, P. Winkler: 1996, 'Comparing Information Without Leaking It', in *Communications of the ACM*, Volume 39, Number 5, pp. 77-85.

[GMR85] S. Goldwasser, S. Micali, C. Rackoff: 1985, 'The Knowledge Complexity of Interactive Proof-Systems', in *Proceedings of the Seventeenth Annual ACM Symposium on Theory of Computing*, Providence, RI, pp. 291-304.

[GMW91] O. Goldreich, S. Micali, A. Wigderson: 1991, 'Proofs that Yield Nothing But their Validity or All Languages in NP have Zero-Knowledge Proofs', in *JACM 38*, pp. 691-729.

[RAB81] M.O. Rabin: 1981, 'How to exchange secrets by oblivious transfer', in *Tech. Memo TR-81*, Aiken Computation Laboratory, Harvard University.

[SCH96] B. Schneier: 1996, 'Applied Cryptography', John Wiley & Sons, New York.

[TEE06] W. Teepe: 2006, 'Proving possession of arbitrary secrets while not giving them away: new protocols and a proof in GNY logic', in *Synthese*, Volume 149, Number 2, Springer, pp. 409-443.

[TSU92] G. Tsudik: 1992, 'Message Authentication with One-Way Hash Functions', in *Proceedings of IEEE INFOCOM 1992*, IEEE Computer Society Press, Los Angeles, pp. 2055-2059.