Proceedings of SIGRAD 2012

Interactive Visual Analysis of Data

November 29–30, 2012

Växjö, Sweden

Edited by

Andreas Kerren & Stefan Seipel

Linköping Electronic Conference Proceedings, No. 81
Linköping University Electronic Press
Linköping, Sweden, 2012
ISBN 978-91-7519-723-4
ISSN 1650-3686 (print)
ISSN 1650-3740 (online)
http://www.ep.liu.se/ecp_home/index.en.aspx?issue=081

Print: Repro, Linnaeus University, Sweden, 2012

Cover illustrations selected from the papers, cf. Page v.

Preface

This year, we are happy to announce the 11th SIGRAD Conference Proceedings. For the first time, SIGRAD 2012, the annual conference of the Swedish Chapter of Eurographics, takes place in the Småland region of southern Sweden and is hosted by the Computer Science Department (ISOVIS Group) at Linnaeus University (LNU), Växjö.

SIGRAD 2012 is the premier Nordic forum for computer graphics and visualization advances for academia, government, and industry. As the years before, this annual event brings together researchers and practitioners with interest in techniques, tools, and technology from various fields, such as visualization, computer graphics, visual analytics, or human-computer interaction. In 2012, the conference solicits novel research ideas and innovative applications especially in the area of interactive visual analysis of complex data sets. Each paper in this conference proceedings was peer-reviewed by at least three reviewers from the international program committee consisting of 23 experts listed below. Based on this set of reviews, the conference co-chairs accepted 12 papers in total (seven full papers and five short papers) and compiled the final program.

Many have contributed to make the conference an enjoyable and beneficial experience. In particular, we would like to express our gratitude to the paper authors, the two invited speakers Alfred Inselberg and Frank van Ham as well as the industrial session speakers Nils Andersson (EON Development AB) and Robert Moberg (IBM Sweden). Without all those great people, this conference would not have been possible. Also, we would like to thank the international program committee as well as the auxiliary reviewers for their commitment and reviewing efforts. Finally, we thank our conference sponsors for generously supporting the conference.

Welcome to SIGRAD 2012. We hope that you enjoy the conference and that you have a great stay in Växjö, Sweden. Be inspired, share experiences, and bring home new fresh ideas.

Andreas Kerren and Stefan Seipel

Auxiliary Reviewers

Daniel Cernea, University of Kaiserslautern
Julian Heinrich, University of Stuttgart
Markus Huber, University of Stuttgart
Markus Kächele, University of Stuttgart
Gabriel Mistelbauer, Vienna University of Technology
Khoa-Tan Nguyen, Linköping University
Johanna Schmidt, Vienna University of Technology
Christian Tominski, University of Rostock
Cagatay Turkay, University of Bergen
Björn Zimmer, Linnaeus University

Cover Image Credits

Conference Sponsors

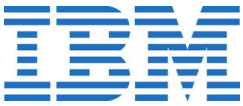


Table of Contents

Invited Talks

SIGRAD 2012
A. Kerren and S. Seipel (Editors)

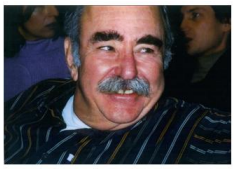
Parallel Coordinates ... are better than they look!

Alfred Inselberg

Tel Aviv University

Abstract

With parallel coordinates the perceptual barrier imposed by our 3-dimensional habitation is breached enabling the visualization of multidimensional problems. The highlights, from the foundations to the most recent results and promising applications to Big Data*, are developed.*
By learning to untangle patterns *from the displays, a powerful knowledge discovery process has evolved. It is illustrated on real datasets together with guidelines for exploration and good query design. Realizing that this approach is intrinsically limited leads to a deeper geometrical insight, the recognition of M-dimensional objects recursively from their $(M-1)$-dimensional subsets. It emerges that any linear N-dimensionsal relation is represented by $(N-1)$ indexed points. The indexing in ||-coords is not well understood and will be demystified. Indexing enables the concentration of relational information into patterns and paves the way for coping with* large datasets.
For example in 3-D, two points with two indices represent a line and two points with three indices represent a plane. There result powerful geometrical algorithms (intersections, containment, proximities) and applications including classification.
A smooth surface is the envelope of its tangent planes. This is equivalent to representing the surface by its normal vectors, rather than projections as in standard surface descriptions. Developable surfaces are represented by curves revealing the surfaces' characteristics. Convex surfaces in any dimension are recognized by the hyperbola-like (i.e. having two asymptotes) regions from just one orientation. Non-orientable surfaces (i.e. like the Möbius strip) yield stunning patterns unlocking new geometrical insights. Non-convexities like folds, bumps, concavities and more are no longer hidden and are detected from just one orientation. Evidently this representation is preferable for some applications even in 3-D. The patterns persist in the presence of errors deforming in ways revealing the type and magnitude of the errors and that's good news for the applications. We stand on the threshold of cracking the gridlock of multidimensional visualization*.*
The parallel coordinates methodology is used in collision avoidance and conflict resolution algorithms for air traffic control (3 USA patents), computer vision (USA patent), data mining (USA patent) for data exploration and classification, multiobjective optimization, decision support, and process control.



Biography: Alfred Inselberg (AI) received a Ph.D. in Applied Mathematics and Physics from the University of Illinois (Champaign-Urbana) remaining there as Research Professor until 1966. He then held senior research positions at IBM, where he developed a Mathematical Model of the Ear TIME Nov. 74, concurrently having joint appointments at UCLA, USC, Technion and Ben Gurion University. Since 1995, he is professor at the School of Mathematical Sciences in Tel Aviv University. AI was elected Senior Fellow at the San Diego Supercomputing Center in 1996, was Distinguished Visiting Professor at Korea University in 2008 and National University of Singapore in 2011. He invented and developed the multidimensional system of Parallel Coordinates for which he received numerous awards and patents (on Air Traffic Control, Collision-Avoidance, Computer Vision, Data Mining). The textbook on Parallel Coordinates: VISUAL Multidimensional Geometry and its Applications, published by Springer, was praised by Stephen Hawking among others.

SIGRAD 2012
A. Kerren and S. Seipel (Editors)

Re-inventing and Re-implementing the Wheel. Visualization Component Reuse in a Large Enterprise

Frank van Ham

IBM Business Analytics

Abstract

Most information practitioners will have implemented an interactive bar chart at some point in their career. In a large enterprise like IBM, we have hundreds of bar chart implementations available, across a number of different platforms. This represents a lot of duplicated work, and a potential maintenance nightmare. Ensuring correct and consistent design across the enterprise becomes much harder as a result. Adding to this problem is that our customers sometimes require new visualization types to be added to our products, or want to existing types tweaked to suit their needs.
In this talk I will talk about some of the practical problems we face in this area. I will also present an IBM rendering and mapping framework that represents a solution to some of these problems. By abstracting two key phases of the traditional information visualization pipeline, we can flexibly insert new mappings into products and facilitate deployment of previously created visualizations across different target platforms. I will outline some of the high level design constraints and present samples where possible.

Biography: Frank van Ham obtained his CS and PhD in Computer Science from the University of Eindhoven, specializing in graph visualization. After his doctorate studies, he joined IBM Research where he was one of the co-creators of the Many Eyes visualization service. Since 2009, Frank is part of IBM's Software division where he helps integrate core information visualization functionality into IBM's software products. He is a current associate editor for IEEE Transactions on Visualization and Computer Graphics (IEEE TVCG) and the 2012 chair of the IEEE Information Visualization conference. Frank's current research interest revolve around visualization frameworks, collaborative visualization, network visualization and general user interface design.

Papers

SIGRAD 2012
A. Kerren and S. Seipel (Editors)

Visual Ontology Alignment System – An Evaluation

V. Sabol[1], W. O. Kow[3], M. Rauch[1], E. Ulbrich[1], C. Seifert[2], M. Granitzer[2], and D. Lukose[3]

[1]Know-Center, Inffeldgasse 13/VI, 8010 Graz, Austria
[2]Faculty of Computer Science and Mathematics, Passau University, Innstrasse 33, 94032 Passau, Germany
[3]Artificial Intelligence Centre, MIMOS Berhad, Technology Park Malaysia, 57000 Kuala Lumpur, Malaysia

Abstract

Ontology alignment is the process of mapping related concepts from different ontologies. A lot of research effort has been invested in development of algorithmic methods supporting automatic discovery of mappings between ontological concepts. However, automatic alignment remains potentially prone to errors especially with large real-world ontologies, demanding intervention of domain experts. We therefore created a semi-automatic tool including algorithmic alignment methods and an interactive visual interface. Visualisation components included in the interface support experts in navigating the concept space and reviewing the automatically generated mapping suggestions. An experiment with 15 test users was performed to evaluate whether, and in which cases the use of visualisation is beneficial compared to a user interface employing standard GUI widgets. The results indicate that users typically executed tasks slightly faster with an interface using standard widgets, but an interface which includes a visualisation component providing overview, filter and narrowing-down functionality achieved higher rates of successful task completion.

Categories and Subject Descriptors (according to ACM CCS): H.5 [Information Interfaces and Presentation]: User Interfaces—User-centered design

1. Introduction

Since the advent of semantic technologies, more and more organisations and people share their knowledge in the form of ontologies. Interoperability issues may arise when ontologies express the same knowledge in different ways, for example due to the use of different terminology, divergent points of views, or differing levels of model granularity. Ontology alignment is the process of finding mappings between related concepts from two or more different ontologies. Incompatibilities between ontologies and the need for interoperability between the rapidly growing number of systems using semantic technologies lead to an increased need for ontology alignment.

There have been a variety of algorithmic solutions proposed for automatic alignment based on different approaches, beginning from simple string matching, over linguistic methods, reasoning, machine learning techniques, to specialised techniques such as similarity flooding. For more details on various algorithmic methods for ontology alignment see [ELBB*04], [ES07] and [GS08]. To compare the performance of different methods, the Ontology Alignment Evaluation Initiative [OAE11] organises a yearly event focusing on evaluation of ontology alignment algorithms and systems. By following the improvement gains on a yearly basis it can be observed that, despite an increasing research effort being invested into new, more complex algorithmic methods, the gains are tangible but diminishing.

Automatic ontology alignment methods may be a viable solution in some cases, but they are likely to underperform in scenarios where deep understanding of a specific application domain is required, large ontologies are aligned and high quality of the computed mappings is expected [Rah11]. In order to overcome these challenges, human experts should be adequately integrated in the alignment process [KL08] to make use of their wide knowledge and rich experience. Semi-automatic ontology alignment systems provide the possibility to improve on the automatic alignment techniques through the involvement of domain experts, who can decide whether an automatically computed mapping is correct or not.

However, domain experts are rarely knowledgeable in semantic technologies and thus require easy to use tools to perform this kind of tasks. Visual methods provide means for exploration and analysis of large amounts of complex infor-

mation by making use of the powerful human visual capabilities [TC05]. Use of visualisation to support ontology alignment has been discussed and advocated by several authors, such as in [LSR*08]. In [GSK*10] a survey of visual ontology alignment systems is given and compared to a list of requirements collected from various user studies, concluding that no system currently fulfils all requirements.

In this paper we present the "Semantic Mediation Tool" (SMT), a semi-automatic visual ontology alignment system using algorithmic methods to compute an initial set of mapping suggestions, which are then reviewed by experts using a visual user interface. The focus of the presented work is twofold: i) design of the user interface with the goal of fulfilling all user requirements, and ii) evaluation of performance of visualisation for semi-automatic ontology alignment tasks.

This paper is structured into eight sections. Section 2 gives a short overview of visual ontology alignment tools, Section 3 provides an architectural overview of our system, and Section 4 outlines the algorithms we used for ontology alignment. Section 5 introduces the list of user requirements and presents the visual interface of our tool. Section 6 describes the investigation procedure we used to find out whether use of visualisation is suitable for performing selected tasks. Evaluation results are reported and discussed in Section 7. Finally, Section 8 sums up the results of our evaluation and discusses future work.

2. Related Work

In visually supported, semi-automatic ontology alignment systems the involvement of humans necessitates an easy to use user interface, where visual components are used to support overview, pattern recognition and navigation functionality. In our previous work [GSK*10] we compiled a survey of semi-automatic, visually supported ontology alignment systems. The majority of available systems can be subdivided into three main categories depending on the visual paradigm employed by the user interface: i) interfaces based on linked tree widgets, ii) interfaces based on graph visualisation, and iii) treemap-based interfaces. In the following we give an brief overview of existing visual ontology alignment systems grouped into these three categories.

Interfaces based on linked trees use the standard tree widget to present the class hierarchy of the two ontologies, with both trees being placed next to each other. Mappings are shown as lines or curves connecting concepts in different trees, such as the AgreementMaker [CSMB07], COMA++ [ADMR05] and COGZ [FS07, FBG09]. The interface of PROMPT [NM03] is similar, however it shows mappings in a table instead of connecting the trees.

Since ontologies are graph structures, graph-based visualisations are a natural fit for visualising ontologies. Tools using graph visualisation methods to represent ontology nodes and generated mappings include Optima [KD08], HOMER [UGM07], PROMPT [NM03], AlViz [LS06], which provides a combination of trees and graph views, and [dSDdMR06] which uses hyperbolic geometry to reduce clutter.

The treemap [Shn91] is a visual representation providing an overview of hierarchical structures, where nodes are represented as nested rectangles. The size and colour of each rectangle encode properties of the corresponding class, such as the number of leaves and the amount of found mappings. COGZ [FS07, FBG09] includes treemaps to provide an overview of the class taxonomy and usees colours to show which regions contain many or few mappings.

Our survey [GSK*10] summarises the findings of user studies, such as [FNS06, FNS07], into a list of requirements for interactive ontology alignment tools (see Section 5 for details). Subsequently, these requirements were compared to the available visual ontology alignment systems and it was concluded that no current system fulfils all requirements. As a consequence we proposed a visual interface which was designed by following the recommendations and requirements compiled in the survey.

In [KSG*11] a very brief overview of an early version of the resulting "Semantic Mediation Tool" (SMT) was presented, but no details were disclosed and no evaluation results were presented. SMT uses an information landscape visualisation to provide an overview of concepts from both ontologies and employs graph visualisation components for showing details. The springScape system [EBJ06] uses an information landscape for visualizing multiple data sources (ontologies) in the context of microarray and contextual bioinformatic data. However, springScape does not address ontology alignment. Also, in contrast to SMT, the employed layout algorithm does not scale well and may produce different results over subsequent runs for the same data set (reproducibility problem). The remainder of this paper gives a detailed description of our system and provides results of the performed user evaluation.

3. Architecture

SMT is conceived as a server-client system implemented in Java. Alignment algorithms are executed on the server and are implemented as so-called Matcher components. Currently two matcher implementations exist, a linguistic and a statistical Matcher (see Section 4 for algorithm details), but due to a standardised Matcher API additional alignment algorithms can be easily added to the system. Matchers have access to ontologies to be aligned through an API which encapsulates triple stores (currently Jena [CDD*02] and AllegroGraph [W3C09] engines are supported). When a pair of ontologies is aligned, the list of concept mappings computed by a Matcher is stored under a user-specified name in the Mediation Repository and can be retrieved and reviewed by the user at a later time. Additionally, a full-text search index and visualisation geometry data (see Subsection 4.2) are

generated and stored in the Mediation Repository together with the list of concept mappings.

The client implements a visual user interface which connects to the server and displays the data stored in the Mediation Repository. The user can review the computed concept mappings and use the visualisations to explore the concept space and the generated mappings. The user interface consists of the following main components:

- Mapping table: a table component for displaying the suggested mappings.
- Ontology browsers: two graph visualisation components for exploring concept properties and navigating in the ontologies.
- Information landscape: a visualisation providing an overview of the entire concept space, supporting narrowing down to regions of interest.
- Ontology trees: two tree components displaying the class hierarchies and concepts of the aligned ontology pair.

A coordinated multiple view framework works behind the scenes to synchronise the different views and ensure that user actions, such as selection or filtering, performed in one component are adequately reflected in other components.

4. Algorithms

This section describes the two ontology alignment algorithms used in our system, a linguistic method and a unsupervised learning-based method. The algorithms employed in the second method are also being used for computing the geometry for the information landscape visualisation. It should be noted that, since the focus of this paper is on visualisation, an into-depth description and evaluation of the algorithms is not included.

4.1. Linguistic Method

This method uses an external taxonomy to measure the semantic distance between the two concepts based on the name of the concept. In this case, we are using the WordNet [Uni10] taxonomy, but the algorithm can be adapted to use any taxonomy. The Wu-Palmer measure [WP94] is used for calculating the similarity values of the mappings. Using this measure, similar concepts (members of the same WordNet synset) would have a score of 1. Concepts that share the same parent will have a score slightly below 1, while concepts that share the same grandparent will have a lower similarity. The distance between the common ancestor and the root of the taxonomy is also taken into account, so that siblings higher up in the tree (more general) will have a lower similarity compared to the more specific ones near the bottom of the tree.

4.2. Unsupervised Learning-based Method

The second alignment method is a machine learning-based approach utilising algorithms implemented in the Know-Center's KnowMiner knowledge discovery framework [KSM*09]. The algorithm consists of three steps: concept vectorisation, concept clustering, and mapping finding. In the first step every concept is converted into one or more feature vectors using following information: i) concept label, ii) concept description (if available), iii) neighbouring concept labels, and iv) relationships connecting to neighbours. Additionally, WordNet is used to extend the label information with synonyms (as well as hypernyms and hyponyms, if desired). In this way up to four different vector spaces are spawned which are used to compute the cosine similarities between concept pairs. It should be noted that, besides concept labels and descriptions, structural information (i.e. relationships and neighbours) and linguistic information (i.e. label synonyms) also contribute to the similarity. A compound similarity value over all spaces allows for adjusting the weight of each space. Currently the weights are fixed, but in a future version of the algorithm they could be automatically adjusted depending on the specifics of the ontologies to be aligned.

Once concept vectors are available, the concepts from both ontologies are clustered using a scalable hierarchical clustering algorithm [MSG10] running in $O(N * log(N))$ time and space, N being the total number of concepts. The algorithm creates a balanced hierarchy of clusters by recursively applying a modified x-means [PM00] clustering method. As a result, similar concepts will be gathered in the same clusters, even when they originate from different ontologies. To avoid comparing all concept pairs when finding mappings, which would lead to a quadratic execution time, mappings are found by inspecting only pairs of concepts assigned to the same cluster (or sub-cluster). By choosing sub-clusters deeper in the hierarchy in such a way that the number of concepts within the branch is smaller than a fixed threshold $C \ll N$, the number of comparisons performed for each concept is O(C) (i.e. constant) resulting in the running time of the mapping-finding step being linear with the total number of concepts N.

The hierarchy produced by the clustering algorithm is also used for computing the similarity layout (i.e. a layout where similar objects are placed spatially close to each other) and the geometry needed for the information landscape visualisation. The hierarchy computation of the 2D similarity layout and cluster area subdivision are performed by a scalable projection algorithm having the same $O(N * log(N))$ time and space complexity as the clusterer [MSG10]. The algorithm proceeds recursively along the cluster hierarchy: First, the top-level clusters are positioned using a force-directed placement method [FR91] producing a layout where similar clusters are placed close to each other. Then, each cluster is assigned a polygonal area using Voronoi area subdivision [Aur91]. The method proceeds recursively by positioning sub-clusters within their parent cluster's area and then assigning Voronoi areas to the sub-clusters. Recursion stops when the bottom of the hierarchy, i.e. the concepts, are

reached and positioned. Scalability of the algorithm is due to the fact that at each hierarchy level the number of direct children of a cluster has a strict upper limit, which is guaranteed by the clustering algorithm. As a results, the force-directed placement method, which is known to compute good similarity layouts but scales poorly, is only applied on small data chunks (consisting of, for example, 20 elements) on which it operates very quickly.

5. Visual User Interface

As already mentioned in Section 2, the design of our system is based on the summary of user requirements for interactive alignment tools provided in [GSK*10]. The requirements are as follows:

1) Presentation of automatically generated mapping suggestions including an estimated confidence for each mapping.
2) Exploration and navigation of ontologies providing details on every concept.
3) Overview of the concept space and the alignment results.
4) Capability to narrow down to the area of interest.
5) Filtering based on features of concepts and mappings.
6) Confirming, rejecting and editing of automatically generated mappings.
7) Collaboration via commenting, tagging, annotating etc.
8) Ability to partition the reviewing task into chunks assignable to team members.
9) Saving and loading of user's changes.

Visual techniques appear particularly useful for addressing Requirements 2, 3 and 4, while Requirements 5, 6 and 8 can also benefit from visualisation methods. Therefore we designed a visual user interface, shown in Fig. 1, providing support for the above requirements. The two aligned ontologies in this example are the **mouse** anatomy ontology (red), and the **human** anatomy ontology (green), both provided by the Ontology Alignment Initiative [OAE11]. The colour used for each ontology is configurable to support users with red-green colour blindness. The user interface consists of the following main components: a table of mappings (up-left) addressing Requirements 1, 6 and 7, two ontology browsers (on right) addressing Requirement 2 and supporting Requirement 6, and an information landscape (in the centre) addressing Requirements 3 and 4 and supporting Requirement 5 and 8. Requirement 5 is fully supported by a search facility (upper-right corner). Requirement 8 is supported by the capability to select a subset of the mappings, typically using the information landscape (see Subsection 5.2), and assigning it as a task to an (expert-)user (second button from the right in the tool bar), who will be able to review and modify only the assigned mappings. Monitoring of task progress is also possible (rightmost button in the tool bar). Finally, saving and loading of user changes (button in the upper left corner of the tool bar) is also provided satisfying Requirement 9. Therefore, in contrast to other visual ontology alignment systems (see Section 2), the proposed user interface, which is described in the rest of this section, fulfils all requirements listed above.

Using an information landscape visualisation in the context of ontology alignment is a novel concept, which needs to be evaluated (see Section 6). In order to compare the information landscape with the standard tree widgets which are typically used in comparable tools, we included the possibility of displaying and navigating ontology class hierarchies using a pair of trees (shown in Fig. 2). These ontology trees address the same requirements as the information landscape.

5.1. Mapping Table

The mapping table (top-left in Fig. 1) displays mappings discovered by the alignment algorithms. Table columns show the name of the mapped concepts (with coloured icons encoding ontology membership), a similarity score between 0 and 1, a status column (suggested, accepted, rejected), and a reviewer column (initially empty). By default, mappings are sorted by their estimated similarity (confidence), but sorting by any other column is possible. Initially all mappings are in suggested state. The user can review the mappings, change their status to accepted or rejected (using buttons in the tool bar), add a comment, and save the performed changes. Navigation in the table takes place by paging in groups of 10, 20 or 50. Filtering of concepts and mappings is possible depending on the mapping state (drop-down list in the tool bar) and by navigation and selection in the information landscape (see next subsection). Concepts and mappings not fulfilling the filter criterion are removed from the mapping table. Additionally, backed by a built-in retrieval subsystem, our tool supports searching as well as fast highlighting and filtering based on full-text and Boolean queries. To optimise the reviewing, various keyboard shortcuts are provided together with a single-click function for accepting the selected concept pair and rejecting all other mappings containing one of the concepts.

5.2. Information Landscape

In the centre of Fig. 1 the information landscape visualisation [SKM*09] can be seen showing an overview of all concepts from the two ontologies. An information landscape is a visualisation paradigm based on a geographic map metaphor, which conveys relatedness in the data through spatial proximity in the visualisation. Concepts are shown as dots with colours encoding ontology membership and spatial proximity between the dots encoding similarity between the concepts. By selecting a mapping in the table, the corresponding pair of concepts will be highlighted in the landscape. As related concepts from different ontologies are grouped together by the similarity layout algorithm, identification of regions rich with mapping candidates becomes easily possible: Identifying promising alignment candidates is as simple as finding dense regions containing dots in different colours, while regions dominated by one colour are

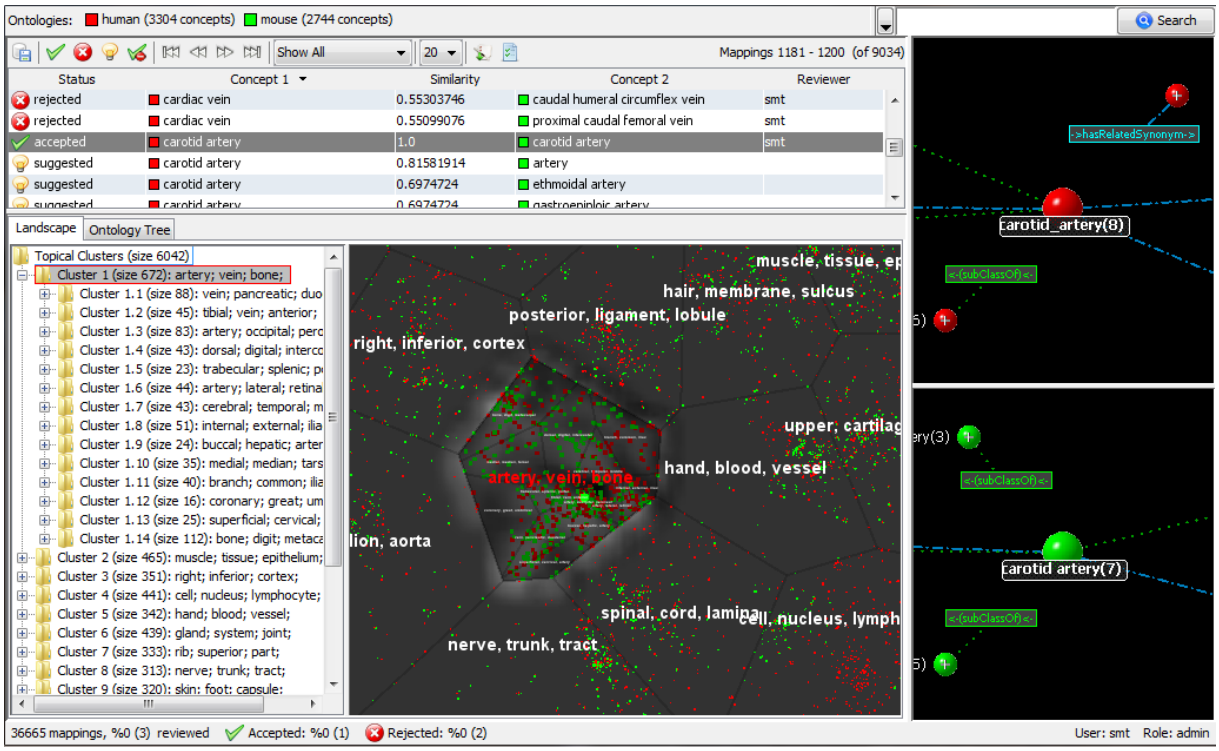


Figure 1: *Semantic Mediation Tool user interface.*

likely to contain no mappings (for example in the bottom-right corner of the landscape).

The landscape organises concepts from the two ontologies into a hierarchy of clusters which are represented as nested polygonal areas. Just like the concepts, cluster areas are also positioned in such a way that similar clusters are placed spatially close together. Each cluster is labelled by several keywords which were extracted as highest weight features from the underlying concept vectors. Labelled cluster areas are useful when aligning large ontologies, because the user can efficiently narrow down to the area of interest: Moving the mouse cursor over the keywords shows the area covered by the cluster (see "artery, vein, bone" cluster in Fig. 1). The user can navigate deeper in the cluster hierarchy by choosing a suitable cluster and clicking on its keywords, which reveals its sub-clusters. Alternatively, free navigation using zooming (mouse wheel) and panning (mouse drag) is also supported. Once the user has narrowed down to the area of interest, lasso- and/or cluster-selection can be used to select a particular group of concepts (highlighted in Fig. 1). On selection, the mapping table is updated to show only mappings containing the selected concepts.

In a classical information landscape hills will arise where the density of visualised items is large, visually emphasising areas with agglomerations of similar items. To highlight the areas being reviewed by the user and separate them visually from areas which are currently not in the focus of interest, we have modified this concept so that only selected concepts, i.e. those for which mappings are currently shown in the mapping table, will contribute to the height of the hills.

The capability provided by the landscape to explore, narrow down and filter is crucial when the number of concepts and mappings is large, enabling the user to identify and focus only on relevant portions of the concept space. Note that, as users may be accustomed to navigating hierarchies using a tree widget, providing an additional tree view showing the cluster hierarchy (on the left from the landscape) proved to be beneficial to the users [GKS*04].

5.3. Ontology Browser

When deciding whether to accept or reject a mapping the user might need additional information on the corresponding pair of concepts. This information is provided visually by a pair of ontology browsers (on right in Fig. 1). When a pair of concepts is selected in the mapping table, each concept, together with the triples containing the concept, will be displayed in the corresponding ontology browser. Literals are shown with a gray 'information' icon, while predicates are

displayed as links with the name of the predicate labelling the link.

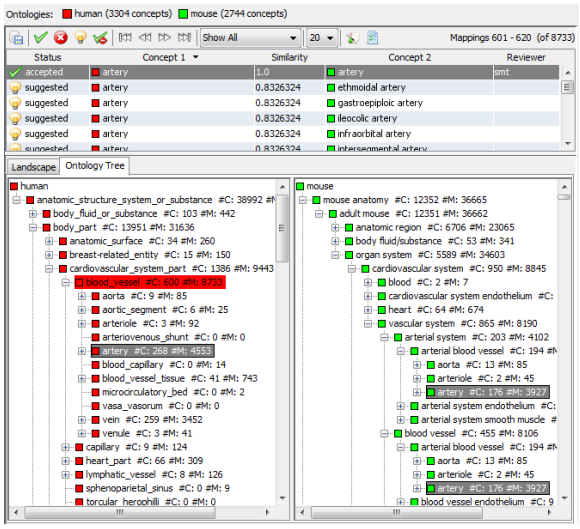


Figure 2: *Trees showing ontology class hierarchies.*

5.4. Ontology Trees

As an alternative to the landscape view, SMT provides a pair of tree views for navigation in the class hierarchies of both ontologies (see Fig. 2). Note that, while the hierarchy in the landscape view is calculated by the clustering algorithm, the ontology class hierarchies are typically man-made. The root element of each tree contains the name of the ontology. Besides the name of the concept, each concept label contains the number of sub-concepts in its sub-tree (*#C*) and the number of mappings for itself and its sub-concepts (*#M*). Narrowing down and filtering of the mappings is supported through navigation and selection. By selecting a concept (Fig. 2: "blood vessel", highlighted in red) the mapping table is updated to show only mappings containing the selected concept or its sub-concepts. Also, when a user selects a mapping in the mapping table, the trees will expand and the concepts will be highlighted (grey background). Note that due to multiple inheritance in the class hierarchy and flattening of the graph structure, it is likely that multiple branches in each tree will be expanded.

6. User Evaluation

We performed a user evaluation of the SMT to find out how our visual tool performs for tasks relevant to ontology alignment. In this evaluation we focused on comparing the usability of two visual representations of the concept space: the information landscape and the ontology trees. Since tree views are standard components in most operating systems, users will certainly be very familiar with them. However, due to multiple inheritance in the class hierarchy and flattening of the ontology graph structure into a tree structure, concepts will be occurring multiple times in different sub-trees. We expect nodes occurring multiple times in different sub-trees to be counterintuitive for users, which might lead to slower interaction with the ontology trees, for example due to increased necessity for scrolling. Thus, our first hypothesis for the user evaluation, which focuses on navigational tasks involving exploration and navigation to a particular concept, is as follows:

H1 Users perform equally well with both representations in navigational tasks.

Information landscapes have been shown to be a useful representation for getting an overview of large data sets and narrowing down to the area of interest [SKM*09]. Thus, our second hypothesis for the user evaluation was the following:

H2 Users perform better with the information landscape in tasks involving narrowing-down and filtering.

During the evaluation we further wanted to find out users' preferences and collect general user feedback to the proposed tool.

6.1. Design

We used a within-subject design with the independent variable being the visual representation of the concept space with two different levels: information landscape and ontology trees. We measured task completion times and task completion success rates. Task completion time is measured as the time the user required from reading the task until the completion of the task or until the timeout for the task was reached. A task is counted as successfully solved if the desired result was achieved within the maximum time limit. The time limit was identified with three pilot-user tests and was set to one minute for the easy (first four) tasks and five minutes for the complex (last four) tasks.

6.2. Procedure

Figure 3 gives an overview of the evaluation procedure. Every participant was given an introduction to ontologies and to the goals of ontology alignment. Then, the participants were asked to fill out a demographic questionnaire. After that participants were introduced to the application and had the possibility to try it out and ask questions. Then each participant performed 8 tasks, where the last 4 tasks were performed twice, once with the information landscape and once with the ontology trees. The first four tasks were designed to be simple with the purpose to familiarise the participants with the terminology and the problem of ontology alignment. These initial four tasks, which were only performed using the mapping table, are as follows: **T1**) finding the total number of mappings, **T2**) identifying concepts for a first

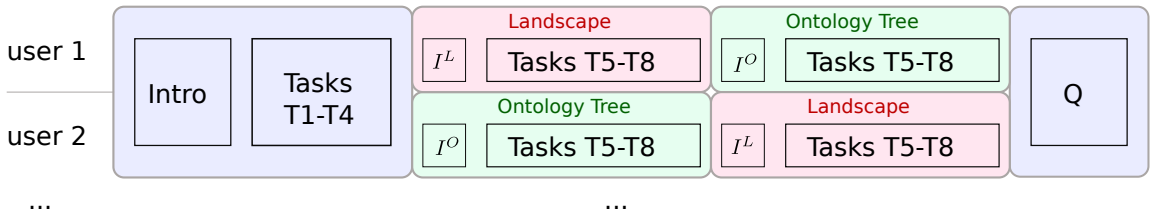


Figure 3: *User evaluation procedure: After a general Introduction (I), all users performed tasks T1-T4. Tasks T5-T8 were performed twice by all users, once using the information landscape and once with the ontology trees. The sequence of the visual representations was altered for subsequent users (I^O - introduction to the Ontology Trees, I^L - introduction to information landscape). Finally, users filled out a questionnaire (Q).*

mapping, **T3**) confirming the highest-, and **T4**) rejecting the lowest-confidence mapping for a given concept.

Each of the next four tasks were performed once with the information landscape and once with the ontology trees. The sequence of the test conditions was altered with each participant. Before beginning with the tasks the respective visual representation (information landscape or ontology trees) and available interaction mechanisms were explained in detail. The user had the possibility to try the interaction mechanisms and ask questions. Each user started testing with exactly the same state of the system for each task. Tasks T5-T8 were defined as follows:

T5 Guided navigation to a concept and mapping counting,
T6 Non-guided navigation and mapping review,
T7 Overview, narrowing-down and elimination of non-relevant mapping subsets,
T8 Narrowing-down and selection of a relevant mapping subset.

Tasks **T5** and **T6** were designed to test hypothesis ***H1*** (navigability), while tasks **T7** and **T8** were designed to test hypothesis ***H2*** (narrowing-down and filtering). After performing the tasks the participants filled out a questionnaire. The questionnaire contained questions about how the participants perceived the interaction with the mapping table, the information landscape and the ontology trees in terms of intuitiveness and ease of use. We used a 5-point Likert scale for these questions. Further, the participants stated which visual representation they preferred for tasks **T5** to **T8**. The questionnaire also contained a comments section about what the participants particularly liked and disliked, and what they think should be improved.

6.3. Test Material

For evaluating the SMT we used two standard ontologies from the *Ontology Alignment Evaluation Initiative* [OAE11], namely the **human** ontology consisting of 2744 concepts and **mouse** ontology containing 3304 concepts. For evaluation we used only the unsupervised learning-based alignment algorithm, the reason being that alignment results and the similarity layout used by the landscape would be more in accord, since they were generated based on the same cluster hierarchy. The alignment algorithm generated 36.665 mapping suggestions, with multiple suggestions per concept enabled and the similarity threshold set to a low value in order to achieve a high recall.

As already mentioned, the cluster hierarchy shown in the information landscape is generated automatically based only on concept similarities. In contrast to that, the tree view displays the inherent hierarchical structure of an ontology by using the existing *subClassOf* relation and *UNDEFINED_part_of* restriction.

6.4. Participants and Environment

For the user evaluation we managed to recruit 15 volunteers, 13 males and 2 females. The age of the participants ranged from 24 to 40 years, with an average of 30.5 years. All participants were experienced computer users. One third (5) of the participants had little or no experience with ontologies and two thirds (10) were familiar or very familiar with ontologies. The participants were tested in a calm environment without noise distractions. The task was performed on a Dell Latitude e650 notebook running Windows XP Professional. The notebook was equipped with an Intel Core Duo 2.26 GHz, 3 GB RAM, a USB mouse and an external keyboard. An external 22 inch display with a resolution of 1680 x 1050 pixels was used.

7. Results of User Evaluation

In this section we present and discuss the results of time measurements, the questionnaire answers, and collected user suggestions.

7.1. Measured Performance

Table 1 shows the mean and standard deviation of the task completion times for tasks T5 to T8. We omit results for tasks T1 to T4, because the goal for those tasks was primarily to make participants familiar with the application. Only 12

out of 15 participants were able to complete task T7 within the given time limit, the results in Table 1 were calculated by omitting these users from the calculation for this task. We tested on equal means with Wilcoxon rank sum test for unpaired samples. The null hypothesis for the test was that the means are equal, we set $\alpha = .05$. We found a statistical difference ($\alpha = 0.05$) between the two conditions for task T5, T6, and T8. We found a statistical difference for task T7, which means users performed slower using the landscape in this task. However, all users were able to solve this task using the landscape within the given time limit. In contrast, 20% of the users were not able to solve this task at all using the ontology trees.

	Completion time [sec]		Success rate [%]	
Task	**L**	**OT**	**L**	**OT**
T5	61 ± 37	47 ± 15	100	100
T6	95 ± 37	89 ± 33	100	100
T7	$\mathbf{179 \pm 73}$	$\mathbf{98 \pm 55}$ [1]	100	80
T8	105 ± 57	112 ± 50	100	100

[1] the three participants who did not complete the task were omitted

Table 1: *Comparing task completion times for tasks T5 to T8 for landscape (L) and ontology trees (OT). Showing mean and standard deviation. Statistically significant differences are marked bold.*

We further were interested, if participants being slower using one visual representation are also slower using the other one. Thus, for Task T5 - T8 we calculated Pearson's correlation coefficient ρ between the completion time participants achieved with the ontology trees and the completion time participants achieved using the landscape. The correlations ranged from 0.11 (T7) to 0.53 (T6). Given the limited amount of samples we conclude that there is no correlation between completion times using the different interfaces.

*Summing up, on our user sample we found no difference in user performance between the information landscape and the ontology trees for navigational tasks (**H1**). Users performed not better with the information landscape in tasks involving narrowing-down and filtering (**H2**), users were able to solve all tasks with the information landscape, but not with the ontology trees.*

7.2. Quantitative Evaluation of the Questionnaire

As mentioned before, each participant filled out a questionnaire after executing the test tasks. We report which visual representation users preferred for which task and how they perceived the interaction with the representation.

Table 2 summarises how participants perceived the visual representations, and Figure 4 shows which one was preferred by the participants for which task. As can be seen in the table, participants found the ontology trees more intuitive and

Property	**Landscape**	**Ontology Trees**
Intuitiveness	4.1 ± 0.8	$\mathbf{3.8 \pm 1.1}$
Ease-of-use	4.2 ± 0.6	$\mathbf{3.3 \pm 0.9}$

Table 2: *User ratings for intuitiveness and ease-of-use of the landscape and the ontology trees on a fife point scale where one is the best and fife the worst value. Showing mean and standard deviation. Better (smaller) values are marked bold.*

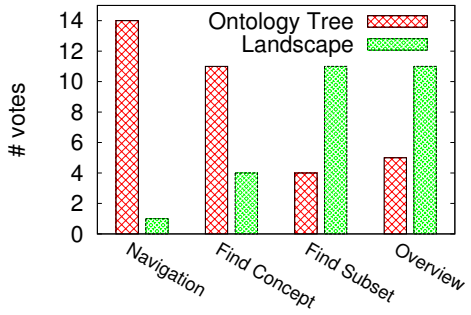


Figure 4: *User voting: Preferred visual representation for tasks relevant to visual ontology alignment.*

easier to use than the information landscape. However, Figure 4 shows a split picture. On the one hand, participants preferred the ontology trees for navigation and finding single concepts. On the other hand, for selecting relevant subsets of ontologies and getting an overview over the concept space, the landscape was the participants' representation of choice.

Summing up, we found out, that participants found the ontology trees more intuitive, easier to use and better suited for navigation tasks, but clearly preferred the landscape for overview and subset selection.

7.3. Qualitative Evaluation of the Questionnaire

Here we give a report on comments provided by test users including positive and negative feedback as well as suggestions for improvement. Due to the large number of comments, we provide insights into the overall distribution of comments and give a summary of the most common issues. A detailed discussion would be outside the scope of this paper and will be provided in a future report. All 15 participants provided at least one comment. The comments could be grouped into five categories which are listed in Table 3. Some of the comments addressed multiple categories and/or multiple visual components. For the following analysis we counted such comments once for each addressed category or component, resulting in 86 comments in total. We evaluated the type of the comments, identifying each comment either as being positive, being negative or being a suggestion. Figure 5 gives an overview of the number of comments, their categories and their types for each component.

Category	Description
Features	Requests for new features, suggestions for improving existing ones
Filtering	Comments on filtering mappings in the mapping table
Navigation	Comments on navigation in ontology trees and landscape
Visual Appearance	Comments about look and feel of different components of the application
Technical Issues	Comments about the technical issues, such as response times or click accuracy

Table 3: *Categories of participants' comments.*

Majority of the comments (27) were about navigation, most of them (12) being about navigation in the ontology trees. Distribution of positive and negative comments on navigation in ontology trees was equal, negative comments arising mostly due to multiple branches being expanded on concept selection in the mapping table. Navigation comments for the landscape (7) are slightly on the negative side (3 positive vs. 4 negative), with negative comments addressing loss of context when zooming and technical issues concerning the mouse interaction. Comments on navigation in the mapping table was predominantly negative (2 positive vs. 4 negative comments), mostly due to complexity arising from the combination of sorting, paging and selection.

In terms of components, the landscape occurred in most comments (32), about 25% of the comments covering its visual appearance. The visual appearance of the information landscape received strongly positive comments (7 positive vs. 3 negative comments). In total, the landscape (ratio positive to negative comments: 13/11) was perceived more positively than ontology trees (ratio positive to negative comments: 3/7).

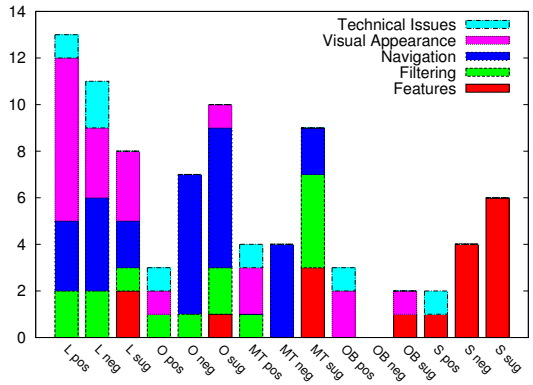


Figure 5: *Positive (pos), negative comments (neg) and suggestions (sug) by component and category. OT - Ontology Trees, L - Landscape, MT - Mapping Table, OB - Ontology Browser, S - Search.*

Summing up, the user rated the landscape more positively than ontology trees, but requested improvements in navigation and fixes for mouse interactions.

8. Conclusion

We presented a semi-automatic visual system for ontology alignment, in which algorithmic methods are used to compute an initial set of mapping suggestions, which are then reviewed by experts using a visual user interface. In a user evaluation we compared the two visual key components, the information landscape and ontology trees, for tasks relevant to visual ontology alignment.

We found no difference in task completion times between the landscape and the ontology trees for navigational tasks. However, landscape appears to be slightly slower, but the results are not statistically significant on the tested user sample. However, the information landscape is better suited for narrowing down, filtering and selection in terms of task completion success rate. On the basis of the comments to the questionnaire users preferred the landscape for overview tasks. However, users would prefer the ontology trees for navigational tasks, but as indicated by user comments this might change if users were provided a better training for the unfamiliar landscape interface.

Concerning our future work, in the short term we will be fixing a number of minor but annoying technical and usability issues, especially those affecting the landscape visualisation. We will also address the most pressing feature requests for improving mapping selection and navigation. In the following, we plan to focus on evaluation and tuning of the alignment algorithms. As a challenging goal for the future, we envision a system where user feedback provided in visual form will be utilised to adapt the model, improve the algorithm performance, and show the adapted results within a short (possibly near real-time) time interval. Finally, after having compared our visual tool against a common tree-based interface, we will compare SMT performance to other visual ontology alignment systems, such as those mentioned in Section 2.

Acknowledgements The Know-Center is funded within the Austrian COMET Program – Competence Centers for Excellent Technologies – under the auspices of the Austrian Ministry of Transport, Innovation and Technology, the Austrian Ministry of-. Economics and Labor and by the State of Styria. COMET is managed by the Austrian Research Promotion Agency (FFG). MIMOS Berhad is funded by the Malaysian government through the Ministry of Science, Technology and Innovation (MOSTI).

References

[ADMR05] AUMUELLER D., DO H.-H., MASSMANN S., RAHM E.: Schema and ontology matching with coma++, June 2005. In Proceedings of the ACM SIGMOD. 10

[Aur91] AURENHAMMER F.: Voronoi diagrams - a survey of a fundamental geometric data structure. *ACM Computing Surveys* (1991), 345–405. 11

[CDD*02] CARROLL J. J., DICKINSON I., DOLLIN C., REYNOLDS D., SEABORNE A., WILKINSON K.: Jena - A Semantic Web Framework for Java. http://jena.sourceforge.net/index.html, 2002. 10

[CSMB07] CRUZ I., SUNNA W., MAKAR N., BATHALA S.: A visual tool for ontology alignment to enable geospatial interoperability. *Journal of Visual Languages and Computing* (2007), 230–254. 10

[dSDdMR06] DE SOUZA K. X. S., DAVIS J., DE MEDEIROS E., ROBERTO S.: Aligning ontologies, evaluating concept similarities and visualizing results. *Journal on Data Semantics V* (2006), 211–236. 10

[EBJ06] EBBELS T. M. D., BUXTON B. F., JONES D. T.: springscape: visualisation of microarray and contextual bioinformatic data using spring embedding and an 'information landscape'. *Bioinformatics 22* (2006), 99–107. 10

[ELBB*04] EUZENAT J., LE BACH T., BARRASA J., BOUQUET P., DE BO J., DIENG R., EHRIG M., HAUSWIRTH M., JARRAR M., LARA R., MAYNARD D., NAPOLI A., STAMOU G., STUCKENSCHMIDT H., SHVAISKO P., TESSARIS S., VAN ACKER S., ZAIHRAYEU I.: State of the art on ontology alignment, 2004. Deliverable of the Knowledge Web Project (IST-2004-507482), Knowledge Web Consortium. 9

[ES07] EUZENAT J., SHVAIKO P.: Ontology matching, 2007. 9

[FBG09] FALCONER S., BULL R., GRAMMEL L.AND STOREY M.-A.: Creating visualizations through ontology mapping, March 2009. In Proceedings of the 2nd International Workshop on Ontology Alignment and Visualization. 10

[FNS06] FALCONER S., NOY N., STOREY M.-A.: Towards understanding the needs of cognitive support for ontology mapping. In *Proceedings of the Ontology Matching Workshop (5th International Semantic Web Conference)* (2006), pp. 25–36. 10

[FNS07] FALCONER S., NOY N., STOREY M.-A.: Ontology mapping - a user survey. In *Proceedings of the Workshop on Ontology Matching (OM2007) at ISWC/ASWC2007* (2007), pp. 113–Ű125. 10

[FR91] FRUCHTERMAN T., REINGOLD E.: Graph drawing by force-directed placement. *Software – Practice & Experience (Wiley)* (1991), 1129–1164. 11

[FS07] FALCONER S., STOREY M.-A.: A cognitive support framework for ontology mapping, November 2007. In Proceedings of International Semantic Web Conference. 10

[GKS*04] GRANITZER M., KIENREICH W., SABOL V., ANDREWS K., KLIEBER W.: Evaluating a system for interactive exploration of large, hierarchically structured document repositories. In *Proceedings of the IEEE Symposium on Information Visualization (InfoVis '04)* (2004), IEEE Computer Society, pp. 127–134. 13

[GS08] GAL A., SHVAIKO P.: Advances in ontology matching. In *Advances in Web Semantics I: Ontologies, Web Services and Applied Semantic Web*. Springer, 2008, pp. 176–Ű198. 9

[GSK*10] GRANITZER M., SABOL V., KOW W. O., LUKOSE D., TOCHTERMANN K.: Ontology alignment - a survey with focus on visually supported semi-automatic techniques. *Future Internet 2*, 3 (2010), 238–258. 10, 12

[KD08] KOLLI R., DOSHI P.: Optima: Tool for ontology alignment with application to semantic reconciliation of sensor metadata for publication in sensormap, August 2008. In Proceedings of the second IEEE International Conference on Semantic Computing. 10

[KL08] KOTIS M., LANZENBERGER M.: Ontology matching: Status and challenges. *IEEE Intelligent Systems 23* (2008), 84–85. 9

[KSG*11] KOW W. O., SABOL V., GRANITZER M., KIENREICH W., LUKOSE D.: A visual soa-based ontology alignment tool. In *Proceedings of the Sixth International Workshop on Ontology Matching (OM 2011)* (2011). 10

[KSM*09] KLIEBER W., SABOL V., MUHR M., KERN R., ÖTTL G., GRANITZER M.: Knowledge discovery using the knowminer framework, iads. In *IADIS International Conference Information Systems* (2009), pp. 307–314. 11

[LS06] LANZENBERGER M., SAMPSON J.: Alviz - a tool for visual ontology alignment. In *In IV '06: Proceedings of the conference on Information Visualization* (2006), IEEE Computer Society, pp. 430–440. 10

[LSR*08] LANZENBERGER M., SAMPSON J., RESTER M., NAUDET Y., LATOUR T.: Visual ontology alignment for knowledge sharing and reuse. *J. Knowledge Management 12*, 6 (2008), 102–120. 10

[MSG10] MUHR M., SABOL V., GRANITZER M.: Scalable recursive top-down hierarchical clustering approach with implicit model selection for textual data sets. In *7th International Workshop on Text-based Information Retrieval in Proceedings of 21th International Conference on Database and Expert Systems Applications (DEXA 10)* (2010). 11

[NM03] NOY N., MUSEN M.: The prompt suite: Interactive tools for ontology merging and mapping. *International Journal of Human Computer Studies 59* (2003), 983–1024. 10

[OAE11] Ontology Alignment Evaluation Initiative. //oaei.ontologymatching.org, 2011. 9, 12, 15

[PM00] PELLEG D., MOORE A.: X-means: Extending k-means with efficient estimation of the number of clusters, 2000. 11

[Rah11] RAHM E.: Towards large-scale schema and ontology matching. In *Schema Matching and Mapping*. Springer, 2011, pp. 3–27. 9

[Shn91] SHNEIDERMAN B.: Tree visualization with tree-maps: A 2-d space-filling approach. *ACM Transactions on Graphics 11* (1991), 92–99. 10

[SKM*09] SABOL V., KIENREICH W., MUHR M., KLIEBER W., GRANITZER M.: Visual knowledge discovery in dynamic enterprise text repositories, 2009. 12, 14

[TC05] THOMAS J., COOK K. (Eds.): *Illuminating the Path: The Research and Development Agenda for Visual Analytics*. IEEE Computer Society, August 2005. National Visualization and Analytics Center. 10

[UGM07] UDREA O., GETOOR L., MILLER J.: Homer: Ontology alignment visualization and analysis. In *6th International and 2nd Asian Semantic Web Conference (ISWC2007+ASWC2007)* (November 2007), pp. 111–112. 10

[Uni10] UNIVERSITY P.: About wordnet. http://wordnet.princeton.edu, 2010. 11

[W3C09] W3C: Allegrograph rdfstore web 3.0's database. http://www.franz.com/agraph/allegrograph/, September 2009. 10

[WP94] WU Z., PALMER M.: Verb semantics and lexical selection, 1994. Proceedings 32nd Annual Metting of the Association for Computational Linguistics (ACL). 11

SIGRAD 2012
A. Kerren and S. Seipel (Editors)

Usability Analysis of Custom Visualization Tools

Mohammad A. Kuhail, Soren Lauesen, Kostas Pantazos, and Xu Shangjin

IT University of Copenhagen, Denmark

Abstract

Many visualization tools allow the implementation of custom (non-standard) visualizations, but they differ in approach. The approaches vary from imperative to declarative programming. Moreover, some tools provide environments that assist designers in implementing visualizations. Which approach supports designers best in implementing custom visualizations? What is lacking? To answer these questions, we compared the approaches of four recent visualization tools that support custom visualizations using an example. Further, we evaluated the approaches using the framework of the Cognitive Dimensions of Notations (CDs). Our findings favour notations that use declarative rather than imperative programming, and environments that allow exploration rather than dialogue-dependant ones.

Categories and Subject Descriptors (according to ACM CCS): H.3.4 [Systems and Software]: Performance evaluation (efficiency and effectiveness)—H.5.2 [User Interfaces]: Evaluation/methodology—

1. Introduction

Charting tools (e.g. MS Excel) support standard visualizations such as line charts, bar charts, etc. Designers only need to select data, and choose their visualizations based on predefined templates. However, *custom visualizations* are tailored to a specific need, and cannot be built like standard visualizations. Further, designers might not be exactly sure about what the desired visualization should look like. They need to explore various avenues to implement the visualization. It is a trial and error approach.

A visualization tool allows the user to set up a graphical presentation of data. Many visualization tools allow the implementation of custom visualizations, but they vary in approach. For instance, some tools [Fek04, HCL05] provide modules (e.g. visual objects, layout mechanisms) that can be used with traditional programming languages. Other tools [BH09] provide declarative domain specific languages that can combine visual objects, and define their properties to show data. Moreover, some tools [KPX13] provide development environments that use cognitive artefacts such as interface builder, direct manipulation, etc. to enhance designers' cognitive ability and ease the process of visualization design.

How do the existing approaches support custom visualizations? To what extent are the approaches sufficiently accessible to designers? What is lacking? To answer these questions, we selected four recent visualization tools: Prefuse, Improvise, Protovis, and Uvis. We investigated the tool approaches, and compared the solutions of the tools to a custom visualization. Furthermore, we used the framework of the Cognitive Dimensions of Notations (CDs) [Gre89], a framework for evaluating a notional system and the environments it is manipulated in, to evaluate the tools and identify areas that need improvement.

Our findings are in favour of notations that use declarative programming rather than imperative programming, and environments that support exploration rather than restrictive dialogue-dependant ones.

2. Related Work

A common approach to evaluating visualization tools is to conduct an experiment or an evaluation study that evaluates how well several tools support tasks . Examples can be found at [Byr99,SEH00,PGB02]. This approach provides some insight about the usability of the evaluated tools, but it is task-specific.

Broad evaluations of visualization tools exist. For instance, a recent study was carried out to evaluate how visualizations are constructed from a user perspective [PL12]. While this approach is insightful, it does not give details

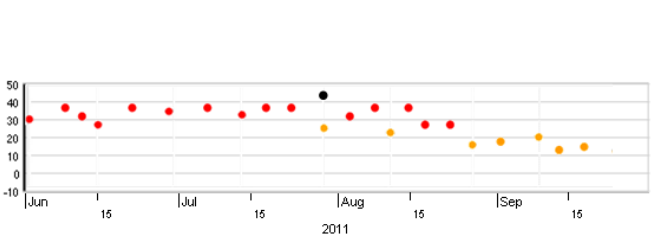


Table HighReading

Id	Temperature	Date
1	30	2-6-2011
4	27	15-6-2011
3	25	28-6-2011
4	37	8-7-2011
...	...	...

Figure 1: *A custom x-y graph based on table HighReading*

about the specifics (e.g. functionality, utility, etc.) of the tools. Another recent study compared the visualization and analysis functionalities, and environments of various visualization tools. However, little critique was given about the usability of the tools [HC12]. The authors of Protovis evaluated the accessibility (the effort required to create or modify a visualization) of Protovis [BH09], Flare [Fla], and Processing [Pro] using a subset of dimensions from the CDs framework [BH09]. The evaluation was brief, but it shed some light on a few aspects of the tools.

3. Evaluation Settings

3.1. Selection of Visualization Tools

We selected four visualization tools for evaluation: Prefuse [HCL05], Improvise [Wea04] , Protovis [BH09], and Uvis [LKP*13, KPL12, KL12]. We selected the tools based on support for custom visualizations, how recent they are, whether they are general-purpose, and difference of approaches.

All the selected tools support the creation of custom visualizations, have been developed in the last decade, and are general-purpose. Also the tools have different approaches to visualization creation. We only selected a representative tool from tools similar in approach or cognitive artefacts. For instance, we excluded Flare [Fla] since it adapted its design from Prefuse. Likewise, we excluded D3 [BOH11] as it borrows a lot of its concepts (e.g. helper functions) from Protovis.

Since they do not have direct cognitive support for visualization creation, we excluded programming languages, Graphics API's and GUI development systems such as Processing [Pro], Java2D, and Piccolo [BGM04].

3.2. Design

We introduce the selected visualization tools, and the support they provide for visual mappings [CMS99]. Visual mapping is a key to visualization expressiveness and effectiveness [SJ07]. It requires four essential *elements*: visual objects that show data graphically, a mechanism to bind visual object properties to data, a mechanism that supports complex arrangement of visual objects, and an environment that helps designers implement visual mappings. We compare the selected tools according to these four elements. Table 1 provides a summary for the comparison.

We compare the selected tools using a custom x-y graph (Figure 1). The example shows high readings of temperature in a given city. The readings are taken from 1 June 2011 to 1 October 2011. The dots represent the readings, and so far it looks like a conventional chart. However, we want to customize the colour of the dots. If the dot is showing the highest temperature, it is black. Otherwise, if the dot is showing a temperature greater than 25, it is red. The rest of the dots are orange. Although the example is simple, it was selected because it can be made with the selected tools without advanced knowledge. Further, it does not favour any of the tools.

We use the CDs framework to evaluate the tool support for custom visualizations. To effectively use the framework, we need to understand the nature of the task of implementing a custom visualization, and which cognitive dimensions are important to look at. The task qualifies as an exploratory task since it is a combination of incrementation (adding information without altering the structure) and modification (changing the existing structure possibly without adding new content), and the desired end might not be known in advance [GB98]. The cognitive dimensions that are important to look at when designing or evaluating tool support for exploratory tasks are: abstractions, hidden dependencies, premature commitment, progressive evaluation, viscosity, visibility, and juxtaposability [GB98]. Based on that, we selected the aforementioned dimensions for the evaluation.

Ideally, systems that support exploratory tasks (e.g. implementing a custom visualization) should have low viscosity, few hidden dependencies, few premature commitments, few abstractions, and high visibility and juxtaposability [GB98]. We use this as a criterion for evaluating how well the tools support custom visualizations.

		Prefuse	**Improvise**	**Protovis** (Protoviewer)	**Uvis** (Uvis Environment)
Environment		N/A	• WYSIWYG • Selection	• WYSIWYG • Selection	• WYSIWYG • Direct manipulation • Error highlighting • Inspector
Visual objects	**Primitive**	Java Shape (Ellipse, etc.)	Glyph (Rectangle, Oval, etc.)	Mark (Dot, Bar, Wedge, etc.)	Ellipse, Triangle, Box, etc.
	Specialized	Graph, TreeMap, etc.	BarChart, MatrixView, etc.	N/A	TimeScale, Spiral, etc.
Binding visual properties to data		• **Class** (Action)	• **Expression** (Projections)	• **Expression** (Anonymous Functions)	• **Expression** (Formulas)
Complex layout		• **Visual objects** (e.g. Tree Map.) • **Layout class** (e.g. force directed.)	• **Visual objects** (e.g. Tree, Graph, etc.)	• **Layout property** (e.g. tree map, force directed.)	• **Visual objects** (e.g. tree map, TreeNode.)

Table 1: *A summary of the selected tool approaches*

4. Approaches

4.1. Prefuse

Prefuse is a visualization toolkit suited for advanced visualizations (e.g. tree maps, sunburst, etc.). It provides modules (e.g. functions, layout classes, etc.) suited for various visualization tasks. To create visualizations, the designer writes Java code that uses the modules.

Prefuse provides primitive geometric visual objects (e.g. rectangles, ellipses, etc.) and specialized visual objects that are suited for specific visualizations such as trees and graphs. Prefuse provides many Action subclasses that bind visual properties to data. The designer can use specialized objects or a Layout subclass to accomplish complex arrangement of visual objects.

Example: Figure 2 shows the specifications of a custom x-y graph with Prefuse. First, a visualization object is created and bound to data (lines 1-3). Prefuse uses an `AxisLayout` abstraction that supports plots (lines 4 and 5). The to-be-visualized fields are passed in the `AxisLayout` constructor.

Actions bind visual properties to data (lines 8-10). There are many types of actions. For instance, `ColorAction` can make a colour property (e.g. border colour or background colour) show data. The `orangeColor` variable makes all visual objects orange (line 8). It sets the `FILLCOLOR` (background colour) of all visual objects to orange. However, the `redColor` variable makes objects that conform to a condition red (line 9). The condition is specified by a predicate that checks if the temperature fields are greater than 25. This predicate is specified at line 6. The actions are attached with the visualization object (lines 11-15).

The axes are positioned using a `RenderFactory` class (lines 17-20), and tick marks of the axes are generated using an `AxisLabelLayout` class (lines 21-24). The tick marks are associated with their corresponding axes (line 25).

Finally, ellipses are chosen as visual objects to represent the temperature readings, and associated with the axes defined previously (lines 26-28).

Summary: There are many abstractions that designers have to know and create (e.g. `AxisLayout`, `RenderFactory`). The separation of actions from the visualizations, predicates, and their properties can facilitate the management of code and allow reuse, but might increase the gap between the problem and the solution (Norman's gulf of execution [Nor86]). A designer might be wondering "which visual object or property does this action relate to?".

```
a
1.  Visualization vis = new Visualization();
2.  Display display = new Display(vis);
3.  vis.add("HighReading", data);

b
4.  AxisLayout x_axis = new AxisLayout(" HighReading ", "Date", Constants.X_AXIS, VisiblePredicate.TRUE);
5.  AxisLayout y_axis = new AxisLayout(" HighReading ", "Temperature", Constants.Y_AXIS, VisiblePredicate.TRUE);

c
6.  Predicate p1 = (Predicate)ExpressionParser.parse("Tempreature > 25");
7.  Predicate p2 = (Predicate)ExpressionParser.parse("Tempreature=MAX(Tempreature)");

8.  ColorAction Orangecolor = new ColorAction("data", VisualItem.FILLCOLOR, ColorLib.getColor(255, 128,0));
9.  ColorAction redColor =new ColorAction("data", VisualItem.FILLCOLOR,p1, ColorLib.getColor(255, 0,0));
10. ColorAction blackColor =new ColorAction("data", VisualItem.FILLCOLOR, p2, ColorLib.getColor(255, 0,0));

d
11. ActionList draw = new ActionList();
12. draw.add(x_axis);
13. draw.add(y_axis);
14. draw.add(redColor); draw.add(orangeColor); draw.add(blackColor);
15. vis.putAction("draw", draw);
16.

e
17. vis.setRendererFactory(new RendererFactory()
18. { AbstractShapeRenderer sr = new ShapeRenderer(7);
19. Renderer arY = new AxisRenderer(Constants.FAR_Right, Constants.CENTER);
20. Renderer arX = new AxisRenderer(Constants.CENTER, Constants.FAR_BOTTOM);});

21. AxisLabelLayout x_labels = new AxisLabelLayout("xlab", x_axis);
22. AxisLabelLayout y_labels = new AxisLabelLayout("ylab", y_axis);
23. draw.add(x_labels);
24. draw.add(y_labels);
25. y_axis.setRangeModel(new NumberRangeModel(-10, 50, -10, 50));

f
26. final Ellipse2D TempreatureEllipse = new Ellipse2D.Double();
27. x_axis.setLayoutBounds(TemperatureEllipse);
28. y_axis.setLayoutBounds(TemperatureEllipse);
```

Figure 2: *Creating a custom x-y graph with Prefuse. a) binding the visualization to data, b) defining time and numeric axes, c) defining a conditional visual mapping, d) associating the visual mappings with the visualization, e) defining tick marks and associating them with the axes. f) defining ellipses representing the temperature readings*

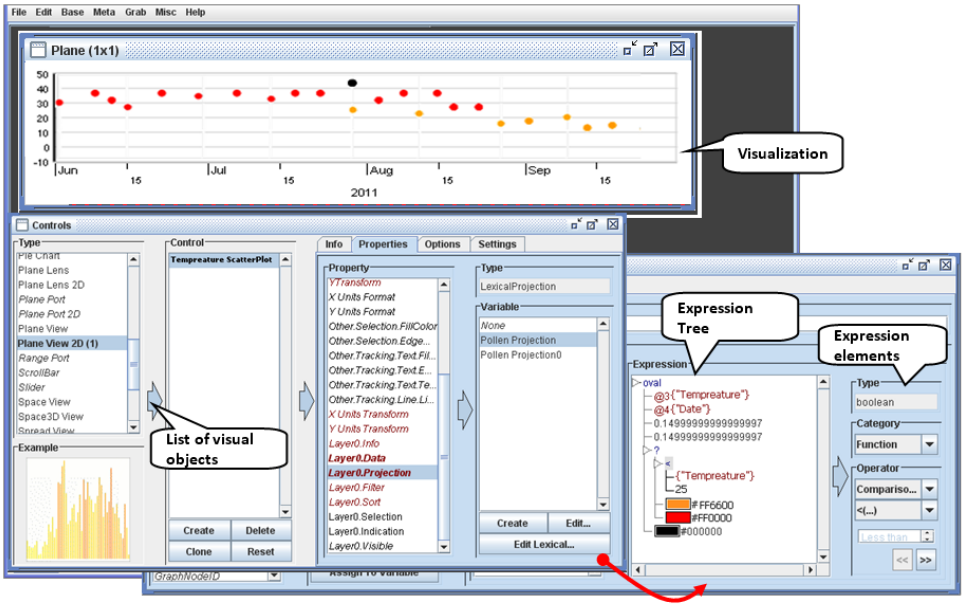


Figure 3: *Creating a custom x-y graph with Improvise*

```
a
1.  var w = 400, h = 400;
2.  var vis = new pv.Panel()
3.  .width(w)
4.  .height(h);

b
5.  var y = pv.Scale.linear(-10, 50).range(0, h),
6.  vis.add(pv.Rule)
7.  .data(y.ticks())
8.  .bottom(y)
9.  .strokeStyle(function(d) d ? "#eee" : "#000")
10. .anchor("left").add(pv.Label)
11. .text(y.tickFormat);

12. var x = pv.Scale.linear(new Date(2011,6,1), new Date(2011,10,1));
13. vis.add(pv.Rule)
14. .data(x.ticks())
15. .left(x)
16. .strokeStyle(function(d) d ? "#eee" : "#000")
17. .anchor("bottom").add(pv.Label)
18. .text(x.tickFormat);

c
19. vis.add(pv.Panel)
20. .data(HighReading)
21. .add(pv.Dot)
22. .left(function(d) x(d.Date))
23. .bottom(function(d) y(d.Tempreature))
24. .fillStyle(function(d) pv.max(data.Tempreature) = d.Tempreature ?
25.  "#000000" : d.Tempreature > 25? "#FF0000" : "#FF6600");
```

Figure 4: *Creating a custom x-y graph with Protovis. a) defining the visualization. b) defining the numeric (temperature) and time scales (axes). c) defining dots and visually mapping them to temperature and date fields according to the scales*

4.2. Improvise

Improvise is a visualization system that mainly supports coordinated visualizations. It provides primitive and specialized properties whose visual properties can show data using expressions. The expressions can be conditional, logical, mathematical, etc. Improvise provides specialized objects that support complex layouts such as trees. Designers use a development environment to create a visualization. They navigate from panel to panel to accomplish visual mappings. Each panel has a distinct purpose. For instance, one panel shows the available visual objects and their properties. Another panel shows the variables that can be used in expressions.

Example: To define a x-y graph, the designer chooses `Plane View 2D` object from the list of visual objects. To define visual mappings for the visual object, the designer chooses `Layer.Projection` from the list of properties. He clicks "Create" to create a new projection (visual mapping). This leads him to a new panel (`Lexicon`) where he can define expressions.

We want to define this expression for the background colour property.

```
Temperature > 25 ? "red": "orange"
```

This expression has to be built step-by-step using combo boxes that provide the available Expression elements (Figure 3). First, the designer creates the conditional part of the expression by choosing `Function` from the `Category` combo box, and `Other` and `?(boolean,Color,Color)`. Improvise shows the result as a conditional expression tree with default colours as results for the true and false expressions.

Second, the designer can manipulate the conditional statement parts by clicking the tree nodes. To create a comparison condition, the designer chooses `Function` from the `Category` combo box, and `Comparison` and `>(...)` from the `Operator` combo boxes. Third, to make one of the nodes refer to the `Temperature` field, the designer clicks the node and chooses `Attribute` from the `Category` combo-box. Improvise displays the available fields, and the designer just selects (clicks) it.

Summary: In general, visual mappings rely heavily on dialogues. For instance, even a simple expression takes long to create. The environment forces the designer to use combo-boxes that have the expression elements. It is not easy to find the expression elements. Moreover, the longer the expression, the harder it is to read.

4.3. Protovis

Protovis is a JavaScript-based visualization toolkit that uses a declarative domain specific language that can map data into primitive visual objects (e.g. bar, dot, etc.) and their properties. Protovis does not provide specialized objects for visualizations with complex layouts, but provides a Layout property that can arrange visual objects in various ways. Protovis

can be extended with a development environment called Protoviewer [Aka11].

Example: Figure 4 shows the specifications of a custom x-y graph with Protovis. First, a visualization object is defined (lines 1-4). Protovis uses non-visual scale classes for creating time and numeric axes (lines 5-11). The designer uses them to generate tick data. `Rule` and `Label` visual objects are used to draw the axes based on the tick data (lines 12-18).

`Dot` objects are bound to data (an array that corresponds to the `HighReading` table) (lines 19-21). The `Left` and `Bottom` properties position the `Dot` objects horizontally and vertically (lines 22 and 23). The designer specified expressions for the two properties that call functions provided by the scales that calculate the positions based on temperature and date fields. Finally, a conditional expression for the `FillStyle` (background colour property) sets the colour of dots that show the highest temperature black. Otherwise, it sets the colour red for dots showing temperature greater than 25 red. Otherwise, they are orange (lines 24 and 25).

Development Environment (Protoviewer): The visualization can be built with the Protoviewer development environment (Figure 5). This has several advantages. Designers can see the resulting visualization immediately as they are modifying the source code. Moreover, clicking a visual object, designers can view the position values (x and y) of the object. This can help inspecting the object.

Summary: Protovis provides non-visual scale classes that facilitate the construction of axes. The axes are not defined directly. Instead, primitive objects such as `Label` and `Rule` are used for drawing the axes. This separation increases flexibility (e.g designers might obtain a custom axis in this way), but increases the steps of such a common task. Unlike Prefuse actions, the declarative expressions for the `Dot` visual objects are not separated from the visual properties. This increases visibility and understandability.

4.4. Uvis

Uvis is a visualization tool that allows creating custom visualizations based on relational data. To construct a visualization, the designer drags and drops visual objects (building blocks), binds their visual properties with data using spreadsheet-like formulas, and the environment shows the resulting visualization in a WYSIWYG fashion. To see properties of a visual object, the designer selects (clicks) the visual object, and the environment shows the visual properties of the object in a property grid. To make a visual property (e.g. `Height`) depend on data, the designer types a declarative spreadsheet-like expression (formula). Uvis supports conditional, logical, and mathematical formulas. Moreover, a formula can refer to data fields, visual properties, and functions. Uvis supports complex algorithms with specialized objects such as tree maps.

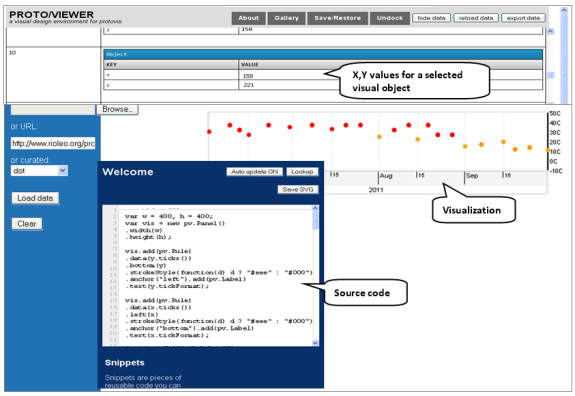


Figure 5: *Creating a custom x-y graph with Protovis environment (Protoviewer)*

```
a
1.  Form: form
2.  Width:600
3.  Height:250

b
4.  HTimeScale: tScale
5.  Range: #1-6-2011#, #1-10-2011#
6.  Canvas: canvas
7.  Width: 530
8.  Top: 150
9.  Left: 10

10. VNumericScale: nScale
11. Range: -10, 50
12. Orientation: Orientation.Right
13. Height: 90
14. Top: 40
15. Left: 10

c
16. Ellipse: Temperature Ellipse
17. Rows: HighTemperature
18. Top: nScale!Position(Temperature) - Height/2
19. Left: tScale(Date)!Position(Date) - Width/2
20. BackColor: MAX(Temperature)=Temperature ? Black :
21. Temperature >25? Red : Orange
```

Figure 6: *The specifications of the custom x-y graph with Uvis*

Example: Figure 6 shows the textual specification of the custom x-y graph with Uvis and Figure 7 shows the environment where the chart was developed.

To create the time and numeric axes, the designer dragged `HTimeScale` and `VNumericScale` visual objects from the toolbox and dropped them on a form. The designer moved and resized them until they looked right. The environment sets position properties (i.e. `Top`, `Height`, etc.) accordingly. To define the range of time and numbers the scales show, the designer typed the value of the `Range` property in the property grid (lines 5 and 11 in Figure 6).

To create dots representing the temperature reading, the designer drags and drops an `Ellipse`. The designer typed formulas for the position properties (`Top` and `Left`). The formulas call position functions provided by the scales to calculate the positions based on temperature and date fields

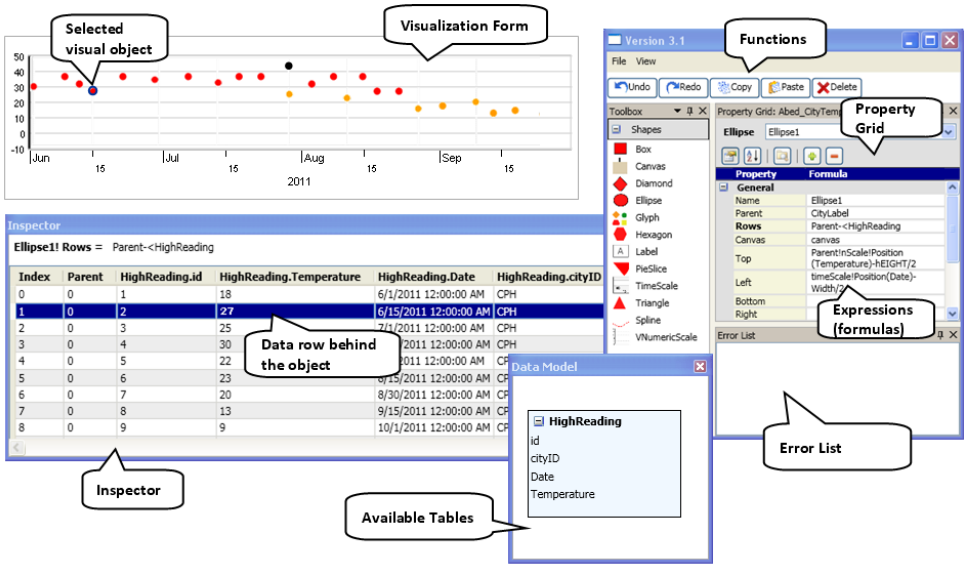


Figure 7: *Uvis environment*

(lines 18 and 19 in Figure 6). Finally, a conditional expression for the `BackColor` (background colour property) sets the colour of ellipses (line 21 in Figure 6).

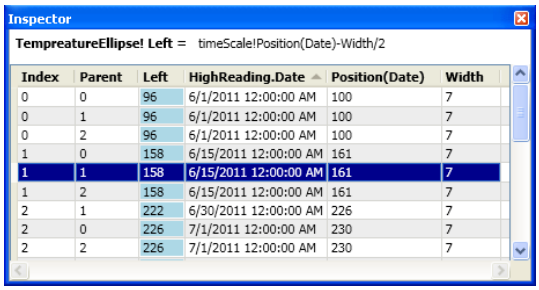
Development Environment: The environment has several advantages. Designers can drag, drop, resize visual objects (Direct manipulation), and they can see the resulting visualization immediately as they are updating the expressions.

The inspector shows data for a bundle of visual objects. It shows the data rows behind the visual objects (Figure 7). Further, it shows the values of an expression and its sub-expressions (Figure 8).

Summary: Unlike Prefuse, Protovis, and Improvise, Uvis deals only with visible visual objects. Like Protovis, Uvis uses declarative expressions that directly define the visual properties, but there is no need to define variables, and the sequence of specifying the expressions is free. Like Improvise, the environment shows the available visual objects, but it allows the designers to drag, drop, and resize them (as long as the position and size properties do not have dynamic expressions) rather than textually setting them.

5. Cognitive Dimensions of Notations

This section evaluates how the selected tools perform in a relevant set of cognitive dimensions. The dimensions themselves are not sufficient to make a judgement. Therefore, we make the judgement based on what is desirable for exploratory tasks such as implementing a custom visualization. For instance, exploratory tasks require high visibility. Hence, tools that have high visibility rate high.



Index	Parent	Left	HighReading.Date	Position(Date)	Width
0	0	96	6/1/2011 12:00:00 AM	100	7
0	1	96	6/1/2011 12:00:00 AM	100	7
0	2	96	6/1/2011 12:00:00 AM	100	7
1	0	158	6/15/2011 12:00:00 AM	161	7
1	1	158	6/15/2011 12:00:00 AM	161	7
1	2	158	6/15/2011 12:00:00 AM	161	7
2	1	222	6/30/2011 12:00:00 AM	226	7
2	0	226	7/1/2011 12:00:00 AM	230	7
2	2	226	7/1/2011 12:00:00 AM	230	7

Figure 8: *The inspector showing property Left expression values*

5.1. Abstractions

The abstractions dimension assesses the abstractions that encapsulate implementation details and the mechanism to manage them. Although abstractions can make the specifications shorter and sometimes fit the domain better, systems that require learning many abstractions have an *abstraction barrier*. Exploratory tasks do not tolerate many abstractions.

Prefuse is an example of a system that has an abstraction barrier. For instance, there are many subtypes of `Layout`, `RenderFactory`, and `Action` to learn. The abstractions can be extended programmatically by Java programming, but this requires in-depth knowledge of Java.

Protovis has fewer abstractions to learn than Prefuse, but some programming abstractions (e.g. variables, anonymous functions) are necessary to learn. Protovis abstractions can be extended programmatically with JavaScript.

Like Prefuse, Improvise has many abstractions. For instance, there are many panels and expression parts (e.g. conditional statements, functions, etc.) and the designers need to be aware of their meaning, and how to manipulate them, etc. Like Prefuse, Improvise abstractions can be extended with Java.

Uvis formulas resemble spreadsheet expressions, but obviously have more abstractions than spreadsheets. For instance, a Uvis formula can refer to data fields, visual properties, etc. However, Uvis has relatively few abstractions. For instance, there are no variables and rendering objects. Uvis does not allow defining new abstractions.

5.2. Hidden Dependencies

The hidden dependencies dimension assesses whether dependencies between entities are hidden or visible. Hidden dependencies slow down information finding and can potentially increase the risk of error. Exploratory tasks tolerate only a few hidden dependencies.

Most Prefuse abstractions have hidden dependencies. For example, the layout action implicitly overrides a specific visual mapping of size and position properties.

Protovis expressions can depend on variables. Such dependencies can be hard to see in textual specifications. More advanced visualizations use layout classes that position visual items implicitly (e.g. tree maps), or some operators such as "Parent" and "Sibling" that have hidden dependencies.

In Improvise, it is hard to derive the elements of an expression, particularly, if the expression contains variables or other sub-expressions. These can be viewed in other panels.

Uvis formulas can depend on other visual properties. The properties can have their own formulas, and so on. When designers change an expression, it is hard to know the implications of such a change. Furthermore, more advanced visualizations such as hierarchical visualizations use operators (e.g. `Parent`) that result in hidden dependencies.

All the surveyed tools except for Uvis do not explicitly show which particular visual property depends on which field. The Uvis environment shows that using the inspector (Figure 8).

5.3. Premature Commitment

The premature commitment dimension assesses whether there are any constraints on the order in which tasks must be accomplished. Premature commitment is harmful for exploratory tasks.

Since the specifications are program-like, Prefuse and Protovis impose constraints on the sequence in which visualizations are defined. For instance, if a property depends on another, the independent one has to be defined first.

Improvise imposes a strict sequence on how some things are done. Constructing the expression step-by-step is an example of strict sequencing, and having to navigate from panel to panel to carry out visual mappings is another one.

Uvis specifications are sequence-free. At run time, the kernel finds out the sequence of execution. If the designer types a formula that refers to a property that does not exist yet, Uvis kernel flags an error, but the application still runs.

5.4. Progressive Evaluation

The progressive evaluation dimension assesses how easy it is to evaluate and obtain feedback on an incomplete task. Progressive evaluation is important for exploratory tasks.

In Prefuse, it is not easy for a designer to obtain visual feedback of the specifications. The source code has to be run in another setting to obtain feedback.

Improvise bridges that gap with an immediate visual feedback feature. However, the visual feedback can be overshadowed with many editing panels.

Protoviewer and the Uvis environment provide a separate design panel that is updated immediately when the specifications are changed. The Uvis environment provides similar kinds of feedback as traditional environments such as highlighting erroneous formula parts, error, and warning lists. In addition, the environment shows the formula values in a separate panel that is updated when the formula changes (Figure 8).

5.5. Viscosity

The viscosity dimension assesses the cost of making small changes. It is costly to make a small change in viscous systems. Viscosity is harmful for exploratory tasks. We consider two types of viscosity. First, *repetitive viscosity* means a single goal-related change which requires many repetitive actions. Second, *knock-on viscosity* means a change in one part affects other related parts.

Prefuse is based on an object oriented language (Java.) Hence, inheritance can reduce repetitive viscosity. For instance, a change can be made in a parent class rather than all inheriting classes. Modern development environments can help with small knock-on changes such as changing a variable name that is used in many places (re-factoring.) Nevertheless, changing Prefuse specifications requires in-depth knowledge of the language constructs and programming concepts.

Like Prefuse, the Protovis language has low-repetitive viscosity since it supports inheritance for visual objects. Moreover, Protovis allows other changes easily, for instance, changing the visual object type. The environment (Protoviewer) does not have support for making changes.

Designers who are experienced with Improvise might find some things easy to change. For instance, variables that are referred to from many expressions can be changed in one setting. Otherwise, Improvise is highly viscous. For instance, changing some specialized visual object types (e.g. Plane View) is not possible. In general, a change in Improvise requires navigating across panels.

Like spreadsheets, simple visualizations in Uvis have low viscosity. However, viscosity grows with size. Uvis does not support inheritance, but designers can add properties that have formulas that other visual objects can refer to. In such a case, a change is only required in the designer property. Since Uvis formulas can refer to other formulas elsewhere, a change in one formula might affect other dependant formulas. The Uvis environment shows errors that result from such a change.

5.6. Visibility and Juxtaposability

The visibility dimension assesses the ability to view data components easily. Juxtaposability assesses the ability to view two similar components side by side. The two dimensions are generally discussed together due to similarity. Both dimensions are important for exploratory tasks.

What data components would a designer want to view when implementing a custom visualization? Many can be considered important. Examples include the currently-designed visualization, the available visual objects and their properties, the visual mappings, the available data, the visualized data, and errors. What needs to be viewed varies from task to task and designer to designer, but a possible solution is to give designers the ability to show or hide components.

Even if Prefuse is integrated with a development environment, only a few components can be visible in one setting. Traditional environments show the source code, the available visual objects, and a list of errors in one setting. However, the designer has to view the currently-designed visualization in another setting.

Protoviewer shows the currently-designed visualization as well as the specifications behind it. Furthermore, designers can view the position property values of a single selected visual object at a time. Protoviewer does not provide support for comparing the specifications of two similar visual objects.

Improvise shows the currently-designed visualization, but it can be over-shadowed by the editing panels. A panel can only show one expression at a time, and it occupies a lot of space. This does not allow comparing many expressions. Further, many data crucial for the task (e.g. data fields) are buried in combo boxes.

Uvis shows the currently-designed visualization, the properties (and the expressions defining them) of a selected visual object, and a list of errors. Upon selecting a visual object, Uvis shows the data behind that particular object. Further, to allow comparison, the data from other visual objects from the same data source are shown as well. It is also possible to see the defining expressions of all properties of a selected visual object. However, it is not possible to see expressions of two visual objects at the same time.

6. Conclusion

We summarize the findings of the comparative analysis and the evaluation with CDs as follows.

- **All the surveyed tools** suffer from low juxtaposability and high hidden dependencies with slightly different degrees.
- **All the surveyed tools except for Uvis** suffer from high premature commitment and low visibility with slightly different degrees.
- **Prefuse** uses a programmatic approach that relies on specialized modules. The main strength of this approach is the breadth of visualizations it can express due to the many modules it provides. However, there are many abstractions to learn even to construct a simple example like a custom x-y graph. Furthermore, even with a development environment, the approach suffers from low progressive feedback.
- **Improvise** uses an approach that is heavily dependant on dialogues (panels). The main strength of this approach is that the tool provides useful visual objects tailored for some tasks. However, the functionalities are not easy to find. For instance, the conditional expression is buried in a combo box item called "Other".
- **Protovis** uses an approach that relies on primitive visual objects and declarative expressions. The main strength of the approach is that the properties of the visual objects are directly specified. No middle-ware objects (e.g. Prefuse actions) are needed to link visual properties with expressions. However, some programming abstractions (e.g. variables) are still needed to learn the language.
- **Uvis** uses an approach that relies on declarative spreadsheet-like formulas for visual mappings, and a dedicated environment with many features (e.g. drag-drop, visual feedback, etc.). The approach has high visibility, low premature commitment, and relatively few abstractions to learn. However, the approach still suffers from high viscosity (especially when it is a large-sized application).

References

[Aka11] AKASAKA R.: Protoviewer: a web-based visual design environment for protovis. In *ACM SIGGRAPH 2011 Posters* (New York, NY, USA, 2011), SIGGRAPH '11, ACM, pp. 85:1–85:1. 24

[BGM04] BEDERSON B. B., GROSJEAN J., MEYER J.: Toolkit design for interactive structured graphics. *IEEE Trans. Software Eng. 30*, 8 (2004), 535–546. 20

[BH09] BOSTOCK M., HEER J.: Protovis: A graphical toolkit for visualization. *IEEE Trans. Vis. Comput. Graph. 15*, 6 (2009), 1121–1128. 19, 20

[BOH11] BOSTOCK M., OGIEVETSKY V., HEER J.: D^3 data-driven documents. *IEEE Trans. Vis. Comput. Graph. 17*, 12 (2011), 2301–2309. 20

[Byr99] BYRD D.: A scrollbar-based visualization for document navigation. In *ACM DL* (1999), pp. 122–129. 19

[CMS99] CARD S. K., MACKINLAY J. D., SHNEIDERMAN B.: *Readings in information visualization - using vision to think*. Academic Press, 1999. 20

[Fek04] FEKETE J.-D.: The infovis toolkit. In *INFOVIS* (2004), pp. 167–174. 19

[Fla] FLARE: Data visualization for the web. http://flare.prefuse.org/. [Online; accessed June-2012]. 20

[GB98] GREEN T., BLACKWELL A.: Cognitive dimensions of information artefacts: a tutorial. *T.R.G. Green and A.F. Blackwell 1*, 2 (1998). 20

[Gre89] GREEN T. R. G.: Cognitive dimensions of notations. In *Proceedings of the fifth conference of the British Computer Society, Human-Computer Interaction Specialist Group on People and computers V* (New York, NY, USA, 1989), Cambridge University Press, pp. 443–460. 19

[HC12] HARGER J., CROSSNO P.: Comparison of open-source visual analytics toolkits. In *SPIEConference on Visualization and Data Analysis* (2012). 20

[HCL05] HEER J., CARD S. K., LANDAY J. A.: prefuse: a toolkit for interactive information visualization. In *CHI* (2005), pp. 421–430. 19, 20

[KL12] KUHAIL M. A., LAUESEN S.: Customizable visualizations with formula-linked building blocks. In *GRAPP/IVAPP* (2012), pp. 768–771. 20

[KPL12] KUHAIL M. A., PANDAZO K., LAUESEN S.: Customizable time-oriented visualizations. In *ISVC (2)* (2012), pp. 668–677. 20

[KPX13] KOSTAS PANTAZOS MOHAMMAD A. KUHAIL S. L., XU S.: Constructing visualizations with a development environment. In *Submitted to: VDA* (2013). 19

[LKP*13] LAUESEN S., KUHAIL M. A., PANDAZOS K., XU S., ANDERSEN M. B.: A drag-drop-formula tool for custom visualization. 20

[Nor86] NORMAN D. A.: *User Centered System Design: New Perspectives on Human-computer Interaction*. CRC Press, 1986. 21

[PGB02] PLAISANT C., GROSJEAN J., BEDERSON B. B.: Spacetree: Supporting exploration in large node link tree, design evolution and empirical evaluation. In *INFOVIS* (2002), pp. 57–64. 19

[PL12] PANTAZOS K., LAUESEN S.: Constructing visualizations with infovis tools - an evaluation from a user perspective. In *GRAPP/IVAPP* (2012), pp. 731–736. 19

[Pro] PROCESSING:. http://processing.org/. [Online; accessed Aug-2012]. 20

[SEH00] SUTCLIFFE A. G., ENNIS M., HU J.: Evaluating the effectiveness of visual user interfaces for information retrieval. *Int. J. Hum.-Comput. Stud. 53*, 5 (2000), 741–763. 19

[SJ07] SEARS A., JACKO J. A.: *The Human-Computer Interaction Handbook: Fundamentals, Evolving Technologies and Emerging Applications*. CRC Press, 2007. 20

[Wea04] WEAVER C.: Building highly-coordinated visualizations in improvise. In *INFOVIS* (2004), pp. 159–166. 20

SIGRAD 2012
A. Kerren and S. Seipel (Editors)

Glint: An MDS Framework for Costly Distance Functions

S. Ingram[†1] and T. Munzner[1]

[1]University of British Columbia, Canada

Abstract

Previous algorithms for multidimensional scaling, or MDS, aim for scalable performance as the number of points to lay out increases. However, they either assume that the distance function is cheap to compute, and perform poorly when the distance function is costly, or they leave the precise number of distances to compute as a manual tuning parameter. We present Glint, an MDS algorithm framework that addresses both of these shortcomings. Glint is designed to automatically minimize the total number of distances computed by progressively computing a more and more densely sampled approximation of the distance matrix. We present instantiations of the Glint framework on three different classes of MDS algorithms: force-directed, analytic, and gradient-based. We validate the framework through computational benchmarks on several real-world datasets, and demonstrate substantial performance benefits without sacrificing layout quality.

Categories and Subject Descriptors (according to ACM CCS): I.3.3 [Human-centered Computing]: Visualization—Visualization systems and tools

1. Introduction

Multidimensional Scaling, or MDS, is a method for positioning the points of a dataset into a user-specified, low-dimensional space. The technique is used when the given description of the points is overly verbose, making visual analysis unwieldy or algorithmic analysis intractable. Input dataset descriptions processed by MDS come in two types: **points**, where each point is described by an equal number of spatial coordinates, or a **distance matrix**, where the rows and columns of the matrix represent a nonnegative value computed by a distance function of the two points.

Plotting the low-dimensional MDS output enables visual analysis of the proximity relationships between data points that are obscured in high-dimensions. This technique is employed in psychophysics, marketing research, and unsupervised learning [BG05, Gre75, HTF09]. MDS visualizations can be readily incorporated into other interactive applications [IMS12], providing a rich overview of the data.

All MDS algorithms work by minimizing an objective function quantifying the distortion of the points in the low-dimensional space relative to their original input configuration. Though the different MDS algorithms compute coordinates in a wide variety of ways, in each case the computational work can be divided into two parts: **distance calculation**, where the inter-point distances are calculated from the input points, and **layout calculation**, which reads the computed high-dimensional distances and positions the points in the low-dimensional space.

The contribution of this paper is Glint, an iterative algorithm framework for automatically minimizing distance calculation in MDS. Structurally, Glint forms an outer loop around a modified MDS algorithm. It starts with an empty distance matrix, densifying the matrix as the outer loop iterates, automatically terminating when the MDS layout is stable. Glint separates the distance calculation portion of the MDS algorithm from layout calculations and provides an automated termination procedure.

The time cost of individual high-dimensional distance calculations have a profound effect on the run time of an MDS algorithm. Even for an efficient metric like the 10-dimensional Euclidean distance function, the time spent calculating high-dimensional distances occupies almost 80% of the algorithm run time using the Glimmer force-directed MDS algorithm [IMO09]. Many real-world problems where MDS is used require more costly distance functions than the

† E-mail: {sfingram,tmm}@cs.ubc.ca

Euclidean case. In these more expensive cases, total distance costs occupy more than 99% of MDS run time using the same algorithm. Thus, an efficient MDS algorithm should seek to minimize the *total* work done, minimizing the sum of both the distance and layout work.

Previous work has assumed that individual distance computations are fast to calculate and thus has not sought to automatically resolve the balance between distance and layout work. Current fast MDS algorithms that handle distance matrices either compute many more distances than necessary [IMO09], or leave the total number of distances to compute as a tuning parameter and so do not have a fully automatic way to terminate [BP07, dST04]. The Glimmer algorithm is an example of the overcomputation shortcoming [IMO09]. It computes an *iterative* MDS approximation using force-directed heuristics. The Glimmer minimization strategy defines a cheap iteration and then iterates until convergence is detected. Within each iteration, both distance calculations and layout calculations are done. Glimmer automatically chooses the number of distance calculations to make before terminating, but computes more than are strictly necessary. The Pivot MDS algorithm is an example of the termination shortcoming [BP07]. It computes a one-step *analytic* MDS approximation. The MDS work is cleanly divided between distance calculation up front followed by a single contiguous layout calculation. Pivot MDS computes all the distances it uses up front, but does not know how many to select.

The above examples motivate a synthesis of the benefits of the two algorithms, keeping the automatic termination of algorithms like Glimmer while separating the distance work from the layout work as in algorithms like Pivot MDS. The goal of Glint is thus to not only compute far fewer distances than the iterative approximation, but also to remove the tuning parameter from the analytic approximation.

To demonstrate the generality and robustness of the Glint approach, our contribution includes Glint instantiations for three very different classes of MDS algorithm: force-directed, analytic, and gradient-based. We present the design of the Glint components for each instantiation, where each is tailored to the requirements of the underlying MDS algorithm. We show that these Glint instantiations drastically reduce total run time on datasets with costly distance functions without penalizing the final layout quality.

2. Distances In MDS

The distances between the points in a low-dimensional MDS solution are intended to closely model those in the high-dimensional input dataset. The core premise of MDS is that the input contains redundant information, allowing for correct output even with an incomplete set of distances as input. Glint exploits this redundancy by iteratively constructing a subset of distances that is as small as possible. This section describes two issues concerning these distances: the existence and effect of expensive distance functions, and how sparse the input distance matrix can be.

2.1. Expensive Distance Functions

Minimizing the total number of distances computed is especially important when the time spent computing distances dominates the time spent computing the layout. Many real-world applications involve datasets with expensive distance functions. Even the straightforward Euclidean distance metric can be costly if the number of dimensions is large enough, for example in the millions. In image processing, the Earth Mover's Distance, or EMD, compares the similarity of color distributions between images and is useful for ranking images for querying and nearest-neighbor-type calculations [RTG00]. Its calculation requires solving a linear program, often a costly operation relative to the layout calculation per point. Computational complexity is not the only reason for distance calculation cost. Distances based on database lookups are costly due to the relative speed of disk I/O to memory reads. Distances that involve elicitation of human judgement can be the most costly of all, because the time scales of human response are so much longer than of automatic computation. Human-elicited distances are of interest in many domains; in a marketing example, a single distance is derived from the averaged similarity judgements elicited from survey takers comparing two items [LMF07]; in a psychophysics example, distances are derived from just noticeable differences in haptic stimuli [TM08].

In all of these cases, distance calculations can comprise well over 99.9% of the total time to compute the MDS layout. We will show that using the Glint framework can drastically reduce the time spent computing distances without compromising the final quality of the MDS layout.

2.2. Experimental Analysis of Sparse MDS Solutions

Spence and Domoney conducted a series of data experiments to determine if there could be an *a priori* way to select an optimal subset of distance matrix entries to compute prior to MDS layout [SD74]. Their experiments investigated the effect of controlling three factors pertaining to layout quality. The first two factors, the amount of noise in the distance measurement and the number of input data points, are given in practice. The last experimental factor they tested, which an algorithm can indeed control in practice, is distance matrix density, or how densely sampled the approximation of the distance matrix is compared to the full version.

The experiments resulted in two key findings that pertain to our work. First, only a fraction of the matrix, ranging from 20% to 60% of the distances on their example data, needed to be computed to accurately approximate the full layout. This finding verifies that the goal of minimizing distance computations is a reasonable one. Second, their results imply that

there is no direct way to assess in advance exactly how many distances need to be computed. We thus designed Glint to run online, determining the optimal number of distances to compute on the fly.

3. Related Work

MDS refers to an entire family of algorithms with different objective functions, computational complexities, and qualitative results [BG05, FC11]. The common thread is that they all minimize objectives that are some function of the difference between the Euclidean distances of the lower-dimensional layout coordinates and the magnitude of the original high-dimensional dissimilarities. Here, we discuss the four major classes of MDS algorithms in terms of their shortcomings in handling costly distances.

3.1. Coordinate-Based Algorithms

MDS input can take the form of a table of coordinates or a distance matrix. When the points are given as coordinates, the number of input dimensions m is often much smaller than the number of points N. The PLMP [PSN10] and LAMP [JCC*11] algorithms build on this assumption to rapidly compute low-distortion layouts for very large datasets. The profound acceleration that the algorithms achieve is hindered when the number of dimensions equals or exceeds the number of points, as is precisely the case when the input format is a distance matrix. Because Glint is designed for the distance matrix use case, coordinate-based algorithms are not suitable as components for Glint.

3.2. Analytic Algorithms

The original MDS algorithm, now called Classic MDS [Tor52], computed a one-step analytic minimum of an objective function called Strain. The algorithm relies on computing the full SVD of a dense N^2 matrix, and is therefore too computationally complex to be suitable for large datasets or problems with costly distance functions.

Several scalable Classic MDS approximation algorithms based on the Nyström approximation of the SVD have been presented [Pla05]. For example, both Pivot MDS [BP07] and Landmark MDS [dST04] work by having the user select a number of "pivot" or "landmark" points. These particular columns in the distance matrix are then computed and processed by the algorithm to map the remaining points into low-dimensional space.

The main drawback to this strategy is the manual nature of selecting the proper number of landmark points. The Pivot MDS authors suggest a human-in-the-loop strategy where the user iteratively adds landmarks until visually determining the stability of the layout. The Landmark MDS authors propose an iterative strategy based on cross-validation, but do not present any benchmarks for this termination criterion.

3.3. Force-Directed Algorithms

The Glimmer [Ing07, IMO09] algorithm and its antecedent, by Chalmers [Cha96], are MDS approximation algorithms that iteratively sample high-dimensional distances and proportionally nudge the layout points in the direction of the residual distances. The movement of the points is controlled using a dampened force-directed simulation heuristic.

While force-directed algorithms typically exhibit a rapid convergence to a minimum, they often suffer from computing more distances than are strictly necessary. The algorithms are designed to compute high-dimensional distances prior to each force simulation time step, regardless of whether enough distance information has already been sampled to achieve a quality layout. This oversampling becomes especially inefficient when distances are costly. As we show later in the paper, force-directed algorithms can sample fewer high-dimensional distances and suffer little to no degradation in quality.

3.4. Gradient Algorithms

Other MDS techniques use exact gradient information to calculate layout coordinates. Some of these algorithms use backtracking gradient descent on the Stress function [BSL*08], while the SMACOF algorithm [dLM09] minimizes a sequence of majorizing quadratic functions. These techniques are more costly but the most flexible, permitting weights and missing values, while also converging to a lower-error minimum than randomized techniques.

As shown in their application to graph drawing [GKN04, KHKS12], gradient techniques can harness a sparsely populated distance matrix as input with good results. However, like analytic approximation techniques such as Pivot MDS, the precise number of distances to compute in advance to converge to a quality minimum is left up to the practitioner.

4. Glint Algorithm Framework

Glint is an algorithm framework: an algorithm with modular components that are themselves algorithms. Figure 1 shows a diagram of these three components; each corresponds to a step in the Glint outer loop. Glint starts with an empty distance matrix and a random layout and then loops over the following three main steps to determine a final layout. First, in the *Densify Matrix* step, it selects a new subset of the distance matrix to compute with the distance matrix densification strategy **DS** and then updates the matrix with the computed values. In the *Lay Out Points* step, it updates the layout using the new distance information as input to the MDS layout algorithm **M**. Finally, in the *Check Convergence* step, it checks to see if the change in the objective function **S** is below a threshold ε. If convergence is detected, the last layout is returned, otherwise the loop repeats.

The MDS layout algorithm **M** takes as input a low-dimensional point configuration as the starting point and a

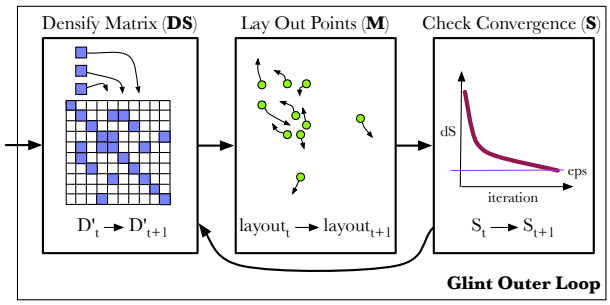


Figure 1: *Diagram of Glint execution.*

Algorithm 1 Pseudocode for Glint, with variable definitions.

function GLINT(ε)

layout ← RANDOMLAYOUT

while !converged **do**

$P_{t+1} \leftarrow \mathrm{DS}(P_t)$
$D'_{t+1} \leftarrow \mathrm{DISTANCE}(D'_t, P_{t+1}, d)$
$\mathrm{layout}_{t+1} \leftarrow \mathrm{M}(D'_{t+1}, \mathrm{layout}_t)$
$S_{new} \leftarrow \mathrm{S}(P_t, D'_{t+1}, \mathrm{layout}_{t+1})$
converged ← $|S_{old} - S_{new}|/S_{old} < \varepsilon$
$S_{old} \leftarrow \mathrm{S}(P_{t+1}, D'_{t+1}, \mathrm{layout}_{t+1})$

return layout

Variable	Description
P_t	the set of computed point pairs at iteration t
D'_t	the sparse distance matrix with nonzeros specified by P_t
S_t	scalar objective function value at iteration t
d	the distance function
t	the current Glint iteration
ε	termination threshold

sparse distance matrix. To qualify for use in Glint, **M** must possess three characteristics. First, it must be able to compute a layout given a distance matrix. Next, it must be able to handle an incomplete – that is, sparse – distance matrix, given the Glint strategy of gradual densification. Finally, **M** must compute a layout from a given starting position rather than starting from scratch each time, so that subsequent outer loop iterations start **M** from a state closer to the final layout configuration. We discuss **M** further in Section 5.1.

Controlling the density rate and pattern of the distance matrix is the job of the densification strategy **DS**. Some MDS algorithms, such as Pivot MDS and Landmark MDS are able to compute layouts with incomplete matrices, but the precise sparsity pattern of the incomplete distance matrix may be constrained. Because matrix sparsity pattern requirements vary from algorithm to algorithm, we must tailor the selection of computed distances **DS** to the MDS algorithm **M**. We discuss **DS** further in Section 5.2.

Glint requires a cheap, monotonic objective function **S** in order to measure layout convergence; it must also be tailored to the MDS algorithm **M**. It should not invoke a costly full stress function that requires computing all the high- and low-dimensional distances, which would obviate all performance benefits of the system. We discuss **S** further in Section 5.3.

4.1. Glint Outer Loop

The Glint algorithm consists of a single threshold-controlled loop, similar to algorithms like gradient descent where the algorithm loops until the change in measured progress becomes very small. Figure 1 lists pseudocode for Glint. The algorithm initializes with a random configuration of points `layout` and then iterates through the main loop. In the main loop, we first call the densification strategy **DS**. On the first call it constructs the initial sparsity pattern P_t of the distance matrix D'_t, and on subsequent calls it densifies the pattern by filling in more nonzero entries. Specifically, the sparsity pattern P_t contains the set of nonzero indices of the D'_t at time t. After selecting the precise entries to change, Glint updates the sparse distance matrix D'_t by invoking the distance function for each pair of points contained in P_t. Next, the MDS algorithm **M** runs to termination with the starting point `layout` and the input distance matrix D'. Glint itself terminates when the change in the objective function **S** is less than the termination threshold ε.

The objective function **S** takes three parameters: the sparsity pattern P that specifies the pairs of points over which we compare high-D and low-D distances, the distance matrix D' from which is read the distances specified by pairs in P, and the low-dimensional layout coordinates `layout` from which we compute the low-D distances. The reason for including the pattern P as an input instead of simply summing over the entirety of D' is subtle, but important. Glint terminates when the objective function converges; that is, when it stops changing between subsequent iterations. Thus, the objective function must compare results at time $t+1$ to results at time t. However, not only do the points in the layout change between iterations, but the number of terms in the distance function changes, because there are more nonzero entries in D'_{t+1} than in D'_t. To properly measure convergence, we need to compare functions with the same number of terms. Including the same sparsity pattern in the objective calculation ensures that we compare objective functions with equivalent terms at each iteration, by specifying which entries of the matrix to use. Thus, in the Figure 1 pseudocode, S_{new} is computed with the sparsity pattern from the previous iteration, P_t, to determine which entries to include in the computation, while using the actual values derived from the current layout at time $t+1$.

5. Glint Instantiations

A Glint *instantiation* substitutes implementations of three concrete components into the abstract framework of the Glint algorithm. We describe three Glint instantiations, one for each of the three different MDS algorithm families described in Section 3: force-directed, analytic, and gradient.

Several of the Glint instantiations require choosing input parameters, as discussed in detail below. Table 1 summarizes the default value of each parameter and our method for selecting it. It also includes our analysis of the tradeoffs, with the results for setting the parameter too small or too big.

5.1. Component M: MDS Algorithm

The **M** component takes as input the low-dimensional input coordinates and places them in a new configuration based on the current distance matrix D' as output. For the analytic instantiation we substituted the Pivot MDS algorithm [BP07] for **M**, and for the gradient implementation we substituted the SMACOF algorithm [dLM09] for **M**. The Pivot MDS algorithm is used without change, but the other instantiations require algorithm parameter choices or internal modifications which we detail in the following subsections.

5.1.1. Gradient-Based Instantiation

For the gradient-based instantiation, the SMACOF MDS algorithm has two tuning parameters: the inner termination threshold, ε, and the maximum number of inner-loop iterations before termination, `numIters`. We use the same value for ε as in the the main Glint algorithm.

We observed that the gradient of the stress function for very sparse input matrices quickly shrinks in proximity to a minimum. Setting `numIters` too large results in over-optimizing with incomplete distance information, while setting it too high leads to computing more distances than are necessary. We select 100 as a good balance over all our benchmarks between these two extremes.

5.1.2. Force-Directed Instantiation

In the force-directed instantiation, we substitute a modified version of the Glimmer [Ing07,IMO09] algorithm for **M**. We used the version of Glimmer that supports distance matrix calculations in addition to handling points [Ing07]. To make the Glimmer algorithm suitable as the **M** component, we must alter the randomized sampling regime used by the algorithm. In Glimmer, sampling is uniform and unconstrained over the entire distance matrix. Glint, however, only feeds a sparse subset of the distance matrix D' to **M** for each outer loop iteration. To compensate, we constrain Glimmer sampling to be uniform over the given nonzeros of the sparse distance matrix D'.

5.2. Component DS: Densification Strategy

The **DS** component determines which distances to compute at each Glint iteration. For each instantiation, we follow a strategy of adding `numDists` new distances per point to the matrix D'. By default, the `numDists` parameter is initially set to $\lceil \log_{10} N \rceil$.

Setting the `numDists` parameter to an overly small value would result in an objective function **S** change that is less than the termination threshold ε and thus an incorrect algorithm termination after the first iteration. A small `numDists` is analogous to performing gradient descent with too small a gradient step-size. To ensure `numDists` is large enough, we follow a simple strategy of doubling `numDists` during the first iteration until we achieve a change in the objective function **S** greater than ε.

The distribution of new distances across the matrix D' varies for each instantiation. We describe these distributions on a per-instantiation basis.

5.2.1. Gradient Instantiation

The gradient instantiation **DS** is the simplest of the densification strategies. At each iteration, the **DS** uniformly samples `numDists` distances per point without replacement.

5.2.2. Force-Directed Instantiation

The force-directed **DS** is similar to the gradient instantiation, except for a single modification addressing Glimmer point hierarchies. The Glimmer algorithm divides points into a pyramid of levels, with the fewest points contained in the top level and increasingly larger sets of points at lower levels [IMO09]. Sampling uniformly without replacement from the distance matrix would often lead to the case that, at the top level, several points will not have any distances computed between any of the other points in the top level, only distances computed to points in lower levels. To solve this problem, the force-directed **DS** samples `numDists` distances without replacement once for the points contained in each level. The sampling for a given level is constrained to be uniform over only the points contained in that level.

5.2.3. Analytic Instantiation

Pivot MDS works by operating on a subset of complete columns of the distance matrix. The uniform sampling of distances per point used by the other instantiations would violate this constraint by allowing zeros within columns. We instead compute `numDists` new columns of the distance matrix at each iteration. New columns are chosen using the *MaxMin* strategy described in the Pivot MDS paper [BP07] starting from a single column chosen uniformly at random.

5.3. Component S: Objective Function

Glint objective functions **S** are fast approximations of the true objective functions **F** that are far more costly. In each of

Parameter Name	Instantiation	Default	Selection Method	If Too Small		If Too Big	
ε	all	0.001	benchmark	T:slower	Q:better	T:faster	Q:worse
`numIters`	gradient-based	100	benchmark	LT:faster	DT:slower	LT:slower	DT:faster
`numDists`	all	$\log N$	parameter doubling	T:faster	Q:worse	T:slower	Q:better
`numRunsF`	force-directed	5	benchmark	LT:faster	Q:worse	LT:slower	Q:better
`numRunsA`	analytic	10	benchmark	LT:faster	Q:worse	LT:slower	Q:better
`trainSize`	force-directed, analytic	3	benchmark	T:faster	Q:worse	T:slower	Q:better

Table 1: *Parameters used in Glint instantiations, their default values, how they were chosen, and the tradeoffs in setting them too small or too big.* T *is total time,* LT *is layout time,* DT *is distance calculation time, and* Q *is layout quality.*

the Glint instantiations, **S** fits the following template:

$$S(hi, lo, sel) = \frac{\sum_{(i,j)\in sel(P)} (lo(i,j) - hi(i,j))^2}{\sum_{(i,j)\in sel(P)} hi(i,j)^2}$$

Here, $hi(i,j)$ and $lo(i,j)$ are functions defining the high and low-dimensional distances between points i and j. The *hi* function varies from dataset to dataset, while *lo* is always the Euclidean distance function. The *sel* function is an index-selection function that selects a subset from set of nonzero distance matrix indices P. Intuitively, this function just measures the normalized sum of distance residuals between the layout points and the data, but only for a small set of point pairs instead of all pairs of points.

Because they are stress-based techniques that minimize distance residuals, the force-directed and gradient-based instantiations use D'_{ij} for $hi(i,j)$ and the low-dimensional Euclidean distance for $lo(i,j)$. The analytic instantiation is *strain*-based, minimizing the inner-product residuals. To measure strain, we set $hi(i,j)$ to be the inner product of ith and jth rows of the double-centered matrix C and set $lo(i,j)$ to be the inner product of the ith and jth layout coordinates. The interested reader should refer to original Pivot MDS paper for more details on the computation of C [BP07].

For the analytic and gradient-based instantiations, the index-selection function *sel* selects the entirety of the nonzero matrix indices P. In contrast, the force-directed instantiation selects a subset of P. The precise subset of P is the set of point indices contained in the union of per-point random sample caches used by the Glimmer algorithm. In the force-directed instantiation, **S** is equal to the sparse stress function computed at the end of each Glimmer run [IMO09].

Each instantiation employs randomized sampling of new distance matrix indices after each Glint iteration, as mentioned in Section 5.2. In the case of the gradient-based instantiation, this random sampling does not impart enough random noise to the observed values of **S** to induce an unexpected termination. However, in the force-directed and analytic cases, we observed enough noise in the sequence of **S** values that early termination was regularly observed. In this section we describe our strategy for creating a smooth **S** from the noisy series of raw objective function values.

A simple approach would be to filter the sequence of raw values using a moving average. Since the noise in the signal is white noise, with equal power across all frequencies, it would manifest itself after filtering any bandwidth, so this approach would not solve the problem.

Fortunately, the observed noise can be modelled by a Gaussian distribution. Stochastic processes where any subset of process samples are normally distributed are known as Gaussian processes and can be accurately modelled by the machinery of Gaussian process regression (GPR) [RW06]. (We confirmed normality with a Shapiro-Wilk test result of $p = 0.55$ [SW65].)

In order to perform GPR we must select the forms of the two functions that completely determine a Gaussian process, the mean and the covariance function. The mean of the Gaussian process encodes information about the shape of the underlying process, for example whether it is linear or constant. We chose a mean prior of zero, indicating that we have no advance knowledge about the signal. We select the squared exponential function, one of the most commonly chosen covariance functions [RW06], because it models smooth transitions between adjacent values of **S**, a behavior that matches our expectations for the convergence curve.

We can improve our smooth estimate of the mean of **S** by increasing the number of samples computed at each outer loop iteration. In the force-directed case, we compute more samples by restarting **M** with the same initial layout and a different random seed. Since the analytic case proceeds deterministically, the same technique cannot be used. To compute a set of random samples for the analytic case, for each sample we select `numDists` columns uniformly at random to leave out of P.

For the parameter designating the number of computed samples per Glint iteration, there is a parameter tradeoff between the fidelity of the estimated mean, which affects the likelihood of observing a false termination, and the speed of algorithm. We empirically find that computing 5 runs for the force-directed `numRunsF` parameter and 10 runs for the an-

alytic and numRunsA parameter yields good results over all our benchmark datasets.

Using GPR requires initialization of the so-called process *hyperparameters* of the squared exponential covariance function. These include the length scale, or degree of smoothness, and the noise level. The hyperparameters can be efficiently learned from a small set of observations computed during the first trainSize iterations of the Glint outer loop, by optimizing a likelihood function using conjugate gradients. We empirically find that using 3 iterations for training yields good results over all our benchmark datasets.

5.4. Instantiation Design Summary

Table 3 summarizes the Glint component design decisions, emphasizing the underlying algorithm features that crosscut the three instantiations. Consideration of these features could guide designers of future instantiations. For example, an algorithm using the entire sparse input distance matrix, like Pivot MDS and SMACOF, can remain unaltered for **M**. Algorithms with objective functions **S** that are noisy, such as Pivot MDS and Glimmer, can employ GPR smoothing.

6. Results

We present the results in terms of a benchmark performance comparison and an assessment of convergence. We first describe the benchmark datasets in detail. We compare the efficiency and quality of Glint instantiations against the standard algorithms in terms of time and stress using these benchmarks. We then discuss convergence issues and demonstrate convergence behavior of each instantiation.

6.1. Dataset and Distance Function Description

The molecule dataset contains 661 points representing polymer-based nanocomposites. The distance function is cheap: it is the Euclidean distance metric where the number of dimensions m is 10. We include this dataset as a baseline where the Glint requirements are not met and unmodified algorithms should be employed instead. The 4000 points in the concept dataset are biomedical terms where the distance function to determine their co-occurrence in journal articles requires running database queries. The Flickr dataset contains 1925 images culled from the first author's public photo collection, with distances computed using the Earth Mover's Distance (EMD) [RTG00]. The BRDF dataset is an example from the computer graphics literature, where computations involving 100 points representing images use the Euclidean distance function. The number of dimensions m is four million [MPBM03]; this function is expensive despite being Euclidean because of the huge number of dimensions. The videogame dataset was created by gathering human judgements in response to survey questions about 96 games. While the exact timing information for the judgments was not reported [LMF07], our conservative estimate is that the sum of the response times of the human participants took an average of 10 seconds for each pairwise comparison. Table 2 summarizes our benchmark distance functions and costs.

d cost (sec)	Distance Calculation	Benchmark
0.00001	Euclidean $m = 10$	molecule
0.001	DB Query	concept
0.01	Earth Mover 8^3 signature	flickr
1.0	Euclidean $m = 4$M	brdf
10.0	Human Elicited	videogame

Table 2: *The cost d of a single distance calculation for the benchmark datasets in seconds rounded to the nearest power of 10. Here m represents the number of dimensions of the input data in the case of using a Euclidean distance function.*

6.2. Benchmark Speed and Quality Comparison

We validate Glint by comparing the benchmark performance of our implementations against the previous work in terms of speed and quality. Speed is measured in seconds to termination and quality is measured in terms of the full objective function **F** using the entire distance matrix D. For the force-directed and gradient-based instantiations, **F** is the full normalized stress function [BG05]. For the analytic instantiation, **F** is the full normalized strain function. We compute **F** only for performance validation; it is never computed in practice. All recorded values are averaged over 5 runs on an Intel Core 2 QX6700 2.66 GHz CPU with 2 GB of memory.

For the original approach in the force-directed and gradient-based performance comparison, we ran the Glimmer and SMACOF algorithms, respectively, with the same ε for these as used in Glint. For the original approach used in the analytic performance comparison, we know of no algorithms with termination criteria. Instead, we used a human-in-the-loop Pivot MDS setup, where the first author added numDists pivots at a time with a keystroke, and manually halted the process after visually assessing layout convergence. The Pivot MDS algorithm is unable to handle incomplete distance matrix columns, so we omit the videogame benchmark, which possesses many missing matrix entries, from the analytic results.

Figure 2 and Table 4 compare the execution time and final layout quality of Glint to the original approaches.

The speedup of the force-directed instantiation ranges from 20 to 115 for the costly target cases, while the original Glimmer algorithm is several times faster for the cheap baseline. The main benefit of the fully automatic analytic Glint instantiation is the elimination of the need for manual monitoring and intervention. The Glint instantiation was faster than Pivot MDS with a manual operator in the loop for molecule and flickr, but slower for concept and

Alg. Class	**M**	**DS**	**S**
force-directed	altered sampling	uniform pointwise for each hierarchy	GPR smoothed stress-based across sample-cache sets
gradient-based	unchanged	uniform pointwise	stress-based across P
analytic	unchanged	uniform columnwise	GPR smoothed strain-based across P

Table 3: *Glint component design summary for each MDS algorithm class.*

Benchmark	Glint **F**	Orig. **F**	Glint Time	Orig. Time	Speed up
Force-Dir.					
`molecule`	0.03	0.03	14	4	0.2
`concept`	0.18	0.18	49	1016	20
`flickr`	0.08	0.09	2.4K	98K	40
`brdf`	0.03	0.04	3K	304K	115
`videogame`	0.45	0.45	23K	482K	20
Analytic					
`molecule`	0.35	0.42	3	23	9
`concept`	0.93	0.94	96	63	0.7
`flickr`	0.48	0.59	1.2K	2.9K	2
`brdf`	0.078	0.233	40K	6K	0.2
Gradient					
`molecule`	0.01	0.03	360	700	1.9
`concept`	0.18	0.18	0.1K	113K	880
`flickr`	0.06	0.04	8K	71M	8.8K
`brdf`	0.008	0.005	4K	859K	200
`videogame`	0.16	0.13	19K	430K	220

Table 4: *Comparison of full objective functions, time (in seconds), and speedup between Glint instantiations and original MDS algorithms.*

`brdf`. The speedup of the gradient-based Glint instantiation is dramatic: several orders of magnitude in the target cases, and a factor of two in the baseline case of `molecule` where the distance function is cheap.

The quality values for Glint are roughly the same magnitude and variability for each benchmark in the force-directed case. For the analytic instantiation, the quality values are equal or better than the manual Pivot MDS method. In the gradient case, most of the final quality values, except `molecule`, are slightly worse than the standard approach using the full distance matrix. The gradient Glint instantiation provides a speed and quality compromise between the extremely costly but accurate full gradient approach, and the fast but approximate force-directed Glint instantiation.

6.3. Convergence

We illustrate the convergence behavior of each Glint instantiation in Figure 3. Each log-scale plot displays two curves: the blue curve represents the value of the full, slow objective function **F** of the layout after each Glint iteration, while the orange curve shows the value of the smoothed, fast objective **S**. For those instantiations that employ GPR smoothing, we also plot the random samples used in the regression as gray dots. Similarly, for those instantiations that employ an iterative layout algorithm **M**, we plot the values of **S** after each **M** iteration. As in the benchmark comparison, **F** is the full normalized stress function for Glimmer and SMACOF, and **F** is the full strain function for Pivot MDS.

The magnitude of the change in the cheap objective **S** approximates that of the change in costly **F** function. In the case of Pivot MDS, the smoothed **S** series is slightly offset from the gray random samples due to the effect of using sparsity patterns from the previous iteration. These benchmarks validate the claim that setting ε to a given termination threshold will terminate Glint when the corresponding change in **F** falls below the threshold modulo some sampling noise.

7. Conclusion

We have illustrated how expensive distance calculations change the efficiency of existing MDS algorithms. Algorithms like Glimmer and SMACOF compute more distances than are required for an existing quality of layout, while analytic algorithms require manually tuning the number of distances to compute as an input parameter. We solve both these problems with Glint, an algorithm framework with three components: a distance matrix densification strategy **DS**, an algorithm **M**, and an inexpensive objective measure **S**. Given these components, Glint samples distances from the distance matrix in fixed batches, updating the low-dimensional layout with new information until the layout quality converges. We show how careful design of termination criteria can overcome the noise effect of random sampling on convergence. We present and validate Glint instantiations for three separate types of previous MDS algorithms: the force-directed Glimmer, the analytic Pivot MDS, and the gradient-based SMACOF.

The Glint instantiations provide essentially equivalent layout quality in all cases. The analytic instantiation was roughly equal in time performance to Pivot PDS, with some cases of speedup and some of slowdown; the main contribution of Glint in this situation is to remove the need for manual monitoring and intervention. The iterative instantiations showed substantial speedups against Glimmer and SMACOF in all of our target cases with costly distance func-

Figure 2: *Comparison of speed (top) and quality (bottom). In each pair, the top blue bar is the original MDS algorithm, and the bottom orange bar is the Glint instantiations. The black lines indicate 95% standard error bars.*

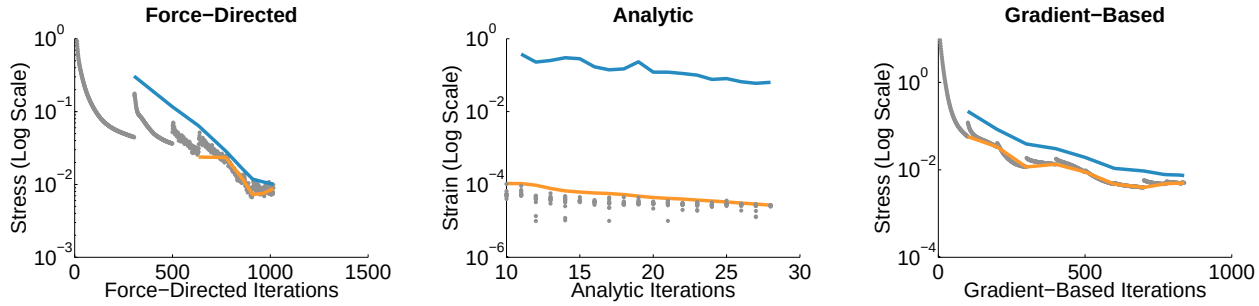


Figure 3: *Log-scale Glint convergence curves on each instantiation generated using the* `brdf` *dataset. The orange* ***S*** *curve is derived from the noisy grey samples.* ***S*** *is designed to match the convergence behavior of the costly* ***F*** *series in blue.*

tions, ranging from 20 to 115 with the force-directed Glint instantiation and from 200 to 8800 with the gradient-based Glint instantiation.

The Glint framework uses modified versions of existing algorithms to more efficiently compute low-dimensional layouts on problems with costly distance functions. Glint reduces the total time spent computing distance information by automatically selecting a reduced number of distance matrix entries to compute based on monitoring layout quality. Glint reduces the time and cost of analyzing distance-based datasets with MDS, opening the door for practitioners to apply MDS to problems with expensive distance functions at an entirely new scale.

8. Acknowledgements

This work was supported through a NSERC Strategic Grant, with partial travel support from UBC ICICS and FOGS.

References

[BG05] BORG I., GROENEN P. J. F.: *Modern Multidimensional Scaling Theory and Applications*, 2nd ed. Springer-Verlag, 2005. 29, 31, 35

[BP07] BRANDES U., PICH C.: Eigensolver methods for progressive multidimensional scaling of large data. In *Graph Drawing*, Kaufmann M., Wagner D., (Eds.), vol. 4372 of *Lecture Notes in Computer Science*. Springer, 2007, pp. 42–53. 30, 31, 33, 34

[BSL*08] BUJA A., SWAYNE D., LITTMAN M., DEAN N., HOFMANN H., CHEN L.: Data visualization with multidimensional scaling. *Journal of Computational and Graphical Statistics 17*, 2 (2008), 444–472. 31

[Cha96] CHALMERS M.: A linear iteration time layout algorithm for visualising high dimensional data. In *Proc. IEEE Visualization* (1996), pp. 127–132. 31

[dLM09] DE LEEUW J., MAIR P.: Multidimensional scaling using majorization: SMACOF in R. *Journ. Statistical Software 31*, 3 (8 2009), 1–30. 31, 33

[dST04] DE SILVA V., TENENBAUM J.: *Sparse multidimensional scaling using landmark points*. Technical report, Stanford, 2004. 30, 31

[FC11] FRANCE S., CARROLL J.: Two-way multidimensional scaling: A review. *IEEE Trans. Systems, Man, and Cybernetics, Part C: Applications and Reviews 41*, 5 (2011), 644–661. 31

[GKN04] GANSNER E. R., KOREN Y., NORTH S. C.: Graph drawing by stress majorization. In *Graph Drawing* (2004), pp. 239–250. 31

[Gre75] GREEN P.: Marketing applications of mds: Assessment and outlook. *The Journal of Marketing* (1975), 24–31. 29

[HTF09] HASTIE T., TIBSHIRANI R., FRIEDMAN J.: *The Elements of Statistical Learning: Data Mining, Inference, and Prediction, Second Edition*. Springer Series in Statistics. Springer, 2009. 29

[IMO09] INGRAM S., MUNZNER T., OLANO M.: Glimmer: Multilevel MDS on the GPU. *IEEE Trans. Visualization and Computer Graphics (TVCG) 15*, 2 (2009), 249–261. 29, 30, 31, 33, 34

[IMS12] INGRAM S., MUNZNER T., STRAY J.: *Hierarchical Clustering and Tagging of Mostly Disconnected Data*. Tech. Rep. TR-2012-01, University of British Columbia Department of Computer Science, May 2012. 29

[Ing07] INGRAM S.: *Multilevel Multidimensional Scaling on the GPU*. Master's thesis, University of British Columbia Department of Computer Science, 2007. 31, 33

[JCC*11] JOIA P., COIMBRA D., CUMINATO J. A., PAULOVICH F. V., NONATO L. G.: Local affine multidimensional projection. *IEEE Transactions on Visualization and Computer Graphics 17*, 12 (2011), 2563–2571. 31

[KHKS12] KHOURY M., HU Y., KRISHNAN S., SCHEIDEGGER C.: Drawing large graphs by low-rank stress majorization. *Comp. Graph. Forum 31*, 3pt1 (June 2012), 975–984. 31

[LMF07] LEWIS J. P., MCGUIRE M., FOX P.: Mapping the mental space of game genres. In *Proc. ACM SIGGRAPH Symp. Video Games* (2007), pp. 103–108. 30, 35

[MPBM03] MATUSIK W., PFISTER H., BRAND M., MCMILLAN L.: A data-driven reflectance model. *ACM Trans. Graphics (Proc. SIGGRAPH 2003) 22*, 3 (2003), 759–769. 35

[Pla05] PLATT J. C.: FastMap, MetricMap, and Landmark MDS are all Nyström algorithms. In *Proc. Intl. Workshop on Artificial Intelligence and Statistics* (2005), pp. 261–268. 31

[PSN10] PAULOVICH F., SILVA C., NONATO L.: Two-phase mapping for projecting massive data sets. *IEEE Trans. Visualization and Computer Graphics 16*, 6 (2010), 1281–1290. 31

[RTG00] RUBNER Y., TOMASI C., GUIBAS L.: The Earth Mover's Distance as a metric for image retrieval. *Intl. Journ. Computer Vision 40*, 2 (2000), 99–121. 30, 35

[RW06] RASMUSSEN C. E., WILLIAMS C. K. I.: *Gaussian Processes for Machine Learning*. MIT Press, 2006. 34

[SD74] SPENCE I., DOMONEY D.: Single subject incomplete designs for nonmetric multidimensional scaling. *Psychometrika 39* (1974), 469–490. 30

[SW65] SHAPIRO S., WILK M.: An analysis of variance test for normality (complete samples). *Biometrika 52*, 3/4 (1965), 591–611. 34

[TM08] TERNES D., MACLEAN K. E.: Designing large sets of haptic icons with rhythm. In *Intl. Conf. Haptics: Perception, Devices, and Scenarios (EuroHaptics)* (2008), Springer LNCS 5024, pp. 199–208. 30

[Tor52] TORGERSON W.: Multidimensional scaling: I. theory and method. *Psychometrika 17* (1952), 401–419. 31

SIGRAD 2012
A. Kerren and S. Seipel (Editors)

Visual-Interactive Preprocessing of Time Series Data

J. Bernard[1], T. Ruppert[1], O. Goroll[2], T. May[1], and J. Kohlhammer[1]

[1]Fraunhofer IGD Darmstadt, Germany
[2]Technische Universitität Darmstadt, Germany

Abstract

Time series data is an important data type in many different application scenarios. Consequently, there are a great variety of approaches for analyzing time series data. Within these approaches different strategies for cleaning, segmenting, representing, normalizing, comparing, and aggregating time series data can be found. When combining these operations, the time series analysis preprocessing workflow has many degrees of freedom. To define an appropriate preprocessing pipeline, the knowledge of experts coming from the application domain has to be included into the design process. Unfortunately, these experts often cannot estimate the effects of the chosen preprocessing algorithms and their parameterizations on the time series. We introduce a system for the visual-interactive exploitation of the preprocessing parameter space. In contrast to 'black box'-driven approaches designed by computer scientists based on the requirements of domain experts, our system allows these experts to visual-interactively compose time series preprocessing pipelines by themselves. Visual support is provided to choose the right order and parameterization of the preprocessing steps. We demonstrate the usability of our approach with a case study from the digital library domain, in which time-oriented scientific research data has to be preprocessed to realize a visual search and analysis application.

Categories and Subject Descriptors (according to ACM CCS): I.3.3 [Computer Graphics]: Picture/Image Generation—Line and curve generation

1. Introduction

Large time series repositories are built up in many scientific disciplines like climate research, genomics research or high-energy physics. Many of these repositories are commonly shared by researchers to search for new findings. In virtually all cases raw time series have to be prepared for effective retrieval as well as for effective exploratory analysis. Data preprocessing is necessary whenever the data does not match the requirements of the following analytical tasks and methods. For example, an algorithm for time series clustering may require uniformly sampled data; another analysis algorithm may not be able to deal with missing values etc. Typically, preprocessing is a combination of different operations arranged in a pipeline.

The outset of our work is that a user is given an analytical task and a large repository of time series data. The preprocessing task is shared by two user roles: The data mining expert is responsible to choose and modify the operations for the pipeline. The domain expert is responsible to define the criteria for useful data. Both experts are responsible to identify and communicate on errors in their respective domain. We assume that the visualization of the process and the data can be the medium for this communication. As soon as the pipeline is set up and tested, the preprocessing is applied as an automated process. If it is not known whether the data matches the requirements of the analysis, it certainly is not advisable to apply an automated preprocess as a 'black-box' operation to see what happens. An interactive adjustment of the pipeline can be used to learn about the characteristic properties of the data, to check the requirements and to test the effects of suitable operations and their parameters.

Kandel et al. [KHP*11] coined the term 'data wrangling' to describe scencarios of this type. They state that the process has to be visible and audible to domain experts to identify and correct potential errors or invalid assumptions. (Domain experts needed to search errors). The coupled interactive visualization of data and process - including all data-fixing efforts - is required to enable domain experts to control the process. Following their recommendation, our work

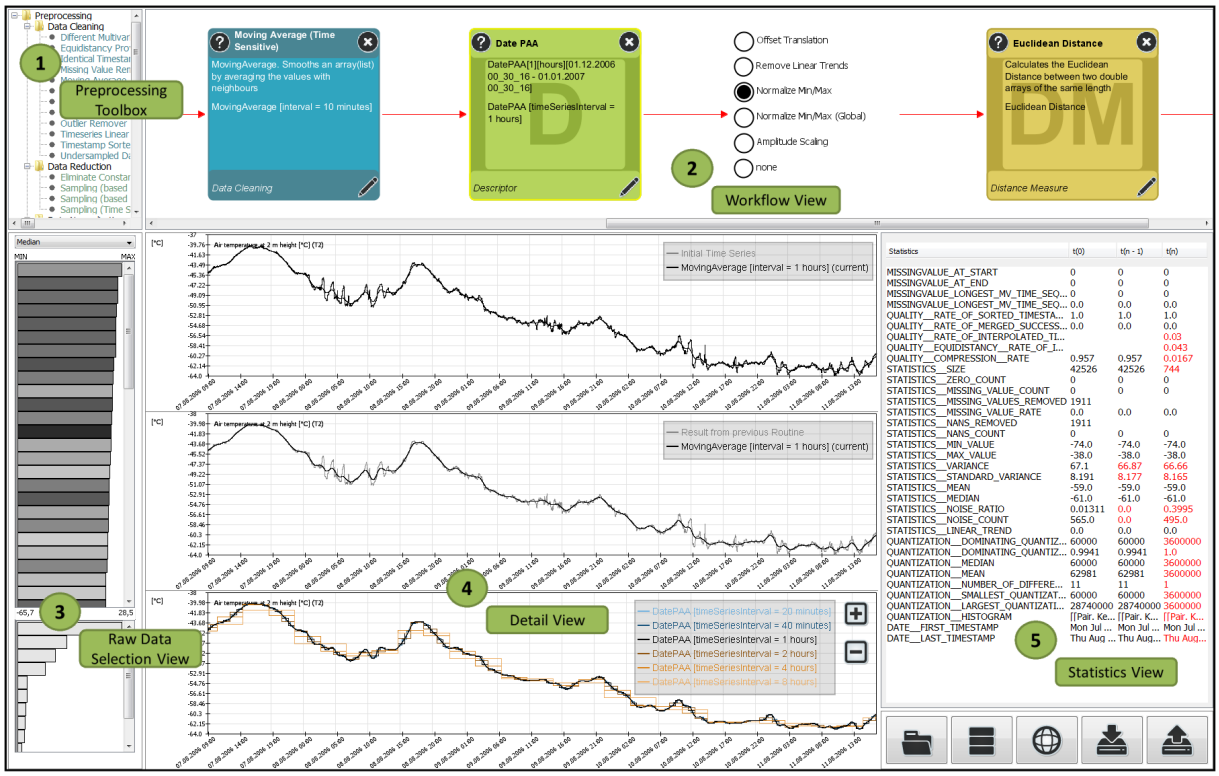


Figure 1: *Preprocessing pipeline creator: a visual-interactive time series preprocessing system. Effects of the Piecewise Approximate Aggregation (PAA) descriptor [KCPM00] are shown in the detail view (4). Statistics are shown in the statistics view (5).*

tightly integrates data diagnostics and data transformation capabilities.

To succeed in her tasks the user must overcome a number of challenges:

The first challenge to the design is to trade-off a compact representation with a faithful representation. A compact representation improves the performance of the analysis on large repositories by eliminating irrelevant parts of the data. It is up to the domain expert to define 'relevance'. Hence, the expert must stay aware if and how a preprocessing step affects the outcome of the analysis.

The second challenge is not to 'overfit' the pipeline. On the one hand, visual inspection and testing is not feasible for a large repository. On the other hand, designing the pipeline using a single series does not always generalize and some important properties of other series might not be processed correctly. Instead the user is required to select a few time series to control and test the design. To increase the robustness of the preprocessing, this selection should at least approximate the diversity of the repository.

The third challenge is the size of the design space. For a typical operation like *sampling* a number of different methods are available. In turn, most of these methods have at least one parameter. Most operations can be combined freely, making the task to search for a suitable, if not optimal, solution as complex as the analysis itself.

The contribution of our work is as follows:

Firstly, we present a system for the interactive design and control of a time series preprocessing pipeline. It provides user support for the composition of preprocessing operations like data cleaning, data reduction an others. These operations can be chosen from a toolkit and arranged freely to define the pipeline. The effects of the preprocessing can be investigated by a visualization of the time series for every step of the pipeline. The approach is designed for expert users and non-expert users alike. In particular, the user is not required to do programming or scripting. The design of the pipeline can be monitored by time series visualizations, showing the effects of every module applied to representative time series. The resulting preprocessing transforms single raw time series into one or multiple descriptors each, depending on the requirements of the following analysis. Since our focus is on the refinement of single time series data, any metadata attached to a raw time series is kept with the corresponding descriptors for further reference. Analytical operations including an aggregation of multiple time series (like clus-

tering, merging of time series etc.) is not considered part of the preprocessing here.

Secondly, the user is supported in the selection of appropriate parameters for every module. An optimal choice of a preprocessing setup actually requires a search in a multi-parameter space. While the user can only change one parameter at a time, we aid her choice by generating an ensemble of alternative parameter values. Alternative results are shown in the time series. An inspection of the alternative effects leaves visual hints for potential optimization.

Thirdly, representative candidates of input time series are suggested to design and test the preprocessing pipeline. At the start of the preprocessing, reliable similarity measures are not available for analysis. Instead we estimate the time series variability by statistical properties. Candidates are suggested for user selection which cover most of this variability to establish the boundaries for the design.

We illustrate the use of our system in collaboration with researchers setting up a digital library for scientific climate data. We selected two tasks with two different preprocessing requirements. For each of these tasks we show how the pipeline is implemented after diagnosing the time series and how adaptions are made to meet the requirements.

The paper is organized as follows: Section 2 gives an overview of existing techniques and toolkits for time series preprocessing. In Section 3 we describe the core of our work, the editor for the preprocessing pipeline including the toolkit and views to try and test methods and parameters. We apply our toolkit to a scientific repository of climate data. In Section 4 we show the design of the preprocessing pipelines for two scenarios posing two different requirements for the analysis and preprocessing of this data. Section 5 summarizes the contribution of our paper.

2. Related Work

In the field of time series analysis, a diversity of analytical tasks exists. Current approaches differ by the targeted user group, the application domain and the analysis goal [AMST11]. For that reason alone the degrees of freedom in *time series preprocessing* are comprehensive [DTS*08, WL05], not only in the choice of methods but also for setting required input parameters [KLR04]. Moreover, many sources of data problems and varying levels of data quality [KHP*11] exacerbate the complexity of time series preprocessing tasks. We review both time series preprocessing techniques and data preprocessing workflows.

2.1. Time Series Preprocessing, Descriptors and Distance Measures

Data cleaning is important to ensure data quality [KCH*03, KHP*11]. Prominent challenges are missing value handling, noise and outlier detection [CBK09] (cf. Figure 3) and avoiding non-equidistant representations [WL05].

Data normalization solves problems with comparing data of different scales or translations. Normalization is essential for natural and subjectively correct similarity calculations [KK03, WL05, GAIM00]. Normalization can have local or global effects, depending on the position within the time series pipeline when applied.

Data sampling is an important method to reduce the amount of data. Data sampling can be defined as a subcategory of data reduction, surveys also describing data reduction in general are given in [Fu11, KK03, WL05]. Data percentage sampling, date-sensitive methods or variants that imply value-specific properties are applied [GAIM00, Fu11].

Data segmentation is applied on time series, since in many indexing, classification and clustering approaches, only subsequences are considered [Fu11, KK03]. Segmentation approaches for patterns with equal length [GAIM00] and unequal length [KCHP01] exist.

Time series descriptors are compact representations of time series raw data, that preserve the relevant information [DTS*08, KK03, LKLC03, Fu11]. Approaches with high compression rates which simultaneously preserve most of the information have been presented. However, distorting the shape of the time series creates problems if compression rates are too high [Fu11], which is also called the trade-off between compression and fidelity [KCHP01].

Similarity measure selection is highly domain- and application-specific process and therefore a challenging task [DTS*08, KK03, WL05, GAIM00]. In many time series applications, the used similarity measures are predefined.

2.2. Visual Analytics Workflows

For the visual analysis of data in general, a great variety of approaches has been presented to date. We refer to the book *Visualization of Time-Oriented Data* [AMST11] for a survey about relevant visualization and visual analysis techniques for time series data. Even though data analysts firstly spent large parts of their time on data cleaning and other preprocessing tasks before actual data analysis tools are executable [KCH*03], comparatively little research advances have been made in how interactive visualization can advance data preprocessing [KHP*11]. We identify works that relate to research on visual-interactive data preprocessing.

Regarding multidimensional data in general, several approaches for visual-interactive analysis exist. For example, an assisted descriptor selection approach based on visual comparative data analysis can be found in [BvLBS11]. In [SS04], the exploratory analysis of multidimensional datasets is supported by a rank-by-feature prism. The user can detect interesting features by choosing statistical ranking criterions (e.g. normality of the distribution) and manually select relevant features in 1D and 2D visualizations. Pretorius et al. [PBCR11] present a technique that supports users

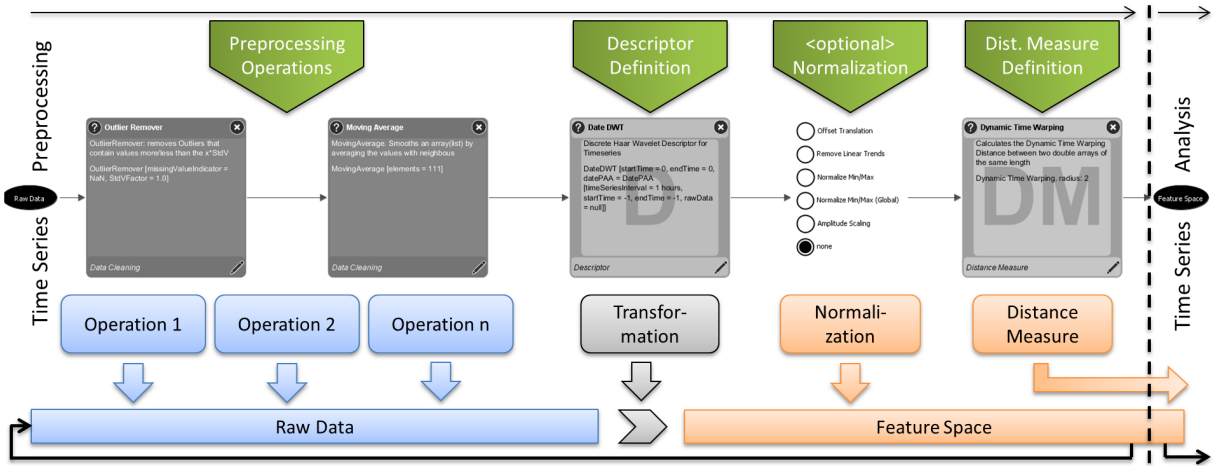


Figure 2: *The time series preprocessing workflow. In the raw data space the user can create a number of operations with the toolkit. A descriptor transforms the raw data into the feature space, the basis for subsequent time series analysis approaches.*

in visually analyzing relationships between the input parameter space and the corresponding output in the field of image analysis. In [IMI*10], a system for visual-interactive dimensional analysis and reduction is introduced. The user can combine a series of analysis steps, change parameters, and get direct visual feedback of the intermediate results. Predefined workflows consisting of several analysis steps can be stored and reused in later analysis processes. These systems guide the users in the analysis of multidimensional data via workflows. However, non of them focus on the specific characteristic of time series data in particular.

An example for the visual-interactive analysis of time series data is realized with the ChronoLenses interface [ZCPB11]. The system focuses on the visual exploration of time series by enabling the user with real-time transformation of selected time series intervals, and pairwise comparison of different time series. However, the authors use cleaned time series data. Therefore, preprocessing in general and the guidance towards appropriate parameter settings is not the focus of their approach. Further approaches for visual-interactive time series analysis are presented in [HDKS05], where importance-driven layouts guide the users in finding interesting time series, and [SSW*12], where the user can compare different parameter setups for the detection of local interest points. In [SBVLK09], clustering of time series trajectory data is performed with enhanced visual-interactive monitoring and controlling functionality.

To the best of our knowledge, there does not exist any technique that focuses on both the visual-interactive definition of an analysis workflow and the user-guidance in setting appropriate parameters for time series preprocessing.

3. Visual Analytics for Time Series Preprocessing

In this section, we introduce our approach for visual-interactive time series preprocessing. Subsection 3.1 presents the idea and the additional value of this work. Each of the following Subsections deals with one contribution, as mentioned in the introduction: user support for the definition of preprocessing scenarios, guidance for the parameter selection for individual preprocessing routines, and supporting the selection of representative testing data.

3.1. A Visual-Interactive Pipeline

We derive three insights from our review of the related work. First of all, we assert that a large number of algorithms for time series preprocessing has been established. However, visual-interactive representations of time series preprocessing operations are scarce, [KCHP01] may serve as an exception. In addition, although visual analysis of time series as such is a popular topic of research [AMST11], hardly any visual-interactive time series preprocessing applications are proposed. In fact, most approaches use preprocessing as a 'black-box'-approach. Finally, we observe that visual preprocessing of other data has become popular, combining algorithmic power with human capability of detecting patterns and steering the preprocessing process [IMI*10, BvLBS11].

We introduce a visual-interactive system for the generation of time series preprocessing pipelines, the conceptual workflow is shown in Figure 2. In the following, we call the preprocessing pipeline a time series *scenario*. We choose a generalizable approach for time series preprocessing. Beginning with the selection of raw data a variety of preprocessing operations can be added to the pipeline and (re-)arranged in arbitrary order. For most scenarios, however, the goal is a compact representation of the time series [KK03], commonly called a *descriptor*. For the definition of a descriptor, the pipeline may include a transformation of the raw data to a feature space, the outcome is a so called feature vector. For example, the Discrete Fourier Transformation may be applied to transform the time series into the signal space.

After an optional normalization step, a distance measure is defined to complete the time series scenario.

We aim to make the different operations as exchangeable and compatible as possible. Hence, the data model of our input time series consists of a list of so-called time-value pairs, each containing a time stamp and a corresponding value. This data model is able to represent virtually all possible characteristics of time series data like non-equidistant time stamps or missing values. Attached attributes like 'location on earth' (meta data) are kept with the produced feature vectors for the subsequent analysis task. Some preprocessing routines split one raw data into many (e.g. segmentation to patterns), which then also holds for the corresponding meta data. Additional qualitative results produced by preprocessing routines (e.g. compression rate) are adhered as additional meta data and displayed in the *statistics view* (5) (cf. Section 3.2). As a consequence of this generalizable approach, our system is not limited to a single time series input format. It only takes the effort of implementing an interpreter interface to make new data sources accessible.

Our intended users are data mining and machine learning experts, but also domain experts with the need to process time series data. Thus, we argue that typically users either have little data domain knowledge or are no experts in scripting interfaces. The visual-interactive approach, combined with user support is aimed to open time series preprocessing workflows for broad user groups. During the selection of preprocessing routines, preview visualizations simplify the selection process. The visualization of alternative parameters helps to discover the impact of distinct degrees of freedom in the time series.

3.2. System and Views

The graphical user interface of our approach consists of five components (see Figure 1). The preprocessing modules needed for the design of the pipeline are provided to the user in a *preprocessing toolbox* (1). The available tools are structured into 6 classes of operations listed below. For each class we implemented a number of alternative approaches.

- data cleaning (e.g. missing values, moving average, etc.)
- data reduction (e.g. sampling methods)
- data normalization (e.g. min-max normalization)
- data segmentation (e.g. subsequence and pattern def.)
- descriptors (e.g. PAA, DTW, DFT, PIP, etc.)
- similarity measure (e.g. Euclidean distance, etc.)

The *workflow view* (2) is the visual representation of the preprocessing pipeline. All modules that possess user adaptive parameters are displayed as rounded rectangles. A module can be added by drag-and-drop from the preprocessing toolbox. Each module glyph contains the name and an external hyperlink for additional information (top), the description and parameter setting (middle), as well as the operation class and an edit button (bottom). Parameter changes can be set when the module is added or afterwards by clicking the edit button. By selecting a module, the preprocessing pipeline is executed on a chosen time series up to this stage. Re-ordering of the pipeline is possible by dragging modules to other positions. Load and Save buttons at the lower right of the system enable to re-edit and branch scenarios at a later time. Since the optional normalization step in the scenario does not require parameters, a radio button-like glyph is used for the representation. Straight forward, the user can select the favored normalization variant with a single click.

The *raw data selection view* (3) is divided in two lists of used and unused raw data represented as bars. The bar size corresponds to raw data values according to a user-definable statistical property (the median in case of Figure 1). The gray value displays the degree of dissimilarity of unused raw data (upper list) in contrast to the ones already visualized. Raw data with black color hold maximum dissimilarity and thus are mostly recommended to visualize. Section 3.4 further describes how this user guidance is algorithmically provided.

In the *detail view* (4) the user can trace the execution by inspecting intermediate preprocessing results of the time series visualized as a line chart. The detail view provides direct feedback on the initial raw data (upper), the last (middle), and the current state of the preprocessed time series (bottom). Section 3.3 describes how the detail view supports the user in setting appropriate preprocessing parameters.

In the *statistics view* (5) statistical information about the selected time series raw data and the preprocessed data is provided to the user. Every change of the statistical information resulting from a preprocessing operation is highlighted in red. By showing this additional meta data, the user can immediately track the effects of the preprocessing.

The proposed workflow is straight forward. The input data is provided in the raw data selection view (3). The user selects the time series to be observed during the analysis. A set of preprocessing modules can be added from the preprocessing toolbox (1) to the workflow view (2), parameter changes, module re-ordering and result-storing is possible at any time. By selecting a module, the user can observe respective modifications on the time series in the detail view (4). Additional statistical information can be monitored at the statistics view (5). After finalizing the scenario by defining a normalization and a suitable similarity measure the preprocessing pipeline completely configured. The scenario is ready for an execution on all time series coming from the current data source, or another data source at a later data. The system supports loading and remodeling of saved scenarios, which enables building branches of preprocessing pipelines.

3.3. User-support for Parameter Setting

Section 3.2 described how preprocessing modules are arranged to an entire time series scenario. In the following, we give details about the parameterization of a single module, which is a problem in itself. Each preprocessing module

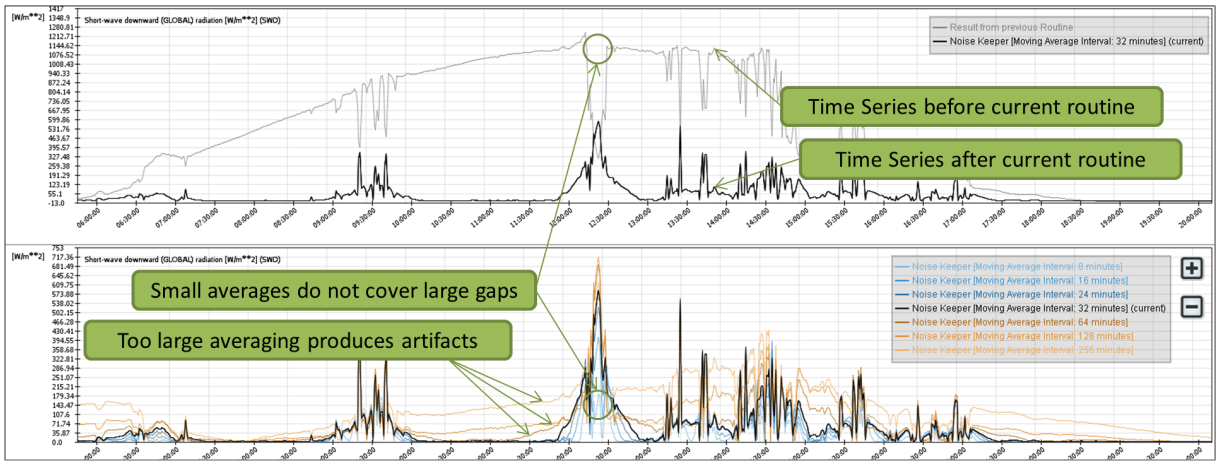


Figure 3: *Emphasizing noisy data by subtracting a moving average. Fine-tuning to a 32 minute moving average interval turns out to be a satisfactory parameterization. The trade off between fidelity and generalizability is accomplished.*

in the system provides ensembles of n alternative parameter values, as appropriate, whereupon n is a user parameter. The time series arising from alternative parameterizations are visualized as line chart bundle in the detail view. Figure 3 demonstrates parameterizations of a *NoiseKeeper* module, alternatively to the examples in the application section. The current parameterization of the particular preprocessing routine is always displayed in black. Curve progressions based on other parameterizations are displayed with bipolar color value from blue to brown. Color brightness is used to illustrate the divergence of each alternative parameterization compared to the current, where darker values are more similar. A colored legend for each parameterization is given at the right of the time series visualization window. In addition, the middle part of the detail view (cf. Figure 1) compares the current time series to the result of the previous preprocessing module (gray) and the upper chart provides a comparison to the raw input data (gray).

We provide two ways for the selection of alternative parameterizations. The user can manually edit the parameterization in the preprocessing pipeline by hand as described in Section 3.2, or choose one of the alternative parameterizations by clicking on the colored legend in the time series visualization window. The number of alternative parameterizations can also be adapted by clicking the 'plus' and 'minus' button at the upper right of the window. Thus the interval of the covered parameter space is also adapted. Incrementing the number n of alternative parameter values (by a larger and a smaller one) increases the parameter interval by the current parameter value times 2 power n.

3.4. Guided Selection of Representative Time Series

We tackle the problem of selecting representatives from a pool of thousands of previously unknown input data. Since scenarios are built to process all raw data, the user needs a method to verify the 'generalizability' of the produced time series preprocessing pipeline based on a small subset of visualized input data. Hence, guidance is needed. To help the user in selecting appropriate time series samples, the system uses a statistics model that estimates the dissimilarity of unused raw data in contrast to the ones already visualized. We define three requirements for the statistical model:

1. Robustness to low quality raw time series data; no preprocessing before the calculation of the statistical model
2. Value and shape-based raw data discrimination
3. Low redundancy between the model features

We use a combined model based on (a) two statistical properties that concern the values of a distinct time series (median, and standard deviation) and (b) three properties that originate from the decomposition of time series (trend, periodicity, and noise). This 5-dimensional feature set is extracted from each input time series, and min-max normalized, afterwards. We recommend to weight each dimension equally in the first iteration of the analysis. The calculation of the Euclidean distance between all used and all unused time series feature sets represented by numerical vectors, produces a single value for each unused time series: the dissimilarity. This property is used for coloring (white is similar and black is unsimilar), to guide the user. The raw data selection view (3) in Figure 1 illustrates our concept.

This similarity concept is independent of the preprocessing operations of the system and of similarity measure chosen at the end of the preprocessing. We are using these parameter-free statistical features to 'boot-strap' the analysis, if no prior knowledge allows for a systematic selection of representatives. After studying hundreds of different input time series, we came to the conclusion that values and shapes are well discriminated with our model. If, however, new insights suggest a refinement of the representative set, the user may change the propery weighting or freely chose an entirely different representative as well.

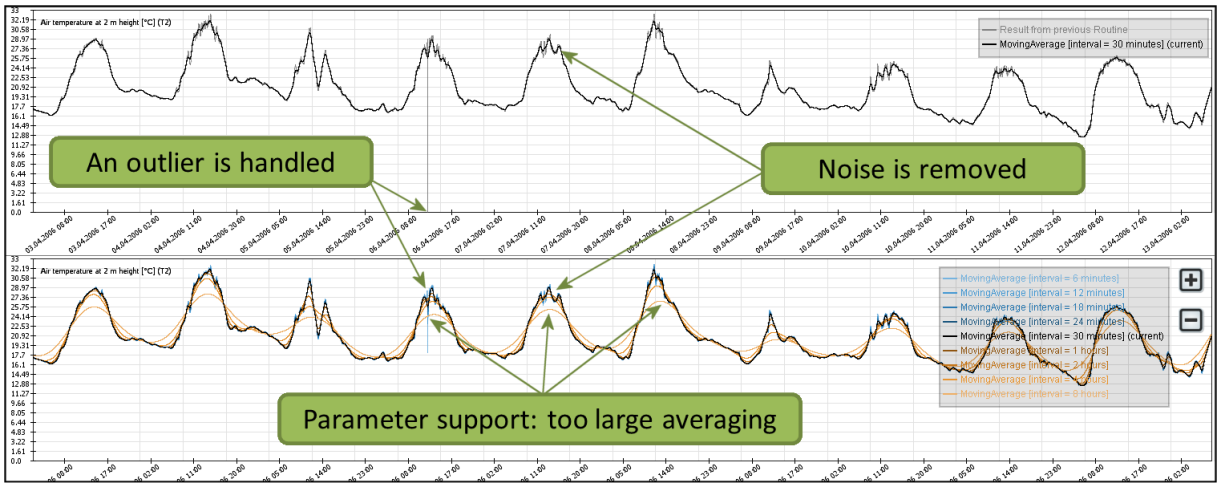


Figure 4: *A 30 minute moving average routine reduces noise too insufficient. For a better result we chose a 1 hour parameterization that smooths the curve yet without too much averaging.*

4. Application

In our case study, we apply our system to tasks coming from the *Digital Library* domain. In the field of digital libraries, scientific research data is considered valuable because research data possibly yields information that has not been discovered yet. It is also challenging because of the size, the heterogeneity and the many degrees of freedom of research data (cf. Section 2.1).

The project *VisInfo* [BBF*11] aims at providing visual search functionality for archived time-oriented research data. In this project it turned out, that domain specialists, librarians, and computer scientists have different time series similarity notions. This complicates the definition of content-based queries for visual search operations in time series data. In a time-intensive case study, we included the researchers and librarians in the definition of preprocessing scenarios for (A) search and clustering (cf. Section 4.2), and (B) compact representations for visualization (cf. Section 4.3). The lack of a system for comprehensive illustration and execution of time series preprocessing impeded the process of defining these final scenarios at the start of the project. In order to provide domain experts with a tool for defining their own time series scenarios with direct visual feedback of its data transformation steps we designed this system.

4.1. Dataset

We focus on *time-oriented scientific research data*, gathered in the scope of the Baseline Surface Radiation Network (BSRN) [ODF*98]. The aim of BSRN is to detect important changes in the earth's radiation field which may be related to climate changes. Thus, up to 100 parameters based on radiation and meteorological characteristics are collected at BSRN stations all over the world, recorded up to a temporal resolution of one measurement per minute. Common physical units include atmospheric pressure, relative air humidity, temperature and a variety of radiation-based measurements like short-wave downward radiation and long-wave upward radiation. The data is archived, published and distributed by the open access library PANGAEA, a data repository operated by the Alfred Wegener Institute (AWI) for Polar and Marine Research in Bremerhaven, Germany. Most of the data is freely available and can be used by domain researchers and interested users to validate hypotheses in earth observations, e.g. regarding climate change.

For our use cases, we choose a data subset from the BSRN data pool [BK12], including 6813 files of monthly measurements originating from 55 BSRN stations on earth. The number of incorporated measurements rises above 500.000.000 data points. Together with the meta data the disc space of the input data is above 20 gigabytes. Thus, data compression will be an important objective in both use cases.

4.2. Use Case A: Extracting Features for Search and Clustering of Time Series Data

Our first use case deals with the extraction of feature vectors from time series data. Assembled in an index structure, the features are used to support fast content-based search and clustering functionality in a web portal for time-oriented research data. Since the user can sketch a time series curve as a query for visual search (or use an example curve), we need to specify a similarity measure for comparing time series data. This similarity has to reflect the similarity notion of climate researchers - the system should find time series that the researcher would expect to. Knowledge of domain expert users has to be included in the time series preprocessing process.

Over a period of two years, we defined a preprocessing pipeline for this purpose within a case study. In the following, we describe a workflow for the creation of time series

scenarios that can be executed within minutes. Here, the pronoun 'we' indicates the cooperation with the domain experts.

We describe the preprocessing workflow for temperature measurements. This data is imported as the raw data source in the pipeline. Next, the statistical information about the raw data is calculated and visualized in the raw data selection view. The measurements differ in a range of values between -70°C and 50°C. The missing value ratio for most raw data is below 10%, the dominating quantization of the measurements is one minute. Most time series have periodic curve progressions with a daily duration, the scientists call this the 'diurnal variation'. We use the raw data selection view and define a test set of raw data with great variabilities regarding median, standard deviation, degree of noise, and periodicity.

We clean the input data to provide a consistent data quality. The *MissingValueRemover* is added to the pipeline and missing values are deleted from the time series. With the *TimeStampSorter* and the *IdenticalTimeStampMerger*, two mandatory routines are established. Since we want to define similarity depending on the overall shape of time series, outliers and local noise can be neglected. We select a *MovingAverageCalculator* to address this task, the results can be seen in Figure 4. By visual comparison we find out that a 30-minute kernel parameterization is too small to reduce noise sufficiently. Hence, we choose a 1-hour interval. The data is now cleaned for subsequent preprocessing steps.

We follow the advice of the domain experts and normalize the time series with a specific *TrueLocalTimeNormalizer*. This method ensures that measurements from all over the globe become comparable, since the time axis is synchronized with the respective time zone. We register that natural periodicities in earth observation mainly have the duration of days and years. We apply the *TimeSeriesSegmentation* routine on our monthly data to receive daily time series patterns, all starting at midnight. Daily patterns with too large gaps compromise the similarity notion. We remove patterns with empty fragments longer than 4 hours with the *UnderSampledDataRemover*. Afterwards, an additional *LinearInterpolation* routine ensures a sampling rate of at least 1 hour.

As a next step we define a descriptor that represents the original time series with respect to the similarity notion of a climate researcher. We choose the PAA descriptor with a quantization of 1 hour as it can be seen in Figure 1. This parameterization is chosen since on the one hand the researcher affirms a sufficient fidelity to the original data and on the other hand the descriptor exhibits a strong compression of 60:1. A higher compression is not feasible, because of too inaccurate results (see the brown line-charts in Figure 1).

Since our domain expert wants to compare (a) absolute values of the measurements (differences between high and low temperatures) and (b) the shape (the slope of the daily temperature curve progressions not depending on absolute values), we decide to branch our pipeline and provide two different preprocessing scenarios. Scenario (b) is additionally normalized with a *MinMaxNormalization* to receive relative temperature curve patterns, while scenario (a) contains absolute values without normalization. Finally we define the *EuclideanDistance* as our similarity measure, since this is a common way for the earth observation scientists to compare their measurements.

We save our preprocessing scenarios. Thus, we are able to modify our preprocessing pipeline if changes in the requirements arise in future.

We assess the outcome of our time series scenario. With this pipeline we achieve a reduction of memory space of more than 98%. The computation of similarities between time series is accelerated considerably. Overall the search results of our prototypical time series search application were judged as very promising by the domain researchers. However, a full evaluation of the preprocessing system and the time series search application is still future work.

4.3. Use Case B: Extracting Features for Visual Representation of Time Series Data

Our second use case deals with the visual representation of large amounts of time series raw data for a web application. A compact representation of each time series has to be stored in a database for quick access. The goal is to optimize the compression of the raw data for minimal memory consumption and quick data transfer. At the same time the important features of the curve progressions have to be preserved for the web visualization. The knowledge of domain experts is crucial for the extraction of representations, since they can define the important visual features to be preserved for a representative line chart visualization.

This use case was developed in the project together with a domain expert from the AWI and the librarians who will host the web application in future. We reconstruct and refine the workflow as follows.

We choose the solar-dependent short-wave downward radiation (SWD) measurements as our data source. Again, the statistical information about the raw data is calculated and visualized in the raw data selection view. In contrast to the first use case the range of values is different. It spans from 0 W/m^2 (at night - no radiation) to approximately 1500 W/m^2 in desert-like environments. The standard deviation and the degree of noise is larger than in temperature curves, which constitutes an additional difficulty for the time series preprocessing pipeline to be designed. SWD measurements contain a variety of spikes, caused by changing terms of cloudiness. This effect (and its geographic dependency) is important, e.g. for photovoltaic power generation. A popular search scenario is the identification of so-called clear-sky conditions, when the sky is free of any clouds. In this case, a curve progression is nearly sine-like as it simply follows the solar altitude. However, our scenario also has to preserve the fidelity

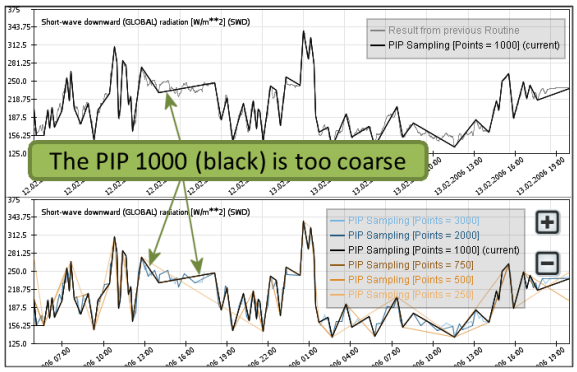


Figure 5: *The PIP 1000 causes losses in fidelity. However, we use the PIP 2000 (dark blue) that has almost the same visual appearance as the raw data (gray).*

of cloudy weather conditions when measurement curves are very noisy (cf. Figure 5).

Firstly we reuse the pipeline from use case A. Subsequent to the *MissingValueRemover*, the *TimeStampSorter*, and the *IdenticalTimeStampHandler*, we create a new branch for use case B. To reduce the fraction of 'micro noise', we apply a 3-minute *MovingAverageCalculator* (the measurements are mostly taken with a one minute quantization).

In this scenario we can neglect the sequence segmentation since we want to visually represent the whole time series in the web application. As a next step we have to define a descriptor that represents the original time series preserving its important visual features, and at the same time optimizes the compression level. We choose the *PerceptionalImportantPoints* descriptor (PIP) as applied in [ZJGK10], since the algorithm is data adaptive and in particular sensitive to the preservation of fidelity. At the start of our project this pipeline has been designed 'by hand'. It compressed the time series to 1000 data points per month. Now, we can see with our system that this is not sufficient in general. The degree of noise in some raw data emerges to be larger than initially expected, an example curve is shown in Figure 5. The refinement of the parameterization to 2000 data points per day reduces losses in fidelity. A benefit of the visual parameter support and the guided selection of input data.

As a result of our preprocessing pipeline we achieve a reduction of memory space of more than 95% with a satisfying visual representation of the time series, again acknowledged by the domain experts. Our prototypical visual time series search system can now be equipped with a linechart visualization that resembles the original raw data.

5. Conclusion and Future Work

The motivation of our work was to make time series preprocessing visible and accessible to domain experts. We presented an approach for the interactive diagnosis, design and control of preprocessing methods used for time series analysis. It allows to compose a pipeline of individual operations drawn from an extensible toolkit. Different steps of the pipeline can be arranged in a very flexible way, because of a common underlying data structure. The composition is guided by a series of line-charts showing intermediate results and statistics. A user learns about the characteristic properties of the data, and how these properties are changed by every step of the pipeline. Many unwanted effects can be pinpointed to a specific operation or parameter. In addition the system provides visual cues for the defensible improvement of parameter settings and the selection of testing data. By exposing the preprocessing and its effects in a visual way, the confidence of the experts to the results have been increased greatly. In the collaboration between experts parameter settings are now open for explanation and discourse. While we illustrated the use of the system with climate data only, the system can be used for time series preprocessing in many other domains. This even applies to fields that use specialized data transformations.

We consider our work a starting point which can be extended with respect to a number of aspects. For example, our system does not include a comprehensive toolkit of transformation methods known from literature. New operations, similarity measures or descriptors can be added to the toolkit, without compromising the system's architecture. A more challenging extension is to expose a relation between the raw data space, the feature space and the results of the analysis. While the first draft of the preprocessing pipeline often draws upon limited knowledge, new analytical findings necessitate further adaptation. Exposing the right leverage points for this adaptation would help to improve the pipeline even faster. Another way to use experts knowledge for data preprocessing is to consider labeled raw data. Labeled data usually serves as a 'ground truth' to measure classification quality. However, since differences and similarities imposed by the labeling should be preserved by the preprocessing, it also could be used as a ground truth to measure its fidelity.

In summary, shifting our attention from visual analysis to visual preprocessing was beneficial. Our collaboration showed that the ability to expose and discuss decisions and their effects is crucial to improve analytical processes and results.

Acknowledgments

We thank the Alfred Wegener Institute (AWI) in Bremerhaven, Germany. The researchers helped us in understanding the data domain, in selecting an appropriate PANGAEA data subset and in creating time series scenarios. We thank Tobias Schreck from the University of Konstanz for a profound and inspiring collaboration. This work was supported by a grant from the Leibniz Association as part of the 'Joint Initiative for Research and Innovation' program.

References

[AMST11] AIGNER W., MIKSCH S., SCHUMANN H., TOMINSKI C.: *Visualization of Time-Oriented Data*. Springer, London, UK, 2011. 41, 42

[BBF*11] BERNARD J., BRASE J., FELLNER D., KOEPLER O., KOHLHAMMER J., RUPPERT T., SCHRECK T., SENS I.: A visual digital library approach for time-oriented scientific primary data. *Springer International Journal of Digital Libraries, ECDL 2010 Special Issue* (2011). 45

[BK12] BERNARD J., KÖNIG-LANGLO G. SIEGER R.: Time-oriented earth observation measurements from the baseline surface radiation network (bsrn) in the years 1992 to 2012 , reference list of 6813 datasets. doi:10.1594/pangaea.787726, 2012. 45

[BvLBS11] BREMM S., VON LANDESBERGER T., BERNARD J., SCHRECK T.: Assisted descriptor selection based on visual comparative data analysis. *Comput. Graph. Forum 30*, 3 (2011), 891–900. 41, 42

[CBK09] CHANDOLA V., BANERJEE A., KUMAR V.: Anomaly detection: A survey. *ACM Comput. Surv. 41*, 3 (July 2009). 41

[DTS*08] DING H., TRAJCEVSKI G., SCHEUERMANN P., WANG X., KEOGH E.: Querying and mining of time series data: experimental comparison of representations and distance measures. *Proc. VLDB Endow. 1*, 2 (Aug. 2008), 1542–1552. 41

[Fu11] FU T.-C.: A review on time series data mining. *Engineering Appl. of Artificial Intelligence 24*, 1 (2011), 164–181. 41

[GAIM00] GAVRILOV M., ANGUELOV D., INDYK P., MOTWANI R.: Mining the stock market: Which measure is best. In *In proceedings of the 6 th ACM Int'l Conference on Knowledge Discovery and Data Mining* (2000), pp. 487–496. 41

[HDKS05] HAO M. C., DAYAL U., KEIM D. A., SCHRECK T.: Importance-driven visualization layouts for large time series data. In *INFOVIS* (2005), IEEE Computer Society, p. 27. 42

[IMI*10] INGRAM S., MUNZNER T., IRVINE V., TORY M., BERGNER S., MÖLLER T.: Dimstiller: Workflows for dimensional analysis and reduction. In *Proceedings of the 5th IEEE Conference on Visual Analytics in Science and Technology (VAST)* (Florida, USA, 2010), IEEE Computer Society. 42

[KCH*03] KIM W., CHOI B.-J., HONG E.-K., KIM S.-K., LEE D.: A taxonomy of dirty data. *Data Min. Knowl. Discov. 7*, 1 (Jan. 2003), 81–99. 41

[KCHP01] KEOGH E., CHU S., HART D., PAZZANI M.: An online algorithm for segmenting time series. In *In ICDM* (2001), pp. 289–296. 41, 42

[KCPM00] KEOGH E., CHAKRABARTI K., PAZZANI M., MEHROTRA S.: Dimensionality reduction for fast similarity search in large time series databases. *Journal of Knowledge and Information Systems 3* (2000), 263–286. 40

[KHP*11] KANDEL S., HEER J., PLAISANT C., KENNEDY J., VAN HAM F., RICHE N. H., WEAVER C., LEE B., BRODBECK D., BUONO P.: Research directions in data wrangling: visuatizations and transformations for usable and credible data. *Information Visualization 10*, 4 (Oct. 2011), 271–288. 39, 41

[KK03] KEOGH E., KASETTY S.: On the need for time series data mining benchmarks: A survey and empirical demonstration. *Data Min. Knowl. Discov. 7*, 4 (Oct. 2003), 349–371. 41, 42

[KLR04] KEOGH E., LONARDI S., RATANAMAHATANA C. A.: Towards parameter-free data mining. In *Proceedings of the ACM SIGKDD int. conf. on Knowledge discovery and data mining* (New York, NY, USA, 2004), KDD '04, ACM, pp. 206–215. 41

[LKLC03] LIN J., KEOGH E., LONARDI S., CHIU B.: A symbolic representation of time series, with implications for streaming algorithms. In *Proceedings of the 8th ACM SIGMOD workshop on Research issues in data mining and knowledge discovery* (New York, NY, USA, 2003), DMKD '03, ACM, pp. 2–11. 41

[ODF*98] OHMURA A., DUTTON E. G., FORGAN B., FRÖHLICH C., GILGEN H., HEGNER H., HEIMO A., KÖNIG-LANGLO G., MCARTHUR B., MÜLLER G., PHILIPONA R., PINKER R., WHITLOCK C. H., DEHNE K., WILD M.: Baseline surface radiation network (BSRN/WCRP): New precision radiometry for climate research. *Bull. Amer. Met. Soc. 79* (1998), 2115–2136. 45

[PBCR11] PRETORIUS A. J., BRAY M.-A., CARPENTER A. E., RUDDLE R. A.: Visualization of parameter space for image analysis. *IEEE Transactions on Visualization and Computer Graphics 17*, 12 (Dec. 2011), 2402–2411. 41

[SBVLK09] SCHRECK T., BERNARD J., VON LANDESBERGER T., KOHLHAMMER J.: Visual cluster analysis of trajectory data with interactive kohonen maps. *Information Visualization 8*, 1 (Jan. 2009), 14–29. 42

[SS04] SEO J., SHNEIDERMAN B.: A rank-by-feature framework for unsupervised multidimensional data exploration using low dimensional projections. In *in Proceedings of IEEE Symposium on Information Visualization* (2004), pp. 65–72. 41

[SSW*12] SCHRECK T., SHARALIEVA L., WANNER F., BERNARD J., RUPPERT T., VON LANDESBERGER T., BUSTOS B.: Visual Exploration of Local Interest Points in Sets of Time Series. In *Proc. IEEE Symp. on Visual Analytics Science and Technology (Poster Paper, accepted for publication)* (2012). 42

[WL05] WARREN LIAO T.: Clustering of time series data-a survey. *Pattern Recogn. 38*, 11 (Nov. 2005), 1857–1874. 41

[ZCPB11] ZHAO J., CHEVALIER F., PIETRIGA E., BALAKRISHNAN R.: Exploratory analysis of time-series with chronolenses. *IEEE Transactions on Visualization and Computer Graphics 17*, 12 (2011), 2422–2431. 42

[ZJGK10] ZIEGLER H., JENNY M., GRUSE T., KEIM D. A.: Visual market sector analysis for financial time series data. In *IEEE VAST* (2010), IEEE, pp. 83–90. 47

SIGRAD 2012
A. Kerren and S. Seipel (Editors)

Real-time Image Based Lighting with Streaming HDR-light Probe Sequences

Saghi Hajisharif[†], Joel Kronander[‡], Ehsan Miandji[§], and Jonas Unger[¶]

Linköping University, Sweden

Abstract

We present a framework for shading of virtual objects using high dynamic range (HDR) light probe sequences in real-time. Such images (light probes) are captured using a high resolution HDR camera. In each frame of the HDR video, an optimized CUDA kernel is used to project incident lighting into spherical harmonics in real time. Transfer coefficients are calculated in an offline process. Using precomputed radiance transfer the radiance calculation reduces to a low order dot product between lighting and transfer coefficients. We exploit temporal coherence between frames to further smooth lighting variation over time. Our results show that the framework can achieve the effects of consistent illumination in real-time with flexibility to respond to dynamic changes in the real environment.

Categories and Subject Descriptors (according to ACM CCS): I.3.3 [Computer Graphics]: Picture/Image Generation—Image based lighting, pre-computed radiance transfer, high dynamic range video

1. Introduction

Image Based Lighting (IBL), [Deb98], is a widely used technique for photo-realistic rendering of virtual objects so that they can be seamlessly composited into still or video footage captured in real scenes. The key idea of IBL is to capture the lighting present in the real scene and use this information to illuminate the virtual objects. The scene lighting in traditional IBL is measured by capturing an omni-directional High Dynamic Range (HDR) image, or HDRi, at a single point in space. Such a panoramic HDRi is generally called a *light probe*. Since the HDRi captures the full dynamic range in the scene (from the direct light sources to the parts of the scene that are in shadow), the light probe can be thought of as a measurement of the scene radiance incident at the point in space where the panorama was captured, and can be used as an approximation of the lighting in the scene during rendering. The ease of use and level of realism attainable have now made IBL a standard tool in most production pipelines, and even in real-time applications (based on approximations).

The key challenge in real-time IBL rendering is that the scene lighting is described as an image with no explicit information about where light sources and other high intensity regions are located. When each fragment in the scene is shaded during rendering, this leads to a significant sampling problem of the spherical radiance distribution described by the light probe image. This has led to the development of techniques for Pre-computed Radiance Transfer (PRT), for an overview see [Ram09]. In PRT, the light transport is approximated using basis projections, e.g. spherical harmonics or wavelets, in which the light probe HDRi, material properties, and local self occlusion on the virtual objects can be represented with only a small number coefficients, and the interaction between lighting, material and geometry can be efficiently computed in the transformed space.

Traditional IBL has been limited to only static lighting environments. This is due to the fact that there are no cameras available on the market, that can capture true HDR images in a single shot. HDR images are commonly captured using exposure bracketing, a series of differently exposed images covering the dynamic range of the scene that are combined into a final HDR image. This limits the capture to

† sagha198@student.liu.se
‡ joel.kronander@liu.se
§ ehsan.miandji@liu.se
¶ jonas.unger@liu.se

static scenes and still images. However, recent developments in sensor and imaging hardware, [UG07, TKTS11, KUG12], have now made it possible to capture HDR-video (HDRv). This in turn also enables the capture of light probe video sequences, and thus IBL with dynamic real world lighting environments and moving light probes.

Contributions In this paper, we present a technique and system overview for real-time IBL using HDRv light probe sequences. Real world scene lighting is recorded using a high quality and high resolution 4Mpixel HDRv camera running at 25 or 30 frames per second (fps). Utilizing a real-time CUDA kernel, the input HDR images are processed and the spherical radiance distribution described by each frame in the video sequence is projected onto a low order spherical harmonics basis. Using precomputed techniques, real-time image based rendering of synthetic objects is achieved. and used for image based rendering of synthetic objects.

2. Background and Previous Work

The way in which illumination varies in a scene as a function of location in space (x,y,z), time t, direction (ϕ,θ) and wavelength λ is described by the plenoptic function $P(x,y,z,\phi,\theta,\lambda,t)$, Adelson and Bergen [AB91]. In computer graphics image synthesis, this corresponds to the radiance distribution $L(\mathbf{x},\vec{\omega}_i)$ incident at a surface point $\mathbf{x}$ from an angular region subtended by a set of directions $\vec{\omega}_i$, as described by the rendering equation, Kajiya [Kaj86]:

$$L(\mathbf{x},\vec{\omega}_o) = \int_\Omega L(\mathbf{x},\vec{\omega}_i)\rho(\mathbf{x},\vec{\omega}_i,\vec{\omega}_o)d\vec{\omega}_i \qquad (1)$$

where ρ describes how the surface material at $\mathbf{x}$ transforms radiance incident from a direction $\vec{\omega}_i$ towards the outgoing direction $\vec{\omega}_o$ from which the point $\mathbf{x}$ is observed. In computer graphics, the time t is usually fixated, and the spectral characteristics, λ, of the plenoptic function, P, are usually described using three spectral bands for red, green and blue colors respectively.

Based on the observation that the lighting plays a key role in the visual realism of computer generated images, Debevec [Deb98] introduced image based lighting; a technique that enables virtual objects to be rendered into real scenes and appear as if they were actually there. In traditional IBL the incident radiance distribution $L(\mathbf{x},\vec{\omega}_i)$ is described by a panoramic HDR image I_j captured in a real scene, where j denotes the linear index for each pixel in the image. Each pixel j in I_j can be thought of as the radiance contribution from the solid angle of direction $\vec{\omega}_i$ subtended by the pixel area, i.e. $L(\mathbf{x},\vec{\omega}_i) \approx I_j$. Since the panoramic image I is captured at a single point in space, the spatial coordinate $\mathbf{x}$ vanishes, and the image I_j describes only the angular variations in the incident radiance distribution. This corresponds to the approximation that the lighting environment captured in I_j is infinitely far away from the virtual objects being rendered.

IBL has traditionally been limited to scene lighting captured at a single point in space and at a single instant in time. The main reason for this is that the HDR image capture, as introduced in the computer graphics community by Debevec and Malik [DM97], has been carried out by combining a series of differently exposed low dynamic range images into an HDR image covering the full dynamic range of the scene. This technique, often referred to as exposure bracketing, requires that several images are captured and can thus not handle dynamic scenes or moving cameras. For an overview of the background of HDR imaging see Reinhard et al. [RWPD06].

Moving beyond conventional cameras, a number of approaches and hardware setups for HDR imaging and even video have been proposed in the literature. In order to minimize the temporal artifacts in the exposure bracketing algorithm, Unger et al. [UG07] presented an HDR video camera, where they programmed a SMART CMOS sensor from SICK-IVP AB to capture the exposures back to back on a per pixel basis in rolling shutter fashion, and do the HDR assembly directly on the sensor chip itself. Another option to the time-multiplexing is to trade spatial resolution for dynamic range. Nayar and Mitsunaga [NM00] placed spatially varying exposure filter array in front of the sensor. This approach was extended to rgb-color image capture [NN05] and even capture of multi-spectral HDR-images [YMIN10]. Currently the best performing approach for single shot and HDR-video capture is based on the idea of internally, inside the camera system, splitting the optical path onto multiple synchronized sensors [AA04, WRA05, MH07, TKTS11, KUG12]. By placing Natural Density (ND) filters with varying density in front of the sensors, e.g. [AA04], or more sophisticated beam splitter optics, e.g. [TKTS11], the different low dynamic range exposures can be captured with full resolution at the same instant in time and with the same integration time. This prevents ghosting artifacts from scene motion and ensures correct motion blur for all sensors. These systems, now, also enable high quality HDR-video capture.

The benefit and impressive rendering results from using IBL has motivated research and development of techniques for real-time rendering. These techniques use a single light probe image captured at a single instant in time, and generally focus on describing $L(\mathbf{x},\vec{\omega}_i)$ and ρ in the rendering equation, Eq. 1, as well as local visibility information using approximations that make it possible to solve the rendering equation in real-time for complex scenes. Ramamoorthi and Hanrahan [RH01] projected the captured illumination onto *Spherical Harmonics* (SH) basis functions, and showed that diffuse materials could be accurately simulated by representing the environment illumination with 9 SH coefficients. Sloan et al. [KSS02, SKS02, SLS05] later extended this technique and introduced *Precomputed Radiance Transfer* (PRT) of glossy materials, performing expensive computations as a pre-computation step. An in-depth overview of PRT-methods is presented in Ramamoorthi [Ram09].

For dynamic environment maps methods like sequential Monte Carlo sampling [GDH06] have also been considered however these methods do not run in real-time and therefore not suitable for our purpose.

Building on this body of work, our method extends IBL and PRT to include also the temporal domain, i.e. in our framework $L(\mathbf{x},\vec{\omega}_i)$ in Eq. 1 becomes $L(\mathbf{x},\vec{\omega}_i,t)$. We also demonstrate how the projection of the captured light probes onto a spherical harmonics basis can be parallelized and computed in real-time for each frame in the input HDR video sequences. This means that, under the assumption of low angular frequency in the illumination, our processing and rendering framework supports dynamic HDR environment maps.

3. PRT for Realtime Rendering with HDR Video Sequences

Considering the illumination as HDR light probes, the underlying assumptions in this paper are that each pixel I_j in a light probe image can be thought of as a radiance contribution from a corresponding incident direction $\vec{\omega}_i$, and that the captured real environment is far enough so that the radiance contribution I_j is parallel over the entire scene.

Rendering objects that are illuminated by such distant environmental lighting requires solving the rendering integral, Eq. 1. The integrand describes the product between the incident radiance distribution and the surface material that includes the cosine falloff factor. To take into account shadowing, we also include a visibility factor $V(\mathbf{x},\vec{\omega}_i)$ that describes the self occlusion at a surface point $\mathbf{x}$ such that: $V(\mathbf{x},\vec{\omega}_i)=1$ if the distant environment is visible in direction $\vec{\omega}_i$, and $V(\mathbf{x},\vec{\omega}_i)=0$ if geometry occludes the environment in the direction $\vec{\omega}_i$. Assuming that the scene is static and only contains Lambertian material, the material ρ will be independent of the viewing angle $\vec{\omega}_o$ and only depend on the diffuse albedo $\rho_A(\mathbf{x})/\pi$ and the cosine between $\vec{\omega}_i$ and the surface normal $\mathbf{N}_x$ at $\mathbf{x}$. Consequently, Eq.1 (with the temporal domain included) is simplified to the following for Lambertian surfaces:

$$L(x,\vec{\omega}_o,t)=\frac{\rho_A(\mathbf{x})}{\pi}\int_{\Omega}L(\vec{\omega}_i,t)V(\mathbf{x},\vec{\omega}_i)(\mathbf{N}_x\cdot\vec{\omega}_i)d\vec{\omega}_i \quad (2)$$

where Ω is the hemisphere centered at the normal $\mathbf{N}_x$. We now define what is referred to as the transfer function T, that includes the visibility and cosine falloff :

$$T(\mathbf{x},\vec{\omega}_i)=V(\mathbf{x},\vec{\omega}_i)(\mathbf{N}_x\cdot\vec{\omega}_i) \quad (3)$$

and the rendering integral becomes:

$$L(\mathbf{x},\vec{\omega}_o,t)=\frac{\rho_A(\mathbf{x})}{\pi}\int_{\Omega}L(\vec{\omega}_i,t)T(\mathbf{x},\vec{\omega}_i)d\vec{\omega}_i \quad (4)$$

Our task is now to compute the rendering integral at high frame rate, while supporting HDR light probe sequences as input. Our algorithm is based on pre-computed radiance transfer [SKS02] and extends these ideas to the temporal domain. We approximate the spherical lighting measurements I_j and local transfer function T using spherical harmonics and utilize the orthogonality of this basis to efficiently solve the rendering integral Eq. 4 as a dot product between SH coefficients. For the sake of the presentation, here we use static scenes and Lambertian materials. It should, however, be noted that our technique applies similarly to previously presented methods in PRT that support both dynamic scenes as well as glossy materials, e.g. [SLS05].

3.1. Spherical Harmonics Representation

Definition Spherical harmonics basis functions, denoted as $y_l^m(\theta,\phi)$ are frequency space basis functions which are defined over the unit sphere. The order, or the number of bands, of the SH functions is indicated by l, and m is the degree where $l>0$ and $-l<m<l$. Using real-valued spherical basis functions, we can define them as follows:

$$y_l^m(\theta,\phi)=\begin{cases}K_l^m P_l^{|m|}(\cos(\theta))cos(|m|\phi) & \text{if } m\geq 0\\ K_l^m P_l^{|m|}(\cos(\theta))sin(|m|\phi) & \text{if } m<0\end{cases} \quad (5)$$

where K_l^m are normalization constants, and $P_l^{|m|}$ are Legendre polynomials. Each band is equivalent to polynomials of the same degree. Since the SH functions are orthonormal, the projection of a spherical function $f(\vec{\omega})$ defined over a domain s onto the SH basis functions is as follows:

$$f_l^m=\int_s f(s)y_l^m(s)ds \quad (6)$$

An n^{th}-order SH basis function is described by n^2 coefficients. It is common practice to truncated the order, n, to a finite number, in the PRT case typically $3,4$ or 5 . An approximation of the projected function $f(\vec{\omega})$ can then be reconstructed from the truncated SH basis functions according to:

$$f(\vec{\omega})\approx\sum_{l=0}^{n-1}\sum_{m=-l}^{l}y_l^m(\vec{\omega})f_l^m \quad (7)$$

Spherical harmonics are rotation invariant and orthonormal. The orthonormality makes it possible to compute the

convolution of two function defined on a spherical domain as a dot, or inner, product of the SH coefficients in the projective space. In the PRT case the transport function $T(\mathbf{x}, \vec{\omega}_i)$ and the spherical radiance distribution $L(\vec{\omega}_i, t)$ are projected onto an SH basis, and the dot product between the SH coefficients corresponds to solving the integral in Eq. 4. For more details about the SH basis functions and their use in PRT, we refer to [RH04], [Ram05] and [Gre03].

3.2. Algorithm Overview

We use an experimental setup consisting of an HDR video camera with a resolution of 2400×1700 pixels capturing the full dynamic range of the environment through the reflection in a mirror sphere (see Section 4). The camera is running at 25 or 30 fps and the processing and rendering is performed in real-time. Our algorithm can be outlined in an off-line pre-processing step, an on-line processing and rendering step:

Off-line pre-processing of the transfer function:
At each vertex on each object in the virtual scene, the transfer function, T, described in Eq. 3,is calculated and projected onto an SH basis. This information is stored on disk and uploaded onto the GPU at runtime.

On-line processing:
For each HDR light probe image streamed to the GPU, the real-time algorithm can be outlined as:

1. *Light probe image processing* - Down-sampling, filtering and processing of the HDR light probe image prior to SH projection.
2. *Lighting SH projection* - The down-sampled image, I_j, is projected onto a SH basis of order $3, 4$ or 5 according to Eq. 6 above. A CUDA kernel is used to accelerate the process and enables real-time performance.
3. *Temporal filtering* - Since we only use a single light probe image at each time step t, the entire virtual scene will be affected by lighting variations that in reality only would affect a small part of it. This may introduce flickering artifacts introduced by strong spatial variations in light probe images. To avoid this, we (optionally) perform filtering of the projected lighting SH coefficients in the time domain.
4. *Lighting reconstruction and rendering* - The SH representation of the pre-processed transfer function, T, of each vertex in the scene and the SH projection of the incident illumination $L(\vec{\omega}_i, t)$ are used to efficiently solve the rendering integral, Eq. 4, at each fragment being rendered. Finally, the full resolution image is projected onto a quad oriented perpendicular to the virtual camera and used as backdrop onto which the rendered objects are composited.

Below, we describe each step in our algorithm in detail, and present an overview of the real-time processing, SH-projections, filtering and reconstruction.

3.3. Off-line Pre-processing of the Transfer Function

The local visibility, V, in Eq. 2 is computationally expensive to compute, and cannot be determined on-line. Therefore the transfer function T in Eq.3, is calculated in a pre-processing step. For each vertex on each surface, the local visibility is sampled on a spherical domain using ray-casting. The dot product between the sample direction and the surface normal is calculated, and the transfer function, T is computed and projected onto SH basis functions as described in Eq.6. The transfer function coefficients , which we denote as c^T_{lm}, are stored to be used during on-line image synthesis. Using Monte-Carlo integration, the integral of Eq.6 can be numerically evaluated as:

$$c^T_{lm} = \int_\Omega T(x, \vec{\omega}_i) {Y_l}^m(\omega_i) d\omega_i \approx \sum_{i=1}^{N} w(\vec{\omega}_i) T(x, \vec{\omega}_i) {y_l}^m(\vec{\omega}_i) \tag{8}$$

where $w(\vec{\omega}_i)$ is a weighting function and N is the number of sample directions. Using stratified sampling over a unit sphere leads to a constant weight function $w(\vec{\omega}_i) = 4\pi/N$. Note that at each shading point we are sampling a spherical domain instead of hemisphere. This is to avoid transformation of global coordinate to local coordinate.

3.4. Light Probe Image Processing

The HDR light probe image available for time step t, is streamed to the GPU. We first iteratively down-sample the image to a lower resolution version for the SH projection. This is reasonable as the order of the SH projection is generally low and thus low pass filter the image. We use a Gaussian blur filter for down-sampling. The result of down-sampling is passed to the projection kernel.

3.5. Lighting SH Projection

The environment lighting is considered to be dynamic and based on HDR video streams. This means that the incident lighting is changing in each frame, and the SH approximation of the radiance distribution needs to be re-computed for each frame. Traditional methods of projecting the environment lighting onto SH basis functions require a considerable processing time not suitable for real-time frame rates and temporal coherency. In order to accelerate this process, we use GPGPU programming to perform these operations in real time.

During the SH projection, we loop over each pixel in the input image, i.e. we perform uniform sampling in image space, and that each pixel corresponds to a solid angle in a certain direction. The light probe images are captured in real-time and mapped with latitude longitude mapping. The image domain is parametrized with $u, v \in [0, 1]$. Thus

the mapping from image coordinates to directions in world coordinates can be described as:

$$(\theta, \phi) = (\pi v, 2\pi u) \tag{9}$$

$$(D_x, D_y, D_z) = (rsin\theta cos\phi, rsin\theta sin\phi, rcos\theta) \tag{10}$$

The incident lighting corresponding to each pixel is then projected onto SH basis functions. As described in Section 3.3, projecting a function, f onto an SH basis requires the integral of the products of f, and the SH basis functions over the domain of f to be computed. The incident lighting, denoted $L(\vec{\omega}_i)$, is projected onto SH basis functions according to:

$$c_{lm}^L = \int_{S^2} L(\vec{\omega}_i) {Y_l}^m(\vec{\omega}_i) d\vec{\omega}_i \tag{11}$$

Using a Riemann sum over the light probe image, Eq.11 can be estimated numerically as follows:

$$c_{lm}^L \approx \frac{1}{N} \sum_{i=1}^{N} w(\theta, \phi) L(\theta, \phi) {Y_l}^m(\theta, \phi) \tag{12}$$

where $N = n_u n_v$, n_u and n_v determines the number of samples on v and u. $w(\theta, \phi) = sin\theta \times 2\pi^2$ is the area measure transformation of sampling from (u, v) space to the direction on the space on the sphere. In order to enable real-time processing we perform these computations using a CUDA kernel.

In order to make use of all available cores on the GPU and to minimize memory latency, CUDA programming requires launching a large number of threads and using fine grained parallelism.

We have designed a CUDA kernel assigning one thread to each pixel in the light probe image. During execution, the threads are grouped together into blocks [NVI11]. The threads are distributed over a set of 2-dimensional blocks. Each block has access to a limited fast shared memory. In the process 16×16 pixels of light probe image are passed to each block where n^2 (for n-band SH basis) coefficients are computed. Each block calculates a partial sum of Eq.12 that is stored in the shared memory. Shared memory is chosen for achieving better performance, since it is the fastest way for the threads within a block to communicate with each other. The result of one block is shown in Fig. 1.

The actual coefficients are then calculated by summing up the projection results from each thread and block. In order to calculate the first coefficient, we need to add up the projection results of N light directions onto the first SH basis function. Therefore, the first element of each thread is added together. This process is applied for the other elements in each block. We use a parallel reduction technique for computing the partial sum in each block, shown in Fig.2. At each step, the addition is reduced in half and this process is repeated until all threads are processed. The result is written into global memory and further reduction is performed between blocks until all partial sums of Eq.12 are calculated. Using reduction is the best way to massively parallelize the computation. In the kernel, samples are converted from uv-coordinate to spherical coordinate using Eq.10 to compute the n^2 SH coefficients.

Spherical harmonics basis functions are well-suited for reconstructing low frequency lighting and shadows. However, when the signal is clamped, this reconstruction is accompanied with a ringing effect known as Gibbs phenomena [HH79]. We use a Hanning window to minimize this problem, see [Slo08] for more details.

3.6. Temporal Filtering

The PRT approximation of distant environments means that we use only a single light probe image per time step. Since a light probe image measures only angular variations in the radiance distribution, it is not possible to capture the spatial variation that would occur in the real world if an object was gradually moving from e.g. strong illumination into shadow. Instead the entire rendered scene will be affected at once. In lighting environments with high spatial variations this may lead to flickering. To avoid such temporal artifacts, we (optionally) perform filtering of the lighting SH coefficients in the time domain. Using the lighting coefficients of the previous frame , $c_{lm}^L(t-1)$, and the current frame, $c_{lm}^L(t)$, the final lighting coefficient is calculated by a linear interpolation as follows:

$$c_{lm}^L(t) = \frac{c_{lm}^L(t) + \alpha c_{lm}^L(t-1)}{\alpha + 1} \tag{13}$$

where α is a non-negative constant. Note that since each two sequential coefficients are interpolated, the coefficients of the previous frames will have a small affection on the current frame's coefficients. This leads to a smooth transition in illumination changing.

3.7. Lighting Reconstruction and Rendering

To relight the scene, according to Eq.2, the projected functions need to be reconstructed. Using orthonormality of SH basis functions, the projections of two functions, T and L over the unit sphere, satisfy:

$$L(\vec{\omega}_i, t) \approx \sum_{k=1}^{n} c_k^L Y_k(\vec{\omega}_i)$$

$$T(x, \vec{\omega}_i) \approx \sum_{k=1}^{n} c_k^T Y_k(\vec{\omega}_i)$$

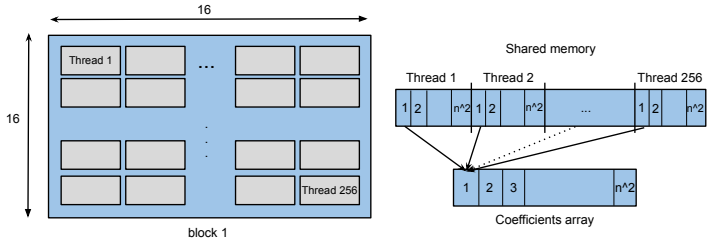


Figure 1: *The arrangements of threads in each block and the shared memory used for storing the result.*

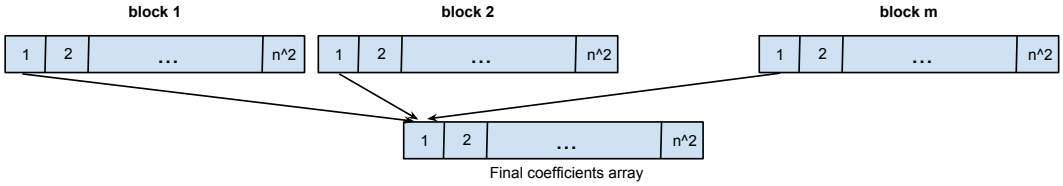


Figure 2: *The result of partial summations in each block is added to the other results using reduction. m is dependent on the width and height of the input light probe image.*

$$L(x,\vec{\omega}_o,t) = \int T(x,\vec{\omega}_i)L(\vec{\omega}_i,t)d\vec{\omega}_i \approx \sum_{i=1}^{n^2} c_i^L c_i^T. \qquad (14)$$

Thus, the rendering integral, Eq.2, is reduced to a scalar product of the projected coefficients.

4. Results and Discussions

In this section, we give an overview of the experimental setup used to test our approach in practice, present a set of example renderings, as well as an evaluation of the algorithm in terms of performance.

4.1. HDR Video Setup

The dynamic HDR light probe input data used for the experiments presented in this paper were captured using an HDR video camera prototype developed in a collaboration between Linköping University, Sweden and the German camera manufacturer SpheronVR AG. The camera system captures images with a resolution of 2400×1700 pixels with a dynamic range in the order of $10.000.000 : 1$ at up to 30 frames per second. For the experiments presented here, the camera is mounted in a real-time light probe setup depicted in Figure 3. In this setup the HDRv camera uses a Zeiss Makro-Planar T* 100mm f/2 ZF.2 lens and images the scene through the reflection in a mirror sphere placed at a distance of 1250 mm from the center of projection along the optical axis. For each captured frame, this yields a close to 360° panoramic HDR image that covers the full dynamic range of the scene. HDR video sequences are stored on disk as individual OpenEXR frames.

Figure 3: *The HDRv camera captures HDR images of* 2400×1700 *pixels exhibiting a dynamic range in the order of* $10.000.000 : 1$ *at up to* 30 *frames per second. Here the camera is displayed in a real-time light probe setup and images the scene through the reflection in a mirror sphere.*

4.2. Results

The test results were acquired on a 3.2 GHz Xeon computer running Linux Ubuntu 11.04 with 23.6 GiB memory and an NVIDIA GeForce GTX580 graphics card. All stages of our method were implemented using C++, CUDA, OpenGL and GLSL respectively. For reducing memory bandwidth cost for data transfer between the GPU and CPU, we are using the OpenGL vertex buffer objects (VBO) extension. We are using diffuse reflecting surfaces and our models are standard models from Stanford university database.

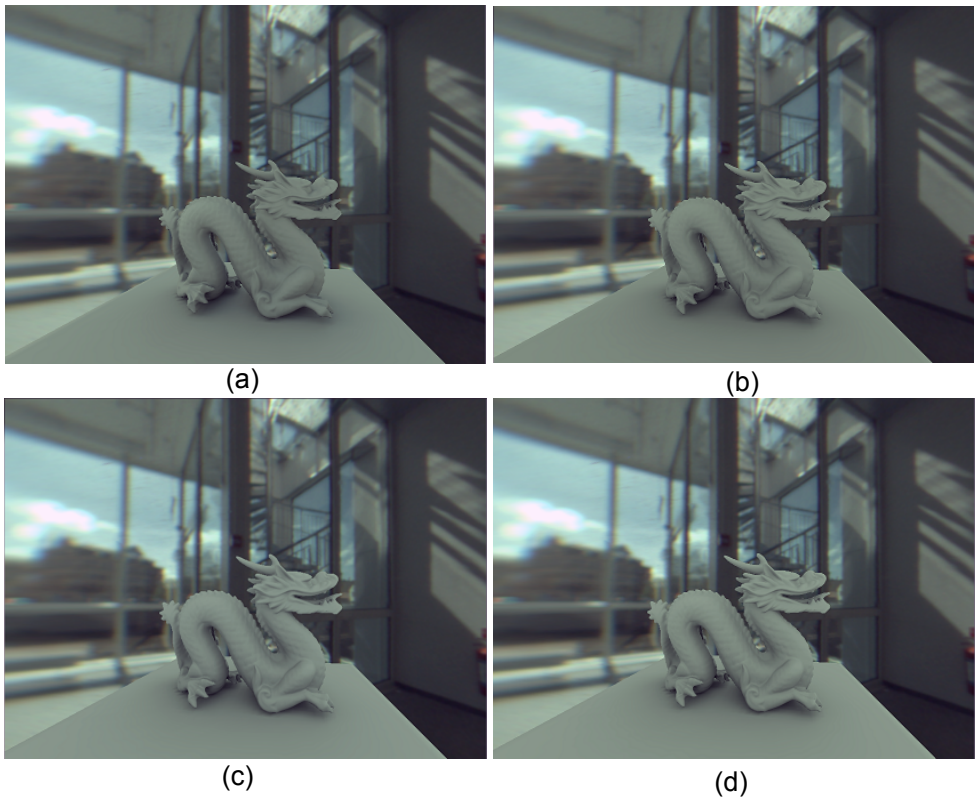


Figure 4: *Testing the shading of the scene with different scaling of the light probe image resolution of a single frame of the environment map. (a)* 1628×814*, (b)* 814×407 *(c)* 203×101 *(d)* 50×25

The CUDA kernel which calculates light's coefficients, utilizes overall $W \times H$ threads, where W is the width and H is the height of the image that is captured in real-time. For each HDR-video frame, the image is uploaded as an OpenGL texture to the graphics memory where it is used as input to the CUDA PRT kernel, as well as for rendering the environment background. For achieving better performance, the environment map input image is down-sampled before SH projection. The actual resolution can be changed in real-time.

Figure 4 displays the dragon model rendered with different resolution of the environment map. As shown in Fig. 4-b, the first iteration of the down-sampling does not have a visible impact on the output shading. However as the procedure continues with higher ratio, the shadows become softer and starting to fade away. Table 2 shows how the performance measured in fps scales according to light probe image resolution.

Table 1 shows timings of the pre-processing phase with visibility on and off. The visibility calculations are multi-threaded. For a test scene containing the Stanford bunny and a ground plane with 7455 vertices, and 14868 triangles, the processing time for computing the transfer function coefficients is 7.17 min with visibility testing and 0.1 sec without it. In Fig. 5 the dragon model is rendered with visibility testing on and off. Despite the time consuming pre-process step that is required, as Fig. 5-b indicates, shadows add to the realism of the results. The number of sample directions per shading point, are also important in calculating the transfer function for shadow testing. Using more samples will result in more accurate shadows and less ringing. This is shown in Fig. 8, where the model is rendered with 100, 900 and 2500 sample directions per vertex. The result shows that 2500 samples are enough for our application.

Using OpenGL vertex buffer objects, the transfer function coefficients are transferred as vertex attributes to the graphics memory. The final reconstruction and rendering is then performed in a GLSL shader. To do so for an object with 6384 vertices and with spherical harmonics with 4 bands, we need a texture of 6348×16 pixels stored in GPU memory. In order to perform the final multiplications according to Eq.14, the coefficients are transferred to the GLSL shader memory. We are using an RGB color model, and for each channel there will be 16 coefficients based on order-4 SH functions. The method was tested for 5^{th} order SH functions, and we found that for lighting a Lambertian surface order-

Table 1: *Pre-process time required for our test scenes with 2500 sample directions, (the plane is included in the calculations)*

Shape	*Vertices*	*Faces*	Time(Visibility off)	Time(Visibility on)
Sphere	6384	12760	0.1 sec	5.19 min
Bunny	7455	14868	0.1 sec	7.17 min
Buddha	54938	109900	0.9 sec	330 min
Dragon	54952	109900	0.9 sec	376 min

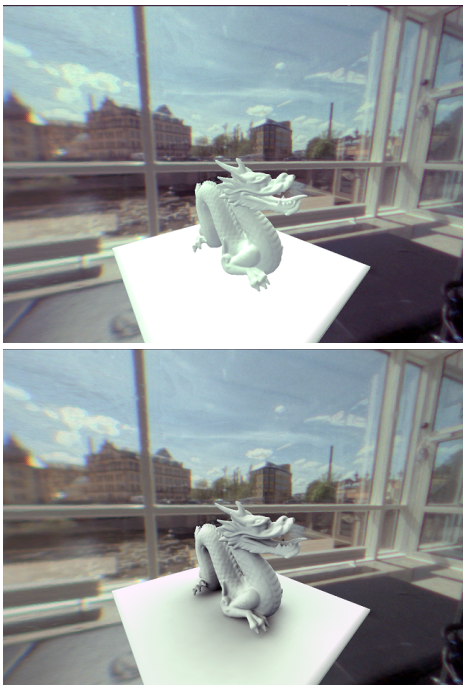


Figure 5: *The Dragon model rendered in the scene (a) without and (b)with visibility testing.*

4 SH functions suffice. However, in order to reduce ringing artifacts, higher order SH bases can be used. Ringing effects can also be minimized using spatial filtering as described.

Finally, Figure 7 shows renderings using input HDR-video sequences from three different scenes, captured both indoors and outdoors. The figure shows the result from the processing, reconstruction and rendering steps in the real-time algorithm described in Section 3.2.

The effect of temporal filtering is to reduce flickering to some extend and by experience we observed that adding more weights to the coefficients of the previous frame can result in a smoother change of illumination in the rendered objects. The result of temporal filtering is shown in the supplementary video attached to this paper.

To simulate specular reflection of the environment in the synthetic object we are using reflection mapping of the environment and the following equation is used for finding the final color of each pixel:

Figure 6: *Specular reflection using environment map and our method.*

$$Color_{final} = C_{diffuse} + c_k * C_{reflection} \tag{15}$$

where $c_k \in [0,1]$ denotes specular reflectivity in the material of the object. The result of this reflection is shown in Fig. 6

5. Conclusion and Future Works

This paper presented a method for real-time rendering of synthetic objects illuminated by real world lighting captured as HDR-video streams. The method is based on pre-computed radiance transfer, where the rendering integral is efficiently computed by projecting the reflectance distribution and spherical lighting environment onto spherical harmonics basis functions. The algorithm was tested using input data from a high resolution HDR-video camera.

In future work, we would like to investigate other basis functions as well as explore augmented reality as an application where real and virtual objects are mixed. We will also further investigate temporal filtering and, based on the coherency between consecutive frames in the input sequences,

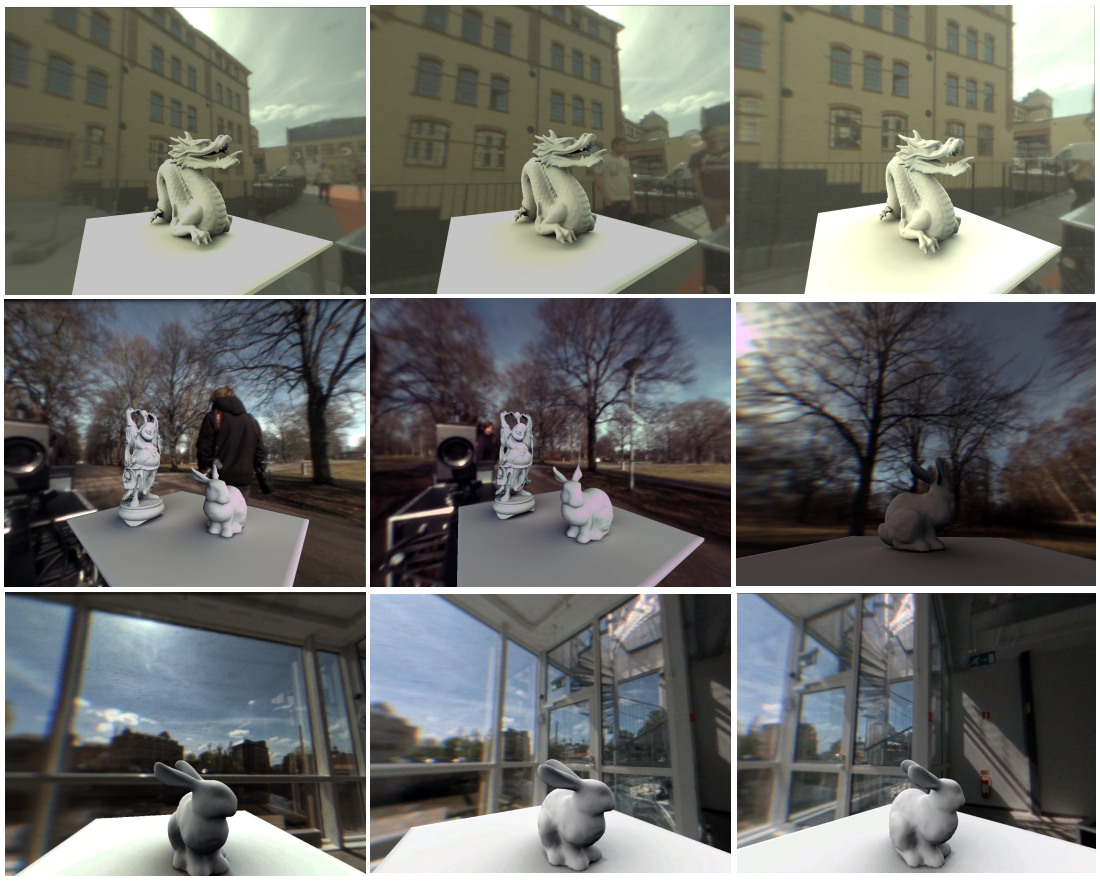


Figure 7: *Synthetic scenes rendered in three different environments.*

techniques for reusing already computed information in the basis projection.

6. Acknowledgement

We would like to thank Per Larsson for assisting with the hardware setups and organizing the lab environment. This work was funded through the Swedish Foundation for Strategic Research, the Linnaeus Environment CADICS, and the Center for Industrial Information Technology CENIIT.

Table 2: *Change of FPS according to light probe image resolution*

Resolution	*FPS*
1628×814	9
814×407	24
407×203	40
203×101	44
101×50	55
50×25	64

References

[AA04] AGGARWAL M., AHUJA N.: Split aperture imaging for high dynamic range. *International Journal of Computer Vision 58*, 1 (2004), 7–17. 50

[AB91] ADELSON E. H., BERGEN J. R.: *Computational Models of Visual Processing*. MIT Press, Cambridge, Mass., 1991, ch. 1. The Plenoptic Function and the Elements of Early Vision. 50

[Deb98] DEBEVEC P.: Rendering synthetic objects into real scenes: bridging traditional and image-based graphics with global illumination and high dynamic range photography. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1998), SIGGRAPH '98, ACM, pp. 189–198. 49, 50

[DM97] DEBEVEC P. E., MALIK J.: Recovering high dynamic range radiance maps from photographs. In *SIGGRAPH 97* (August 1997), pp. 369–378. 50

[GDH06] GHOSH A., DOUCET A., HEIDRICH W.: Sequential sampling for dynamic environment maps. In *ACM SIGGRAPH 2006 Sketches* (New York, NY, USA, 2006), SIGGRAPH '06, ACM. 51

[Gre03] GREEN R.: Spherical harmonic lighting: The gritty details. *Sony Computer Entertainment America* (2003). 52

[HH79] HEWITT E., HEWITT R. E.: The gibbs- wilbraham phe-

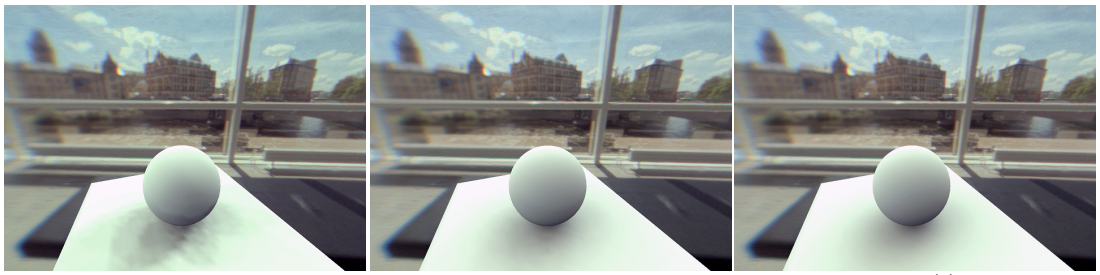


Figure 8: *The effect of under-sampling in calculating transfer function: a) Number of sample directions = 100, b) 900 and c)2500.*

nomenon: An episode in fourier analysis. *Arch. Hist. Exact Sci. 21* (1979), 129–160. 53

[Kaj86] KAJIYA J.: The rendering equation. In *SIGGRAPH 86* (1986), pp. 143–150. 50

[KSS02] KAUTZ J., SLOAN P.-P., SNYDER J.: Fast, arbitrary brdf shading for low-frequency lighting using spherical harmonics. In *EGRW '02: Proceedings of the 13th Eurographics workshop on Rendering* (Aire-la-Ville, Switzerland, Switzerland, 2002), Eurographics Association, pp. 291–296. 50

[KUG12] KRONANDER J., UNGER J., GUSTAVSON S.: Real-time hdr video reconstruction for multi-sensor systems. *Siggraph 2012 Posters* (August 2012). 50

[MH07] MCGUIRE M., HUGHES J. F.: Optical Splitting Trees for High-Precision Monocular Imaging. *IEEE Computer Graphics And Applications 27*, April (2007), 32–42. 50

[NM00] NAYAR S., MITSUNAGA T.: High dynamic range imaging: Spatially varying pixel exposures. In *Proc. of CVPR* (2000), pp. 472 – 479. 50

[NN05] NARASIMHAN S. G., NAYAR S. K.: Enhancing Resolution Along Multiple Imaging Dimensions Using Assorted Pixels. *IEEE Transactions on Pattern Analysis and Machine Intelligence 27*, 4 (2005), 518–530. 50

[NVI11] NVIDIA CORPORATION: NVIDIA CUDA C programming guide, 2011. Version 4.2. 53

[Ram05] RAMAMOORTHI R.: Modeling illumination variation with spherical harmonics. In *In Face Processing: Advanced Modeling and Methods* (2005), Academic Press. 52

[Ram09] RAMAMOORTHI R.: Precomputation-based rendering. *Found. Trends. Comput. Graph. Vis. 3*, 4 (Apr. 2009), 281–369. 49, 50

[RH01] RAMAMOORTHI R., HANRAHAN P.: An efficient representation for irradiance environment maps. In *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 2001), ACM, pp. 497–500. 50

[RH04] RAMAMOORTHI R., HANRAHAN P.: A signal-processing framework for reflection. *ACM Trans. Graph. 23*, 4 (Oct. 2004), 1004–1042. 52

[RWPD06] REINHARD E., WARD G., PATTANAIK S., DEBEVEC P.: *High Dynamic Range Imaging – Acquisition, Display and Image-Based Lighting*. Morgan Kaufmann, San Francisco, CA, 2006. 50

[SKS02] SLOAN P.-P., KAUTZ J., SNYDER J.: Precomputed radiance transfer for real-time rendering in dynamic, low-frequency lighting environments. In *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 2002), ACM, pp. 527–536. 50, 51

[Slo08] SLOAN. P.-P.: Stupid spherical harmonics (sh). *Microsoft Corporation.* (2008). 53

[SLS05] SLOAN P.-P., LUNA B., SNYDER J.: Local, deformable precomputed radiance transfer. In *SIGGRAPH '05: ACM SIGGRAPH 2005 Papers* (New York, NY, USA, 2005), ACM, pp. 1216–1224. 50, 51

[TKTS11] TOCCI M. D., KISER C., TOCCI N., SEN P.: A versatile hdr video production system. *ACM Trans. Graph. 30*, 4 (July 2011), 41:1–41:10. 50

[UG07] UNGER J., GUSTAVSON S.: High-dynamic-range video for photometric measurement of illumination. In *Proceedings of Sensors, Cameras and Systems for Scientific/Industrial Applications X, IS&T/SPIE 19th International Symposium on Electronic Imaging* (Feb 2007), vol. 6501. 50

[WRA05] WANG H., RASKAR R., AHUJA N.: High dynamic range video using split aperture camera. In *Proc. of OMNIVIS* (2005). 50

[YMIN10] YASUMA F., MITSUNAGA T., ISO D., NAYAR S. K.: Generalized Assorted Pixel Camera : Postcapture Control of Resolution , Dynamic Range , and Spectrum. *IEEE Transactions on Image Processing 19*, 9 (2010), 2241–2253. 50

SIGRAD 2012
A. Kerren and S. Seipel (Editors)

ESSAVis: A Framework to Visualize Safety Aspects in Embedded Systems

Ragaad AlTarawneh[1], Jens Bauer[1], Patric Keller [2], Achim Ebert[1], and Peter Liggesmeyer[2]

[1]Computer Graphics and HCI Group, University of Kaiserslautern, Germany, {tarawneh,j_bauer, ebert}@cs.uni-kl.de
[2]Software Engineering:Dependability Group, University of Kaiserslautern, Germany, {pkeller, liggesmeyer}@cs.uni-kl.de

Abstract

In this paper, we present a framework, called Embedded-System Safety Aspects Visualization (ESSAVis), that is a system prototype designed to analyze the safety aspects of embedded systems. The ESSAVis prototype provides a 3D environment that aids in detecting infected components in the hardware of the target embedded system. The prototype also provides an abstract representation for the failure mechanisms of the target embedded system, including the software structure and failure path information for the underlying safety scenarios at a certain moment in the system's life. The results indicate clearness of our proposed method over existing techniques and promise acceleration in performance of the failure detection process in embedded systems for critical applications.

Categories and Subject Descriptors (according to ACM CCS): I.3.3 [Computer Graphics]: Picture/Image Generation—B.7.2 [Design Aids]: Graphics—Simulation D.2.6 [Programming Environments]: Graphical environments—

1. Introduction

Embedded systems are widely used in our daily activities. Some examples of such systems are control systems in cars, airplanes, rail-road crossings and even washing machines. Normally, embedded systems consist of both hardware and software components. Therefore, most of them have complex structures and are not centralized in one component but are distributed among a set of components, which represent the system parts. These components communicate with each other via a set of hardware and software interfaces. These features help in developing relatively complex systems since many small-embedded systems, although physically separated, can be grouped together to construct larger systems. [LS10].

For many embedded systems, the safety and reliability aspects are very important. Consequently, as the complexity of such systems increases, the task of detecting or analyzing failures of the system becomes increasingly difficult [KGF07, KLM03]. The process of analyzing failures is necessary, in order to trace the reasons that lead to a specific hazard of the system's life. Subsequently, many techniques have been proposed to trace the failure propagation paths amongst the set of cooperating components in the failure. The Fault Tree Analysis (FTA) technique is one of the most common modeling techniques, which helps in understanding failure mechanisms in embedded systems. FTA emphasizes the logical relations amongst the set of basic failures which could lead to a specific undesired state of the system. That is called the Top-Event (TE) state [KGF07, KLM03].

As a descendant of this technique, the Component Fault Tree (CFT) concept provides an extension of the FTA concept by introducing additional information about the system's structure. CFT is used to depict the failure scenario in complex embedded systems as a directed acyclic graph [KLM03, KGF07]. Usually, results of the CFT model of complex embedded systems could be very large. Therefore, the process of analyzing and tracing the failure paths in such complex models becomes a tedious task. In this work, we aim to give an overview of the set of components involved in a given failure scenario. Moreover, we aim to provide a mean for visually comparing the components with respect to the system safety aspects. This work also provides an insight into the ways in which components might contribute to an undesired system failure. To achieve the above mentioned goals for our framework, we map extracted information from the CFT to visual elements in our final layout. As

described above, the underlying embedded system consists of hardware parts, originally modeled as in a CAD model, which can be intuitively visualized in a 3D world.

We accomplish this by providing a Virtual Reality (VR) environment that allows exploring certain safety scenarios in the 3D model of the hardware components. The 3D model is integrated with an *abstract representation* that describes the hierarchy of failure relations among cooperating components in the current failure scenario. We visualize the abstract representation using the radial layout algorithm [Ead92], because the underlying CFT model is inherently depicted as a tree in most of the cases. We use the radial layout algorithm to help in utilizing the screen space more efficiently and for showing the hierarchical relations of the failure among the infected components. Our work attempts to provide an effective means of identifying and classifying system artifacts like software and hardware components, with respect to their participation in safety critical system outages. Of particular interest in this work is the development of visualizations, which are able to support the analysts in answering questions like:

- What are the critical system artifacts and how do they contribute to a system failure?
- How severe is their influence in terms of probability of occurrence?
- Which artifacts influence each other and how strong are those influences?
- Do the causes occur within SW or HW components or in both?

Our goal is to support common analysis by providing representations integrating different views about safety critical embedded systems. The rest of the paper is organized as following: Section 2 provides a list of the related work. Section 3 provides the background information about the application domain. Section 4 highlights the proposed ESSAVis framework in details. While in Section 5, we present the results of the brief evaluation for our proposed framework. Finally, we conclude the work in Section 6.

2. Related Work

The visualization step is considered as one of the most important steps increasing the cognitive level of users. This is due to the facilities and the techniques which are offered to support the user in getting the correct insight about the current situation of the data only by watching or interacting with it. As mentioned earlier, this work's goal is the diagnosis of system status faster and more accurate by providing two linked-views of the CFT model and the 3D CAD model, where they are attached to each other, in a 3D environment. Therefore, we categorize the related work into two categories: Sub-section 2.1 presents the related work with regard to visualizing the FT and the CFT in general while Sub-section 2.2 presents the related work with regard to visualizing the data into the 3D world.

2.1. Visualization in Safety Domain

Depicting the failure relations among the system parts is critical to understand the failure mechanism. Many existing tools try to visualize this by showing a tree structure of the failure. Most of these tools visualize the fault tree in 2D representations like ESSAREL [Sof12], UWG3 in [KLM03], and Cecilia OCAS in [BBC*04]. In these tools the node-link diagram is used to show the relations between the infected system parts. While the simple primitive shapes are used to show the components of the Fault Tree (FT) e.g. small circles to represent basic events and a small rectangle to show the gate (the logical connector) between two basic events. These tools also use color or/and the text to depict other information such as the gate type. These kinds of tools are useful to model the failure relations between the system parts, but they do not provide options for analyzing the failure path or the set of the critical parts in the underlying system. In spite of the facility of editing and modifying the FT structure in these tools, generally they lack the abilities of analyzing the FT itself and the presentation of an overall view of the current failure mechanism. However, amongst them the ESSAREL tool provides a textual description of some safety aspects of the FT (like, the set of the minimal-cut-set), which is unreadable in most of the cases due to the data size and the file format.

There are some other tools that analyze the safety aspects of software systems. For example PLFaultCAT [DL06] has been used to reduce the effort needed to safely reuse the software requirements and to customize the product-line for software fault tree analysis (SFTA) during product-line engineering. [YKL*12] a visualization system has been proposed to support the engineers in identifying proper solutions for the system visually. The proposed visualization integrates the fault tree and a plot which represents the cause-effect relationship between the solutions of the system failures and the resulting risk reduction of the system. Moreover, the authors tried to associate the component fault tree view with the plot diagram to allow maintaining helpful context information about the current state of the system.

An interesting visualization system, called SViT (Safety Visualization Technique), was proposed by [KSZ09] showing the status of the digital home. SVit helps the homeowner to know the current safety level of the home and the reasons behind this level. It provides multiple interfaces for each device at home. It also automates the safety computation process of the safety information for each device. This helps homeowners to determine the required safety levels of their homes.

2.2. 3D Visualization Approaches

The 3rd dimension has been utilized in many information visualization techniques for many purposes. For example, the hyperbolic layout has been proposed by [Lam96,LRP95,

Mun97, Mun98, MB95] to show the information structure in the 3D world, while [CM00] presented the cone tree layout method as a pure 3D layout algorithm for hierarchical structures. Another example is semNet, which can be found in [FPF88]. These examples were based on using the animated 3D visualization and the lightening to show the depth perception. Collins et al. [CC07] provided many examples for showing the usability of the third dimension in encoding different aspects of the data. They achieved it by isolating a subset of a 2D graph representation to a separate layer, thus a 2.5D would be created. In this case, they proved that the third dimension could be used to encode some aspects of the data instead of the relations between nodes. The third dimension in [BDS03] has been used to show the evolution of the graph over time where each separated layer is used to represent the graph at a certain time step. Eades et al. [EF96] also used the third dimension to depict the hierarchical nesting of clusters in the graph using a set of transparent layers.

The current state of the art in graph visualization shows some interest in using the 3D layout in few cases. For a general introduction to graph visualization techniques, we refer the reader to the work of Herman et al. [HMM00]. In our work, we use the third dimension to represent the hardware components of the real model of the underlying embedded system, linked with the abstract representation of the safety scenario, which shows the important safety information of the current Top-Event. Initially, the abstract representation is laid out in a 2D plane that is integrated in a 3D world with the 3D model of the underlying embedded system.

3. Background and Motivation

The Fault Tree Analysis (FTA) technique is a top-down approach. It supports modeling the safety scenarios of complex systems [KGF07, KLM03] in which the relations between the set of faults (basic events) leading to a specific undesired event (Top-Event) are depicted graphically. The basic event is defined as one of the reasons causing the current Top-Event with a certain failure probability to occur [KLM03]. In the Fault Tree model, sets of basic events are represented as the tree leaves. The FTA depicts logical relations between sets of basic events that lead to a specific Top-Event, as shown in Figure 1. The Component Fault Tree (CFT) technique is a safety and reliability modeling technique that supports a hierarchical decomposition of large systems. The difference between CFT and FTA is that FTA offers a decomposition of the system into modules, which is a breakdown process of the system regarding the hierarchy of failure influences rather than the system architecture. Whereas, CFT represents each component by an extended fault tree that contains input and output ports besides basic events and gates. By connecting the set of components via set of ports, components can be integrated into a higher-level system model [KGF07, KLM03].

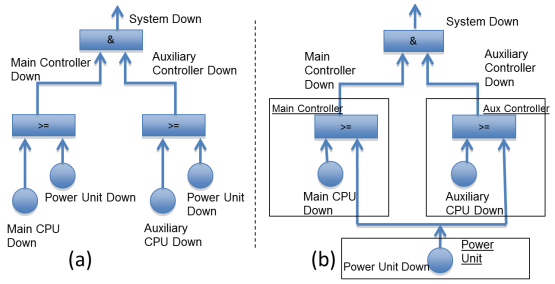


Figure 1: Fault Tree Concept vs. Component Fault Tree Concept.

The CFT modeling technique is useful for relatively large systems because all components can be developed independently and then can be saved in separate files, which provides a way to integrate the components and reuse them independently. The CFT extends the tree structure of the FTA into a Directed Acyclic Graph (DAG). This feature gives the ability of producing compact representation for large failure scenarios [KGF07, KLM03]. Although the safety scenario in our framework can be modeled as a DAG structure; currently, our prototype provides only the basic functionalities of the system. Therefore, at the moment we model only the tree structures as they are often used to model safety scenarios in such applications. However, for future work we would like to visualize more realistic models that can be visualized as a DAG structure to reflect the CFT concept.

Visualizing the safety aspects of embedded systems is comparatively a new field. As the complexity of the underlying embedded system is increased the size of corresponding fault tree is also increased, which makes it difficult to handle the failure detection process. Moreover, it makes maintenance of the underlying system more costly and time consuming. The information visualization can play an important role in speeding up the system developing process because it eases the step of finding the important information from both the system and the user's perspectives. Also, it helps in reducing the errors made by human in searching the relevant information. For example; a study, conducted by IBM and Industry Studies [BV10], shows that 30% of people time is wasted on looking for the important information. Moreover, the visualization helps in interacting with the data and automates the steps of information extracting, which helps in detecting the relevant information more accurately. Furthermore, the visualization can provide the integration between different parts of the complex system. Like in our case, it helps in integrating the hardware part and the abstract safety view, which is useful in conveying the whole story to the end users.

Due to the interaction between safety analysts and systems engineers, the task of analyzing the system is an iterative task. We summarize this process in the following steps: first, system engineers and safety engineers built the FT for

a specific Top-Event in a way that reflects the system design; then the relevant information is analyzed from the safety engineers' perspective. Whereas in our case, we are interested in showing the most critical components of the system from both the hardware and the software perspectives; then system engineers react to fix the infected components based on the analyzed output from the previous step; and finally the corresponding FT is updated accordingly as it is shown in Figure 2.

4. The Embedded System Safety Visualization (ESSAVis) Framework

The proposed ESSAVis framework provides a two-views layout where the first view shows a 3D model of the hardware part while the second view shows an abstract representation of the underlying safety scenario as shown in Figure 3. To achieve this, we analyze the safety scenario that represents a critical status of the underlying embedded system. In this work, we analyze the critical situations of a robot, called RAVON [Pro10]. The safety scenario is modeled using the Component Fault Tree (CFT) technique. We analyze this safety scenario to produce the required data structure we need in computing the set of critical components for the scenario. This set of critical components consists of all those components that have a high probability to fail. For each component, we compute the criticality based on many factors such as the number of basic events and their probabilities or the number of failure paths that pass through each component. We visualize the abstract information using the Radial Layout algorithm [Ead92]. The main task of the abstract representation is to reveal the failure relations between the collaborating components in a specific hazard of the system at a certain time. In this abstract representation, the set of nodes represents components set while the set of edges reflects logical relationships among components of the CFT model (as shown in the left-side of the Figure 3).

ESSAVis also provides a 3D model of the hardware part of the underlying embedded system. For example, the 3D model of RAVON is dedicated to convey right impression of the real physical model of hardware components. To do this, ESSAVis parses VRML files of physical components of RAVON and assembles them together to produce the full model of RAVON. Then the two views are rendered in a 3D world. The ESSAVis framework synchronizes between the two-views to let users trace the set of changes in both views faster. For example, we highlight infected components in both views in red color according to the underlying safety scenario (as shown in both views of Figure 3).

4.1. The ESSAVis Pipeline

To achieve the desired goals, we developed a tool that reads a safety scenario of the underlying system at a certain time. Then we extract the relevant safety information to help

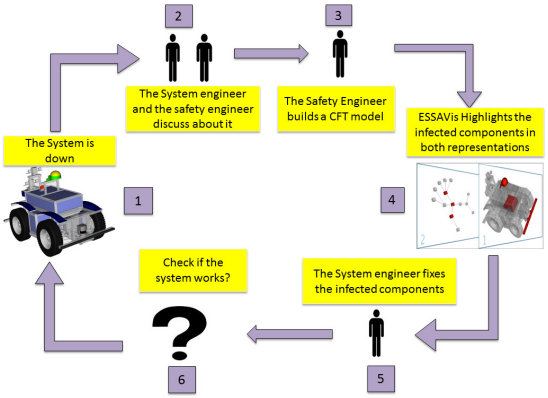


Figure 2: The embedded systems safety analysis cycle using ESSAVis.

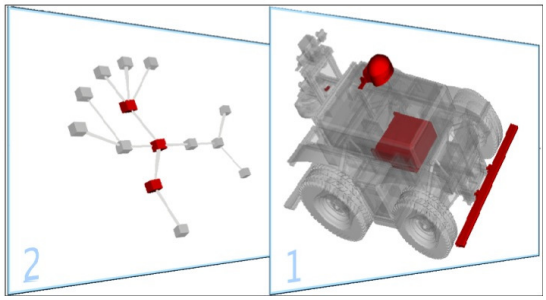


Figure 3: Integrated view showing infected hardware components (right) and the corresponding failure path in the abstract layout (left).

safety engineers in defining the set of critical components. As mentioned earlier, the criticality of components is based on many factors, such as the number of failure paths passing through each component or the accumulated probabilities of the contained basic events [KGF07]. We generate the required safety scenario using a specialized safety-modeling tool, called Essarel [Sof12]. The Essarel tool supports quantitative analysis of safety, availability, and reliability of embedded systems. Also, it provides the feature of exporting the model into an xml file, which contains a description of the safety scenario, or into the component fault tree model and corresponding required safety attributes. For example, the set of minimal-cut-set in the current scenario is included together with the information about the related set of components and failure connections among them. This xml file is the input of our framework, where we extract the safety data related to each component and then arrange the set of components to be ready for the visualization process.

For visualizing data, we use Vrui package [Kre12], which is a toolkit to provide a virtual reality development environment. Vrui package shields the application developing process from the particular configuration of the VR environ-

ment. Moreover, it works in many different hardware platforms. Therefore, it is possible to render the application in many different environments such as 3D displays, PowerWall displays, or CAVE systems. This feature increases the scalability and portability of our framework into different environments, which also helps in showing the depth cue in the 3D world more precisely. We render the 3D model of the RAVON robot using the Vrui package as it supports the construction of 3D world using a scene graph data structure. Then we link the safety information, generated using the Essarel tool, with the 3D model of RAVON. This allows us to highlight directly infected components in the hardware model, which helps system engineers to identify faults quicker than the traditional way of the textual description. Figure 4 shows how the different toolkits in our framework are interacting together.

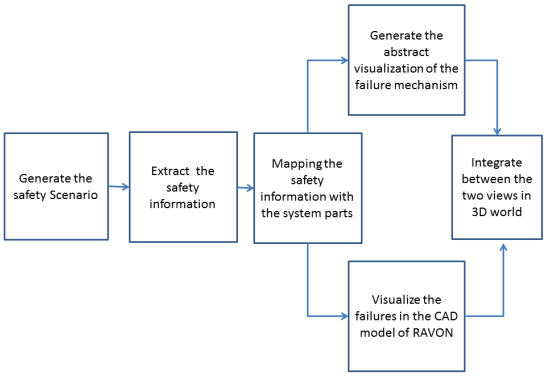


Figure 4: The ESSAVis pipeline.

4.2. The Two-Linked View

In this subsection, we describe the proposed integration model between the two data representations. As mentioned earlier, ESSAVis synchronizes between the abstract layout and the hardware model in a 3D world. Initially, both representations are visible side by side as shown in Figure 5. ESSAVis gives the possibility to toggle between different views on-demand. It provides three different arranged views to link between the two layouts. The initial one is the *side-by-side view*, in which both representations are displayed next to each other with equaled sizes for both and at the same distance from the viewer. Users can change this by selecting the toggle view option from the ESSAVis main menu. The available options are the *big-small view* option and the *layered view* option.

The big-small option provides the facility to users scaling up of one of the representations while scaling down the other one, as it is shown in Figure 7. This option is useful in few cases, e.g. safety engineers, who are interested in knowing more about some failures in the system, can decide to view the abstract view larger than the 3D model (as shown in Figure 7) or system engineers can go for the opposite case. While in the third view, the depth is used to show importance degree of the representation. Initially, the two-views are arranged in the same depth value that is zero in our case (as shown in Figure 5). However, this arrangement can be modified by changing the view through the layered-view option. For example, if user is interested in the hardware model, the view of hardware model pops up to a layer closer to the viewer point and the second view will be stacked behind the current active view in the first layer as it is shown in Figure 3. In this example case, the second layer represents the abstract representation while the first layer represents the 3D model of RAVON robot respectively. The user can toggle between the different views on demand. If he/she is interested in the hardware model then this view will be the dominant in the screen. The user then can change the visual cues of this view accordingly, e.g. changing the transparency, size, or orientation. The toggle view option is provided via one of the main menu options. Currently, users can interact directly with the prototype using a pop-up menu that includes the basic functionalities of the framework. This menu has a set of sub-menus that are used to list the extracted information from the current safety scenario. These submenus allow users to map the needed information with the actual 3D hardware components and components in the abstract visualization (as shown in Figure 6).

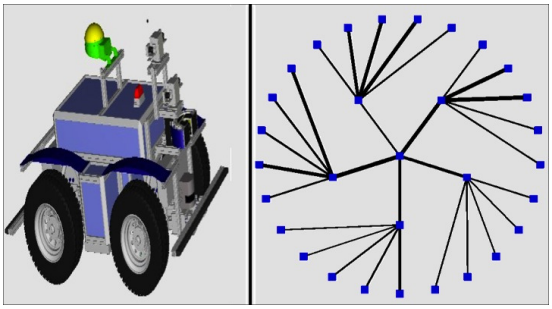


Figure 5: The initial integrated layout of hardware model and the abstract representation.

The menu in the prototype is triggered by clicking the right mouse button at any free space in the 3D world. ESSAVis also supports the basic interaction techniques in the 3D world, like zooming, panning or rotation. One of the options that are provided by the Vrui itself is scale bar option, which helps user to indicate the size of the object in the 3D scene and quickly adjusts the display's zooming factor to a standard ratio, like 1:10 or 50:1. The zooming ratio is between the physical space length and the navigational space length of the object [Kre12]. As mentioned earlier, the abstract visualization is used to encode relations between the infected components in the corresponding safety scenario. For example, in Figure 8, we show an abstract representa-

tion of a component fault tree consisting of 30 components. The top-event node represents the main hazard in the system that is depicted as the root of the tree, which is the central node in the radial tree layout. The leaf nodes are those components that have the set of basic events, which contribute in a specific Top-Event.

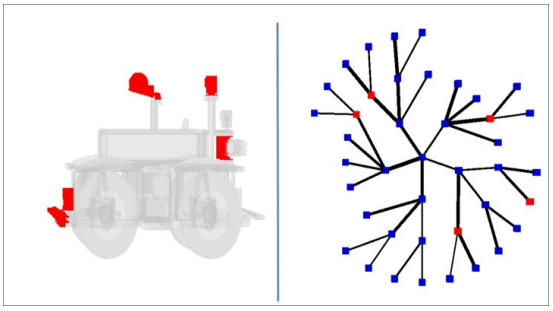


Figure 6: An integrated *side-by-side view* showing the correlation between the infected components in the hardware model and the abstract representation.

ESSAVis provides a list of options for analyzing the safety aspect of the corresponding safety scenario. This helps in answering the set of safety engineers' questions, already mentioned in Section 1. For example, users can highlight infected components in both representations by selecting the component name from the submenu. The current prototype also allows users to get information about each involved component in the scenario such as basic events for each component. Figure 8a shows mapping between the selected basic events and the related components. This function is crucial in helping safety engineers in finding the criticality of the component according to the probability of basic events that it has. It is worth mentioning that each basic event has a probability value, which indicates the occurrence probability of such an event. The current prototype provides information about the basic event probability via a label containing the value that is attached to the basic event name in the list. We intend to use other visual cues to encode the probability value to help users getting more insight about such an important metric from one glance.

One of the important aspects about our framework is showing of the Minimal-Cut-Set (MCS) information. The MCS indicates the least number of Basic Events that are required to cause a top-event in the system. Showing the MCS is a strong indicator of the infected components influence in the current scenario. ESSAVis provides a mechanism to show the relations between lists of MCSs in the current scenario with correspondence components in the 3D hardware model. This facility aids safety engineers in diagnosing the critical components in the current scenario. We give users the ability to interact directly with the list of MCSs and to select one of the entries of the underlying scenario. Due to user selection, the set of infected components in the current

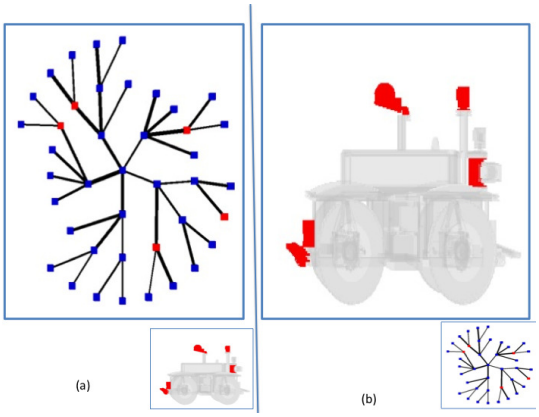


Figure 7: A big-small view to integrate the different views.

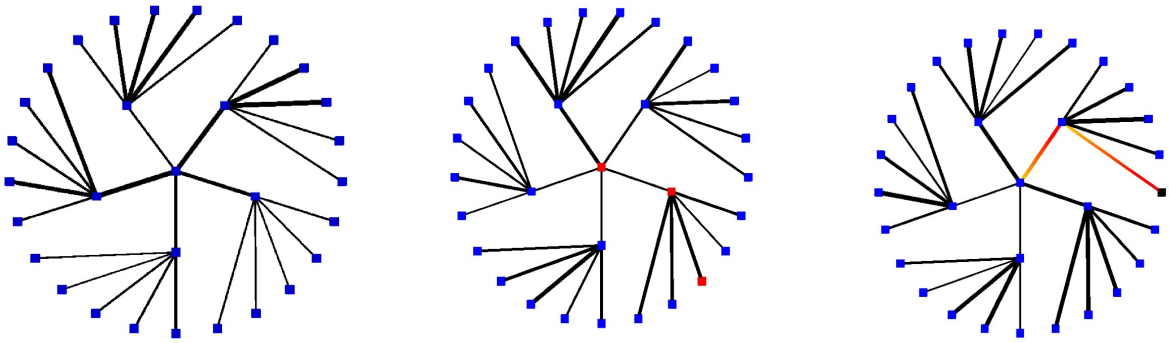
scenario are highlighted in red color. This is applied on both views, the hardware view and the abstract view, as shown in Figure 8b. This helps system engineers to diagnose the system status quickly; hence, it speeds up the maintenance process of the underlying system.

Fault trees for large systems are considerably complicated and difficult to read. Therefore, it is quite harder to achieve in them the process of tracing the path failure between any component and the root component. Our prototype provides an option to show paths between the infected components that collaborate in a Top-Event in the system based on the current configuration. This feature provides the safety engineers the ability of finding out the set of influenced components through the selected ones. This option facilitates them an insight of those components that might fail next according to the current selected components. ESSAVis also provides the facility to animate the failure path among the infected components by selecting one component from the abstract layout to the top-event node. As a result, the set of edges, which contributes in the selected failure, are highlighted in orange color. We use a simple animation for showing the propagation of the failure path. This case is illustrated in Figure 8c. Beside the failure propagation path, edges between the set of components are also used to show the criticality of the path between two components. Different thickness are used to depict the path criticality, as shown in Figure 8a.

5. Framework Evaluation

A brief evaluation was conducted for the proof of the validity of our proposed ESSAVis solution.

- *Environment settings:* we carried out the evaluation experiment using a Zalman stereoscopic desktop display equipped with polarized 3D glasses. The participants in the evaluation were from two different user groups. The first group consisted of 6 safety experts while the second group consisted of 3 robotics experts. A safety scenario,



(a) (b) (c)

Figure 8: (a) Abstract view for a safety-scenario in a certain moment. (b) Red colored components represent the set of infected components by a specific MCS. (c) The failure path depiction between the selected component and the Top-Event and the edge thickness indicates the path criticality.

consisted of 40 components aligned side-by-side to the 3D model of RAVON, was presented to both groups.

- *Hypothesis:* we assumed that the ESSAVis solution increases the detection process speed of infected components of the current scenario. The detection speed is compared with the detection speed of the traditional tools like Essarel tool.
- *Tasks:* the participants were asked to answer the following questions:
  - Detect the infected components in both representations; the abstract representation and the 3D model representation of RAVON using our prototype.
  - What are the basic events that are related to a specific component.
  - What is the failure path between a specific component and the top-event node.

The preliminary results of the evaluation show the usability of our framework especially from the system engineers' side that work with the hardware model of RAVON robot. They agreed that our given framework could help them in defining and detecting the infected components in the current state faster than the textual description of the failure. We discuss the accuracy in terms of users' groups to accomplish the task successfully. The accuracy of system engineers' group answers was 94% with average time of 24 seconds. Also, it was easy for them to infer those components that have the possibility of failure based on the current configuration. While the safety group mentioned that our given prototype could help them in detecting the relations between the infected components beside in tracing the failure propagation paths among the infected components. 5 out of 6 safety group participants were able successfully to find out the set of most critical components approximately within 18 seconds with accuracy reached to 85%.

The results also indicate the need of training for both groups to read the new representation and the 3D environment that is used for the rendering process. Beside this, we found that some visualization aspects need to be optimized more, especially the path tracing option as shown in Figure 8c. In this regard, participants were not able to map the changes in the abstract view with respect to the changes in the 3D model. So, there is a need to provide a solution for tracing the changes between the different views of our framework. Moreover, one limitation of the current prototype is performance of the rendering speed of the 3D model of RAVON. It is comparatively large model and consists of 395 vrml files. Therefore, there is need to apply some filtering techniques to such a large model for speeding up the rendering in order to increase the response time of ESSAVis.

6. Conclusions

We presented a framework, called ESSAVis, to visualize the failure mechanism of embedded systems. ESSAVis shows the infected components in parts of any embedded system, the hardware parts or the software parts. Based on a specific safety scenario at a certain time step, it visualizes the hardware components as 3D objects and highlights the infected components in red color. For showing the failure mechanism among the different types of components in the underlying component fault tree, it provides an abstract visualization to depict the failure relations among the system parts. To prove the validity of the framework, a brief evaluation took place using a stereoscopic desktop display to show the real 3D impression of our work. Results show the intuitiveness and clearness of our technique over the traditional techniques that currently provide a textual description of the system's safety aspects.

As a future work, we intend to conduct a more extended version of the evaluation by comparing performance of our prototype with other available systems, which are currently used to analyze the safety aspects of embedded systems, with different scenarios and different data sizes. As mentioned earlier, the real data is more complex than the normal tree structures and has many different possible scenarios that are included in our current version of the prototype. Therefore, we intend to include those cases, such as DAG case, in our visualization. Moreover, as we mentioned the designing of an immersive environment for visualizing safety aspects of embedded systems; so, we also intend to evaluate the accuracy of the proposed framework in different environments, like using CAVE system, Power-Wall, and stereoscopic desktop display.

Acknowledgements: This work is part of ViERforES2 project and partially funded by IRTG 1131 (DFG) and BMBF. Many thanks go to the Software Engineering and Dependability Group at University of Kaiserslautern headed by Prof. Dr.-Ing. habil. Peter Liggesmeyer for their support. Also, we would like to thank the Robotics Research Lab at University of Kaiserslautern for their collaboration and support.

References

[BBC*04] BIEBER P., BOUGNOL C., CASTEL C., P. HECKMANN J., KEHREN C., SEGUIN C.: Safety assessment with altarica - lessons learnt based on two aircraft system studies. In *18th IFIP World Computer Congress, Topical Day on New Methods for Avionics Certification* (2004), p. 26. 60

[BDS03] BRANDES U., DWYER T., SCHREIBER F.: Visualizing related metabolic pathways in two and a half dimensions. In *Graph Drawing* (2003), pp. 111–122. 61

[BV10] BOZZANO M., VILLAFIORITA A.: *Design and Safety Assessment of Critical Systems*. CRC Press (Taylor and Francis), an Auerbach Book, 2010. 61

[CC07] COLLINS C., CARPENDALE S.: Carpendale s: Vislink: revealing relationships amongst visualizations. *IEEE Trans Vis Comput Graph 2007* (2007). 61

[CM00] COCKBURN A., MCKENZIE B.: An evaluation of cone trees. In *University of Sunderland* (2000), Springer-Verlag, pp. 425–436. 61

[DL06] DEHLINGER J., LUTZ R. R.: Plfaultcat: A product-line software fault tree analysis tool. *Automated Software Engineering 13*, 1 (2006), 169–193. 60

[Ead92] EADES P.: Drawing free trees. *Bulletin of the Institute for Combinatorics and its Applications, pp. 10 -36* (1992). 60, 62

[EF96] EADES P., FENG Q.-W.: Multilevel Visualization of Clustered Graphs. In *Proc. Graph Drawing, GD* (Berlin, Germany, 1996), no. 1190, Springer-Verlag, pp. 101–112. 61

[FPF88] FAIRCHILD K. M., POUTROCK S. E., FURNAS G. W.: SemNet: Three-dimensional graphic representation of large knowledge bases. In *Cognitive Science and its Applications for Human-Computer Interaction, R. Guinon, Editor. 1988, Lawrence Erlbaum: Hillsdale NJ* (1988), pp. 201–233. 61

[HMM00] HERMAN I., MELANCON G., MARSHALL M. S.: Graph visualization and navigation in information visualization: A survey. *IEEE Trans on Visualization and Computer Graphics 6*, 1 (2000), 24–43. 61

[KGF07] KAISER B., GRAMLICH C., FORSTER M.: State/event fault trees: A safety analysis model for software-controlled systems. *Reliability Engineering System Safety 92*, 11 (2007), 1521–1537. 59, 61, 62

[KLM03] KAISER B., LIGGESMEYER P., MÄCKEL O.: A new component concept for fault trees. *Reproduction 33* (2003), 37–46. 59, 60, 61

[Kre12] KREYLOS O.: Vrui virtual reality toolkit, July 2012. "`http://idav.ucdavis.edu/~okreylos/ResDev/Vrui/index.html`". 62, 63

[KSZ09] KUMAR P., SUBRAMANIAN N., ZHANG K.: Savit: Technique for visualization of digital home safety. *ACIS International Conference on Computer and Information Science* (2009), 1120–1125. 60

[Lam96] LAMPING J.: The hyperbolic browser: A focus+context technique for visualizing large hierarchies. *Journal of Visual Languages Computing 7*, 1 (1996), 33–55. 60

[LRP95] LAMPING J., RAO R., PIROLLI P.: A focus+context technique based on hyperbolic geometry for visualizing large hierarchies. In *Proc. of CHI '95* (1995), ACM, pp. 401–408. 60

[LS10] LEE E. A., SESHIA S. A.: *Introduction to Embedded Systems - A Cyber-Physical Systems Approach*, 1 ed. Lee and Seshia, 2010. 59

[MB95] MUNZNER T., BURCHARD P.: Visualizing the structure of the world wide web in 3d hyperbolic space. *Proc. of the first symposium on Virtual reality modeling language VRML 95* (1995), 33–38. 60

[Mun97] MUNZNER T.: H3: laying out large directed graphs in 3d hyperbolic space. In *Proc. of InfoVis '97* (Washington, DC, USA, 1997), IEEE Computer Society, pp. 2–. 60

[Mun98] MUNZNER T.: Drawing large graphs with h3viewer and site manager (system demonstration). In *Proc. of Graph Drawing'98, number 1547 in Lecture Notes in Computer Science* (1998), Springer-Verlag, pp. 384–393. 60

[Pro10] PROETZSCH M.: *Development Process for Complex Behavior-Based Robot Control Systems*. RRLab Dissertations. Verlag Dr. Hut, 2010. ISBN: 978-3-86853-626-3. 62

[Sof12] SOFTWARE ENGINEERING RESEARCH GROUP: DEPENDABILITY KAISERSLAUTERN UNIVERSITY, ESSAREL TOOL: Embedded systems sadety and reliability analyer, July 2012. "`http://essarel.de/index.php?site=backgroundtext`". 60, 62

[YKL*12] YANG Y., KELLER P., LIVNAT Y., LIGGESMEYER P., HAGEN H.: Improving safety-critical systems by visual analysis. *Dagstuhl Follow-Up series* (2012). 60

SIGRAD 2012
A. Kerren and S. Seipel (Editors)

Visual Parameter Optimization for Biomedical Image Analysis: A Case Study

A.J. Pretorius[1], D. Magee[1], D. Treanor[2,3], and R.A. Ruddle[1]

[1]School of Computing, University of Leeds
[2]Leeds Teaching Hospitals Trust
[3]Leeds Institute of Molecular Medicine, University of Leeds

Abstract

The conventional approach for parameter optimization of biomedical image analysis algorithms is to tweak parameters by trial-and-error. This presents a challenge: parameter space is often inadequately explored and, consequently, output quality suffers. Interactive visualization can alleviate this problem but has not been widely adopted. Moreover, few examples of the successful application of visualization for parameter optimization of image analysis algorithms have been published. To address this and to illustrate the potential usefulness of interactive visualization, we present a case study. A multidisciplinary team developing novel image segmentation software for histopathology was observed. Within the context of our study, our hypotheses were confirmed: (1) using interactive visualisation, participants considered larger parts of parameter space than they had previously by trial-and-error; (2) participants gained a better understanding of their algorithm (an unknown logic error and errors in its implementation were discovered); and (3) participants achieved higher quality output. Our work is also an example of the value of case studies in iterative design. We describe how a valuable additional requirement was revealed (the importance of derived measures) and how our visualization method was extended to cater for this.

Categories and Subject Descriptors (according to ACM CCS): H.5.2 [Information Interfaces and Presentation]: User Interfaces—Visualization; I.3.8 [Computer Graphics]: Applications—Biomedicine

1. Introduction

Parameter optimization is often encountered in applied science and engineering and presents a non-trivial challenge: given a system with several input parameters that produces one or more outputs, how do users find suitable parameter values such that certain requirements hold for the outputs? In particular, the application of segmentation algorithms to biomedical images requires users to find parameter values to accurately detect objects, such as cells, or higher-level structures, such as regions of tissue. The conventional approach, which we refer to as stepwise iterative refinement, is to find suitable values through a process of manual parameter tweaking by trial-and-error. Users supply parameter values, initialize and wait for algorithms to execute. The output is inspected, parameter values are changed, and the process repeated until satisfactory output is produced.

A second category of approach is to find optimal parameter values by using interactive visualization. Although some promising visualization methods for parameter optimization of image analysis algorithms have been proposed, they have not been widely adopted. This can be partly attributed to few examples of the application of such methods in real-world image analysis scenarios.

To address this, the primary contribution of this paper is a case study of a multidisciplinary team using interactive visualization to develop novel histopathology image segmentation software. We hypothesized that using visualization, as opposed to stepwise iterative refinement, our participants would: (1) analyze larger parts of parameter space; (2) gain a better understanding of the underlying algorithms; and (3) achieve higher quality results. Our study provides anecdotal evidence in support of our hypotheses and illustrates the potential usefulness of interactive visualization for parameter optimization purposes (albeit in a limited context). It also serves as an example of users' reactions to the introduction of interactive visual analysis into the established

field of biomedical image analysis. As a secondary contribution, our work is an example of the role of case studies in iterative design. We discuss how an additional requirement was discovered and addressed. We also share preliminary results of how this changed user behavior.

2. Related Work

Many techniques exist for automating parameter optimization. Numerical optimization methods apply mathematical or statistical techniques to minimize (or maximize) an objective function defined over parameter space [Onw00]. Well-known examples include linear programming and gradient descent. Another trend is to apply artificial intelligence techniques (such as genetic algorithms) to optimization [AH97].

Automated techniques usually seek to minimize a single objective function. However, users often have multiple conflicting requirements. Moreover, end-users of image segmentation algorithms find it a daunting task to rigorously and mathematically formalize optimization criteria. This leads to human-in-the-loop approaches, which we describe below.

2.1. Stepwise Iterative Refinement

The most frequently encountered approach is to require users to search for optimal parameter values without additional support. Stepwise iterative refinement involves manual parameter tweaking by trial-and-error. It is a drawn-out procedure where users change parameter values and invoke a system or algorithm to produce the corresponding output. Output is judged qualitatively, input settings are changed, and the process repeated until satisfactory output is produced. For example, CellProfiler is a popular biomedical image analysis tool [CJL*06]. The incorporated algorithms are parameterized and users have to try different parameter values until they are satisfied with the quality of the output.

Stepwise iterative refinement is recognized as a bottleneck in parameter optimization [PBCR11, TWSM*11]: many iterations of the above process are typically required and users rely on memory recall to compare current output with previous results. Consequently, parameter space is inadequately explored, which negatively impacts the quality of output.

2.2. Interactive Visual Analysis

Several visualization methods have been developed to address the deficiencies of stepwise iterative refinement.

Guided navigation. Piringer et al. developed a technique for guided parameter optimization in a system called HyperMoVal [PBK10]. It shows multiple linked visualizations of the local neighborhood of a high-dimensional focal point in parameter space. This supports an iterative process where the user is guided to optimal parameter values by showing residual errors of estimated output of sampled parameters compared to known pre-computed output. In subsequent work, Berger et al. apply a similar approach for exploring parameter space [BPFG11]. The user is provided with visual guidance to potentially optimal regions in parameter space. Again, approximations of output are shown in visualizations of the local neighborhood of a user-selected focal point. Uncertainties associated with predicted output are also shown.

Torsney-Weir et al. present an approach to systematically explore the entire parameter space of a segmentation algorithm [TWSM*11]. Their system, Tuner, combines guided exploration with several linked visualizations of parameter space and output space. First, parameter space is sampled sparsely and segmentation outputs are computed for these points. Outputs are then evaluated with respect to a ground truth image (an optimal segmentation marked up by an expert) to compute quality measures. Next, a statistical model is used to estimate the quality of the entire parameter space. Regions likely to yield high quality outputs are highlighted. The user is guided to resample and investigate high-potential regions of parameter space until parameter values that yield suitable outputs are identified. In related work, Bergner et al. introduce an approach for interactive exploration of parameter space for multivariate simulation models [BSN*11]. The user is visually guided to discover regions of parameter space that are qualitatively different. There is also assistance for resampling regions of interest more finely.

The above methods are powerful but they require users to understand complex mathematical and statistical notions to interpret visualizations. Although there are niche users for which this is applicable (automotive engineers, for instance [TWSM*11]), end-users such as biomedical researchers are likely to find this challenging. The methods also rely on the existence of objective functions, quality measures, or ground truths, which are not always available.

Interactive exploration. A number of techniques enable users to explore parameter space and evaluate output qualitatively by visual inspection. Parameter space is typically high-dimensional, so standard multidimensional visualization techniques can be used [WB97]. Examples include parallel coordinates and scatterplot matrices [Ins85, Har75]. A number of techniques have been developed specifically for parameter visualization, however.

Several approaches, including those above, employ multiple coordinated visualizations. In the Influence Explorer, Tweedie et al. show histograms of input parameters and of outputs [TSDS95]. Users specify an active range for each histogram and data items for which one or more dimensions fall outside these ranges are dimmed. In later work, Tweedie and Spence extend the approach to prosection matrices [TS98]: 2D scatterplots, each with an associated third dimension for which users can specify an active range. Data items with one or more dimensions outside these ranges are hidden. With both methods, users discover relationships between parameters and outputs interactively. Dynamic query-

ing enables them to explore different scenarios and identify parameter combinations that meet or fail requirements.

A few methods focus on changes in parameter values, and corresponding outputs, during the optimization process. Ma shows the parameter search process as a directed graph to help users find rendering parameter settings for computer graphics scenes [Ma99]. When parameter values are changed a new scene is rendered and a thumbnail representation is connected to the previous rendering with an edge. Callahan et al. use a similar approach for history management in VisTrails [CFS*06]: a visual history tree is maintained as users generate different visualizations of a dataset. Both methods use the visualization of the evolution of outputs as an external memory aid. Search paths are shown and existing outputs can be selected and refined further.

Some techniques emphasize the structure of parameter or output space. Jankun-Kelly and Ma use a spreadsheet metaphor to let users investigate the influence of rendering parameters on computer graphics scenes [JKM00]. Pairs of parameters define the x- and y-axes of a table and renderings that correspond to a pair of settings are shown at the intersection of the two coordinates. In Design Galleries, Marks et al. structure their visualizations on perceptually recognizable differences of outputs [MAB*97]. Parameter values are chosen so that outcomes are perceptually distributed. Several outputs illustrating these differences are shown to enable users to identify suitable regions of parameter space. Bruckner and Möller sample parameter space and apply a novel clustering method to identify key changes in the output space [BM10]. Both the clusters and output are visualized.

The visualization system considered in this paper supports interactive exploration of parameter space and was designed to analyze relationships between input parameters and outputs of biomedical image analysis algorithms. The approach does not rely on objective functions or ground truths as they cannot be assumed to exist in our target domain. An overview of this system is deferred to Section 3.2.

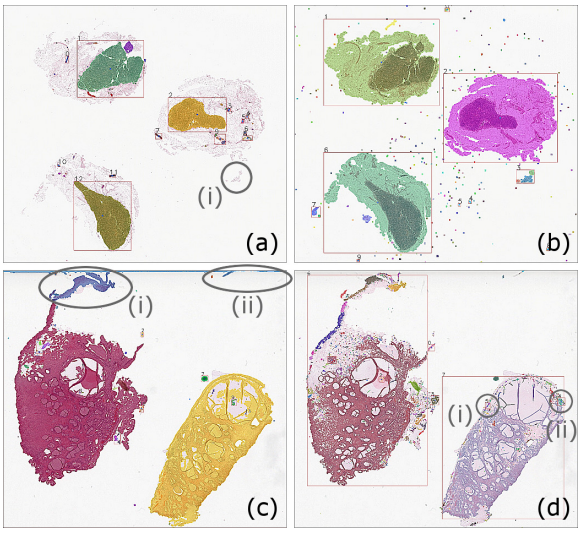


Figure 1: *Segmentation of histopathology images. Detected contiguous regions are filled with a random colour and outlined with a bounding box. (a) Poor quality segmentation of a lymph node section resulting from default parameter values identified through stepwise iterative refinement: multiple smaller parts of the three main contiguous regions are detected. (b) High quality segmentation of a lymph node section resulting from more optimal parameter values identified by interactive visual analysis with Paramorama. (c) Poor quality segmentation of a cervix section resulting from default parameter values: in addition to the coloured regions, the entire image is also detected as a contiguous region. (d) High quality segmentation of a cervix section resulting from more optimal parameter values identified by visual analysis. Histopathology images often contain noise resulting from sectioning ((a)(i) and (c)(i)), and artifacts on the glass slides ((c)(ii)). A subtle error in the original implementation of the algorithm, where smaller regions, such as (d)(i) and (ii) were not combined with larger regions that contain them, was also identified by visual analysis.*

3. Parameter Optimization for Image Analysis

To compare stepwise iterative refinement with interactive visual analysis, we conducted a case study. Before it is discussed, we first describe the software systems that we used.

3.1. Proprietary Segmentation Software

Histopathology images are very high resolution scans of microscopic-scale tissue slices (typically 80,000 × 100,000 pixels). Consequently, a user may be viewing morphological structures at a very high zoom level. This makes it difficult to navigate the images efficiently as context is lost. The software we considered is being developed as part of a set of methods to assist in (semi-) automatic navigation between distinct regions of tissue in histopathology images. It forms part of an ongoing initiative, independent of the present study, to develop virtual microscope software [TJOH*09].

The goal is for the software to accurately detect contiguous regions of tissue in an image. This is achieved by several image processing steps. For instance, a thresholding step distinguishes foreground and background based on pixel-level intensity differences. Each processing step requires the specification of values that indicate, for instance, the amount of thresholding to apply. As we describe in Section 4, a set of default values had been identified for these parameters by stepwise iterative refinement and were subsequently hard-coded in the software. For a particular input image, the output is an image where each detected region is filled with a color and marked by a bounding box (see Figure 1(a)).

3.2. Paramorama

We also considered a visualization prototype designed for parameter optimization of image analysis algorithms in a biomedical context. This technique differs dramatically from the software described in Section 3.1 in the way that parameter optimization is approached by providing interactive visual support. It is an example of using visual analysis for parameter optimization, as discussed in Section 2.2.

In previous work we presented a design study for interactive visual optimization of parameters for biomedical image analysis algorithms [PBCR11]. Our initial design was informed by interviews and discussions with domain experts at the Broad Institute of MIT and Harvard and observations of users at a hands-on introductory workshop on image analysis. Our technique was developed to visually analyze a Cartesian sampling of parameter space. This has the benefit of being conceptually simple: for each parameter a user wishes to sample, they simply specify a range and the number of samples to compute. To facilitate this, we originally implemented a sampling plugin for CellProfiler [CJL*06]. Our approach is not tied to CellProfiler or any particular algorithm, however. As we will demonstrate, users of other tools can extend these to sample any parameters of interest and analyze their results with our prototype. Requirements analysis revealed that users are usually interested in sampling 3–7 parameters at 3–7 intervals each [PBCR11].

As shown in Figure 2(a), we visualize sampled parameter space as a tree, using a node-link depiction oriented left-to-right. Every level represents a parameter and users can interactively change the order to suit their analysis needs. In the figure, the first parameter was sampled four times, shown by four second-level nodes (the root contains all samples). For each of these values, the second parameter was sampled four times, and so on. Users can identify, navigate to, and select particular combinations of parameter values. When they select regions of parameter space (subtrees in the node-link view) thumbnails of the actual image-based output are displayed in the detail view (Figure 2(b)). We developed a layout with the following properties [PBCR11]: each selection is shown in a distinct region; the top-to-bottom ordering of regions preserves the ordering of selections in the overview; the hierarchy of each selected subtree is shown by nesting; and within each region, the numerical order of sampled parameter values is shown left-to-right. The layout is parameterized and users can specify a parameter for which values are positioned top-to-bottom to make comparisons easier.

Users are often interested in contextual information in the form of a reference image to interpret the results of segmentation algorithms. Typically, they want to compare outputs to the original input image to determine whether correct objects, and not noise, were identified. We supply a reference image view as shown in Figure 2(c). When users move the cursor over any thumbnail, it is superimposed on the reference image. Our prototype provides standard interaction capabilities such as linked views and filters. A popular feature is to tag regions of parameter space that are associated with high- or low quality outputs. When a user tags regions, they are marked with a green (good) or magenta (bad) background in the overview and detail view (Figure 2(a) and (b)).

4. Histopathology Case Study

The software described in Section 3.1 requires parameters to be set to ensure accurate region detection. We hypothesized that the visualization method implemented in Paramorama (see Section 3.2) would enable users to: (1) consider larger parts of parameter space; (2) better understand their algorithms and the influence of parameter values; and (3) achieve higher quality segmentation. We observed three participants as they used Paramorama to analyze the results of the segmentation software using sampled parameter values.

4.1. Method

Participants. Adam is an undergraduate research assistant who had implemented the histopathology segmentation algorithm described. Bob is an academic member of staff and an active member of the image processing research community. He has developed several biomedical segmentation algorithms in the past and served as Adam's supervisor. Charles is a practicing histopathologist with deep domain knowledge in the analysis of histopathology images. He has been involved in several research projects where image analysis methods were applied in a histopathology context. Charles served as a domain expert during the development of the algorithm. None of the participants had prior experience with our visualization prototype. All were initially skeptical about the suitability of interactive visualization methods for the analysis and optimization of segmentation results.

Materials. Three tissue types were considered: biopsies from the oesophagus of patients with Barretts' esophagus, resections of the cervix, and excised lymph nodes. For each type, two images were selected. The first was a "textbook" example with one to three clean contiguous regions of tissue. The second image, which contained more noise, was emblematic of what often happens in practice. Histopathology images are high-resolution scans of thin sections of tissue fixed to glass slides. Noise is introduced when tissue tears, separates, and distorts during the sectioning process (see Figure 1(a)(i) and (c)(i)), or by artifacts on the glass (Figure 1(c)(ii)). All sections had been treated with hematoxylin and eosin stain (H&E), which is widely used in histopathology. It colors cell nuclei blue while cytoplasm is colored pink/purple. The study was conducted on a desktop workstation (2.4GHz CPU; 12GB RAM) running Windows 7 with a 30-inch flat panel monitor (2,560 × 1,600 pixels; 60Hz).

Procedure. An initial discussion with Adam and Bob revealed the overall goal of the segmentation software being

Figure 2: *Visual analysis of parameter space with Paramorama. (a) Sampled parameter space is visualized as a tree where, left-to-right, every level represents a parameter. (b) For selected regions in parameter space, the corresponding image-based output is shown as thumbnails in the detail view. Users can tag results as good (green) or bad (magenta). (c) A reference image, typically the original input image, is supplied to visually compare the results to. The visualized data is of segmentation results for a histopathology image of lymph node tissue.*

developed and the research context in which this was happening (see Section 3.1). It was a challenge to identify how the implementation of the algorithm had been parameterized. After discussions with Adam and Bob, it was decided that some of the hard-coded values in the algorithm might better be considered as parameters and that different values for these might be more appropriate under different circumstances. Adam's approach had been to run the algorithm on a single input image and to identify suitable parameter values by stepwise iterative refinement. Once identified, these values were hard-coded in the implementation of the algorithm. Adam and Bob were confident about the robustness of these values and were not convinced that they could be improved on by exploring the parameter space further.

After a code-review with Adam, five parameters were identified for closer scrutiny. These were *smoothing* (how much noise reduction to apply), *thresholding* (the pixel intensity at which to differentiate foreground and background), *dilation* (how much to expand the foreground to "fill" holes), *expansion* (how much to expand foreground regions before detecting intersections), and *merging* (a threshold for discarding or combining intersecting regions). Adam modified his code so that it could be called with arguments to specify the parameter values to use. He then wrote a script to perform Cartesian sampling of the parameter space and to call the algorithm with all unique combinations of the sampled parameter values. Next, suitable intervals and the number of samples to compute over each interval were identified: parameters were sampled four times each, yielding 1,024 unique combinations. Corresponding segmentation output for the six input images were computed overnight.

After a training session with our prototype Adam was given an opportunity to identify regions of parameter space that produce high-quality segmentation results. In a second session, Bob was given training to use our prototype and then asked to review Adam's results. Finally, in a third session, the functionality and features of our prototype were demonstrated to Charles before being given an opportunity to try it. All participants were comfortable using Paramorama after 30 minutes of hands-on training.

4.2. Results

Adam. Our first participant required 10-30 minutes to analyze each dataset for a total analysis time of 1 hour 21 minutes for all six. The first two datasets, respectively, required

25 and 32 minutes but this dropped to between 9 and 11 minutes for the final three datasets.

Using the node-link visualization of parameter space (see Figure 2(a)), subtrees were systematically selected starting from the top and working to the bottom. Thumbnail versions of the corresponding outputs were viewed in the detail view (Figure 2(b)). In most cases Adam tagged all displayed thumbnails based on an initial impression formed by scanning the thumbnails ("good" or "bad"). Tagging was done in one operation by using the nested representation of the detail view to select all outputs. Most regions of parameter space that Adam thought yielded poor results were identified like this. Individual results of which he was uncertain were viewed in more detail using the reference image view. This often resulted in flicking "bad" tags to "good" (and vice versa). We noticed that Adam devised a rule-of-thumb for each of the six input images during his analysis. This involved scanning the output to determine whether key regions had been identified. This was often vocalized, for example: "yes, the five major regions are correctly detected" or "too many regions are identified, it's picking up noise".

Adam made a number of important discoveries. For all six input images, more optimal parameter values that differed from the hard-coded values were identified. Using the number of segmentations as a crude quality metric, the default parameter values resulted in 21 regions being erroneously segmented for Barrett's noisy, 22 for cervix noisy, 26 for lymph clean, and 16 for lymph noisy. Visual inspection also revealed that the boundaries of regions detected for Barrett's clean and cervix clean were closer to what a human expert would have picked. In particular, two parameters (*smoothing* and *thresholding*) were discovered to have an important influence on the quality of the outputs produced by the segmentation algorithm. Figure 1 contrasts the results of applying the segmentation algorithm to two of the datasets with the original hard-coded parameter values (Figure 1(a) and (c)) versus the values identified with Paramorama (Figure 1(b) and (d)). Even casual visual inspection shows that there is a vast improvement in quality for the newly identified values. Adam also came to a more general conclusion: suitable parameter values are extremely context sensitive. Each of the six images required a different set of parameter values to yield the best results. This convinced Adam that different types of tissue and the amounts of noise in the images require different parameter values to ensure high-quality segmentation.

Bob. Upon reviewing Adam's tagged results, Bob did not perform an exhaustive analysis of parameter space, but focused on a few regions that had been mostly tagged as "good" or as "bad". Bob next scanned the thumbnails for these regions in the detail view. For many results he deemed the thumbnails sufficient to confirm his agreement with the categories that had been assigned by Adam. Although not as vocal about the number of regions that the algorithm had segmented, Bob did pay attention to object count. Many results were rejected by noting, for example, "far too many artifacts". However, there were a number of individual results where Bob proceeded to view the segmentation results in more detail using the reference image view (see Figure 2(c)).

By reviewing his colleague's work, Bob identified a subtle logic error. The algorithm detects small contiguous regions and should progressively combine these to form larger regions, provided certain criteria hold. However, in many cases overlapping regions of dissimilar size were not being combined (see Figure 1(d)(i) and (ii)). Bob concluded that either the heuristics for deciding when to combine smaller regions were insufficient, or the implementation was incorrect. This sparked follow-on discussions between Adam and Bob that confirmed the existence of a logic error. It also led to a code review where additional implementation errors were discovered. The logic and implementation errors were flagged as high priority changes for a next release of the software.

Charles. When Charles reviewed Adam's results, he spontaneously changed the order of parameters in the node-link representation of parameter space. He explained that his aim was to identify those parameters that have the most "striking" influence on segmentation quality, using Adam's tags as an indication of quality. He remarked that our prototype made these relationships "very clear". Using an approach similar to that of Bob, many cases were found where Charles disagreed with the initial tagging of the outputs. This was put down to inexperience and a lack of in-depth domain knowledge on the part of Adam. Charles recognized this as a common problem when trying to get results quickly and emphasized the need for more expert reviews in the future.

4.3. Discussion

The above results provide anecdotal evidence to support our hypotheses (within the context of our study). First, participants explored larger parts of parameter space using our visualization prototype than they had previously using stepwise iterative refinement. All participants agreed that it would have been prohibitively time-consuming to analyze the number of parameter combinations they considered with Paramorama using their conventional approach. Second, visualization helped all participants to achieve new insights and learn more about the relationships between input parameter values and the resulting output. They were also more confident in their understanding of the algorithm than before. Third, using interactive visualization, participants achieved higher quality output that differed from those produced by hard-coded parameter values.

In our experience, our case study is a good example of the reaction of users to the introduction of visual analysis as an alternative to an established approach (stepwise iterative refinement of parameters for image analysis algorithms, in this case). Despite initial skepticism, participants were soon able

to apply the new method to great effect. As a case in point, an important logic error and implementation errors were discovered using our visualization prototype. These had previously gone undetected despite the analysis of segmentation output and the identification of what were believed to be optimal parameter values. Also, all participants were far less certain about the robustness of the algorithm and the hard-coded parameters after using Paramorama than they had been originally. This suggests that visual parameter optimization may be able to facilitate critical reflection better that stepwise iterative refinement.

Visual analysis also highlighted flaws in participants' assumptions and quality control during algorithm implementation. In particular, it emphasized the gap that exists between "good" results for different stakeholders (with technical versus domain knowledge). Our prototype facilitated discussion between stakeholders that would have been difficult without the ability to visually compare and refer to different outputs interactively. The threefold drop in analysis time by Adam over the six datasets suggests that there is an associated learning effect, but once users become familiar with visualization tools, such as Paramorama, they should be able to analyze large parts of parameter space in shorter periods.

The case study also revealed an important shortcoming of our approach. In terms of analysis strategy, Adam systematically considered all outputs corresponding to sampled parameter values. However, he often referred to the number of objects detected when judging whether an individual result is acceptable. The other two participants also mentioned object count while analyzing the outputs. From this we concluded that, had a facility been available to select or filter results based on object count, Adam would probably not have analyzed all outputs exhaustively but would have discarded large parts of parameter space where associated object counts were unreasonable. We also suspect that such a facility would have enabled Bob and Charles to target their detailed analysis to smaller subsets of the generated output. When we raised the issue, our participants agreed on the potential advantages of augmenting our approach with derived measures such as *object count* (the number of objects identified by segmentation) and *area occupied* (the sum of regions occupied by detected objects). They felt that this would allow users with a "gut feeling" of what suitable quality involves to identify high-potential parts of parameter space.

5. Follow-on Work

To investigate the role of derived measures, we extended our prototype to show a histogram of the distribution of values for each derived measure associated with the data (see Figure 3(a)). Inspired by the work of Tweedie et al. [TSDS95], this metric view is linked with all other views (Figure 3(b)–(d)): when the user selects one or more bars in a histogram the corresponding regions of parameter space are selected in the overview and the output shown in the detail view.

We were interested to see whether, and how, users who had not previously used our prototype would apply this new feature. To do so, we performed a preliminary study with eight students majoring in biological sciences. Because histopathology images were considered too specialist, we used the sampling plug-in we had developed for CellProfiler to compute segmentations for two 640×640 pixel photomicrographs of human osteosarcoma cells (cancerous bone tumor) [PBCR11, Bro]. Both images had been stained with DRAQ, which highlights cell nuclei as light gray on a dark background. Five parameters were sampled four times each to yield 1,024 different outputs. After training, each participant was presented with the results of one of the two images and given 10 minutes to find a region of parameter space where cell nuclei are most accurately detected.

All participants used the metric view to converge on a region of parameter space where they thought the object count was correct (see Figure 3(a)). In most cases, they started with a low guess for object count, selected the corresponding histogram in the metric view, and scanned the output thumbnails in the detail view (Figure 3(c)). Repeating this process, they revised the object count upward until settling for a value. As observed during the case study (Section 4.2), participants then considered the thumbnails in the detail view to analyze individual outputs (Figure 3(d)). Parameter values were inspected with the node-link view (Figure 2(b)).

This is a positive result that shows an improvement on our previous design, but there are further opportunities to explore. First, the inclusion of several derived measures could assist users in further reducing the subset of outputs to review, as with faceted search [Tun09]. Other measures, such as the total area covered by identified objects, could be used for this (our prototype already caters for an arbitrary number of measures). Second, the output of an objective function for each point in parameter space could be shown to more closely resemble guided navigation (see Section 2.2). We note, though, that all domain experts we interviewed were reluctant to introduce more complex measures. A third option is to visualize the distribution of outputs per parameter for a meaningful measure (using histograms, for example) and to enable users to directly filter the outputs to view in more detail using a scented widgets approach [WHA07].

6. Conclusion

The presented case study serves as an example of the potential usefulness of interactive visualization to identify suitable parameter values for biomedical image analysis algorithms. As hypothesized, using our visualization prototype, participants were able to: (1) analyze larger parts of parameter space; (2) obtain a better understanding of the underlying algorithms; and (3) achieve higher quality results than they had previously with stepwise iterative refinement. The study also serves as an example of users' reactions to the introduction of interactive visual analysis into the established field of

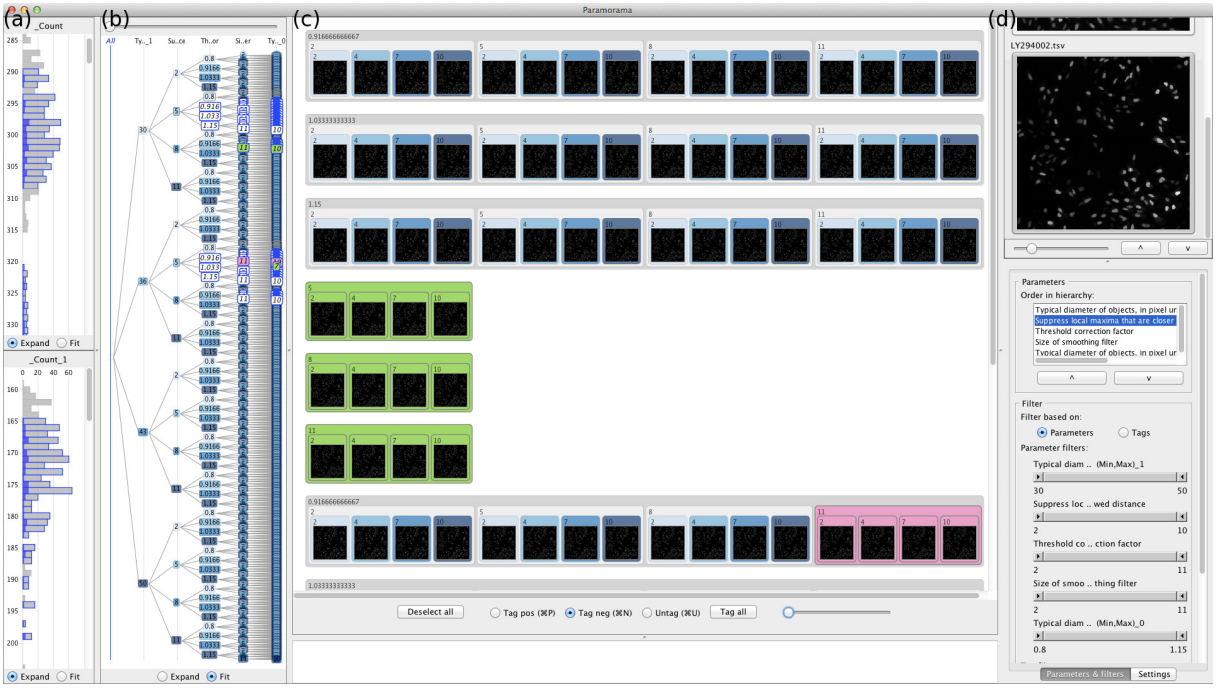


Figure 3: *(a) Interactive histograms that show the distribution of output for one of more derived measures were added to our prototype after a case study revealed this as a requirement. These are linked with (b) the parameter space view, (c) the detail view, and (d) the reference image view. The visualized data is of segmentation results for photomicrographs of osteosarcoma cells (cancerous bone tumour).*

biomedical image analysis. Despite initial skepticism, participants discovered an unknown logic error in their algorithm and errors in its implementation.

Our work also serves as an example of how case studies can serve as an important part of iterative design. Collaboration with experts revealed an important additional requirement: facilitating analysis of derived measures. Our preliminary results suggest that interactive visualization of derived measures further reduces user effort during parameter optimization. Based on this initial work, we anticipate that users will benefit further from methods that help them effectively select an even smaller subset of outputs for detailed analysis. Our prototype is available for download at [Pre].

We anticipate that the visual analysis design principles implemented in Paramorama should generalize to different image analysis algorithms and input images (in biomedicine and beyond). Indeed, the case study and follow-on work discussed in this paper were based on different segmentation algorithms and considered different input images (histopathology versus photomicrographs). To investigate this and to generalize the findings summarized above, more empirical evidence is needed. This presents a number of challenges.

Rigorous user evaluation studies require a large participant sample size while biomedical data sets encountered in the real world often require scarce specialist skills to interpret and analyze. Moreover, our findings suggest that participants would require sufficient time for training and we anticipate that the value of interactive visualization methods will emerge with repeated use over extended periods of time. This implies a longer evaluation timeline than feasible with a typical laboratory-based investigation. Designing and implementing user evaluation studies that meet all these requirements is a significant challenge similar to those encountered by other visualisation researchers [Pla04]. In this light, a more realistic goal may be for researchers to conduct more longitudinal investigations, to accumulate, and to present more anecdotal evidence, such as the present study [SP06].

Acknowlegements

We thank Anne Carpenter and Mark Bray (Broad Institute of MIT and Harvard) for crucial input and John Lau (University of Leeds) for his collaboration. This work was funded through WELMEC, a Center for Excellence in Medical Engineering funded by the Wellcome Trust and EPSRC, under grant number WT088908/Z/09/Z.

References

[AH97] ANSARI N., HOU E.: *Computational Intelligence for Optimization*. Springer, 1997. 68

[BM10] BRUCKNER S., MÖLLER T.: Result-driven exploration of simulation parameter spaces for visual effects design. *IEEE Transactions on Visualization and Computer Graphics 16*, 6 (2010), 1468–1476. 69

[BPFG11] BERGER W., PIRINGER H., FILZMOSER P., GRÖLLER E.: Uncertainty-aware exploration of continuous parameter spaces using multivariate prediction. *Computer Graphics Forum 30*, 3 (2011), 911–920. 68

[Bro] BROAD INSTITUTE OF MIT AND HARVARD: Broad Bioimage Benchmark Collection, SBS Bioimage CNT. http://www.broadinstitute.org/bbbc/sbs_bioimage_cnt.html. Last visited 31 March 2012. 73

[BSN*11] BERGNER S., SEDLMAIR M., NABI S., SAAD A., MÖLLER T.: Paraglide: interactive parameter space partitioning for computer simulations. *Computing Research Repository abs/1110.5181* (2011). 68

[CFS*06] CALLAHAN S. P., FREIRE J., SANTOS E., SCHEIDEGGER C. E., SILVA C. T., VO H. T.: Managing the evolution of dataflows with VisTrails. In *Proceedings of the International Conference on Data Engineering Workshops* (2006), pp. 71–75. 69

[CJL*06] CARPENTER A. E., JONES T. R., LAMPBRECHT M. R., CLARKE C., KANG I. H., FRIMAN O., GUERTIN D. A., CHANG J. H., LINDQUIST R. A., MOFFAT J., GOLLAND P., SABATINI D. M.: CellProfiler: image analysis software for identifying and quantifying cell phenotypes. *Genome Biology 7*, R100 (2006). 68, 70

[Har75] HARTIGAN J. A.: Printer graphics for clustering. *Journal of Statistical Computation and Simulation 4*, 3 (1975), 187–213. 68

[Ins85] INSELBERG A.: The plane with parallel coordinates. *The Visual Computer 1* (1985), 69–91. 68

[JKM00] JANKUN-KELLY T. J., MA K.-L.: A spreadsheet interface for visualization exploration. In *Proceedings of the IEEE Conference on Visualization* (2000), pp. 69–76. 69

[Ma99] MA K.-L.: Image graphs - a novel approach to visual data exploration. In *Proceedings of the IEEE Conference on Visualization* (1999), pp. 81–88. 69

[MAB*97] MARKS J., ANDALMAN B., BEARDSLEY P. A., FREEMAN W., GIBSON S., HODGINS J., KANG T., MIRTICH B., PFISTER H., RUML W., RYALL K., SEIMS J., SHIEBER S.: Design Galleries: a general approach to setting parameters for computer graphics and animation. In *Proceedings of the Conference on Computer Graphics and Interactive Techniques* (1997), pp. 389–400. 69

[Onw00] ONWUBIKO C.: *Introduction to Engineering Design Optimization*. Prentice-Hall, 2000. 68

[PBCR11] PRETORIUS A. J., BRAY M. A. P., CARPENTER A. E., RUDDLE R. A.: Visualization of parameter space for image analysis. *IEEE Transactions on Computer Graphics and Applications 17*, 2 (2011), 2402–4211. 68, 70, 73

[PBK10] PIRINGER H., BERGER W., KRASSER J.: HyperMoVal: interactive visual validation of regression models for real-time simulation. *Computer Graphics Forum 29*, 3 (2010), 989–992. 68

[Pla04] PLAISANT C.: The challenge of information visualizations evaluation. In *Proceedings of the Working Conference on Advanced Visual Interfaces* (2004), pp. 109–116. 74

[Pre] PRETORIUS A. J.: Paramorama website. http://www.comp.leeds.ac.uk/scsajp/applications/paramorama/. Last visited 1 September 2012. 74

[SP06] SHNEIDERMAN B., PLAISANT C.: Strategies for evaluating information visualization tools: multi-dimensional in-depth long-term case studies. In *Proceedings of the AVI Workshop on Beyond Time and Errors* (2006), pp. 1–7. 74

[TJOH*09] TREANOR D., JORDAN-OWERS N., HODRIEN J., QUIRKE P., RUDDLE R. A.: Virtual reality powerwall versus conventional microscope for viewing pathology slides: an experimental comparison. *Histopathology 55*, 3 (2009), 294–300. 69

[TS98] TWEEDIE L., SPENCE R.: The prosection matrix: a tool to support the interactive exploration of statistical models and data. *Computational Statistics 13*, 1 (1998), 65–76. 68

[TSDS95] TWEEDIE L., SPENCE B., DAWKES H., SU H.: The influence explorer. In *Proceedings of the Conference on Human Factors in Computing Systems* (1995), pp. 129–130. 68, 73

[Tun09] TUNKELANG D.: Faceted search. *Synthesis Lectures on Information Concepts, Retrieval, and Services 1*, 1 (2009), 1–80. 73

[TWSM*11] TORSNEY-WEIR T., SAAD A., MÖLLER T., WEBER B., HEGE H. C., VER-BAVATZ J. M., BERGNER S.: Tuner: principled parameter finding for image segmentation algorithms using visual response surface exploration. *IEEE Transactions on Computer Graphics and Applications 17*, 2 (2011), 1892–1901. 68

[WB97] WONG P. C., BERGERON R. D.: 30 years of multidimensional multivariate visualization. *Scientific Visualization: Overview, Methodologies, and Techniques* (1997), 3–33. 68

[WHA07] WILLETT W., HEER J., AGRAWALA M.: Scented widgets: improving navigation cues with embedded visualizations. *IEEE Transactions on Visualization and Computer Graphics 13*, 6 (2007), 1129–1136. 73

Short Papers

SIGRAD 2012
A. Kerren and S. Seipel (Editors)

Visualization of Text Clones in Technical Documentation

Morgan Ericsson, Anna Wingkvist, and Welf Löwe[1]

[1]Linnaeus University, Sweden

Abstract

An initial study of how text clones can be detected and visualized in technical documentation, i.e., semi-structured text that describe a product, software, or service. The goal of the visualizations is to support human experts to assess and prioritize the clones, since certain clones can be either intentional or harmless. We study some existing visualizations designed for source code, and provide initial and limited adaption of these. A major difficulty in this adaptation is to manage the semi-structured technical documentation compared to structured source code.

Categories and Subject Descriptors (according to ACM CCS): I.3.8 [Computer Graphics]: Applications—

1. Introduction to Clones and Clone Detection

Duplicated and redundant text, so called (text) clones, can be a problem in technical documentation. Technical documentation is semi-structured text that describes a product, software, or service. We consider the text to be semi-structured, since it generally contains markup, such as XML, but this structure is defined by the authors of the document.

Clones increase the size of the documentation, which adds to the cost to store, translate, print, etc. while more effort is required to maintain the documentation, for example due to update anomalies. An update anomaly occurs when not all clones are changed to reflect an update to the content, e.g., an instruction is only changed in certain parts of the documentation. However, clones can also improve the quality of a documentation, by increasing readability and understandability. Duplicated text can help to create context and familiarity, and reduce the need for cross-references. The dual nature of clones is a problem to assess the quality of a documentation and there is a need to identify clones. Further, how human experts faced with this problem can be supported by visualizations. We ask the following question: how can visualization help to (quickly) identify clones?

Roy et al. [RCK09] introduce the foundations of clone detection. A text fragment F is any sequence of text characters. It can be uniquely identified, e.g., by its file, starting, and ending position. A fragment F_1 is a *clone* of another fragment F_2 if they are considered similar. We can define (different) function(s) f to determine similarity. If $f(F_1, F_2)$ is *true*, F_1 and F_2 are clones, and F_1 and F_2 form a *clone pair*. We assume f to define an equivalence relation. Hence, we can classify all text fragments using f and the fragments $F_1, F_2, \ldots, F_3$ of the same class form a so-called *clone set*.

Clone detection is automated extraction of similar text fragments. Approaches are in the domain of similarity function f, which could be based on plain text sequences, their syntactic structure, e.g., induced by XML markup, and their meaning (the latter is undecidable in general). For a given set of documents (files) and a given similarity function f, we can compute the clone sets. As the clone fragments of different clone sets may overlap or be included in each other, post-processing is necessary to deduce the actual document uniqueness or similarity. Moreover, to avoid noise, thresholds for too small text fragments and to disregard simple or automated text transformation, pre-processing could filter and abstract from the original text sequences, for example by replacing tabs and linefeeds with spaces.

Text cloning could happen when writing with haste, but also intentionally, e.g., when similar text fragments are required in different documents, or as a result of the writing process itself, e.g., when documents come in variants or versions. In any case, awareness of clones and understanding of their genesis is necessary for assuring the quality of documentations. It helps fixing typos and spelling errors consistently, uncovering plagiarism, understanding related document sets and their evolution, and identifying text fragment that may be generated automatically. Even awareness of absence of expected clones, for example a common header that should be present in all documents, can be helpful.

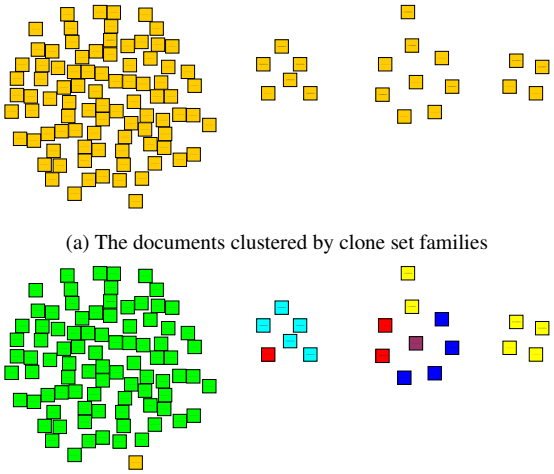
2. Visualization of Clones

There are plenty of research on how to visualize clones in source code. However, it is not clear how many of these visualizations are suitable for technical documentation. It is also not clear if the aim of these visualizations are to simply detect clones or also help to classify them. Here, we outline different kinds of visualizations and discuss how they can be adapted to technical documentation and if adapted versions provide enough information to help classify the clones. We base our discussion on the summary by Jiang et al. [JHH06]. They divided the visualizations on what relation they focus on; clone pairs or clone sets and since we want to classify clones we only focus on clone set visualizations.

The six visualizations of clone sets are: Metric Graphs [UKKI02], Hasse Diagrams [Joh94], Hyper-linked Web [Joh96], Linked Editing [TBG04], Duplication Aggregation Tree Map, and Clone Class Family Enumeration [RDL04]. Metric Graphs presents a set of metrics for each clone set, such as *%coverage* or *radius* (maximal distance from a common ancestor). Hasse Diagrams can be used to display a partial order based on the subset relation between files in clone sets. With positioning in a 2D space it shows relations between and properties of the clones, e.g., the degree of similarity. Hyper-linked Web can be considered a hypertext version of a Hasse diagram, where each clone set is displayed as a document with metrics such as size and links to super- and sub-clone sets as well as the actual files. Linked Editing shows clones in an editor view, where cloned parts are marked and can be edited simultaneously, i.e., a change can be made to all instances of a clone.

Duplication Aggregation Tree Map is modified to allow empty space. Each directory represent a box that contains smaller boxes (files). The size of the file boxes shows the number of internally (width) and externally (height) cloned lines. The size and shape of the directories mimic the amount and ratio of internal and external clones. Clone Class Family Enumeration groups clone sets if they contain the same set of files. In a 2D space, the clone class families and source files are positioned with respect to lines of cloned code, number of source files, lines of code and how many clone class families the source file is a member of, respectively.

Hyper-linked Web, Linked Editing and Hasse diagrams have interesting properties, but they are not suitable to quickly identify clones. Metric graphs rely on various metrics calculated on clones and [KKI02] put forward the need for more discussion on what metrics can be used to determine related clones. Duplication Tree Map Aggregation and Clone Class Family Enumeration provides new and improved ways to show dependencies between files, however at least Duplicate Tree Map Aggregation relies on properties of source code (package/class/directory structures, etc.). Based on our experience with documentation, it can be difficult to apply these structure properties to documentation, since the structure is not as controlled as it is in software where a method or a class have specific rules and purposes.



(a) The documents clustered by clone set families

(b) The clusters clustered by clone set with each document colored by which of the eight parts of the documentation it belongs to

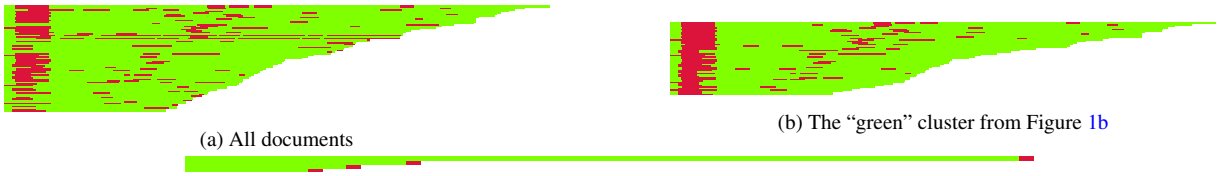
Figure 1: Visualizations of clusters of documents (square). Each cluster represents a group of document that are part of the same clone set family(ies). Nodes are not weighted and edges are removed to make the grapher clearer.

3. Experiment

We implemented a set of visualizations and applied them to the result of clone detection. The clone detection is performed on 300 documents from a real-world documentation of a telecom product that consists of about 600 documents. We define an information extractor that removes the XML markup from the documents, normalizes whitespace and converts all text to lower-case. We define a simple similarity function that recognizes identical sequences of words in sequences that are at least 100 words long. Text fragments that are identical form clone sets and are presented using a triple that consists of file name, line number of the first, and last line of the fragment, e.g., $(doc1, 4, 19)$. The total size of the 300 documents is approximately 90,000 lines.

The clone detection results in more than 500 clone sets, and approximately 25,600 lines (28%) of the documentation are clones of some other part. Compared to real-world software systems, where it is not uncommon that 7-20% of the source code are clones [RC08], this may seem high, but compared to our previous results for documentation (e.g., [WELL10], where we find in the range of 40-50% cloned text), this is low. On average, each clone set contains 7 document, and the largest contains 87.

Figure 1a shows document similarity: each node is a document grouped by similarity. Each of the four clusters depicts

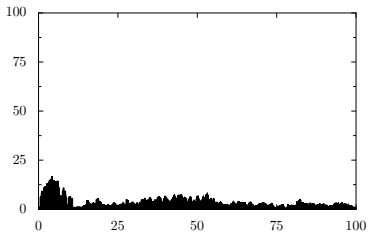


(a) All documents

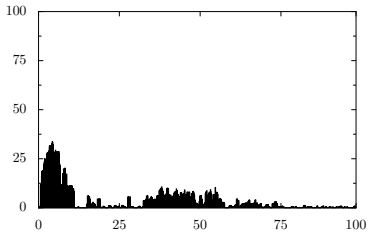
(b) The "green" cluster from Figure 1b

(c) The four node "yellow" cluster from Figure 1b

Figure 2: Pixelmap representations of three different clone sets. Each pixel represents a line, red pixels represent clones and green pixels represent non-clones. Each pixelmap is restricted to documents between 100-400 lines for clearer figures.



(a) All documents



(b) The "green" cluster from Figure 1b

Figure 3: Histograms depicting the percentage of clone sets (y-axis) that a line (x-axis) is part of. The documents are normalized to the length of the largest document. Clones are positioned relatively, but their length is not normalized. The pattern in (b) is more distinct than in (a) due to the common structure of the documents in (b).

a Clone Class Family. It illustrates where the clones exist, which is generally not enough information to gain an understanding and prioritize which clones to remove. For example, pair-wise comparison of 87 documents using a diff-tool or further visualizations is a cumbersome task. We applied a coloring based on which documentation purpose a document belongs to (extracted from file name conventions), and two clusters shows a strong correlation between purpose and similarity (cf. 1b).

The largest cluster contains documents describing a function or command. The document similarity indicates that a common template is used and clones are intentional. If we consider a pixelmap (cf. Figure 2) of a subset of the documents (documents between 100 and 400 lines, to help reduce the image size), we find that the clones (red pixels) are placed in the beginning of the documentation and in a band across the middle that mirrors the length of the document. If we investigate the cluster with four documents, where documents have another purpose, we can see that each of them shares a common postscript.

To gain more structure insights than the pixel maps provide, we project the clone sets w.r.t. one document and calculate a hierarchy based on textual containment; Figure 4 shows such a hierarchy (tree). Each tree shows the structure of clones in respective document, and each direct child of the root represents a red sequence of pixels in Figure 2 If we project and overlay the clone sets w.r.t. several documents and represent the tree nodes using different colors per document (and appropriate color mixtures for nodes representing clone sets of several documents), we can see similarity in the clone structure between the documents. This idea is similar to Jiang's et al. visualization of super clones, i.e., combinations of clones, using code execution structure (basic-block graphs) to relate different source files. Since we do not have such structure in documentation, we have to rely on simpler methods to relate different documents. One approach is to normalize the length of a document in order to make it comparable. Figure 3 shows the relative number of clone sets a line is part of. So, if the first line of all the documents are part of 10 of a total of 100 clone sets, the position 1 would have a value of 10%.

4. Discussion and Conclusions

Our main finding is that even though we rely on clone detection and extraction for source code, most of the visualizations are not a good fit for documentation. A major challenge is that documentation lacks some of the structure that is available in code. Our initial attempts suggest that further work is needed on both extraction and visualization of text clones. Most of the documentation we analyze contains text and markup, even if most of the discussion here is focused on text. We can extract certain structural properties from the markup, such as bullet lists, headers, etc., but they are not as well defined nor used as purposefully as structure and (static) semantics information of a programming language. The company that produce the documentation often provide structure, rules, and guidelines, and these could be used to provide additional semantic information. For exam-

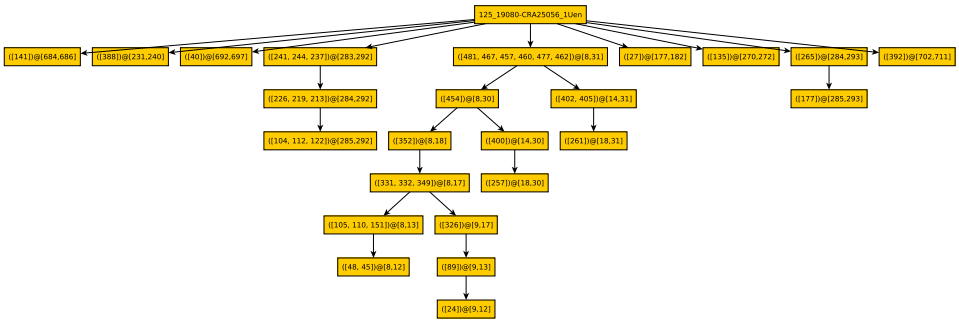


Figure 4: A tree that depicts clone sets and subsets that are present in a single document. Each box shows the clone sets the clone is part of and the line numbers in the document. The tree is built using subset containment, i.e., a parent contains the subsets of all its children. Each red sequence in a single document in Figure 2 is a direct child of the root.

ple, our coloring of the document similarity graph, based on purpose (file name conventions) helps to explain one of the large clusters of clones. However, such semantic information is based on written guidelines or convention, which might not be used consistently. Consider, e.g., the four documents with the same postscript; these all have the same purpose, but there are 15 other documents with the same purpose that are not part of the cluster.

The syntactic and semantic structure of source code is also a benefit when you want to visualize super clones, i.e. a combination of (all) clone sets, because these provide a structure that can be used to normalize the different clone sets. To achieve the same effect on documentation/text, we need to find ways to normalize the different file lengths and structures. So to visualize super clone information, we scale the file length, and position the clones relative to the start of the file. The scaling works well, if the positions of the clones are relative to the file (e.g., pre- and postscript). But scaling can affect the positioning of a clone, which can mask actual patterns and create non-existing ones. The pixelmaps are an alternative to super clone visualization. However, we still experience problems with file lengths, since patterns that are not relative are again difficult to represent.

Since we currently only consider the text when we visualize the position of the clone sets, we plan to map these back to the XML representation and see if we can use this information to improve the normalization of the documents. This might affect both the super clone and pixelmap representations. We also plan to investigate more advanced data normalizations techniques and determine if these are applicable. Another, related track is to investigate if relative positioning is exact enough, if we improve the resolution.

We will look into other types of visualizations, where we do not focus on position within the document or clusters of clone sets: for example graphs that focus on the order of the clone sets within the files, and individual differences between clone sets that are present in the same range of files. Here, the visualizations we adapt focused on source code clones, but we will study other types of software and information visualizations. Even if our experiment shows that properties of the documentation, such as types of documents, etc. can be a risk to include when clustering documents that are similar, we see if we can find a better set of properties, for example author, creation date, etc. Also, we need to improve our tools and the quality of the visualizations produced.

References

[JHH06] JIANG Z. M., HASSAN A. E., HOLT R. C.: Visualizing clone cohesion and coupling. In *APSEC* (2006), pp. 467–476. 80

[Joh94] JOHNSON J. H.: Visualizing textual redundancy in legacy source. In *Proc of the 1994 conf of the Centre for Adv Studies on Collaborative research* (1994), pp. 9–18. 80

[Joh96] JOHNSON J. H.: Navigating the textual redundancy web in legacy source. In *Proc of the 1996 conf of the Centre for Adv Studies on Collaborative research* (1996), pp. 7–16. 80

[KKI02] KAMIYA T., KUSUMOTO S., INOUE K.: CCFinder:a multilinguistic token-based code clone detection system for large scale source code. *IEEE Tran Soft Eng 28*, 7 (2002), 654–670. 80

[RC08] ROY C. K., CORDY J. R.: An empirical study of function clones in open source software. In *WCRE* (2008), Hassan A. E., Zaidman A., Penta M. D., (Eds.), IEEE, pp. 81–90. 80

[RCK09] ROY C. K., CORDY J. R., KOSCHKE R.: Comparison and evaluation of code clone detection techniques and tools: A qualitative approach. *Sci. Com. Prog. 74*, 7 (2009), 470–495. 79

[RDL04] RIEGER M., DUCASSE S., LANZA M.: Insights into system-wide code duplication. In *Proc of the 11th Working Conf on Reverse Eng* (2004), IEEE Computer Society, pp. 100–109. 80

[TBG04] TOOMIM M., BEGEL A., GRAHAM S. L.: Managing duplicated code with linked editing. In *Proc of IEEE Symp on Visual Languages-Human Centric Comp* (2004), pp. 173–180. 80

[UKKI02] UEDA Y., KAMIYA T., KUSUMOTO S., INOUE K.: Gemini: Maintenance support environment based on code clone analysis. In *Proc of the 8th Int Symp on Software Metrics* (2002), IEEE Computer Society, pp. 67–76. 80

[WELL10] WINGKVIST A., ERICSSON M., LÖWE W., LINCKE R.: A metrics-based approach to technical documentation quality. In *Proc of the 7th Int Conf on the Quality of Info and Com Tech* (2010), pp. 476–481. 80

SIGRAD 2012
A. Kerren and S. Seipel (Editors)

Analytical Semantics Visualization for Discovering Latent Signals in Large Text Collections

C. Stab, M. Breyer, D. Burkhardt, K. Nazemi, and J. Kohlhammer

Fraunhofer Institute for Computer Graphics Research IGD, Germany

Abstract

Considering the increasing pressure of competition and high dynamics of markets, the early identification and specific handling of novel developments and trends becomes more and more important for competitive companies. Today, those signals are encoded in large amounts of textual data like competitors' web sites, news articles, scientific publications or blog entries which are freely available in the web. Processing large amounts of textual data is still a tremendous challenge for current business analysts and strategic decision makers. Although current information systems are able to process that amount of data and provide a wide range of information retrieval tools, it is almost impossible to keep track of each thread or opportunity. The presented approach combines semantic search and data mining techniques with interactive visualizations for analyzing and identifying weak signals in large text collections. Beside visual summarization tools, it includes an enhanced trend visualization that supports analysts in identifying latent topic-related relations between competitors and their temporal relevance. It includes a graph-based visualization tool for representing relations identified during semantic analysis. The interaction design allows analysts to verify their retrieved hypothesis by exploring the documents that are responsible for the current view.

Categories and Subject Descriptors (according to ACM CCS): H.5.2 [Information Interfaces and Presentations]: User Interfaces—Graphical user interfaces (GUI), Interaction styles H.3.1 [Information Storage and Retrieval]: Content Analysis and Indexing—Abstracting methods

1. Introduction

The early detection of novel developments and trends is crucially important and confers companies a significant advantage over their competitors in global competition. Concepts known from the field of strategic management like weak signals [Ans75] and environmental scanning [Agu67] constitute the theoretical foundation for detecting early warnings and systematically scanning the environment of an organization for relevant information [Sch05]. Today, different research areas focus on adopting these concepts in different kinds of software systems that aim at (semi-) automatically identifying knowledge for facilitating strategic decision making. On the one hand, business intelligence platforms consider mainly internal data, whereas competitive intelligence platforms aim at gathering semi- or unstructured data from the external environment of an organization with the aim of supporting the optimization of strategic decision making. The latter are also known as *Strategic Early Warning Systems* (SEWS).

In contrast to SEWS, common information systems usually provide keyword searches that make intensive use of information retrieval technologies. These search mechanisms are primarily focused on providing the users with easy access to information of their interest and deal with the access to information items and resources [BYRN10]. The underlying assumption of information retrieval platforms is an *initial information need* of the user that is usually expressed by a keyword query. In response to the given query the user receives a sorted list that is often supplemented by *Key-Word-In-Context* (KWIC) snippets. In contrast to typical search tasks, weak signal discovery rarely starts with an initial intention expressed as a set of keywords but rather with the exploration of key topics and trends that are latent in an underlying collection of text documents. After getting an overview, analysts dive deeper into a specific topic and achieve new hypotheses and impulses by identifying novel knowledge artifacts and recognizing relations between them.

In this paper we present an SEWS called *Signal Tracing* that

is based on analytical semantics visualizations for discovering latent signals in large text collections. The approach utilizes a backend system that provides semantic analysis and data mining tools for summarizing and querying the underlying document collection. Starting at an initial state, the interactions of the analyst are translated into different queries whose results are visualized in several visualizations. In the following section we briefly introduce the general process of a SEWS followed by a detailed description of the visualization and interaction techniques that are used in Signal Tracing for discovering latent signals in large text collections.

2. The SEWS Process

Usually a SEWS passes through three general and repeating process steps (Figure 1), namely (1) information gathering, (2) analysis & diagnosis and (3) reporting and decision making. In the first step, data is gathered and selected either fully automated or manually by detecting key issues in the environment of a company. In most cases the collected data contains mainly text documents like news feeds, company websites, newsletters, customer ratings, etc. but it may also contain pictures and video material that requires additional preprocessing (e.g. image recognition, speech2text, etc.) for extracting the main content. The gathered text collection is analyzed during the second step of the process. After some preprocessing steps like stop-word removal and stemming, different techniques like document clustering, classification, key term extraction, topic detection and tracking (TDT) and more domain specific methods like those described in [ZS06, MZ05] can be utilized to summarize the collection and to extract latent signals. Beside the described data mining methods, information visualizations are increasingly used to communicate and summarize the discovered information. In addition to common visualization techniques like pie, line and bar charts there are also a number of more sophisticated methods like topic-based visualizations [LZP*12, DWCR11], interactive maps [FMG05, PM06] or graph-based methods [ZS06, GM04] for exploring latent knowledge in text collections. Finally, in the third step the inferred signals are used to evaluate strategic possibilities, to formulate potential reactions and to assess the consequences of uncovered trends and topics.

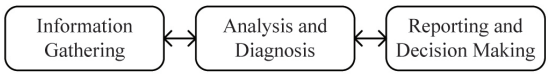


Figure 1: *Process of Strategic Early Warning Systems*

3. Visual Discovery of Latent Signals

The user interface of Signal Tracing is based on several visualization tools that cover different analytical aspects. These components are connected by brushing and linking techniques for providing multiple interactive and aspect-oriented perspectives on the underlying text collection. Altogether the user interface (Figure 2) integrates six different components: (1) Entity Explorer, (2) Trend Visualization, (3) Topic and Keyword Explorer, (4) Relation Graph, (5) Document Browser and (6) Reporting Tools.

The actual analysis and diagnosis follows a prolonged interactive loop of querying, exploring and refining using the tools provided by the Signal Tracing user interface and the analytical, semantic backend system. So the analyst is able to develop new hypotheses and to gain novel insights into the gathered sources and to infer strategic decisions. This interactive cycle includes the following tasks: (a) Entity Exploration and Selection, (b) Trend Discovery, (c) Topic Detection and Exploration, (d) Relation Analysis and (e) Validation and Verification. Thanks to the design of the user interface and its juxtaposed components, the sequence of these analytical tasks is not strictly defined but analysts are able to combine the tools in different orders. For example it is possible to insert a unknown topic identified in the Topic and Keyword Explorer as a new entity for identifying its temporal correlation with existing entities. Hence, it is possible to consider new acquired knowledge from the one perspective in another perspective to visually correlate novel insights with already existing entities in an iterative loop. A detailed explanation of how the components of the Signal Tracing user interface are related to these tasks and how the components are interactively connected is provided in the next subsections.

3.1. Entity Exploration and Selection

The Entity Explorer presents known entities and their corresponding categories that are either identified during the information gathering step of the SEWS process or emerged during the semantic analysis of the given document collection. For each entity the view includes the number of occurrences for providing an overview of the quantitative distribution. The Entity Explorer is directly connected with the adjacent Trend Visualization so that the temporal characteristics of selected entities are also visible. Thus the Entity Explorer also serves as a kind of filtering instance for selecting entities that are relevant for the further analysis. It also allows the definition of custom entities that can be incorporated in subsequent analysis steps.

3.2. Trend Discovery

The awareness of emerging and disappearing trends constitutes a decisive factor for evaluating current strategies and for accurate strategic decisions. The Signal Tracing user interface includes a Trend Visualization that represents the temporal occurrence of selected entities in an extended stacked-graph visualization. By exploring the temporal distribution, analysts are able to identify correlations between

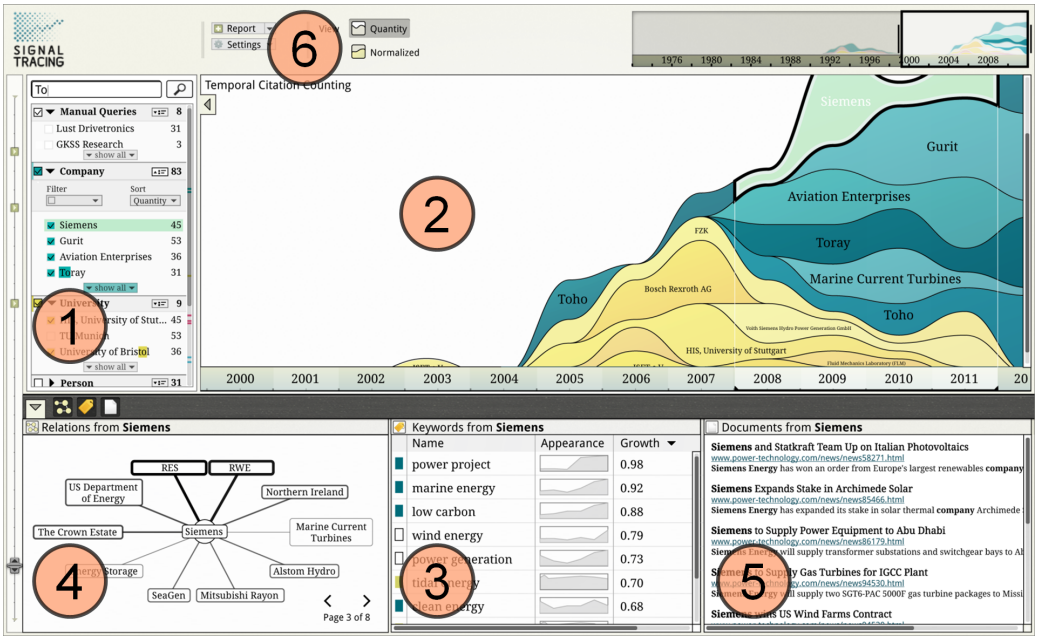


Figure 2: *The user interface of Signal Tracing incorporates the following components for discovering latent signals in large text collections: (1) Entity Explorer, (2) Trend Visualization, (3) Topic and Keyword Explorer, (4) Relation Graph, (5) Document Browser and (6) Reporting Tools.*

selected entities. The advanced overlay techniques (3.4) integrated in the stacked graph enable the exploration of topic specific and temporal co-occurrences in the given document collection.

3.3. Topic Detection and Exploration

For obtaining an overview of the content, the Topic and Keyword Explorer summarizes either the whole corpus or the current entity selection. Therefore, the background system analyses all documents that are related to the current state of the visualizations. The extracted topics and keywords are ordered according to their relevance and presented in a list view. Next to each entry, the view also includes sparklines for indicating the temporal development of each keyword and the documents that are related to the current state are included next to the Topic and Keyword Explorer in the Document Browser.

3.4. Relation Analysis

Signal Tracing provides two different types of relation discovery: Analysis of entity relations and the analysis of topic-specific and temporal co-occurrences. The first type of relations is visualized in a graph representation that shows relations between entities. The strength between the entities is indicated by the thickness and the color intensity of visible edges. So analysts are able to identify unknown relations e.g. between competitors that are often mentioned together in the documents. The graph also contains a paging feature for preventing visual clutter and switching between pages of related entities. The interactive linkage with the entity explorer allows the selection of related entities for enabling subsequent trend analysis.

The second type of relation analysis enables analysts to identify relations between topics and selected entities and to inspect their strengths over time. Visually this feature is implemented by overlays for each entity in the stacked graph. By selecting an entry in the Topic and Keyword Explorer, a query is generated for each visible entity in the Trend Visualization. The result of each query is the number of co-occurrences of an entity and the selected keyword over time that is visualized as a visual overlay in the Trend Visualization (Figure 3). The strength of the relation is indicated by the intensity of the overlay for each time interval. So it is possible to explore the evolution of a certain topic in relation to selected entities. For instance, an analyst may be interested if selected competitors are involved in a novel technology and to identify increasing or decreasing activities in this specific sector.

3.5. Validation and Verification

Finally, the Document Browser includes all documents that are responsible for the current selection and the state of the visualization respectively. So analysts are able to validate

and to verify their hypotheses by means of the sources and to collect additional data for reporting and strategic decisions.

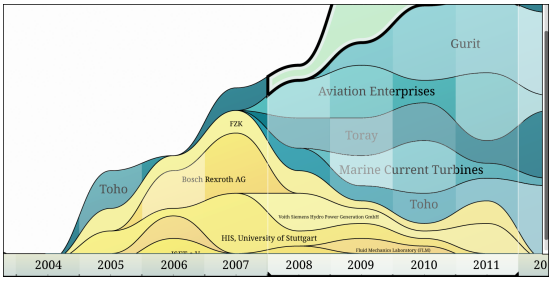


Figure 3: *The overlay technique used in the stacked graph reveals the strengths of the relations between competitors and selected topics (e.g. technologies or market sectors) over time for identifying latent trends in the underlying document collection.*

4. Related Work

Current approaches to mining strategic knowledge differ not only in the type of representation but also in the applied analysis methods. For instance Pulse [GACoR05] uses a tf-idf based clustering algorithm and sentiment analysis for extracting categories that are represented in a treemap for identifying critical issues in customer opinions. Other approaches e.g. [ZS06] [GM04] utilize co-occurrences of named-entities or keywords for providing a graph-based exploration. [ZS06] also utilizes taxonomic background knowledge for generating semantic profiles of competitors which are compared using a graph-based visualization. Tiara [LZP*12] is a visual text summarization tool that is based on Latent Dirichlet Allocation (LDA) for extracting topics whose strengths are determined at different time periods and visualized in a stacked graph but it does not contain tools to inspect co-occurrences over time. Map-based approaches [PM06, FMG05] utilize dimensionality reduction methods like Multi-dimensional Scaling (MDS), Principal Component Analysis (PCA) or Latent Semantic Indexing (LSI) for plotting high dimensional document vectors in a two-dimensional space. The resulting maps provide an overview even for very large document collections but do not reveal trends or entity relations. ParallelTopics [DWCR11] is a visual analytics system for analyzing large text corpora. It includes a document distribution view that presents the probabilistic distribution of documents across topics, a temporal view, a topic cloud and a document scatterplot. However it does not include tools for discovering latent relations in the text collection.

5. Conclusion & Future Work

In this paper we introduced Signal Tracing, a strategic early warning system for interactively discovering latent knowledge in large text collections. Besides the common process for SEWSs we presented several visualization approaches incorporated in Signal Tracing and the interactive design that fosters the identification of weak signals for strategic decisions. For the future work we plan to extend the current prototype with additional tools e.g. a visual comparison of keyword distributions might help analysts to compare different players and to apply the prototype to additional corpora.

Acknowledgements

This project (HA project no. 290/11-35) is funded in the framework of Hessen ModellProjekte, financed with funds of LOEWE - "Landes-Offensive zur Entwicklung Wissenschaftlich-ökonomischer Exzellenz, Förderlinie 3: KMU-Verbundvorhaben" (State Offensive for the Development of Scientific and Economic Excellence). We thank Dr. Rainer Vinkemeier (C21 Consulting GmbH) and Joachim Caspar (Conweaver GmbH) for the inspiring discussions and the provision of the backend system.

References

[Agu67] AGUILAR F. J.: *Scanning the business environment*. Collier-Macmillan, 1967. 83

[Ans75] ANSOFF I. H.: Managing Strategic surprise by response to weak signals. *California Management Review 18*, 2 (1975), 21–33. 83

[BYRN10] BAEZA-YATES R., RIBEIRO-NETO B.: *Modern Information Retrieval*, 2nd ed. Addison-Wesley Publishing Company, 2010. 83

[DWCR11] DOU W., WANG X., CHANG R., RIBARSKY W.: Paralleltopics: A probabilistic approach to exploring document collections. In *Visual Analytics Science and Technology (VAST), 2011 IEEE Conference on* (oct. 2011), pp. 231 –240. 84, 86

[FMG05] FORTUNA B., MLADENIC D., GROBELNIK M.: Visualization of Text Document Corpus. *Informatica Journal 29*, 4 (2005), 497–502. 84, 86

[GACoR05] GAMON M., AUE A., CORSTON-OLIVER S., RINGGER E.: Pulse: Mining customer opinions from free text. In *Proc. of the 6th International Symposium on Intelligent Data Analysis* (2005), pp. 121–132. 86

[GM04] GROBELNIK M., MLADENIC D.: Visualization of news articles. In *SIKDD 2004 at multiconference IS* (2004). 84, 86

[LZP*12] LIU S., ZHOU M. X., PAN S., SONG Y., QIAN W., CAI W., LIAN X.: Tiara: Interactive, topic-based visual text summarization and analysis. *ACM Trans. Intell. Syst. Technol. 3*, 2 (Feb. 2012), 25:1–25:28. 84, 86

[MZ05] MEI Q., ZHAI C.: Discovering evolutionary theme patterns from text: An exploration of temporal text mining. In *Proc. of ACM SIGKDD* (2005), KDD '05, ACM, pp. 198–207. 84

[PM06] PAULOVICH F., MINGHIM R.: Text map explorer: a tool to create and explore document maps. In *Information Visualization, 2006. IV 2006.* (july 2006), pp. 245 –251. 84, 86

[Sch05] SCHWARZ J. O.: Pitfalls in implementing a strategic early warning system. *Foresight - The journal of future studies, strategic thinking and policy 7*, 4 (Apr. 2005), 22–30. 83

[ZS06] ZIEGLER C.-N., SKUBACZ M.: Towards automated reputation and brand monitoring on the web. In *Proc. of IEEE/WIC/ACM* (2006), WI '06, pp. 1066–1072. 84, 86

SIGRAD 2012
A. Kerren and S. Seipel (Editors)

Analyzing Multiple Network Centralities with ViNCent

Björn Zimmer, Ilir Jusufi, and Andreas Kerren

Linnaeus University, School of Computer Science, Physics and Mathematics (DFM), ISOVIS Group,
Vejdes Plats 7, 351 95 Växjö, Sweden

Abstract

The analysis of multivariate networks is an important task in various application domains, such as social network analysis or biochemistry. In this paper, we address the interactive visual analysis of the results of centrality computations in context of networks. An important analytical aspect is to examine nodes according to specific centrality values and to compare them. We present a tool that combines exploratory data visualization with automatic analysis techniques, such as computing a variety of centrality values for network nodes as well as hierarchical clustering or node reordering based on centrality values. Automatic and interactive approaches are seamlessly integrated in one single tool which provides insight into the importance of an individual node or groups of nodes and allows quantifying the network structure.

Categories and Subject Descriptors (according to ACM CCS): E.1 [Data Structures]: —Graphs and networks H.5.2 [Information Interfaces and Presentation]: User Interfaces—Graphical user interfaces (GUI) I.3.6 [Computer Graphics]: Methodology and Techniques—Interaction techniques

1. Introduction

Network centralities are used to discover the relative importance of nodes within a complex network topology. Especially the comparison of several different centrality values in larger networks is an open problem. Identifying communities and central actors in social networks or the calculation of the importance of specific nodes in biochemical networks are some examples where multiple network centralities are used. Typical tasks during network centrality analysis include: finding nodes with high/low centrality values or finding nodes with high values in several centralities across a large number of nodes. Especially the latter task raises challenging analytical problems.

In this paper, we propose analytical extensions for our tool ViNCent [KKZ12] that offers solutions to interactively visualize networks together with their centralities. Our previous work addressed mainly visualization problems, such as the visual representation of the network itself together with the computed centralities, the interactive filtering of (non-)interesting nodes by using histogram brushing, or the minimization of clutter by the implementation of several edge bundling methods that work without a node hierarchy. Here, we discuss a set of extensions which support the analysis process. These are *node reordering* in the circular network drawing for selected centrality measurements, automatic *cluster analyses* over a set of centralities to group similar nodes together, adding a node-link view to provide the analyst with traditional topological network information, and a complete reimplementation of our tool using *OpenGL* to speed up the analysis process for larger networks.

Background This paragraph provides a brief overview of network centralities in order to facilitate the understanding of the rest of the paper. Because of the page limit, we abandon the inclusion of standard graph definitions and centrality formula. Instead, we refer the reader to various works, such as [GPQX07, DBETT99] for graph definitions or [JKS06, DHK*06, JKL*05] for centrality measurements.

A network centrality C is a function that assigns a value $C(u)$ to a node $u \in V$ of a given graph $G = (V,E)$. In order to compare network centralities according to their importance, u is more important than v if $C(u) > C(v)$, with $u, v \in V$. By using network centralities, analysts are able to better understand the structure of networks and to identify central actors. Typical examples are *degree*, *eccentricity* or *random walk betweenness* [New10]. Whereas the *degree* of a node is a very simple centrality that orders the nodes according to their degree values, *eccentricity* is calculated by using the distance (based on shortest paths) between nodes of an undirected, connected graph. More central nodes have therefore a higher value. Our tool uses the CentiBiN plug-in [JKS06] that computes up to 17 different network centralities. Note that not every centrality measure can be applied

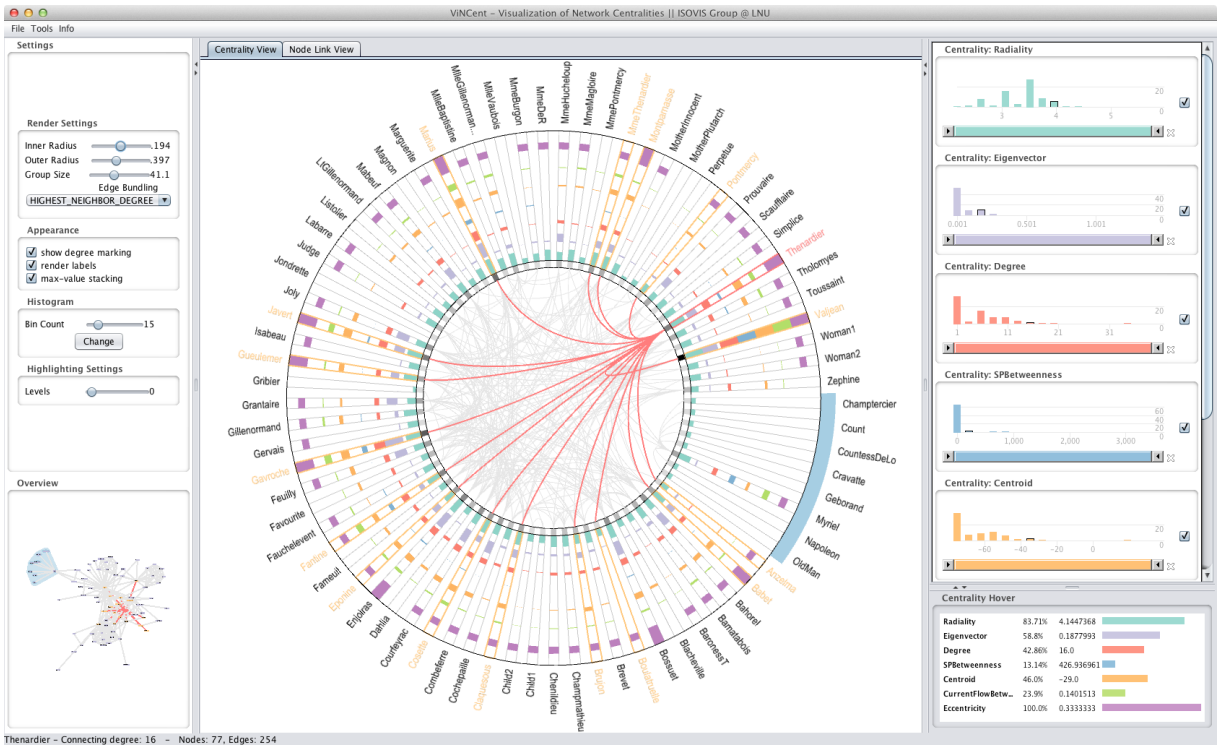


Figure 1: *Overview of ViNCent. The center shows the radial centrality view of the* Les Miserables *network. Alternatively, another view with a node-link drawing can be displayed. The right side displays the corresponding histograms of the network centralities as well as detailed values of the network centralities for the currently hovered node "Thenardier". The left panel allows changing the render settings and displays an overview of the respective node-link layout of the network. A node group has been manually created in the node-link view and is shown as a light-blue stripe along the outer circle in the centrality view as well as in the overview (bottom left) by using a background region of the same color.*

to every graph. The problem of choosing the right centralities differs from network to network. For the computation of suitable centralities, data about the functional properties of networks is often missing. This data would allow to choose the "right" centrality measures. Therefore, this analysis is usually done by visual comparison of centrality values on the networks [DHK*06].

Related Work The visualization of network centralities was not much discussed in the visualization literature so far. Typical methods are the use of correlations, scatter plots, and parallel coordinates. These solutions have disadvantages when used with networks, since they do not show where those correlations occur within the network. Dwyer et al. [DHK*06] describe three techniques to visualize network centralities in context of network drawings. These techniques and other related works are described in our previous work [KKZ12].

In the field of information visualization, there is much related work in context of general network analysis, especially in the field of social networks. Some tools include the analysis of centrality measurements, for example, SocialAction [PS08]. It allows users to rank nodes by centralities and to display them in coordinated views as ordered lists or to compare them by scatterplots. A recent overview of techniques and tools for social network analysis is given by Correa and Ma [CM11].

2. Approach

ViNCent provides multiple, coordinated views on the input data, see Figure 1 for an overview. Concretely, it supports a radial centrality view, a standard node-link view, interactive histograms for filtering as well as a view to display the individual centrality values of hovered nodes. The node-link view uses a standard force-directed layout algorithm to show the general network structure. The centrality view is used to display and analyze all calculated centralities of the input network. Each network node is represented by a small quadrangle that is positioned on a circle. The quadrangle is gray-scaled according to the corresponding node's degree. Its connections to the other nodes (i.e., the edges) are laid out inside of this circle. The centralities are stacked as additional quadrangles on each node, providing a good comparability of all relative centrality values. In order to distinguish

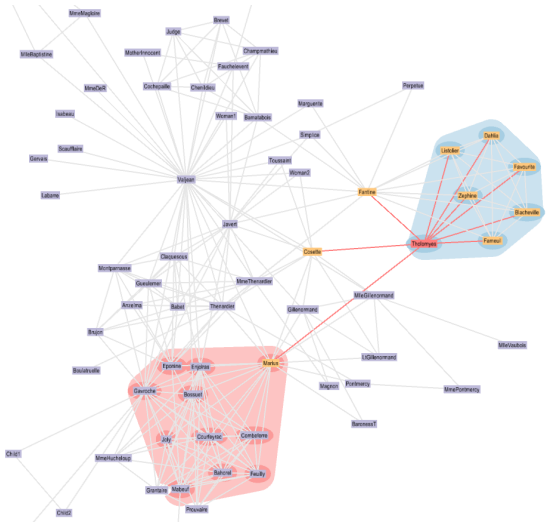
between single centrality values, a specific color schema is employed. A more detailed explanation about the visualization and interaction approaches used in our tool can be found in [KKZ12]. For our sample screenshots, we use the network of characters in the novel *Les Miserables* [Knu93]. It consists of 77 nodes which represent characters and 254 edges which connect characters that appear in the same chapter of the novel.

2.1. Interactive Visual Analysis Methods

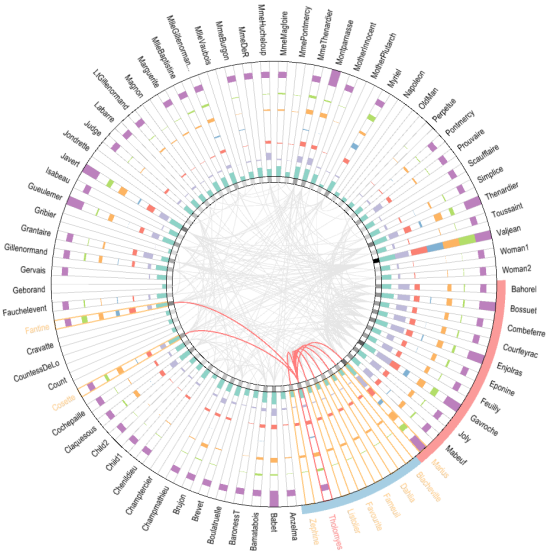
All views of ViNCent are connected to each other via brushing and linking. Hovering an object in the network centrality view, node-link view or histogram bars highlights the respective node or its corresponding data in all other views. Filtering of nodes or centralities is possible by clicking on a histogram bar, by changing the range sliders of the histograms or by clicking on a node in the centrality view. To better facilitate the analysis of centralities in a graph, our reimplementation of ViNCent offers new options to group and to order all or selected nodes in the centrality view based on their centrality values.

Node Grouping If the initial layout of a graph displayed in the node-link view reveals a couple of nodes that look interesting, the user can manually select these nodes and group them together. All grouped nodes will be drawn side by side in the centrality view, increasing the comparability of all centralities in a group. The group markers are drawn as small colored stripes along the outer circle in the centrality view. Figure 2 shows an example with two manually selected groups. The user can also automatically group all nodes by using a k-Means clustering algorithm [Mir05]. For this, he/she chooses which centralities should be used for the clustering algorithm and how many clusters (k) are desired. Automatic clustering is especially useful while analyzing large networks, where it is difficult to see central nodes by just looking at the node-link representation. Figure 3 shows the centrality view after clustering the *Les Miserables* graph.

Node Ordering By clicking on a rectangle of a specific centrality value in the centrality view, all nodes are arranged by their corresponding values in descending order. This makes it easier to compare centrality values within the same magnitude. Figure 3 shows a small use case based on our sample data set. In Figure 3(a), the user has clicked on the *centroid* centrality for the node representation of the character "Marius". All nodes are now aligned in descending order of their respective *centroid* centrality values, whereas the ordering is done for each group individually. In this case, "Marius" has the highest *centroid* centrality and is therefore drawn at the first position of his group. This rule applies for all following groups on the circle, providing the possibility to quickly identify the highest/lowest centrality values for each group individually. Figure 3(b) shows the same data set with a node ordering based on the *eigenvector* centrality.



(a) Node-link view with two groups manually created by the user.

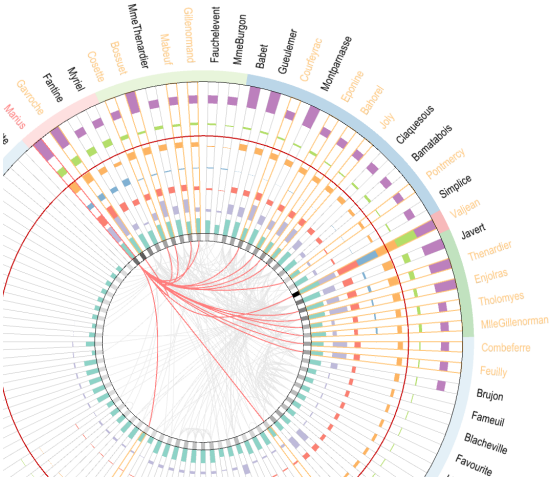


(b) Centrality view showing the selected groups. The group markers are drawn as colored stripes along the outer circle.

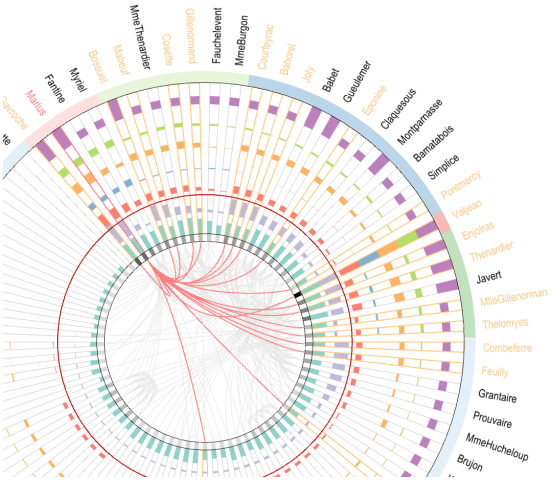
Figure 2: *Two manually grouped nodes displayed in the node-link (a) and centrality view (b). Color brushing is used to identify the groups in each view. The name "Tholomyes" is hovered and simultaneously highlighted in red color within both views. All direct neighbors of the corresponding node, i.e., in graph-theoretic distance of one, are highlighted in orange color together with the connecting edges.*

3. Conclusions and Future Work

The additional node-link view enables the user to better perceive the network's topology in order to interactively preselect interesting nodes. It also supports node grouping to analyze the nodes in the centrality view afterwards. With the help of our new node ordering functionality, nodes with



(a) Centrality view ordered by *centroid* centrality (orange).



(b) Centrality view ordered by *eigenvector* centrality (lavender).

Figure 3: *Both centrality views show the groups created by a k-means clustering algorithm and the node ordering for two different centrality values. Here, nodes with low centrality values are in the largest group on the lower part of the circle, whereas all nodes with higher centralities are grouped into five smaller clusters on the upper right part. Thus, node grouping and ordering provide an easy way to analyze and compare the highest centrality values. While a centrality of a node is hovered (here: "Marius") a red circle is drawn at the corresponding centrality value to allow a better comparison of this value with those of other nodes.*

similar centralities can be arranged side by side to increase the comparability of their specific network centralities. Also, the lack of immediate feedback (less than five frames per second while exploring larger networks) in our previous work [KKZ12] was solved by using OpenGL to render the centrality and node-link visualizations. This increased the performance to 25 frames per second on a 2.2 Ghz CPU using a Radeon HD 6750M graphics card. For scalability reasons, we plan to embed centrality views into the node-link representation to either manually or automatically group nodes and replace them by our circular centrality view, similar to the NodeTrix approach [HFM07]. Using hierarchical clustering layouts [NL05] in this context should decrease the number of overlapping groups in the node-link view. Additionally, we will evaluate our tool together with domain experts in biological and social networks to clarify arising challenges while analyzing the centralities of large networks.

References

[CM11] CORREA C. D., MA K.-L.: Visualizing social networks. In *Social Network Data Analytics*, Aggarwal C., (Ed.). Springer, 2011, pp. 307–326. 88

[DBETT99] DI BATTISTA G., EADES P., TAMASSIA R., TOLLIS I. G.: *Graph Drawing: Algorithms for the Visualization of Graphs*. Prentice Hall, 1999. 87

[DHK*06] DWYER T., HONG S.-H., KOSCHÜTZKI D., SCHREIBER F., XU K.: Visual analysis of network centralities. In *Proceedings of the 2006 Asia-Pacific Symposium on Information Visualisation (APVis'06)* (Darlinghurst, Australia, 2006), Misue K., Sugiyama K., Tanaka J., (Eds.), Australian Computer Society, ACM International Conference Proceeding Series, vol. 164, pp. 189–198. 87, 88

[GPQX07] GÖRG C., POHL M., QELI E., XU K.: Visual Representations. In *Human-Centered Visualization Environments* (2007), Kerren A., Ebert A., Meyer J., (Eds.), LNCS Tutorial 4417, Springer, pp. 163–230. 87

[HFM07] HENRY N., FEKETE J.-D., MCGUFFIN M. J.: Nodetrix: a hybrid visualization of social networks. *IEEE Transactions on Visualization and Computer Graphics (IEEE Visualization Conference and IEEE Conference on Information Visualization) Proceedings 13* (2007), 1302–1309. 90

[JKL*05] JACOB R., KOSCHÜTZKI D., LEHMANN K. A., PEETERS L., TENFELDE-PODEHL D.: Algorithms for centrality indices. In *Network Analysis*, Brandes U., Erlebach T., (Eds.). Springer, 2005, pp. 62–82. 87

[JKS06] JUNKER B., KOSCHUTZKI D., SCHREIBER F.: Exploration of biological network centralities with CentiBiN. *BMC Bioinformatics 7*, 1 (2006), 219. 87

[KKZ12] KERREN A., KÖSTINGER H., ZIMMER B.: Vincent - visualisation of network centralities. In *Proceedings of the International Conference on Information Visualization Theory and Applications (IVAPP '12)* (2012), INSTICC, pp. 703–712. 87, 88, 89, 90

[Knu93] KNUTH D. E.: *The Stanford GraphBase: a platform for combinatorial computing*. ACM, New York, NY, USA, 1993. 89

[Mir05] MIRKIN B.: *Clustering for Data Mining: A Data Recovery Approach*. Chapman & Hall/CRC, Boca Raton, FL, USA, 2005. 89

[New10] NEWMAN M. E. J.: *Networks: An Introduction*. Oxford University Press, 2010. 87

[NL05] NOACK A., LEWERENTZ C.: A space of layout styles for hierarchical graph models of software systems. In *Proceedings of the 2005 ACM symposium on Software visualization* (New York, NY, USA, 2005), SoftVis '05, ACM, pp. 155–164. 90

[PS08] PERER A., SHNEIDERMAN B.: Systematic yet flexible discovery: guiding domain experts through exploratory data analysis. In *Proceedings of the 13th International Conference on Intelligent User Interfaces* (New York, NY, USA, 2008), IUI '08, ACM, pp. 109–118. 88

SIGRAD 2012
A. Kerren and S. Seipel (Editors)

Towards Interactive Visual Analysis of Microscopic-Level Simulation Data

Martin Luboschik, Christian Tominski, Arne T. Bittig, Adelinde M. Uhrmacher, and Heidrun Schumann

Institute for Computer Science, University of Rostock, Germany

Abstract

In this work, we aim at facilitating the analysis of spatial simulations of particles at the microscopic level. This level poses significant challenges to interactive visual analysis tools. On the one hand, the data may contain up to 100.000 data points, and on the other hand, the data exhibit Brownian motion. As a first step to deal with these challenges, we apply well-accepted techniques to visualize the data and to allow analysts to interact with the data and their visual representation. Preliminary results from a spatial simulation of protein–lipid-raft interaction indicate that interactive visual solutions are indeed a useful addition to the modeling and simulation toolbox.

Categories and Subject Descriptors (according to ACM CCS): I.3.6 [Computer Graphics]: Miscellaneous—Visualization of simulation data

1. Introduction

Interactive visual analysis tools have the potential to significantly aid in modeling and simulation of biological phenomena. Current research investigates spatial simulation at the *microscopic* level, a level that imposes significant challenges to be addressed by a visualization solution. A primary concern is the visualization of thousands of moving entities associated with multiple attributes evolving in space and time. On top of that, the visualization has to deal with chaotic movement, because entities moving at the microscopic level are governed by Brownian motion.

We present our ongoing work to facilitate the simulation of spatial phenomena at the microscopic level under consideration of the above challenges. We show how common visualization approaches can be applied as a first step to support analyzing microscopic-level simulation data.

In the next section, we will provide some more details about the application background, the involved simulation methods, and the generated simulation data. Based on that we describe major challenges to be tackled by interactive visual analysis tools. Then we illustrate how the current simulation practice can be improved by means of interactive visualization of the simulated particles. The results achieved with our preliminary solution attest to the advantages of interactive visual analysis.

2. Simulation of Proteins and Lipid Rafts

Spatial simulation of cell biological phenomena at the microscopic level can reveal previously unknown behavior, even for well-studied components such as the MAPK pathway [TTNtW10]. New modeling and simulation approaches follow this promising avenue. In this work, we deal with membrane surface dynamics of receptor proteins and lipid rafts [NBPH06], which are relevant to cellular signaling (e.g. in the cancer-related Wnt pathway [KYS09]). To gain new insights into the interplay of proteins and lipid rafts, the simulation experts apply a rule-based model description and hybrid simulation approach, called ML-Space, which combines (mesoscopic) reaction-diffusion system and (microscopic) particle-based simulation [BHMU11].

Here the focus is on the particle-based simulation. The simulated models comprise up to two thousand proteins with a diameter of approximately 2 nm and up to a few hundred lipid rafts with diameters varying between 6 and 50 nm. The particles are simulated in continuous 2D space.

During the simulation, the following rules are applied. Based on a diffusion factor, the positions of the particles are updated randomly such that the particles exhibit Brownian motion [CCFV05]. Potential collisions are handled differently depending on the involved particles. When a protein and a lipid raft collide, the protein is placed just inside the

raft's boundary and its diffusion factor is reduced. If two entities of the same type are about to collide (for which no applicable reaction is defined in the model), a new random position is drawn until the collision is resolved. Moving a raft implies taking along all the proteins it includes. That is, the protein positions are updated by their own and their encompassing raft's update vector. When a protein inside a raft moves against the raft's boundary, it is placed right outside of the raft and its original diffusion factor is restored.

The simulation is started with all proteins outside rafts and stopped when the number of proteins inside rafts has reached a steady state. The simulated time frame then is between fractions of a second and a few minutes .

3. Visualization Challenges

Developing interactive visual tools to assist in the simulation is challenging for reasons related to the generated data and for reasons related to the tasks and goals of the analysts.

The data can become quite large with thousands of particles evolving over hundreds of time steps, resulting in data sets that can easily exceed 100,000 data points. On top of that, the simulated particles follow Brownian motion due the physical laws in force at the microscopic level. Therefore, any movement analysis will have to face undirected high frequency motions as the fundamental components of the data and heavily self-intersecting movement trajectories. So visual clutter will be a serious concern.

Within this chaotic motion of many particles, our collaborators seek to gain insight into the complex spatial interplay of different types of particles. In the first place, the scientists need to confirm certain characteristics to validate the simulated model and the simulator itself. Secondly, the data of valid simulation runs need to be analyzed for new findings regarding the simulated biological phenomenon. This analysis involves several facets.

Understanding spatio-temporal characteristics is at the heart of any spatial simulation. This requires appropriate communication of the spatial and temporal dimensions of the simulation as well as the particles' location and movement in space and time. Although this is nothing uncommon, it becomes difficult when several attributes associated with each entity evolve during the movement and have to be visualized accordingly. While the visualization of a single attribute along self-intersecting trajectories is complex for its own, it becomes even more challenging when two or more related attributes need to be shown together.

Besides these primary challenges, there are further aspects concerning the exploration of data. For example, flexible interaction techniques are required to enable the user to focus on specific parts of the data. In order to extract characteristic behavior, it should be possible to compare potentially interesting regions.

Developing tools that address these challenges is our ongoing research. Next we present first steps towards this goal.

4. Visualization of Protein–Lipid-Raft Interaction

Our initial solutions consists of a preprocessing step to enhance the data, a dynamic animation of moving particles and a static visualization of movement trajectories, as well as interaction techniques to filter the data. The following paragraphs will provide the details.

Data Enhancement The raw data hold only sparse information about the simulated proteins and lipid rafts. There are tuples (i,t,p,e), where i is an identifier, t is a time-stamp, p is the particle's position, and e is an optional event flag. Events occur when proteins entered or exited a lipid raft, and when the simulator resolved a potential collision.

Because the time domain is continuous, the simulator reports particle positions and events separately for individual particles at arbitrary time-stamps. In order to ease the data enhancement process, we first transfer the data to a unified time line, where missing data points are computed via linear interpolation. In a second step, we derive data attributes $a_1,\ldots,a_n$ providing further details on the particles movement and thus form tuples $(i,t,p,e,a_1,\ldots a_n)$. Following the suggestions for potentially interesting attributes by Andrienko et al. [AAH*11], we compute speed, direction, and curvature as derived attributes. Furthermore, we take the special simulation background into account and compute for each protein the distance and the relative difference in direction to the closest lipid raft. The enter/exit events are translated to a *containment* attribute. Beside storing attributes on a per-tuple basis, we additionally compute cumulative values per particle (e.g., average speed). This comes in handy when searching for interesting particles through dynamic queries.

Visualization Design Due to the multi-faceted character of the simulation data and the analysis goals, it makes sense to investigate different visualization techniques. Our initial solution is a two-fold design: a dynamic representation of the particle movement and a static representation of derived movement attributes.

The dynamic representation shows animated particles for live monitoring of the simulation. As illustrated in Fig. 1, each particle is represented by a circle whose size corresponds to the particle's diameter. Color is used to encode the type of the particle: lipid rafts are green, proteins inside lipid rafts are red, and proteins outside lipid rafts are blue. To include a bit of the history of the particle movement, old positions are indicated as dimmed circles.

The static representation in Fig. 2 shows the particles' movement as trajectories which encompass all positions a particle has visited during the simulation. We use a straightforward polyline visualization of the trajectories. Along the polylines we color-code a user-selected attribute. That cod-

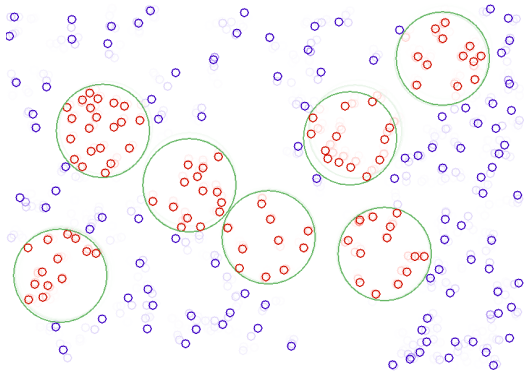


Figure 1: *Snapshot of the particle animation.*

ing is based on an appropriate classification of the attribute's value range and on adapted color schemes from [HB03].

Interaction To address the chaotic character of the Brownian motion, we provide Gaussian kernels of different size. These can be applied and adjusted interactively depending on the concrete data and the desired smoothing results. They have to be used carefully, since smoothing implies a loss of high frequencies in the data. On the other hand it is beneficial for example when examining the overall movement direction of lipid rafts and interior proteins. To further address the serious clutter of trajectories, we provide several interactive filter sliders. These can be used to reduce the time range to be shown or to focus on particles of different type. Filtering out class intervals of the visualized attribute via an interactive legend further helps in focusing on those parts of the trajectories relevant to the task at hand. Common zoom and pan operations and smooth viewport transitions facilitate overview and detail exploration.

Although being rather straight-forward solutions, the insights that could be gained with the help of them are already quite promising.

5. Preliminary Results

The interactive visual approach was successfully applied to confirm that the simulated models are mostly valid. Additionally, new and unexpected findings could be revealed using the interactive visualization.

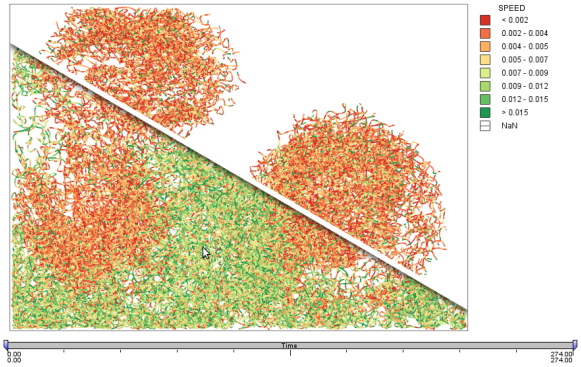
Dynamic Representation The particle animation (Fig. 1) communicates fundamental characteristics of the spatial simulation. When looking at the animation as a whole, one can confirm that the Brownian motion mechanism performs well: All particles move constantly resulting in a fully dynamic animation without any suspicious behavior such as halting particles or preferred moving directions. Another important result is the correctness of the inclusion mechanism. At the beginning, the animation shows empty lipid rafts, which continuously collect proteins during the simulation. The color coding helps in distinguishing outside and inside proteins, with the latter constantly growing in numbers. Finally the lipid rafts reach a kind of steady state, dropping and collecting proteins in equal measure. When looking at individual animation frames, the current particle positions and the particle distribution can be seen without clutter. Since no conspicuous features like overlapping proteins appear, it was concluded that the basic position updates work correctly.

On the other hand, the animation suffers from intense flickering due to Brownian motion. It is difficult to judge particle characteristics like speed, general direction, or collisions. The same applies for the particles' evolution in terms of their movement and their associated attributes. Showing dimmed circles for previous positions and providing interaction techniques to navigate in time are only first approaches to overcome this drawback. Still this issue requires further investigations.

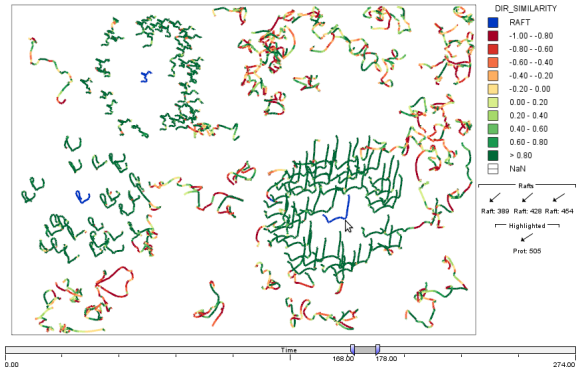
Static Representation The static representation primarily focuses on the evolution of the particle movement. All particle trajectories are shown condensed in one view (Fig. 2). Alternately selecting a derived attribute to be color-coded along the trajectories helps in confirming further aspects of the simulation. For example, Fig. 2(a) (bottom-left) shows the evolution of the particle speed. Although individual particles are difficult to extract, it is overall clearly visible that proteins slow down (red) in certain regions. To find out why, an interactive filter (operating on the containment attribute) is used to focus on protein's trajectory segments that lie inside of lipid rafts. The filtered visual representation in Fig. 2(a) (top-right) reveals that the slow speeds occur exclusively inside lipid rafts. This is the expected behavior as modeled by the rule to reduce the diffusion factor when proteins enter a lipid raft.

The trajectory based visualization is also useful to confirm the synced movement of lipid rafts and interior proteins (Fig. 2(b)). Interactive filters are applied to limit the visible time span and to create appropriately small trajectory extracts. This way, visual clutter is reduced and individual trajectories become visible allowing for the comparison of movement directions. Analyzing the synced movement is supported by a color map that encodes the derived similarity of movement directions (from green: equal direction to red: opposite direction, blue: lipid raft). The colored trajectories clearly show that the lipid rafts and their included proteins follow similar paths, as it was intended by the model's rules.

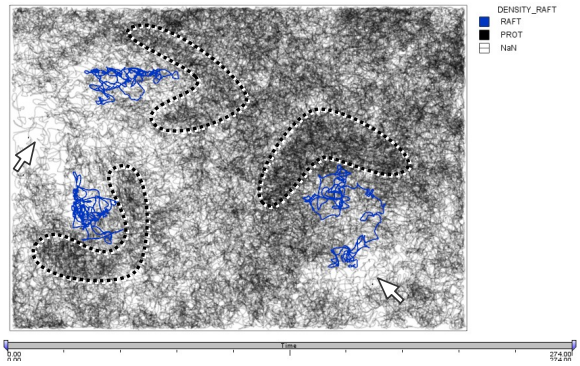
Besides confirming characteristics defined by the model, the trajectory approach also unveiled new findings. Although the dynamic visualization approach revealed no suspicious behavior concerning the distribution of proteins at the level of individual time points, the trajectory visualization does so for the *whole* simulation. In Fig. 2(c) only the particle trajectories (black: proteins, blue: lipid rafts) are shown. What we can see is an uneven distribution of particles: Dense (marked areas) and sparse (arrows) regions emerge near the



(a) Speed of particles.



(b) Similarity of direction.



(c) Dense and sparse regions.

Figure 2: *Color-coded trajectory visualization.*

lipid rafts. This pointed the simulation experts to reconsider their assumptions. A peculiarity of the model was that a moving lipid raft takes up proteins it collides with (given free space inside), but does not leave a "trail". It was reasoned that a possible gap "behind" a moving raft would be filled rather quickly by other particles diffusing into that sparse space. The lighter regions with few trajectories passing through strongly suggest that this is not the case given the relatively low particle density of the test models. This finding prompted investigations of models with different diffusion factors and protein numbers.

But as with the animation approach, the trajectory-based approach has some drawbacks. Some simulation characteristics are not visible at a glance. This includes for example the position of particles at a certain time, the particles' extent, as well as collisions or the inclusion of proteins. Therefore, dedicated attributes and interactive filters have to be used.

6. Conclusion

The challenges identified in this work are significant. Our initial approach to address them is to apply basic visualization techniques combined with appropriate interactive tools. This straight-forward solution already helped simulation experts in their work.

However, more research is required to come up with full-fledged approaches dealing with the Brownian motion, multiple evolving attributes and with larger simulation data. In future work, one could investigate mechanisms to abstract from the Brownian motion in order to allow for a more abstract visual representation. This would also open up possibilities to visualize multiple attributes and to handle larger datasets. Another promising avenue is to follow a feature-based approach to automatically extract interesting regions in the data (like those we marked manually in Fig. 2(c)).

References

[AAH*11] ANDRIENKO G., ANDRIENKO N., HURTER C., RINZIVILLO S., WROBEL S.: From Movement Tracks through Events to Places: Extracting and Characterizing Significant Places from Mobility Data. In *Proceedings of the IEEE Symposium on Visual Analytics Science and Technology (VAST)* (2011), IEEE Computer Society, pp. 161–170. 92

[BHMU11] BITTIG A. T., HAACK F., MAUS C., UHRMACHER A. M.: Adapting Rule-based Model Descriptions for Simulating in Continuous and Hybrid Space. In *Proceedings of the International Conference on Computational Methods in Systems Biology* (2011), ACM, pp. 161–170. 91

[CCFV05] CECCONI F., CENCINI M., FALCIONI M., VULPIANI A.: Brownian Motion and Diffusion: From Stochastic Processes to Chaos and Beyond. *Chaos: An Interdisciplinary Journal of Nonlinear Science 15*, 2 (2005), 026102–1–026102–9. 91

[HB03] HARROWER M. A., BREWER C. A.: ColorBrewer.org: An Online Tool for Selecting Color Schemes for Maps. *The Cartographic Journal 40*, 1 (2003), 27–37. 93

[KYS09] KIKUCHI A., YAMAMOTO H., SATO A.: Selective Activation Mechanisms of Wnt Signaling Pathways. *Trends in Cell Biology 19*, 3 (2009), 119–129. 91

[NBPH06] NICOLAU D. V., BURRAGE K., PARTON R. G., HANCOCK J. F.: Identifying Optimal Lipid Raft Characteristics Required To Promote Nanoscale Protein-Protein Interactions on the Plasma Membrane. *Molecular and Cellular Biology 26*, 1 (2006), 313–323. 91

[TTNtW10] TAKAHASHI K., TĂNASE-NICOLA S., TEN WOLDE P. R.: Spatio-temporal Correlations Can Drastically Change the Response of a MAPK Pathway. *Proceedings of the National Academy of Sciences 107*, 6 (2010), 2473–2478. 91

SIGRAD 2012
A. Kerren and S. Seipel (Editors)

Interactive Visualization for Real-time Public Transport Journey Planning

Josua Krause, Marc Spicker, Leonard Wörteler, Matthias Schäfer, Leishi Zhang, and Hendrik Strobelt

University of Konstanz, Germany

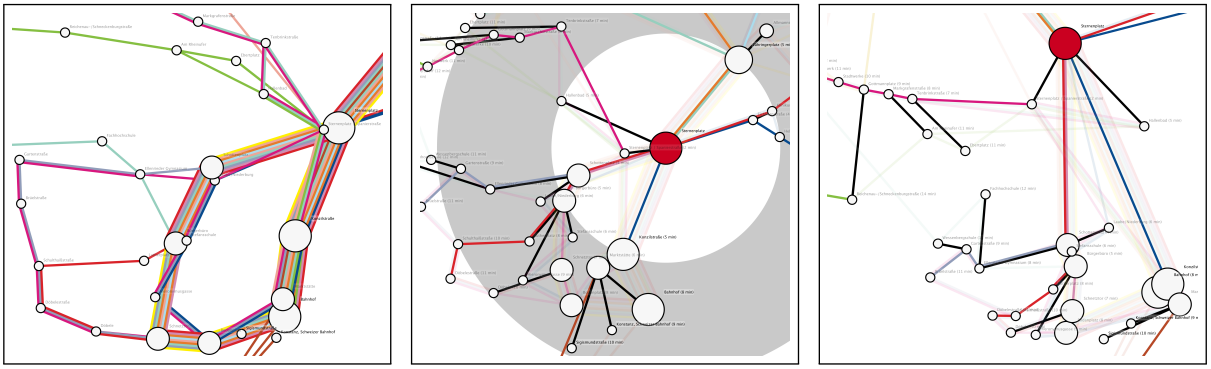


Figure 1: *Visualization of the bus transportation system of Konstanz (Germany) created with* BusVis. *Left: Geographical layout. Center: Radial layout. Right: Stress-majorization layout.*

Abstract

On-line journey planners encourage the use of public transport systems by easing the task of finding "optimal" routes between start and destination locations. Most of the tools lack the flexibility of comparing journeys that end at different (but maybe nearby) destinations. In this paper, we propose a novel visualization tool for public transport journey planning that integrates graph layout techniques to allow visual comparison of travel time and journey directions. Our tool uses data from the bus system of Konstanz. Taking this data as example, use cases describe the tool's applicability to trip planning with route alternatives.

1. Introduction

The motivation of using public transport is often hindered by inflexibility in journey planning. Without careful planning a traveler might end up wasting a lot of time waiting for the next connection or walking a long distance to reach another station. To solve this problem much effort has been devoted to develop on-line journey planning tools, for example, the Transport for London journey planner [Lon], the Public Transport Victoria journey planner [PTV], or the Rail travel planner Europe [Rai]. Most of these systems allow the user to enter a starting point and destination of a planned journey to find optimal routes between them. Some systems additionally allow for setting preferences (e.g., change times or types of vehicles) to refine results accordingly. However, most of them lack the ability to set multiple destinations to compare different journeys.

In this paper we propose a novel visualization system that provides overviews of a transport system in terms of distance and travel time, as well as easy comparisons between different routes and journeys. We exemplify our approach by showing our tool, *BusVis*, for planning bus journeys within the city of Konstanz, Germany. The system focuses on visualizing travel time between different locations and supports comparison of multiple destinations. Two modes of visualization display travel time with focus on a position selected by the user.

2. Related Work

Map drawing is a well established discipline ranging from prominent examples like Beck's map [Gar94] to recent techniques allowing automatic creation of metro map layouts [SRMOW11, NW11]. While being a generally broad topic, we want to focus strongly on related approaches for journey planning.

Early evidence for superior efficiency of using spatial maps over tabular data for choosing bus routes has been given by Bartram [Bar80]. Bogen *et al.* [BBZ10] analyze map examples, both historic and current, to open up space for innovative map designs, blending art history and computer science. In this context, the authors propose distortion and details-on-demand techniques to help the user focus on relevant areas rather than the complete map. Böttger *et al.* [BBDZ08] use such a technique to warp maps. A semantic zoom provides the possibility to explore the nearby area of a station without distortion while the overview is warped to represent the schematic map.

Hoar [Hoa08] proposed a web based geographic information system that displays schedule times of buses. The user can interactively query the system by entering bus stops or geographic locations and see the optimal route highlighted. Visualizing network lines on top of a geographic map has advantages in reflecting distances between stations. In crowded central areas of a map lines may be occluded. Wang and Chi [WC11] addressed this occlusion problem and proposed a focus and context technique that allows highlighting of routes while the remaining context is deformed. In contrast to standard focus lens techniques their deformation method is guided by Beck's map constraints. Both approaches, like many existing on-line journey planners, allow only planning of one-start-to-one-end connections.

Our system was encouraged by a visualization of the London Underground by Tom Carden [Car12]. His system allows to focus on a station and recalculate travel time from this station to other stations. Time is projected as distance to the selected station and angles reflect directions on the original map. One drawback of the system is that it uses time independent routing which ignores the schedule and possible waiting times.

3. Application

We design *BusVis* as interactive visualization tool for journey planning in transit systems exemplified on the bus network of the city of Konstanz, Germany. Our goal is to provide a flexible system that allows travelers to search for optimal routes between locations while still being able to compare different routes. The example data set consists of 123 bus stations, 18 bus lines and 16,088 edges between stations with departure times and travel duration. Based on this, the system finds fastest routes from one starting location to multiple destinations and maps travel time to visual distance. Visualization techniques are applied to show query results in a way that allows intuitive understanding, interaction, and comparison between different routes.

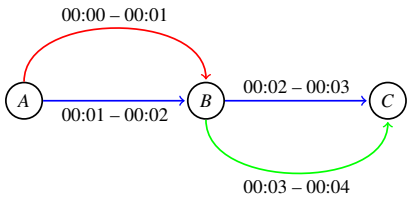


Figure 2: *Change times allow suboptimal intermediate steps to result in an optimal solution. From station A the fastest path to B is the red line. Considering a change time of two minutes, it is impossible to catch the blue line at B. Therefore the green line has to be taken. However, when a traveler takes the blue line at A he reaches B later than with the red line but arrives earlier at C.*

BusVis provides an interactive graphical user interface to support dynamic queries and shows the results visualized in a dynamic node-link diagram where nodes represent stations, edges show connections between them, and the size of a node depicts the number of lines. This allows rapid interpretation of overall connectivity and easier path tracing than adjacency matrices such as regular bus plans [GFC04, Bar80]. Edges of bus lines connecting same stations are drawn next to each other distinguished by line color. Walking edges are drawn in black when used for traveling. The interface allows the user to set a starting location and starting time, optional destinations of a planned journey, maximal tolerated walking time, and an estimated duration of changing bus lines. Automatic selection of the current time allows real-time updates. The results are shown in two different visualizations that allow either local correctness or global stress minimization for the distance mapping. Transitions are smoothly animated and the initial layout shows the geographic positions to retain the mental map [MELS95]. The schematic overview plan in the top left of Figure 3 provides a stable layout for easy station localization. Semantic zooming adaptively adds or removes station labels to make the visualizations less cluttered.

4. Routing

The travel time between two stations may vary throughout the day because of waiting periods and irregularities of the schedule. We modified the Dijkstra algorithm [Dij59] in order to determine routes on a graph with time dependent edge weights. Starting on a node at time t the optimal travel time lies in the interval $[t, t + w(t)]$ where w is the edge weight function. Using the partial periodicity of the schedule, bus lines can be split up into tours, which can be defined as an identification of a bus in its line. Later tours can be omitted because using a later bus of the same line never leads to a

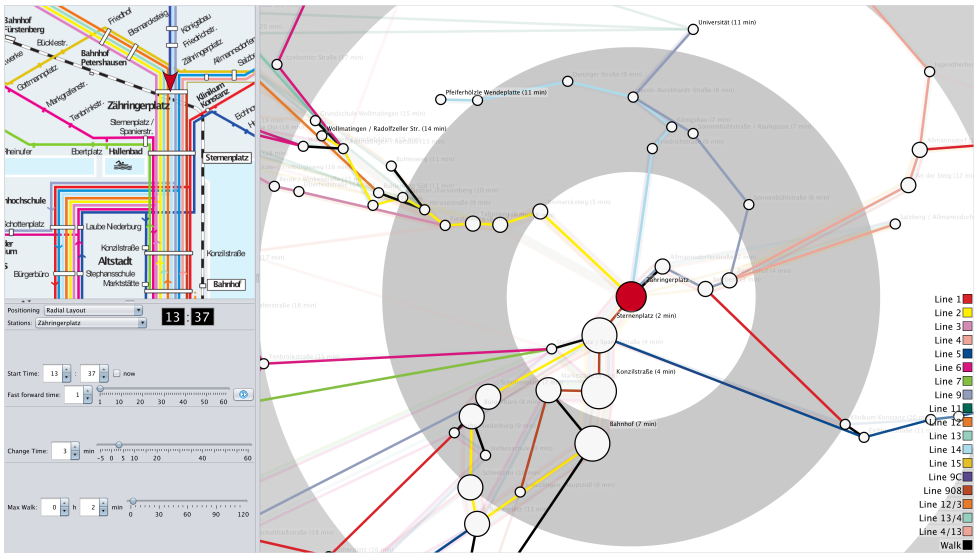


Figure 3: *The* BusVis *graphical user interface. Note the preservation of the geographic relations in the radial layout (center), as seen on the schematic overview (top left). The start location is indicated red and a color legend for bus lines is provided in the bottom right corner. The distance of a node to the origin is mapped to the travel time needed to reach the station.*

faster route. Changing bus lines requires time so that suboptimal intermediate steps can still lead to an optimal route. In Figure 2 the optimal route from A to C uses a suboptimal edge to B when assuming a change time of two minutes. Keeping track of the complete route from the starting location avoids this problem and also allows to reconstruct the final route. The worst-case runtime is similar to Dijkstra's algorithm although practically below, because most of the edges are not considered.

5. Visualization Techniques

We display the routing results in two different layouts. The radial layout (Figure 3) arranges stations concentric around the starting location while visual distance between any station and the center exactly reflects the fastest travel time to reach the station. To preserve the mental map we chose the direction to align with the geographic positioning. Concentric rings represent five-minute intervals and help the user compare distances in different directions. Resulting overlaps may occlude information and make the visualization less visually appealing. We resolved this by slightly changing the direction of overlapping nodes which does not affect visual distances. However, visible edges between two stations do not necessarily show the time needed to travel on them because only distances to the center are correct.

The stress-majorization layout reflects distances between all stations by finding an optimum in which every edge approximately has the length proportional to the time it takes to travel on it. This is an optimization problem which does not result in an exact solution. We used the *SMACOF* algorithm [dL77] to minimize the global stress function. Geographic positions of the stations are chosen as initial layout in order to roughly maintain the shape of the bus network. While travel times along edges are represented more accurately than in the radial layout, the mental map is not as well preserved. Overlaps cannot be resolved since node movements result in higher stress. However, stations connected by the same bus line can easily be identified as seen in Figure 4.

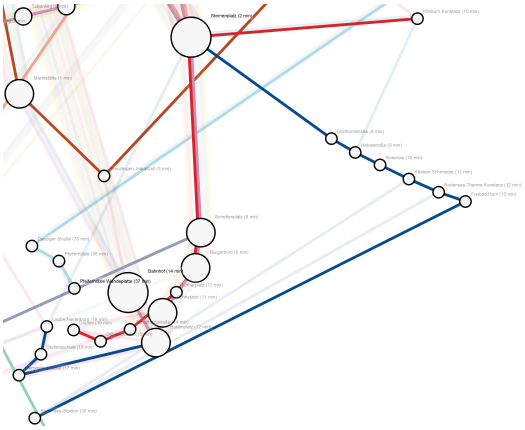


Figure 4: *Stress-majorization layout. The edge lengths between stations approximately represent travel time. Stations connected by the same line are lined up and can easily be identified (blue and red line).*

6. Use Cases

Our tool is useful for different user groups. Bus schedule planners can use the radial layout to get an overview of how well stations are connected throughout the day. To identify long waiting times when changing bus lines on a route, the stress-majorization layout can be used. Also frequent walk suggestions or outlier stations in one of the layouts indicate bad reachability.

The system can also be used to easily compare travel times. For example in the city of Konstanz, the bus stations *Universität* and *Egg* are geographically very close despite having no connecting street and therefore no bus connection. This raises the question when it is rewarding to chose one destination over the other. The user can simply choose the station that is closer to the center in the radial layout. Figure 5 shows that the optimal choice can change throughout the day. A similar scenario would be to find the fastest reachable restaurant for dinner from the current position. The corresponding stations could be integrated into the system. Another task is for citizens to find a flat with fast access to basic needs like grocery stores, doctors, and workplaces. The possibility to experiment with different times enables the user to check how the reachability of those places changes throughout the day.

A physical display at stations is thinkable as public time table. This display would use the radial layout with the station as starting point and the current time to help travelers at the station plan their routes, find out the arrival time, and get an estimation of needed waiting times.

7. Conclusion

We presented a novel visualization tool that allows the user to plan bus journeys within a city by exploring the shortest routes in the transportation network at a given starting time. The user can navigate the visualization and modify parameters with instant feedback via an interactive interface. Two graph layout techniques are integrated to visualize routing query results. The radial layout is especially useful at transit stations, where the starting point of the layout is the transit station itself. The tool is easily adaptable to support journey planning in other cities and networks with different means of transportation, only the data at the back-end has to be exchanged. We demonstrated the usefulness of the tool by a number of use cases that involve different tasks and user groups. As future work we want to show applicability of our approach on larger, more complex data sets.

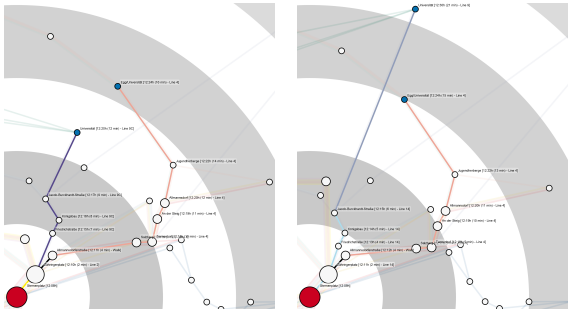


Figure 5: *Routes from* Sternenplatz *(big red dot) to* Universität *and* Egg *(small blue dots) at different times. On the left* Universität *(blue line) is closer than* Egg *(orange line), vice versa on the right.*

The authors thank Feeras Al-Masoudi for his support.

References

[Bar80] BARTRAM D.: Comprehending spatial information: The relative efficiency of different methods of presenting information about bus routes. *Journal of Applied Psychology 65*, 1 (1980). 96

[BBDZ08] BÖTTGER J., BRANDES U., DEUSSEN O., ZIEZOLD H.: Map Warping for the Annotation of Metro Maps. *IEEE Computer Graphics and Applications 28*, 5 (2008), 56–65. 96

[BBZ10] BOGEN S., BRANDES U., ZIEZOLD H.: Visual Navigation with Schematic Maps. In *Visual Information Communication*. Springer US, 2010, pp. 65–84. 96

[Car12] CARDEN T.: Travel Time Tube Map. http://www.tom-carden.co.uk/p5/tube_map_travel_times/applet/, August 2012. 96

[Dij59] DIJKSTRA E. W.: A Note on Two Problems in Connexion with Graphs. *Numerische Mathematik 1* (1959), 269–271. 96

[dL77] DE LEEUW J.: Applications of Convex Analysis to Multidimensional Scaling. In *Recent Developments in Statistics*. North Holland Publishing Company, Amsterdam, 1977, pp. 133–146. 97

[Gar94] GARLAND K.: *Mr. Beck's Underground Map*. Capital Transport Publishing, 1994. 96

[GFC04] GHONIEM M., FEKETE J.-D., CASTAGLIOLA P.: A Comparison of the Readability of Graphs Using Node-Link and Matrix-Based Representations. In *Proceedings of the IEEE Symposium on Information Visualization* (2004), INFOVIS '04. 96

[Hoa08] HOAR R.: Visualizing Transit Through a Web Based Geographic Information System. In *World Academy of Science, Engineering and Technology* (2008), vol. 22, pp. 180–185. 96

[Lon] Transport for London – Journey Planner. http://journeyplanner.tfl.gov.uk/. 95

[MELS95] MISUE K., EADES P., LAI W., SUGIYAMA K.: Layout Adjustment and the Mental Map. *Journal of Visual Languages & Computing 6*, 2 (1995), 183 – 210. 96

[NW11] NOLLENBURG M., WOLFF A.: Drawing and Labeling High-Quality Metro Maps by Mixed-Integer Programming. *IEEE Transactions on Visualization and Computer Graphics 17*, 5 (May 2011), 626–641. 96

[PTV] Public Transport Victoria – Journey Planner. http://jp.ptv.vic.gov.au. 95

[Rai] Rail Europe – Rail travel planner Europe. http://www.raileurope-world.com/. 95

[SRMOW11] STOTT J., RODGERS P., MARTINEZ-OVANDO J. C., WALKER S. G.: Automatic Metro Map Layout Using Multicriteria Optimization. *IEEE Transactions on Visualization and Computer Graphics 17* (2011), 101–114. 96

[WC11] WANG Y.-S., CHI M.-T.: Focus+Context Metro Maps. *IEEE Transactions on Visualization and Computer Graphics 17* (2011), 2528–2535. 96