

Robyn Taylor  
Pierre Boulanger  
Antonio Krüger  
Patrick Olivier (Eds.)

LNCS 6133

# Smart Graphics

10th International Symposium on Smart Graphics, SG 2010  
Banff, Canada, June 2010  
Proceedings



 Springer

*Commenced Publication in 1973*

Founding and Former Series Editors:

Gerhard Goos, Juris Hartmanis, and Jan van Leeuwen

## Editorial Board

David Hutchison

*Lancaster University, UK*

Takeo Kanade

*Carnegie Mellon University, Pittsburgh, PA, USA*

Josef Kittler

*University of Surrey, Guildford, UK*

Jon M. Kleinberg

*Cornell University, Ithaca, NY, USA*

Alfred Kobsa

*University of California, Irvine, CA, USA*

Friedemann Mattern

*ETH Zurich, Switzerland*

John C. Mitchell

*Stanford University, CA, USA*

Moni Naor

*Weizmann Institute of Science, Rehovot, Israel*

Oscar Nierstrasz

*University of Bern, Switzerland*

C. Pandu Rangan

*Indian Institute of Technology, Madras, India*

Bernhard Steffen

*TU Dortmund University, Germany*

Madhu Sudan

*Microsoft Research, Cambridge, MA, USA*

Demetri Terzopoulos

*University of California, Los Angeles, CA, USA*

Doug Tygar

*University of California, Berkeley, CA, USA*

Gerhard Weikum

*Max-Planck Institute of Computer Science, Saarbruecken, Germany*

Robyn Taylor Pierre Boulanger  
Antonio Krüger Patrick Olivier (Eds.)

# Smart Graphics

10th International Symposium, SG 2010  
Banff, Canada, June 24-26, 2010  
Proceedings

Volume Editors

Robyn Taylor  
Pierre Boulanger  
University of Alberta  
Advanced Man-Machine Interface Laboratory, Department of Computing Science  
Edmonton, Alberta, Canada  
E-mail: {robyn, pierreb}@cs.ualberta.ca

Antonio Krüger  
German Research Center for Artificial Intelligence (DFKI)  
Saarbrücken, Germany  
E-mail: krueger@dfki.de

Patrick Olivier  
University of Newcastle upon Tyne,  
School of Computing Science, Culture Lab  
Newcastle upon Tyne, United Kingdom  
E-mail: p.l.olivier@ncl.ac.uk

Library of Congress Control Number: 2010927761

CR Subject Classification (1998): I.4, H.3-5, I.3.7, I.2, I.5

LNCS Sublibrary: SL 6 – Image Processing, Computer Vision, Pattern Recognition,  
and Graphics

ISSN 0302-9743  
ISBN-10 3-642-13543-9 Springer Berlin Heidelberg New York  
ISBN-13 978-3-642-13543-9 Springer Berlin Heidelberg New York

This work is subject to copyright. All rights are reserved, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, re-use of illustrations, recitation, broadcasting, reproduction on microfilms or in any other way, and storage in data banks. Duplication of this publication or parts thereof is permitted only under the provisions of the German Copyright Law of September 9, 1965, in its current version, and permission for use must always be obtained from Springer. Violations are liable to prosecution under the German Copyright Law.

[springer.com](http://springer.com)

© Springer-Verlag Berlin Heidelberg 2010  
Printed in Germany

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India  
Printed on acid-free paper 06/3180

# Preface

The 10th International Symposium on Smart Graphics was held during June 24-26, 2010 in Banff, Alberta, Canada. Smart Graphics brings together researchers and practitioners from the fields of computer graphics, artificial intelligence, cognitive science, graphic design and fine art.

In response to the overwhelming success of Smart Graphics' conception as a AAAI Spring Symposium in 2000, its organizers decided to turn it into a self-contained event. Since then, annual Smart Graphics symposia have been held internationally, in such diverse locations as Heidelberg, Kyoto, Vancouver, Rennes, and most recently in 2009 in Salamanca, Spain, where Smart Graphics was intertwined with the 5th International Arts Festival of Castilla y Leon. Connecting last year's symposium to Salamanca's arts festival intensified the creative flavor of the Smart Graphics sessions, and sparked lively debate and discussion during the course of the symposium, inspiring exploration of the interconnected relationships between the disciplines of arts and sciences. Building upon the momentum begun last year, it was suggested that the 2010 symposium would place specific emphasis on the integration of scientific research and digital media technology into the design of aesthetic experience and artistic practice.

In 2010 we had the opportunity to collaborate with the Banff New Media Institute to present the symposium at Canada's renowned Banff Centre, a facility known worldwide for its creative excellence in arts and culture research, education and practice.

Banff New Media Institute's mandate is to bring together members of the artistic, scientific, and technological communities in order to encourage collaboration, inquiry and the production of interdisciplinary work. Their research shares the vision of Smart Graphics and provided a perfect venue to encourage the Smart Graphics community to explore and share ideas using both creative and analytical thought.

We were pleased to receive a high standard of paper submissions to the symposium, with a range of topics including virtual reality and simulation, sketch-based interfaces, visual analytics, and camera planning. The common thread running through this diverse collection of research was that the Smart Graphics community has framed their investigations in a human-centered way, presenting content that engages the user, effectively supports human cognition, communication, and collaboration and is aesthetically satisfying.

In addition to the paper presentations, the 2010 symposium featured an expanded arts track, presenting a public exhibition of performance, visual art, and installation, showcasing the work of an international community of artists who incorporate technological innovation and research into their creative practice. We were pleased to invite notable guest speakers John Bowers and Maria Lantin to join us in Banff this year. Both these individuals espouse methodologies that

are truly interdisciplinary in nature, drawing from both the art and science worlds to frame their research practices in a human-centered way, making them ideal candidates to inspire and stimulate discussion amongst the Smart Graphics community.

We would like to thank all the authors, reviewers, speakers, exhibitors and conference facility staff for bringing this event together and reuniting the Smart Graphics community for another year of refreshing, cross-disciplinary perspectives on practice and research.

June 2010

Robyn Taylor  
Patrick Olivier  
Antonio Krüger  
Pierre Boulanger

# Organization

## Organizing Committee

Robyn Taylor	University of Alberta, Canada
Pierre Boulanger	University of Alberta, Canada
Andreas Butz	University of Munich, Germany
Marc Christie	IRISA/INRIA Rennes, France
Brian Fisher	Simon Fraser University, Canada
Antonio Krüger	German Research Center for Artificial Intelligence, Germany
Patrick Olivier	University of Newcastle Upon Tyne, UK

## Program Committee

Elisabeth André	University of Augsburg, Germany
William Barres	Millsaps College, USA
Marc Cavazza	University of Teesside, UK
Luca Chittaro	University of Udine, Italy
Sara Diamond	Ontario College of Art and Design, Canada
David S. Ebert	Purdue University, USA
Steven Feiner	Columbia University, USA
Knut Hartmann	Flensburg University of Applied Sciences, Germany
Phil Heslop	Newcastle University, UK
Hiroshi Hosobe	Tokyo National Institute of Informatics, Japan
Christian Jacquemin	LIMSI/CNRS, France
Tsvi Kuflik	University of Haifa, Israel
Rainer Malaka	University of Bremen, Germany
Shigeru Owada	Sony CSL
W. Bradford Paley	Digital Image Design
Bernhard Preim	University of Magdeburg, Germany
Thomas Rist	University of Applied Sciences, Augsburg, Germany
Mateu Sbert	University of Girona, Spain
John Shearer	Newcastle University, UK
Shigeo Takahashi	University of Tokyo, Japan
Roberto Therón	University of Salamanca, Spain
Lucia Terrenghi	Vodafone Group R&D
Massimo Zancanaro	ITC-irst Trento, Italy

## Reviewers

Ajaj, Rami	Krüger Antonio
André, Elisabeth	Kuflik, Tsvi
Bares, William	Lioret, Alain
Bartindale, Tom	Malaka, Rainer
Bischof, Walter	Olivier, Patrick
Boulanger, Pierre	Owada, Shigeru
Buchanan, John	Preim, Bernard
Butz, Andreas	Ranon, Roberto
Cavazza, Marc	Rist, Thomas
Chittaro, Luca	Sbert, Mateu
Christie, Marc	Schöning, Johannes
Diamond, Sara	Shearer, John
Ebert, David	Takahashi, Shigeo
Feiner, Steven	Taylor, Robyn
Fisher, Brian	Terrenghi, Lucia
Heslop, Phil	Therón, Roberto
Hosobe, Hiroshi	
Jacquemin, Christian	

## Supporting Institutions

The Smart Graphics Symposium 2010 was organized and sponsored by the Advanced Man-Machine Interface Laboratory at the University of Alberta and the Banff New Media Institute at the Banff Centre. Additional support was provided by Newcastle University. The Symposium was held in cooperation with the Eurographics Association (EG), the American Association for Artificial Intelligence (AAAI), ACM SIGGRAPH, ACM SIGCHI and ACM SIGART.

# Table of Contents

## Sketching 1

Sketch Based Volumetric Clouds .....	1
<i>Marc Stiver, Andrew Baker, Adam Runions, and Faramarz Samavati</i>	
Applying Mathematical Sketching to Sketch-Based Physics Tutoring Software .....	13
<i>Salman Cheema and Joseph J. LaViola Jr.</i>	
A Sketch-and-Grow Interface for Botanical Tree Modeling .....	25
<i>Nordin Zakaria</i>	
Animated Volume Texture Mapping for Complex Scene Generation and Editing .....	33
<i>Rui Shen</i>	

## Physics and Simulation

A Novel Three-dimensional Collaborative Online Platform for Bio-molecular Modeling .....	44
<i>Kugamoorthy Gajananan, Arturo Nakasone, Andreas Hildebrandt, and Helmut Prendinger</i>	
The Effects of Finger-Walking in Place (FWIP) for Spatial Knowledge Acquisition in Virtual Environments .....	56
<i>Ji-Sun Kim, Denis Gračanin, Krešimir Matković, and Francis Quek</i>	
Interactive Design and Simulation of Net Sculptures .....	68
<i>Grigore D. Pintilie, Peter Heppel, and Janet Echelman</i>	
A Cross-Platform Framework for Physics-Based Collaborative Augmented Reality .....	80
<i>Damon Shing-Min Liu, Chun-Hao Yung, and Cheng-Hsuan Chung</i>	

## Camera Planning 1

Accurately Measuring the Satisfaction of Visual Properties in Virtual Camera Control .....	91
<i>Roberto Ranon, Marc Christie, and Tommaso Urli</i>	

VEX-CMS: A Tool to Design Virtual Exhibitions and Walkthroughs That Integrates Automatic Camera Control Capabilities .....	103
<i>Luca Chittaro, Lucio Ieronutti, and Roberto Ranon</i>	

Automatic Speed Graph Generation for Predefined Camera Paths .....	115
<i>Ferran Argelaguet and Carlos Andujar</i>	

## Sketching 2

Example-Based Automatic Font Generation .....	127
<i>Rapee Suveeranont and Takeo Igarashi</i>	

Sketch-Based Interfaces: Exploiting Spatio-temporal Context for Automatic Stroke Grouping .....	139
<i>Lutz Dickmann, Tobias Lensing, Robert Porzel, Rainer Malaka, and Christoph Lischka</i>	

A Method for Reconstructing Sketched Polyhedral Shapes with Rounds and Fillets .....	152
<i>Pedro Company, Peter Ashley, and Clifford Varley</i>	

Pressure-based 3D Curve Drawing .....	156
<i>Chan-Yet Lai and Nordin Zakaria</i>	

## Imaging

An Interactive Design System for Water Flow Stains on Outdoor Images .....	160
<i>Yuki Endo, Yoshihiro Kanamori, Jun Mitani, and Yukio Fukui</i>	

Automated Hedcut Illustration Using Isophotes .....	172
<i>SungYe Kim, Insoo Woo, Ross Maciejewski, and David S. Ebert</i>	

## Visual Analytics

Articulate: A Semi-automated Model for Translating Natural Language Queries into Meaningful Visualizations .....	184
<i>Yiwen Sun, Jason Leigh, Andrew Johnson, and Sangyoon Lee</i>	

Visual Analysis of Time-Motion in Basketball Games .....	196
<i>Roberto Therón and Laura Casares</i>	

Event Line View: Interactive Visual Analysis of Irregular Time-Dependent Data .....	208
<i>Krešimir Matković, Alan Lež, Denis Gračanin, Andreas Ammer, and Werner Purgathofer</i>	

## Camera Planning 2

Automatic Blending of Multiple Perspective Views for Aesthetic Composition .....	220
<i>Kairi Mashio, Kenichi Yoshida, Shigeo Takahashi, and Masato Okada</i>	
Focus and Context in Mixed Reality by Modulating First Order Salient Features .....	232
<i>Erick Mendez, Steven Feiner, and Dieter Schmalstieg</i>	
An Interactive Interface for Lighting-by-Example .....	244
<i>Hai Nam HA, Christophe Lino, Marc Christie, and Patrick Olivier</i>	

## Art

Stroking a Cymbidium.....	253
<i>Young-Mi Kim and Jong-Soo Choi</i>	
WAVO: An Interactive Ball to Express Light Waves with Wave Equation.....	257
<i>Kazushi Mukaiyama</i>	
OverWatch: Real-Time Narrative Visuals from Live Performance .....	261
<i>Guy Schofield, Rachel Casey, and Patrick Olivier</i>	
Sticking Point .....	265
<i>Tom Schofield</i>	
Phantasmagoria: Composing Interactive Content for the humanaquarium .....	269
<i>Robyn Taylor, Guy Schofield, John Shearer, Pierre Boulanger, Jayne Wallace, and Patrick Olivier</i>	
Self Portraits with Mandelbrot Genetics .....	273
<i>Jeffrey Ventrella</i>	
Smart Graphics - Art 101: Learning to Draw through Sketch Recognition .....	277
<i>Tracy Hammond, Manoj Prasad, and Daniel Dixon</i>	
ColourVision—Controlling Light Patterns through Postures .....	281
<i>Alexander Wiethoff and Andreas Butz</i>	
Interaction with a Virtual Character through Performance Based Animation .....	285
<i>Qiong Wu, Maryia Kazakevich, Robyn Taylor, and Pierre Boulanger</i>	
<b>Author Index .....</b>	<b>289</b>

# Sketch Based Volumetric Clouds

Marc Stiver, Andrew Baker, Adam Runions, and Faramarz Samavati

University of Calgary

**Abstract.** Like many natural phenomenon, clouds are often modeled using procedural methods, which may be difficult for an artist to control. In this paper, a freehand sketching system is proposed to control the modeling of volumetric clouds. Input sketches are used to generate a closed mesh, defining the initial cloud volume. Sketch analysis as well as the elevation at which the cloud is drawn is used to identify the cloud type and then generate a mesh with the appropriate characteristics for the determined cloud type. The cloud volume can then be edited using Boolean operations that allow for addition and removal of material from existing clouds. The proposed modeling system provides an intuitive framework for generating individual clouds and entire cloud fields, while maintaining the interactive rates necessitated by the sketch-based paradigm.

## 1 Introduction

The visual aspects of clouds add a great deal to interactive outdoor scenes. The inclusion of fully volumetric clouds through the use of particle engines adds even more to a scene by providing a realistic feel. However, the variable nature of clouds makes them very difficult and time consuming to model by conventional means. Thus, a faster and more intuitive interface for modeling clouds would be beneficial. In particular, a sketch-based approach is ideal. The words sketch-based tend to bring to mind some of the more familiar sketch interfaces that have been introduced; most of which address mesh based construction and manipulation. However, in some instances meshes may be inappropriate or insufficient to supply the information required to achieve a desired effect. Particle systems, for example, are frequently used to model



**Fig. 1.** Using the binary cutting tool the artist is quickly able to shape a cloud as they wish

objects and phenomena that meshes are unable to portray. Due to their volumetric nature, and the wide variety of shapes they exhibit, clouds are often represented using particle systems [7,9,11,11,13].

We propose a sketch-based particle placement system for the creation and manipulation of clouds. Using our interface artists can create clouds by drawing them directly into a 3D environment. An adjustable canvas is used to determine the initial depth of the cloud. To further the natural feel of the interface we allow the artist the ability to sketch multiple strokes during the cloud creation process. The input strokes are decomposed and analyzed to determine the type (and number) of convex sections the input sketch is composed of. This, in conjunction with the altitude at which the cloud is drawn, is used to determine a cloud-type. A mesh is then created using the artist's sketch and the pre-determined cloud-type. Finally, the mesh is filled with particles to create a fully volumetric cloud. To achieve a realtime method for filling the mesh, we hash the mesh into a 2D grid which is used to facilitate real time in/out testing for the particles. Once a cloud is created the artist then has the ability to alter the clouds through the use of simple Boolean operations such as cutting and adding to a selected cloud (see Fig. II).

The organization of the paper is as follows. In section 2 we summarize previous works related to volumetric clouds and sketch-based interfaces and provide a brief overview of cloud meteorology. Section 3 describes our method and gives an overview of the interface and the techniques that are used in the proposed cloud modeling system. Section 4 presents results, and section 5 gives our conclusion and potential future work.

## 2 Background and Previous Work

### 2.1 Volumetric Cloud Modeling

Volumetric cloud creation approaches can be separated into the categories of procedural and simulation based methods. Procedural based cloud generation provides a more viable solution for real time implementations and therefore we focus our discussion on procedural techniques.

Harris et al [4] uses a particle system to create clouds and then dynamically renders the clouds to imposters that are bill boarded to face the camera. This allows for a real time cloud environment. Visual artefacts sometimes occur when an object enters a cloud, these artefacts are reduced by using multiple layers of imposters. A similar approach is used in the system implemented by Microsoft Flight Simulator 2004; clouds at a distance are rendered to billboards in an octagonal ring and clouds near the camera are rendered using particles [16]. Groupings of cuboids are used to place the textured particles for the clouds. Both these systems work well in real-time; however neither system specifies a fast and interactive way to model clouds within their systems.

In contrast to these approaches Schopok et al, [14] demonstrate an effective cloud creation method using volume textures which are divided into layers with respect to the light vector and the view vector; spheroids are used in the

placement of the textures. Harris et al [5] demonstrates cloud dynamics by using volume textures in a similar system. These systems make use of advanced graphics hardware for improved performance and the methods used to place the textured particles are either slow for large quantities of clouds or done through simulation.

Cellular automata are frequently used in the rendering of dynamic clouds. Dobashi et al [2] uses voxels that correspond to cells of a cellular automata; each cell is set to either a 1 or a 0, and so can be expressed by Boolean operations. Liao et al [10] uses cellular automata to determine density distributions over time for each voxel within the simulation volume. These processes are generally slow and to our knowledge real time implementation is, at present, impossible on graphics hardware.

Dobashi et al [3] describe a method for cumuliform formation using fluid mechanics that takes into account physical processes to form the cloud to a user defined curve. This work is very related to our method, however fluid mechanics are not suitable for our purposes as in general they do not run in real time.

In a recent paper on cloud modeling [17], we see the creation of precisely specified cloud objects using sphere primitives. The artist is able to quickly create specific shapes using a sketch based interface. Specifically, the system employs an interface similar to that proposed by Igarashi et al. [7], which may not be ideal for clouds (this is discussed in greater detail in section 3.1). The work is on the creation of cumulus clouds and does not mention any other types of clouds or cloud scapes. The physical formation process of clouds is also not taken into account as they are more interested the creation of shapes and not the physical placement of particles.

## 2.2 Sketch Based Modeling

Teddy [7] is perhaps the most well known sketch based modeling system; it allows the creation of a surface by inflating regions defined by closed strokes. Strokes are inflated, using the chordal axis transform, so that portions of the mesh are elevated based on their distance from the stroke's chordal axis. This system creates models with a plush toy feel that are difficult to obtain using traditional interaction techniques.

Plushie [11] is another sketch based interface that creates a three-dimensional object from an artist's sketch by use of procedural modeling. The specific intent of Plushie however is the creation of stuffed toys and facilitates this by creating two-dimensional patches that can later be sewn together to create a plush toy.

Other procedural modeling systems include work done on more flexible objects such as clothing. Igarashi and Hughes [6] present a method of cloth manipulation where a two dimensional clothing model is placed onto a three-dimensional model, or body, through the use of artists strokes. The placement and manipulation of the cloth is controlled via interactive techniques such as surface dragging and pushpins.

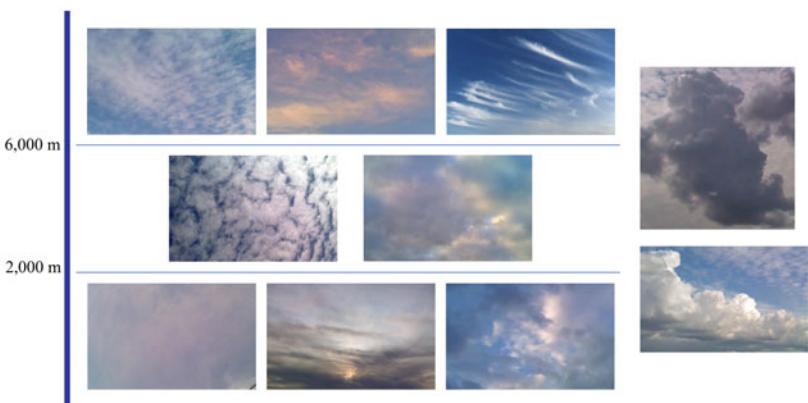
Other mesh creation algorithms include of those proposed by Cherlin et al., [1] specifically, rotational and cross sectional blending. Rotational blending defines a

parametric surface which extends surfaces of revolution. Two curves are blended together circularly about their local midpoints; the output created by this process can easily be converted to high quality meshes. Meshes generated by this technique describe certain objects well, however not all objects are suited to it (e.g. objects with variable cross-sections). Cross sectional blending permits the cross-section of the rotational blend to be modified from a circle to an arbitrary curve.

As was stated before, for the most part sketch-based systems focus on meshes as they are the most common medium for geometric modeling. Our system relies on this connection between sketch-based system and meshes for the particle placement phase. For this portion of our interface we borrow the techniques presented by Cherlin et al (rotational and cross-sectional blending), as we have found them to produce suitable meshes for most cloud types.

### 2.3 Cloud Meteorology

Elevation plays a key role in the formation of clouds in nature and we have found it very useful for differentiating between cloud types within our interface. High-level clouds are formed above 6,000 meters; these clouds are composed primarily of ice crystals and are typically thin when compared to other clouds. Mid-Level generally form between 2000 meters and 6000 meters, whereas low-level clouds tend to form below 2000 meters; these clouds are composed mainly of water droplets and generally have more vertical development than high-level clouds (See Fig. 2). Clouds that span multiple levels such as cumulonimbus also exist however we have yet to integrate them into our interface. Adding to the cloud classifications, we have cumulus clouds which are associated with convection, and stratus clouds that result from forced lifting of air. Cumulus are generally more full and fluffy where as stratus tend to be more thin and wispy. Both cloud types can occur at any of listed altitudes.



**Fig. 2.** The 10 different cloud types; From left to right; Top Row: Cirrocumulus, Cirrostratus, Cirrus; Middle Row: Altocumulus, Altostratus; Bottom Row: Stratus, Nimbostratus, Stratocumulus; Side (Top to bottom): Cumulonimbus, Cumulus

### 3 Methodology

In this section we outline the cloud creation and manipulation processes in our system, along with some of the features of our interface. A mesh is created from the artist's input sketch, height of the sketch and drawing style (See figure 3 for the cloud creation process). The mesh is then filled with particles using an accelerator grid.

#### 3.1 Creation of a Cloud

Clouds come in a multitude of shapes and sizes, as such creating all the different types of cloud using one methodology is difficult. In order to give our interface the ability to generate a diverse set of clouds, the creation of meshes and selection of textures changes based on how and where the artist draws their strokes. The elevations combined with the general features of the strokes are analyzed. By using three separate elevation levels and two sub-types we are able to define six of the ten recognized cloud types [8].

Two horizontal semi-transparent planes are used to allow the artist to see the level at which they are sketching the cloud (Fig. 4 displays the interface). These elevations are used to determine the clouds base-type; a low-level base-type is created by sketches that originate below both planes; sketches drawn above the lower plane will create a mid-level base-type and sketches above both planes will create a high-level base-type. The sub-type of the cloud is determined by analyzing the users input sketch and can roughly be paralleled with a subtype of cumulus or stratus.

**The Canvas.** As with most sketch-based interfaces we must address the problem of inferring a three dimensional object from two-dimensions. The inherent

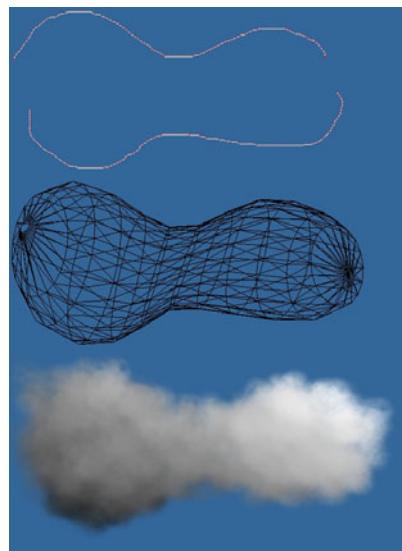


Fig. 3. The cloud creation process

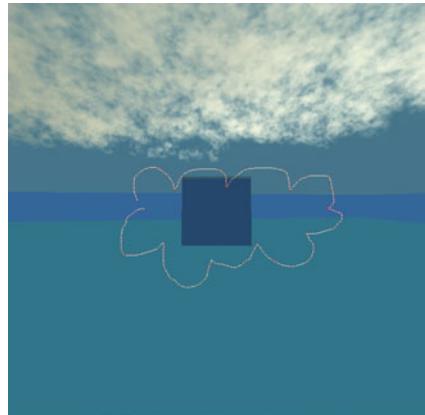


Fig. 4. The canvas is seen at the center of the image, the two horizontal planes specifying cloud height are also shown

The sub-type of the cloud is determined by analyzing the users input sketch and can roughly be paralleled with a subtype of cumulus or stratus.

ambiguity in the process makes it difficult to determine where the artist intends to place the object. In order to reduce this ambiguity, and enable the artist to place clouds more precisely, a sketch plane has been added to allow the artist to specify depth of placement. The sketch plane is designed to be in view at all times. When the artist sketches, the cloud is created at the depth of the canvas.

**Mesh Creation.** As with the placement of the cloud itself, inferring depth for the mesh is a difficult task. There are a multitude of different meshes that can be made and for the most part every sketch interface yields different outputs for the same input strokes. The type of mesh used in our application is determined by using the cloud base-type and the cloud sub-type. A cloud in our interface has a Cumulus or a Stratus sub-type as well as a high, mid or low level base-type.

Two separate mesh interpretations are used based on these given parameters; we have borrowed the rotational and cross-sectional blending techniques presented by Cherlin et al [1]. Rotational blending is used for mid and low-level clouds in our application, whereas cross-sectional blending is used for the high-level clouds. A rotational blend has a similar surface

to that of a surface of revolution; however a surface of revolution rotates about a line axis creating a uniform surface, whereas rotational blend rotates about a curved axis creating a blended surface between two curves. Any type of mesh creation can be used with our interface, however we have chosen the rotational (or circular) blending because the meshes that are created by it are similar to many low level and mid level clouds. The circular blend is a good interpretation of a cloud and we have found it to be more visually appealing than other mesh creation methods (such as inflation [7], see Fig. 5 for a comparison).

Cross-sectional blending is used for the high level clouds in our system. Similar to the rotational blend, a cross-sectional blend interpolates between two curves to form a surface. However, where the cross-section of the rotational blend would parameterize a circle, the cross-sectional blend uses predefined functions to determine cross-sections. This allows us to obtain a mesh that is thin, long and flat which turns out to be ideal for placing sparse sections of clouds as seen in



**Fig. 5.** The top two images display the use of Teddy style inflation within our application. Top left is the same orientation as the sketch was drawn and top right is the above view of the created cloud. The bottom two images use the same camera orientations for a similar sketch using the rotational blending algorithm. Teddy's mesh creation tends to produce a more squashed mesh when viewed perpendicular to the canvas; our method uses a more uniform interpretation which we have found to be more appropriate for clouds.

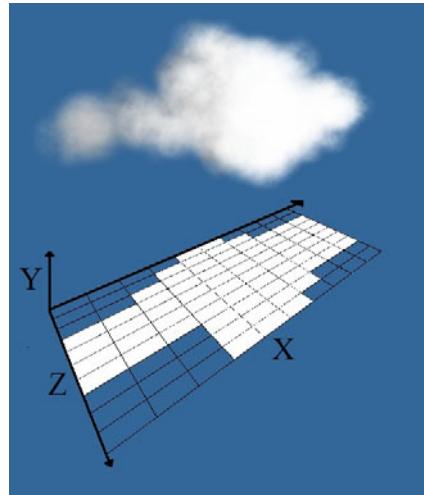
figure 8. Additional cross-sections can be specified by the artist, thereby providing more control over the cloud volumes.

### Filling The Mesh With Particles.

The cloud mesh is then filled with particles. These particles naturally create noise on the boundaries of the cloud which would otherwise require the use of a very high resolution mesh. An axis aligned bounding box is used in the initial placement of potential particles. The size of the particles used is based on the cloud type determined at the mesh creation stage. The bounding box is filled randomly with points; the density of these points is determined by the volume of the bounding box and the size of the particles. Each particle is then tested to determine whether it is within the mesh. Since the mesh may not be uniform a ray casting algorithm can be used for this purpose [15]. However, this approach does not allow for interactive rates to be achieved, which we require for our interface. Therefore we have found a more efficient approach.

Determining the particles location can ultimately be reduced to a series of triangle in and out tests by use of an accelerator grid. During the mesh creation process the triangles of the mesh are hashed into a two-dimensional grid. To make use of the accelerator grid the vertices, the maximum and minimum height, and the normal of each triangle are calculated and stored in the appropriate grid cell. The proper location is found for each triangle by projecting it onto the accelerator grid (see Fig. 6).

The position of each particle is then hashed into the accelerator grid and checked against all the triangles registered in that location. For inside/outside tests, a ray is projected from each particle, along the axis perpendicular to the accelerator grid. If there are an odd number of hits we conclude that particle is inside the mesh, otherwise the particle is removed. This optimization makes the filling process very quick, however since we reduce the problem to two dimensions some ambiguities arise in the third dimension. In particular, a particle may be above the minimum height of a triangle and below the maximum height, as the triangles exist in three dimensions. In this situation we do not know if the particle is inside the mesh or not, as a hit will be registered either way. This situation is resolved by comparing the normal of the triangle with the vector between the triangle and the particle in question as follows:



**Fig. 6.** The perimeter of the cloud is projected onto the accelerator grid to demonstrate where the triangles of the mesh would be hashed

$$\text{Orientation} = \begin{cases} \text{Inside}, & \text{if } M_n \cdot A_p \geq 0 \\ \text{Outside}, & \text{if } M_n \cdot A_p < 0 \end{cases} \quad (1)$$

where  $M_n$  is the normal of the triangle  $n$  and  $A_p$  is the normalized vector from the particle  $p$  to the center of the triangle  $n$ . By using this calculation ray casting is avoided entirely which permits the algorithm to be run more efficiently.

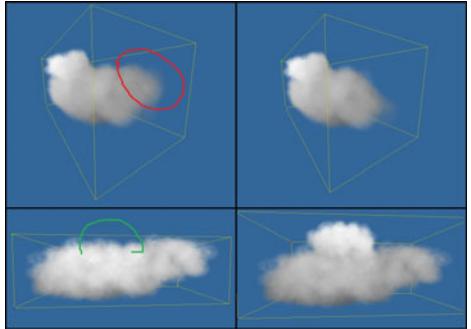
### 3.2 Editing

Since our end result is a grouping of particles we also define a set of particle based editing operations. The nature of a particle based structure allows us to implement some basic Boolean operations in order to manipulate the overall appearance of the cloud. Sketch-based techniques are once again used for sculpting and extending the cloud models.

**Sculpting.** We use a closed sketched curve (see Fig. 7 top row) for cutting away sections of the cloud. After a closed stroke has been drawn each particle within the selected cloud is temporarily projected to the screen and particles within the stroke are removed. The technique which we use for efficiently filling the mesh with particles can be easily extended to cut away particles in two-dimensions.

**Extending.** To add to the cloud in a given direction an extending operation has been provided to the artist; the interface determines that an extend operation is requested if an open curve is drawn on the cloud. A new mesh is then generated from the artist's edit stroke based on the selected clouds type. This new mesh is then filled to create an addition to the pre-existing cloud. By adding to the cloud in this manner there is the possibility of creating an area of the cloud that is very dense with particles. This may cause issues, as overlapping particles may cause irregularities. To avoid this overlap, each particle of the pre-existing cloud is checked to see if it resides inside the new mesh and is deleted if this is the case. The new mesh is then filled using the accelerator grid.

The initial depth of the addition is determined by temporarily projecting the particles to the screen and comparing the start of the sketch to these projected particles. The closest particle to the start and the end of the sketch are used as depth markers for the new mesh.



**Fig. 7.** The editing operations; Top Left: artist sketching a closed edit stroke. Bottom Left: artist sketching an open curve. The results of these operations are displayed in the top-right and bottom right respectively.

## 4 Results

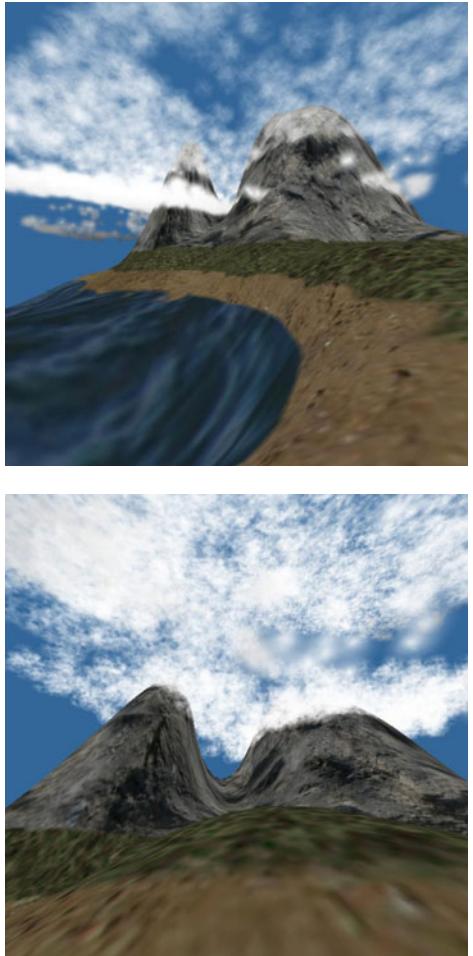
Figure 3 displays the cloud creation process from start to finish, the artist uses the canvas to select a proper location/altitude, and then proceeds to sketch their cloud. The cloud is then lit and displayed for the artist to observe and edit, all at interactive rates. The mesh used for filling is never made known to the artist.

In order to demonstrate the effectiveness of our interface we have implemented a simple one-pass light scattering algorithm. Although better lighting algorithms have been proposed [12][10][4][5][16], we have found this simplified lighting model produces reasonable results while permitting interactive frame-rates.

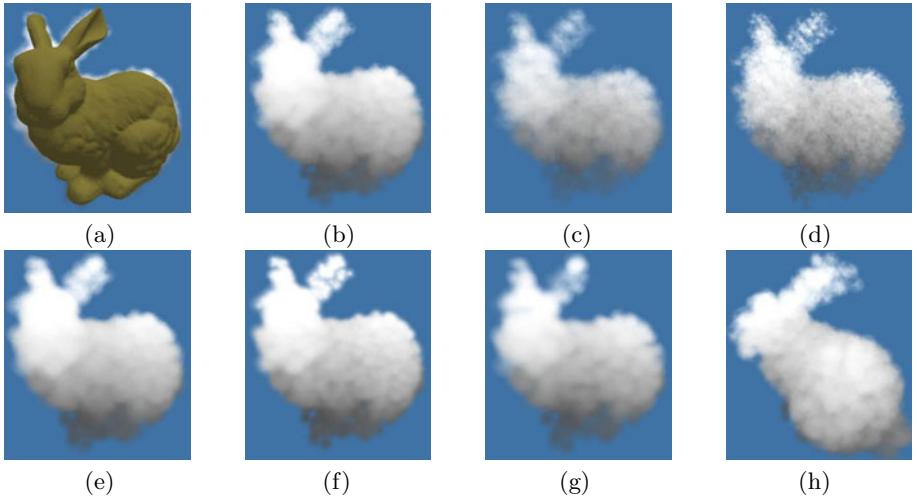
Figure 8 shows high level clouds created by our interface in just a few minutes. These high level clouds are created in our system using a flat and wide cross-sectional blending surface with larger, more spaced out particles. To our knowledge these types of clouds have not been attempted by any other sketch-based interface. Complex scenes such as these can be created without having to rigorously place shapes or containers.

Different particle types, based on the sub-types of the cloud, can be seen in figure 9 which also shows our technique being used on a pre-defined mesh. This allows complex forms to be used as the starting point for interactive editing (see Fig. 1).

All our algorithms run in real time on a 2GHz computer for particle groupings of less than 10,000 particles. Cloud edges can be improved by increasing the mesh resolution, however we use low resolution to facilitate speed when adding particles to the cloud. Since the mesh is used as a placement tool, we find that this sacrifice in accuracy is acceptable as the particles provide the variation that an artist may be trying to achieve (see Fig. 3). However, if more accurate clouds with respect to the sketch are required a one-time sacrifice in speed when filling the mesh would also be acceptable. The placement algorithm starts to slowdown



**Fig. 8.** Result created by our interface in a couple minutes, mostly high level clouds are displayed



**Fig. 9.** (a) The Stanford bunny that has been filled with particles using our algorithm. (b-g) Different textures are used on the same particles to achieve different effects. (h) A rotated view of the formed cloud.

when the number of particles being placed is relatively large compared to the processing speed of the computer (we noticed the slowdown at around 10,000-20,000 particles on a 2.0GHZ computer). A reduction in this slow down may be accomplished by fine tuning the accelerator grid for more complex operations.

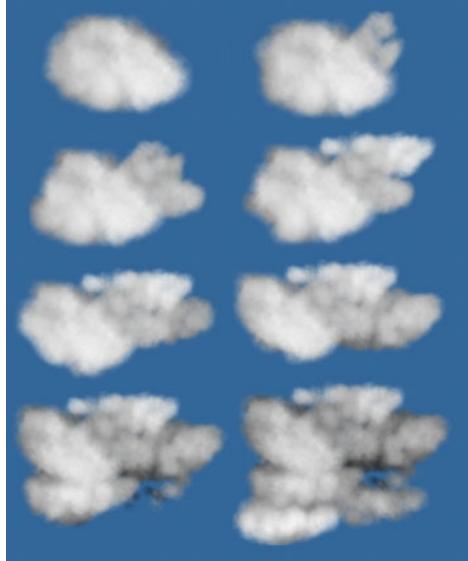
## 5 Conclusion

In this paper we have introduced a novel sketch based cloud modeling interface by use of particle placement and manipulation. The user draws a sketch and a mesh is then created using a rotational blending or cross-sectional blending surface. The mesh is populated with particles using an accelerator grid in conjunction with a triangle in-out test. Using this method we are able to place particles for clouds in a more efficient manner than was previously possible.

Our system is able to take a wide range of sketches as inputs, which are then interpreted based on sketch stylization and altitude. Basic Boolean operations also allow clouds to be manipulated once they have been created and placed. Such tools, to our knowledge, have not been proposed previously in the field of cloud modeling. The interface proposed by Wither et al [17] demonstrates some similarity to our interface, however we have included a methodology to create several different cloud types at interactive rates; as well as the ability to quickly define cloud fields. The content artists in games or flight simulators may benefit from the speed and flexibility of our approach when creating or manipulating a cloudy scene.

It is also worth noting that all the results presented in this paper were obtained without hardware acceleration and in general the proposed method operates in real time for reasonable sized clouds.

In our system, we emphasize an artistic interface for the creation and manipulation of clouds. The ease of using the interface provides a simple and manual yet direct approach for reshaping and deforming the base cloud at different time points, thus providing a means for specifying the dynamics of clouds (See Fig. 10). However, a more intelligent system that supports the dynamics of cloud deformation by taking in to consideration the exact physical model may be a better approach. The challenge is how to provide a natural interface appropriate for artists to control this physical model and its parameters. Exploring sketch-based interfaces for interactive control of physically-based deformation of clouds is a topic for future work.



**Fig. 10.** A sequence of frames (left-to-right, top-to-bottom) manually created using our interface

## 6 Acknowledgments

We would like to thank Steve Longay for the use of his terrain in our images and Kipp Horel for helpful discussions regarding modifications to the user interface. We would also like to thank Gordon Richardson for the use of his cloud images. This research was supported in part by the National Science and Engineering Research Council of Canada and GRAND Network of Centre of Excellence of Canada.

## References

1. Cherlin, J.J., Samavati, F., Sousa, M.C., Jorge, J.A.: Sketch-based modeling with few strokes. In: SCGG 2005: Proceedings of the 21st spring conference on Computer graphics, pp. 137–145. ACM, New York (2005)
2. Dobashi, Y., Kaneda, K., Yamashita, H., Okita, T., Nishita, T.: A simple, efficient method for realistic animation of clouds. In: SIGGRAPH 2000: Proceedings of the 27th annual conference on Computer graphics and interactive techniques, pp. 19–28. ACM Press/Addison-Wesley Publishing Co., New York (2000)
3. Dobashi, Y., Kusumoto, K., Nishita, T., Yamamoto, T.: Feedback control of cumuliform cloud formation based on computational fluid dynamics. ACM Trans. Graph. 27(3), 1–8 (2008)

4. Harris, M.J., Lastra, A.: Real-time cloud rendering. In: Chalmers, A., Rhyne, T.-M. (eds.) EG 2001 Proceedings, vol. 20(3), pp. 76–84. Blackwell Publishing, Malden (2001)
5. Harris, M.J., Baxter, W.V., Scheuermann, T., Lastra, A.: Simulation of cloud dynamics on graphics hardware. In: HWWS 2003: Proceedings of the ACM SIGGRAPH/EUROGRAPHICS conference on Graphics hardware, pp. 92–101. Eurographics Association, Aire-la-Ville (2003)
6. Igarashi, T., Hughes, J.F.: Smooth meshes for sketch-based freeform modeling. In: I3D 2003: Proceedings of the 2003, symposium on Interactive 3D graphics, pp. 139–142. ACM, New York (2003)
7. Igarashi, T., Matsuoka, S., Tanaka, H.: Teddy: a sketching interface for 3d freeform design. In: SIGGRAPH 1999: Proceedings of the 26th annual conference on Computer graphics and interactive techniques, pp. 409–416. ACM Press/Addison-Wesley Publishing Co., New York (1999)
8. Kindersley, D.: Earth. Dorling Kindersley Ltd. (2003)
9. Levet, F., Granier, X.: Improved skeleton extraction and surface generation for sketch-based modeling. In: GI 2007: Proceedings of Graphics Interface 2007, pp. 27–33. ACM, New York (2007)
10. Liao, H.-S., Chuang, J.-H., Lin, C.-C.: Efficient rendering of dynamic clouds. In: VRCAI 2004: Proceedings of the 2004, ACM SIGGRAPH international conference on Virtual Reality continuum and its applications in industry, pp. 19–25. ACM, New York (2004)
11. Mori, Y., Igarashi, T.: Plushie: an interactive design system for plush toys. In: SIGGRAPH 2007: ACM SIGGRAPH 2007 papers, p. 45. ACM, New York (2007)
12. Nishita, T., Dobashi, Y., Nakamae, E.: Display of clouds taking into account multiple anisotropic scattering and sky light. In: SIGGRAPH 1996: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques, pp. 379–386. ACM, New York (1996)
13. Owada, S., Nielsen, F., Nakazawa, K., Igarashi, T.: A sketching interface for modeling the internal structures of 3d shapes. In: SIGGRAPH 2007: ACM SIGGRAPH 2007, courses, p. 38. ACM, New York (2007)
14. Schpok, J., Simons, J., Ebert, D.S., Hansen, C.: A real-time cloud modeling, rendering, and animation system. In: SCA 2003: Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation, pp. 160–166. Eurographics Association, Aire-la-Ville (2003)
15. Sealy, G., Novins, K.: Effective volume sampling of solid models using distance measures. In: CGI 1999: Proceedings of the International Conference on Computer Graphics, p. 12. IEEE Computer Society, Washington (1999)
16. Wang, N.: Realistic and fast cloud rendering. Journal of graphics tools 9(3), 21–40 (2004)
17. Wither, J., Bouthors, A., Cani, M.-P.: Rapid sketch modeling of clouds. In: Eurographics Workshop on Sketch-Based Interfaces and Modeling, SBM (2008)

# Applying Mathematical Sketching to Sketch-Based Physics Tutoring Software

Salman Cheema and Joseph J. LaViola Jr.

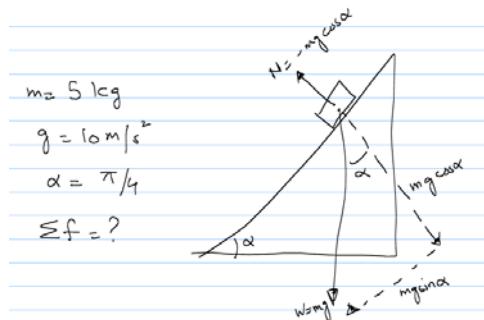
University of Central Florida  
School of EECS  
Orlando, FL 32816  
`{salmanc, jj1}@cs.ucf.edu`

**Abstract.** We present a prototype sketch-based physics tutoring system that combines mathematical sketching, an interaction paradigm that supports construction of dynamic illustrations using the association of handwritten mathematical expressions with drawings to govern animation behavior, and a custom physics engine. We highlight key features of our core system that focus on correcting approximately drawn sketches and maintaining the correspondence between imprecise handwritten drawings and precise mathematical specifications. We describe the behavior and design of the system in detail and finally, present two example scenarios illustrating its possible uses in an educational setting.

## 1 Introduction

Mathematics and physics teachers often use diagrams to present scientific concepts in visual form and as an initial step in problem solving [12][13]. In addition, to solve a physics or mathematics problem, students usually condense the information given in the problem statement by drawing a diagram. Students annotate these initial diagrams with mathematical expressions to make links between abstract concepts and concrete diagram elements. Often, the answer to a problem is numeric or symbolic and it is unclear how it would affect the drawing, requiring students to rely on their imagination to visualize the underlying concepts in action. To alleviate this issue, we postulate that providing meaningful animation of student-drawn sketches based on the associated mathematics used to solve a problem is required in order to impart better understanding of physics concepts. Such animation can also help students in intuitive verification of their answers to problems. Our research goal is to construct sketch-based intelligent tutoring systems for mathematics and physics that capture the essence of pen and paper diagrams while leveraging the power of computation by providing meaningful animation to enhance student learning.

We have developed a prototype physics tutoring system that is based on mathematical sketching [7], an interaction paradigm that supports construction of dynamic illustrations using the association of mathematical expressions with drawings to govern animation behavior. We took initial steps toward this goal with a proof-of-concept system in [3]. Our current prototype more fully integrates



**Fig. 1.** A typical inclined plane diagram drawn by a student

the advantages of mathematical sketching with an underlying physics engine and leverages domain knowledge more efficiently to help it infer how to animate a sketch at different levels of mathematical specification.

Hand drawn sketches are often approximate in nature. For example, Figure 1 represents an inclined plane problem. The triangle drawn by the user is approximately right-angled. Likewise, the inscribed angle  $\alpha$  is not exactly  $\pi/4$ . Such approximations are acceptable with pen and paper diagrams because the user relies on his imagination to see concepts in action. However, these inaccuracies can cause problems for the underlying physics engine in 3 due to ambiguity between the precise mathematical specifications and the imprecise drawings. A sketch-based tutoring system must be able to correct such diagrams (i.e., perform rectification) to allow students to sketch diagrams in a natural manner. In this paper, we describe the drawing rectification features that enable our tutoring system to deal with approximate sketches. We present a discussion of its feedback mechanisms that allow users to more directly observe changes in diagram state over time. We also highlight scenarios where understanding the user's intent based on physics domain knowledge allows us to present a better dynamic illustration.

## 2 Related Work

Pen-based systems for understanding hand-drawn sketches and providing visual feedback have been developed in the past for a number of specific goals. Alvarado [1] and Kara [5] have constructed systems for recognizing and animating diagrams in specific domains such as mechanical design and vibratory systems. Both utilize a simulation back-end to facilitate animation of problems understood in terms of basic primitive shapes. However, they are limited in scope because they do not allow the user to write mathematics to govern animation behavior. MathPad<sup>2</sup> [8] provides a better mechanism for animating sketches by allowing users to write down mathematics to describe aspects of animation. MathPad<sup>2</sup> is limited in its scope because users must specify all aspects of animation.

Existing physics tutoring systems, such as the Andes Physics Tutoring System [11], provide step by step guidance in problem solving, but rely on WIMP interfaces and do not provide users with the ability to write down their solutions and sketches as if using pen and paper. Newton's Pen [9] is a pen-based tutoring system that aids student learning in a few selected classes of physics problems. It does not permit free-form sketching of diagrams and does not include the ability to make associations in order to perform visual linkages between mathematics and diagram. PenProof [4] is a pen-based geometry theorem proving system that is able to recognize and understand the correspondence between geometric constructs and proof steps. This system is able to leverage the correspondence between geometry and proof steps to provide visual feedback to the user. At the same time, its limitation is that users are forced to disambiguate between geometry and proof steps explicitly and must rely on the system to associate proof steps with the figure.

### 3 System Features

Users sketch diagrams and write mathematics by means of a stylus on a tablet computer. Recognized diagrams can be annotated by making associations with mathematical expressions. Expressions used to make associations can either denote initial conditions or mathematical equations governing behavior.

The system parses a sketch when instructed to do so. In the past, we experimented with real-time sketch recognition but discarded it because it did not allow flexibility in terms of making/changing associations. On-demand parsing of the user's sketch is more natural because in a pen and paper scenario, students first sketch the diagram and then annotate it with mathematical expressions. The system is capable of recognizing the following diagram components: convex shapes(Circles,Polygons), springs, and wires. Realistic values are assigned to each component's attributes upon recognition based on the component's spatial characteristics. For example, a shape's initial mass is assigned proportional to its enclosed area. This approach allows us to animate sketches realistically even when the user does not specify any initial values. Assignment of initial attributes in this manner also allows the user interface to be flexible because it does not necessitate the input of all initial conditions by the user. Users can alter attributes by writing mathematical expressions to reflect proper initial conditions and associating them with diagram components. A lasso gesture is used to select expression(s). The association is completed by tapping the desired component. Existing associations can be viewed by hovering the stylus over a recognized component.

Mathematical expressions can either be constant expressions (e.g.  $f_1 = (0, 5)$ ) or equations (e.g.  $v_x = k\sin\theta$ ). Equations can denote scalar quantities (e.g. magnitude of normal force due to an incline) or vector quantities. Vector quantities can be specified in terms of x and y component equations or as a single vector equation. When x and y components are not specified, we use physics domain knowledge to determine if the variable in the left-hand side of the equation

denotes a vector or a scalar. Equations may be written in terms of numbers (e.g.  $f_1 = -0.5v_A$ ) or in terms of symbolic constants (e.g.  $f_1 = -\mu v$  where  $\mu = -0.5$ ). The system also uses domain knowledge to establish possible errors regarding associations and diagram components. This takes an understanding of physical properties of recognized diagram components. For example, it is illogical to associate an expression specifying a spring constant (e.g.  $k = 0.8$ ) with a shape or to associate a tension with a spring.

Writing and associating mathematics allows users to apply initial values and modify behavior as needed for a given problem and also provides intuitive feedback. Recognition errors or mistakes in equations can cause incorrect animation that conflicts with the user's intuition. A reset mode is provided that lets users debug and correct existing associations. This allows users to experiment with different initial values and gain better insight into the working of underlying concepts. We also allow a user to directly observe an attribute on a movable shape with respect to time. Observable attributes are velocity, acceleration, net force, all forces, x-displacement, y-displacement and position. When a particular attribute is under observation, most other information on the shape's animation is replaced by the selected attribute in bold. The user can also choose to view a real-time graph of a several selected attribute at any time.

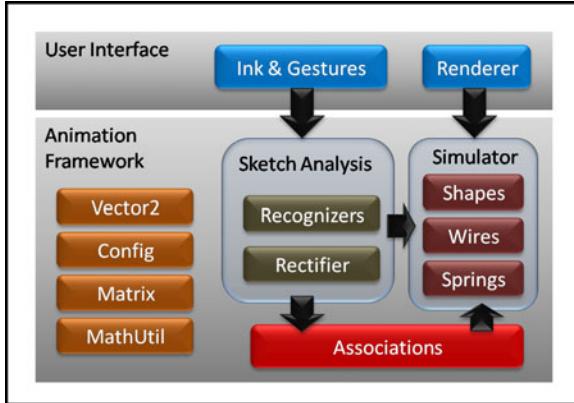
In keeping with our research goal to capture and enhance the essence of problem solving using pen and paper, the system has the capability to preserve existing associations along different system modes. If users want to alter part of an expression that is already associated with a diagram component, they do not have to make the association again. We believe that this approach minimizes unnecessary work on a user's part and lets him focus on the problem at hand rather than being encumbered by the user interface.

## 4 System Design

Figure 2 shows a high level view of the various components of the system. The following sections will highlight salient aspects of important components.

### 4.1 User Input

Sketches and written mathematics are acquired as digital ink strokes. A gesture recognition module recognizes three gestures: lasso, scribble-erase, and tap. The gesture set is limited thus to reduce interface complexity. The lasso gesture is used to select both ink and diagram components. Users can drag and reposition the selection on the screen. The scribble-erase gesture is used to erase ink and diagram components. If mathematical expression(s) are selected and a tap gesture is performed on a diagram component, an association is made. If no mathematics is selected, the tap gesture will cause a recognized shape to become immobile. The interface includes options to save/load ink. Instant recognition feedback for mathematical expressions is provided by means of an online mathematics recognizer [14]. Mathematical recognition errors can be corrected



**Fig. 2.** Overview of system components

by erasing the offending part of the expression by the scribble erase gesture and rewriting it. The feedback results are also used to make the association which improves performance by avoiding unnecessary work and minimizes recognition errors by using only correctly recognized mathematics.

## 4.2 Sketch Recognition

Sketch recognition is the first step in converting a user's sketch into components that will be animated by the underlying physics engine. All ink strokes are filtered and re-sampled prior to recognition to remove noise and ensure equal spacing of stroke points. We use a custom cusp detector to count the number of cusps in a stroke. A threshold is applied to the distance between the last and first cusp of the stroke to check for closure. A closed stroke is classified as a polygon if it has more than 2 cusps, otherwise it is possibly a circle. As a measure of circularity, we compute the standard deviation of the angle subtended at the stroke's centroid by each line segment in the stroke. Since the points are evenly spaced, the deviation will be extremely low for circular shapes. Strokes that have 2 cusps and are not closed are possibly springs or wires. A stroke is a spring if it has three or more self intersections. A line segment intersection test is employed for counting self intersections in the ink stroke. Wires are simply approximate straight lines. Any ink strokes that do not meet the above criteria are considered to be mathematics.

Every ink stroke is classified as either a diagram component or part of a mathematical expression. It is possible to mis-classify parts of a mathematical expression as diagram components (e.g. zeros as circles). We use the following rules to disambiguate between diagram components and mathematics. Shapes are disambiguated by ensuring that all convex components enclose a minimum area. For springs and wires, the end point of each must be attached to a shape. Hence recognition proceeds in the following order: Shapes, Springs, Wires, and lastly mathematics.

### 4.3 Associations

Associations between mathematics and diagram components are used to modify aspects of the animation. Associated mathematical expressions are either constants or equations. Constants can modify attributes of a diagram component such as mass, velocity, acceleration, etc. Constants are applied once, at the start of the animation. The values of associated constants can be changed in reset mode.

Equations can be similarly used to modify attributes of diagram components. During each animation step, existing equations are populated with their parameter values and evaluated to guide the animation. When evaluating equations with mathematical errors (e.g.  $f = -\mu v_A$  is valid only if  $\mu$  is specified), any unrecognized/unspecified parameters are assigned zero. This ultimately serves as visual feedback indicating an error in input being the cause of abnormal animation. In [3], equations could either use values associated with a single component or use global values (e.g. gravity and time). We have extended this scheme to include symbolic constants associated with any component in the diagram. Such a mechanism is necessary to solve the inclined plane problem discussed in Section 6.1.

### 4.4 Physics Engine

Recognized diagrams are animated by a custom 2D physics engine that is based on principles described in [10]. The default animation behavior of all diagram components depends on the standard equations of motion. Each shape's position is updated by computing net acceleration and performing numerical integration twice for position. Collision detection/resolution are performed after the position update. Earlier we used the method proposed by [2] to deal with resting/sliding contacts. But upon experimentation, we found that keeping track of every shape's net motion via exponential smoothing defined by

$$\begin{aligned} Motion_{frame} &= \|v\|^2 + \omega^2 \\ Motion_{net} &= (1 - \alpha)Motion_{net} + \alpha Motion_{frame} \end{aligned}$$

where  $\alpha = 0.8$  works fairly well and is faster.

After collision processing, a inference step is applied to deal with unspecified circumstances such as whether wires will break or not. Important attributes of shapes (e.g. velocity) are rendered as arrows. To provide visual feedback regarding how the magnitudes and directions of important quantities change over time, the length of rendered arrows are adjusted in proportion to the magnitude of the attribute represented.

Attributes of recognized components can be altered by making associations with written mathematics. It is also possible for a user to specify position or velocity update equations for a diagram component as replacement for standard equations of motion. If only velocity is associated, it is integrated once to update position. If the position is associated, no integration is required. Observable attributes (see Section 3) are computed in every frame even if there must be computed indirectly (e.g. compute velocity when position equations are associated).

Interesting situations arise when default behavior is overridden by custom behavior. Consider the following example. Shape A moving under standard equations of motion collides with shape B moving under user-defined position update equations. It is clear, that after the collision is resolved, the user-defined position update equation for B no longer applies. When such objects collide, the user defined equations are disabled and the physics engine takes over to resolve the collision. After the collision, the collided objects' position is updated by the computation of net force and acceleration and double integration with respect to time.

## 5 Correction of Approximate Sketches

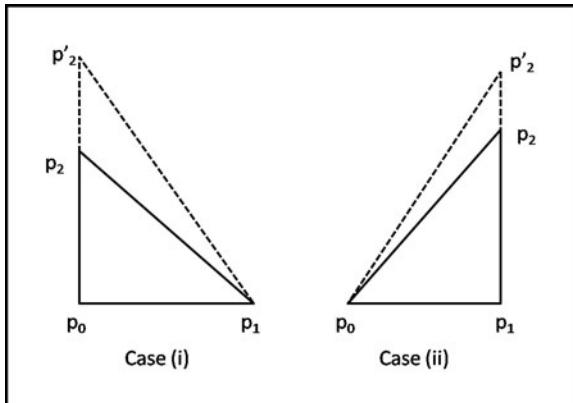
Correction of approximate sketches can take three forms. First, components may be corrected individually to conform to a user's intended sketch. Second, groups of shapes may need to be corrected. For example, the user may intend for one shape to rest on top of the other, but his actual sketch places them a small distance apart. During animation, the underlying physics engine will not treat the two shapes as if they were in contact. Lastly, making an association may change the appearance of a diagram component. Consider an inclined plane problem. The user draws a triangle and then writes an expression for the incline angle. When the association is made, the triangle should be adjusted to reflect the correct incline angle. This is necessary because the underlying physics engine will resolve contacts between shapes based on their spatial characteristics. For example, an incline with actual incline angle  $\pi/4$  is associated with the following mathematical expression  $\theta = \pi/6$ . If a ball is placed on the incline, the user will calculate the parallel and perpendicular components of weight in terms of  $\theta = \pi/6$ . When the association is made and the animation run, the ball will fly off the incline instead of sliding down the incline. This happens because the direction of the net force will be slightly divergent from the incline instead of parallel to it.

### 5.1 Shape Correction

Sketched polygons can have approximately horizontal and vertical edges and may thus need individual correction. Each edge of a polygon is examined to see if its slope is nearly vertical or horizontal. Approximate vertical or horizontal edges of a polygon are thus corrected. In [3], users were required to draw polygon vertices in anti-clockwise order. This constraint has now been lifted by examining the sign of the cross product between adjacent polygon edges. If any of the cross products is positive, then the ordering of vertices is reversed to ensure anti-clockwise winding. This allows users to sketch polygons in an unconstrained manner.

It is possible for the user to draw two or more shapes close to each other with the intent that the shapes be in physical contact. Upon recognition, the shapes may be a small distance apart. To correct these situations, the minimum distances between all pairs of shapes are computed. If the distance between two

shapes is below a threshold value, then they are moved into contact. Between circles, movement occurs along the axis connecting the centers. If either shape in a pair is a polygon, then a vector corresponding to the minimum distance is computed. This is always perpendicular to exactly one edge of one of the polygons. This vector serves as the axis along which movement occurs. The direction of movement is from the shape with lower mass toward the more massive shape.



**Fig. 3.** Two cases for Triangle Rectification

Associating an angle with a right-angled triangle changes the incline of the triangle. Two possible cases are shown in Figure 3. The only difference is whether the incline is uphill or downhill. In case (i), the new vertex  $p'_2$  is computed as

1. Rotate  $p_0$  clockwise by  $\theta$  about  $p_1$  to get  $p'_0$
2. Compute direction  $v = \frac{p'_0 - p_1}{\|p'_0 - p_1\|}$
3. Compute  $p'_2 = p_1 + k v$ , where  $k = \frac{\|p_0 - p_1\|^2}{v \cdot \frac{p_0 - p_1}{\|p_0 - p_1\|}}$

In case (ii), the new vertex  $p'_2$  is computed as

1. Rotate  $p_1$  anticlockwise by  $\theta$  about  $p_0$  to get  $p'_1$
2. Compute direction  $v = \frac{p'_1 - p_0}{\|p'_1 - p_0\|}$
3. Compute  $p'_2 = p_0 + k v$ , where  $k = \frac{\|p_0 - p_1\|^2}{v \cdot \frac{p_1 - p_0}{\|p_1 - p_0\|}}$

## 5.2 Wire and Spring Correction

Wire and spring correction can also take three forms. The first case relates to the length of the wire/spring. During sketching, users may accidentally draw them as ending within a shape rather than attached to its boundary. In order to capture the user's intent, the correction mechanism clips the endpoint of the wire/spring against the shape it is attached to. This ensures that forces due to

wires/springs act at their proper locations and result in a correct animation. Secondly, if the distance between the endpoint of a wire/spring and a polygon vertex is below a threshold, the endpoint is aligned with the vertex to conform to the user's intent.

The last case relates to the angle that wires/springs may make with horizontal/vertical axes. If an association involves an angle, the system determines if the association is made near the start or the end of the wire/spring. An angle association completed near the end shape implies that the angle with the horizontal axis is being altered. An angle association completed near the starting point of the wire/spring indicates that the angle being altered is the angle with the vertical axis. We assume that wires/springs endpoints are always drawn in top-down ordering. Once the axis and the proper endpoint are determined, a rotation about the other endpoint is required to alter the wire/spring into the desired location in conformity with the user's intent.

## 6 Example Scenarios

### 6.1 Example 1: Inclined Plane

Figure 4 represents a problem that involves a ball rolling down an inclined plane. The information given to a student is the angle of the incline and the mass of the ball. The student is asked to work out the magnitude of the normal force acting on the ball. The student sketches a triangle to represent the inclined plane. He then draws a circle on the incline to represent the ball. From the given information he works out that the magnitude of the normal force acting on the ball is  $f_1 = \|W\| \cos \theta$ .

To verify the answer, the student decides to animate the sketch with the solution as input. Upon analysis, the incline and ball are replaced with corrected

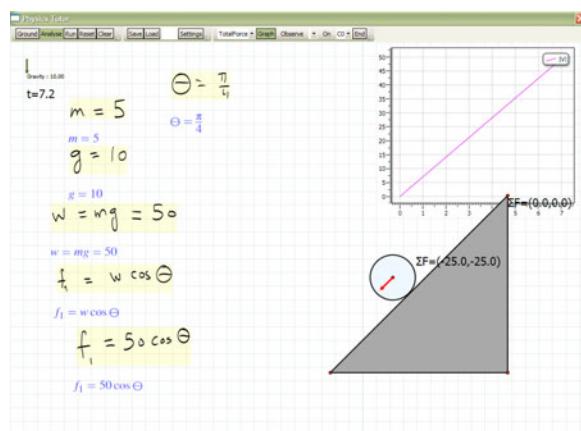


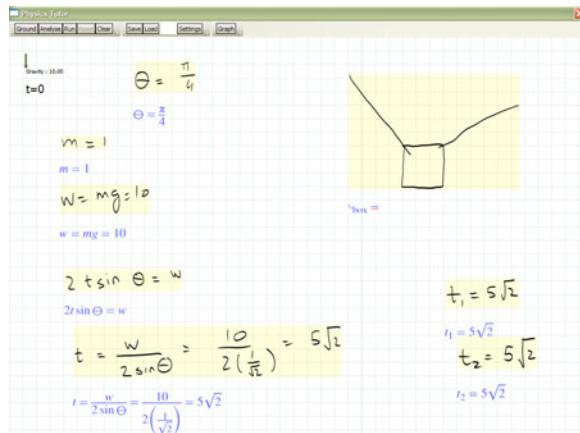
Fig. 4. An inclined plane problem

diagram components (triangle and circle). The triangle is made immobile by tapping it once. The student associates the expression for incline angle with the triangle. This causes the triangle to change appearance and the circle on it to be repositioned accordingly. The student associates the expressions for mass and the normal force ( $f_1$ ) with the circle and animates the sketch.

If the computed answer is correct, the ball will have a resultant force that will cause it to roll down the incline. If the expression is wrong, then the ball will either bounce off the incline (normal force too low) or fly off (normal force too great). An incorrect animation therefore provides the user with hints about possible errors in his calculation. The student also chooses to view a graph of how the magnitude of the velocity changes with time for the ball. It shows a linear increase in velocity with time, implying a constant acceleration. Notice that the student worked out the answer as he would on paper. The information associated with recognized diagram components is minimal and is a subset of the student's solution. The student did not have to specify the position update equations to govern the ball's motion down the incline. He simply worked out the magnitude of the normal force acting on the ball due to the inclined plane. Also, the expression derived by the student is just the magnitude of the normal force and by itself is insufficient to animate the diagram. The system infers from the shapes in contact and the association of the angle expression with the triangle that an inclined plane problem is being animated. The unit direction of the normal force is computed as perpendicular to the incline edge of the triangle and multiplied by the magnitude (associated by the user) to get the normal force. Note also that the expression for the normal force is written in terms of  $\theta$ . Our previous implementation [3] did not allow writing of vector equations and would have required the specification of the normal force in terms of x and y coordinates as  $f_x = W_x \cos \theta$  and  $f_y = W_y \cos \theta$ . The above equations are incorrect from a physics point of view and would only confuse a student should he have to use them to specify the animation's initial conditions. We have removed this cause of confusion in our current prototype.

## 6.2 Example 2: Wires Holding an Object in Equilibrium

Figure 5 represents a problem where a box is being held in equilibrium by means of two wires attached at an angle to it. The student is given the mass of the box and the angle that each wire makes with the horizontal axis. He must calculate the tension in each wire that will keep the box suspended in equilibrium. Upon reflection, the student realizes that the sum of the vertical components of tension in each wire must equal the weight of the box. With this insight, the student works out that the magnitude of the tension in each wire is  $5\sqrt{2}$ . As before, to verify the result, he instructs the system to analyze the sketch. The system recognizes and replaces the box with a square. The wires are clipped properly against the square's boundary. The student alters the mass of the square and associates the tension with each wire. Notice that the sketch is approximate and that the wires are not truly at an angle of  $\pi/4$  with the horizontal. Upon making



**Fig. 5.** An equilibrium problem modeled in our system

the association ( $\theta = \pi/4$ ), the system alters each wire so that it is at an angle of  $\pi/4$  with the horizontal.

When the animation is run, the system shows the box being held in a stationary position. If the student had computed the tension in any one of the wires incorrectly, the animation would have resulted in a net force on the box. The box's resulting motion would cause one or both wires to break, confirming to the student that his answer was incorrect.

## 7 Conclusion

We have presented a prototype sketch-based physics tutoring system that fuses mathematical sketching with an underlying physics engine. We have described key features of our prototype that allow it to deal with several cases of approximate user sketches through sketch rectification, enhance its capabilities in terms of accepting flexible input, provide visual feedback, and support associations between drawings and mathematics. Our current prototype is limited to simple linear kinematics and can only provide visual verification. In the future, we plan to include problems from other areas such as work, energy, rotational kinematics, simple harmonic motion, and planetary motion. We also plan to support numerical verification of a user's answer to a problem. Other possible avenues of future work include implementation of a dimensionality analysis tool and a tool to convert between different systems of units. We plan to do a general study of the techniques and ways that university physics students employ to solve different types of physics problems. We also plan on conducting technical and user evaluations of our system at a suitably mature stage to help us get important feedback to improve student learning.

## Acknowledgments

This work is supported in part by NSF CAREER Award IIS-0845921.

## References

1. Alvarado, C.J.: A Natural Sketching Environment: Bringing the Computer into early stages of Mechanical Design. Master's Thesis, Massachusetts Institute of Technology (2000)
2. Baraff, D., Witkin, A.: Physically Based Modeling: Principles and Practice, Siggraph Course Notes (1997)
3. Cheema, S., LaViola, J.: Towards Intelligent Motion Inferencing in Mathematical Sketching. In: Proceedings of IUI 2010, pp. 289–292 (2010)
4. Jiang, Y., Tian, F., Wang, H., Zhang, X., Wang, X., Dai, G.: Intelligent Understanding of Handwritten Geometry Theorem Proving. In: Proceedings of IUI 2010, pp. 119–128 (2010)
5. Kara, L.B., Gennari, L., Stahovich, T.F.: A Sketch-based Tool for Analyzing Vibratory Mechanical Systems. Journal of Mechanical Design 130(10) (2008)
6. LaViola, J.: Advances in Mathematical Sketching: Moving Toward the Paradigm's Full Potential. IEEE Computer Graphics and Applications 27(1), 38–48 (2007)
7. LaViola, J.: Mathematical Sketching: A New Approach to Creating and Exploring Dynamic Illustrations. PhD Thesis, Brown University (2005)
8. LaViola, J., Zeleznik, R.: MathPad<sup>2</sup>: A System for the Creation and Exploration of Mathematical Sketches. ACM Transactions on Graphics (Proceedings of Siggraph 2004) 23(3), 432–440 (2004)
9. Lee, W., de Silva, R., Peterson, E.J., Calfee, R.C., Stahovich, T.F.: Newton's Pen: a pen based tutoring system for statics. In: Proceedings of SBIM 2007, pp. 59–66 (2007)
10. Millington, I.: Game Physics Engine Developement. Morgan Kaufmann, San Francisco (March 2007)
11. VanLehn, K., Lynch, C., Schulze, K., Shapiro, J.A., Shelby, R., Taylor, L., Treacy, D., Weinstein, A., Wintersgill, M.: The Andes physics tutoring system: Lessons Learned. International Journal of Artificial Intelligence and Education 15(3), 1–47 (2005)
12. Varberg, D., Purcell, E.J.: Calculus with Analytical Geometry. Prentice-Hall, Englewood Cliffs (1992)
13. Young, H.D.: University Physics. Addison-Wesley Publishing Company, Reading (1992)
14. Zeleznik, R., Miller, T., Li, C., LaViola, J.: MathPaper: Mathematical Sketching with Fluid Support for Interactive Computation. In: Butz, A., Fisher, B., Krüger, A., Olivier, P., Christie, M. (eds.) SG 2008. LNCS, vol. 5166, pp. 20–32. Springer, Heidelberg (2008)

# A Sketch-and-Grow Interface for Botanical Tree Modeling

Nordin Zakaria

Computer & Info Science Department, Universiti Teknologi PETRONAS,  
Malaysia  
[nordinzakaria@petronas.com.my](mailto:nordinzakaria@petronas.com.my)

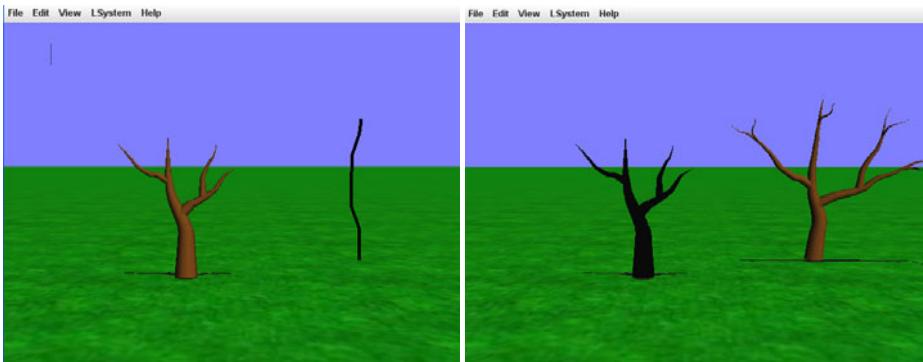
**Abstract.** We introduce in this paper an interface for botanical tree modeling that allows the user to sketch a tree, and then to automatically reproduce it at various stages of ‘growth’. The technique works by first inferring as best as possible the L-System growth rule that represent the tree model drawn by the user. The user can then reproduce the tree model by drawing a *growth stroke*. The growth stroke indicates the shape of the main axis, and the height up to which the tree should be grown, and determines the number of iterations of applications of the L-system rule. We show in this paper how the technique can be used as a simple yet effective way to produce variants of a tree structure.

**Keywords:** sketch-based modeling, tree-modeling, L-System.

## 1 Introduction

Botanical trees lend themselves well to sketching. Branches correspond to strokes, and one can create strokes that begin from some other strokes in order to create a hierarchy of branches. A stroke can have various shapes and length, corresponding to the complexity and variance of an actual tree branch in nature. On the other hand, a sketch of a tree lacks the depth and compactness of an L-system tree grammar formalism. An L-system represents an attempt to precisely model the structure and the sequence of morphological changes in a tree. Combination of both sketch and L-System into a single interaction framework promises to combine the intuitiveness of sketching with the expressive power of L-system, and has been the pursuit of recent research works [2,7]. In this paper, we contribute to this line of approach by proposing a user interface that derives L-system from the user input sketch and reproduce it to depict variants of the sketched tree in various forms and stages of growth.

We implement our proposed approach as a user interface feature in *TreeSketch*, a sketch-based tree modeling application described in [18]. The interaction mechanism for our sketch-and-grow functionality is as shown in Figure 1. When a tree is already drawn and a user draws a new stroke that starts from the ground, the new stroke will be interpreted to indicate the shape of and the height up to which the tree should be duplicated and grown. Internally, a parametric L-System [15] is used to represent the structure of the user-drawn tree. We map from the user sketch to an L-System description by scanning the geometry of the sketch to infer the L-System template,



**Fig 1.** A *growth stroke* and its corresponding tree structure

and then using a genetic algorithm to infer the numerical parameters. With the L-System internally representing the structure of the user sketch, our application is then able to animate the ‘growth’ of the tree represented by the sketch, resulting in what we call a *sketch-and-grow* interface.

We organize the rest of the paper as follows: We discuss on related work in tree modeling and L-System inference in section 2. In section 3, we elaborate on the algorithms involved in our sketch-and-grow approach, and the results obtained. Finally, in section 4, we conclude the work and project future development.

## 2 Tree Modeling and L-System Inference

Various techniques for tree-modeling have been proposed over the years. These include approaches based on growth parameters [16], L-Systems [15], geometric parameters [18], sketch-based user interfaces[7,14,19], and image-based techniques [12]. We chose to integrate the representational power of L-System into the intuitive sketch-based framework of Okabe et al [14]. There has been similar integration work by Ijiri et al [7] and Anastacio et al [2]. However, in both work, it is necessary for the user to specify the L-System. The core of our approach is the inference of a parametric L-system from user sketches. Hence, the user does not have to provide the grammar specification.

The inference problem for L-Systems involves finding the axioms and rewrite rules for a given branching structure. The problem is nontrivial because the mapping from an L-System to the final string is multidimensional, nonlinear and discontinuous. Random modification of production rules generally gives little insight into the relationship between L-Systems and the figures they produce [15].

The work of Ferraro et al [3] suggests an approach for the automation of the process, based on analyzing the tree isomorphism of the input structure. However user sketches are often sparse, and it is, in general, difficult to detect isomorphism in such settings. Nevill-Manning and Witten [13] proposed SEQUITUR, an algorithm that can infer a hierarchical structure from a sequence of discrete symbols. Though SEQUITUR has been shown to be applicable in discovering structure in an L-System string or L-String, the rules obtained are not necessarily syntactically correct.

Other approaches view the L-System inference as an optimization problem. Due to the nature of the problem, many works build on top of some variants of evolutionary algorithms [4]. In [8] and McCormack [11], a rule is encoded in an individual genome. The advantage of this approach is that the rule can be arbitrarily long. However, the search space is very big, even for genetic algorithms, and it is difficult to obtain good results in a reasonable amount of time. Kokai et al [9] describe a parallel processing approach to evolve an L-system for the modeling of blood veins of the human retina. They evolve subpopulations of L-Systems, with each subpopulation comprising of individuals with the same rule structure but with different parameters.

### 3 L-System Derivation

Our L-System derivation process comprises of the following basic steps:

- 1) architectural scan - gathers the branching types in a tree sketch, creates the corresponding L-Rules, and computes the *instruction list*. The instruction list is simply an array of indices to the L-Rules. It is a form of L-System control mechanism [15], meant to ensure that a tree grows to look similar to the user input. The resulting L-System can be thought of as a template – a parameteric L-System with the parameters still unknown.
- 2) Derivation of parameters – with the L-System template computed, the derivation of the numerical parameters can be posed as an optimization problem. The task is to find the parameters that will make the L-System generated tree similar to the original tree.

#### 3.1 Architectural Scan

Given a user sketch, as shown in Figure 2a, the branches are first clustered based on the proximity of the anchor points. The nodes are then categorized according to the number of lateral branches attached to it as well as to whether or not it is an apex node (i.e. with no further growth along the same branch). For each category, an L-Rule is created using the following L-Symbols: F (forward), R (rotation), [ (start of a new branch), ] (end of a branch), and G (further growth). The L-Rules corresponding to the input sketch in Figure 2a is as shown in Figure 2d. Note that other forms of L-Rules can be used instead of those shown in the figure. Further, note that G in Figure 2d is a variable index pointing to one of the 3 L-Rules, depending on the corresponding index value in the instruction list to be computed.

In the next stage, firstly, each node,  $n_i$ , stores a link to the first node on each of the lateral branches. Secondly,  $n_i$  stores the link to the next node on the same branch. If a lateral branch has no node, we assume a *null* link for  $n_i$  to that branch. Similarly, if  $n_i$  is not an apex (the tip of the branch) but there is no further node on the same branch, we appended a null link to  $n_i$ . The result of the node linking process is a tree data structure, as shown in Figure 1b. Note that in the figure, a null link terminates with a diamond shape, while a non-null link with an arrow shape. Each link is then associated with an index (starting from 1 in the example) to the corresponding L-Rule, or -1 if it is a null link. Finally, a breadth-first traversal of the tree data structure is performed, collecting the indices stored with the links, resulting for the input sketch in Figure 1a, the instruction list shown in Figure 1e.

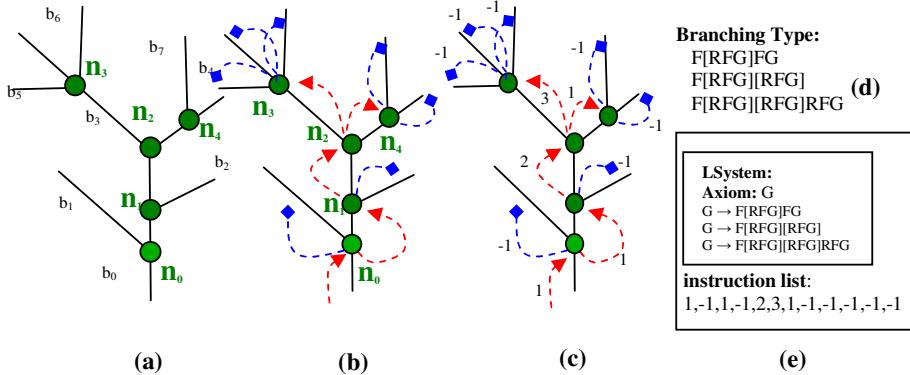


Fig. 2. Architectural Scan

### 3.2 Derivation of Parameters

The architectural scan described in the previous section computes a parametric L-System with the parameters still unknown. Finding the parameters, especially for non-trivial input sketches, can be posed as an optimization problem. We need to find the sequence of values that when mapped to the parameter tuple  $T$  within an L-System would result in a structure matching as best as possible that drawn by the user.

The parameter tuple for an L-System is formed by concatenating the symbols or scalar variables within the parameter expression of the L-Chars (i.e. L-System symbols) in a linear top-down, left-right manner, starting from the axiom. As an example, in our implementation, assuming the parameteric form of the L-System on the left to be that on the right:

$$\text{Axiom: } G \\ G \rightarrow F[RFG]FG$$

$$\text{Axiom: } G(f, r_1, r_2, r_3) \\ G(f, r_1, r_2, r_3) \rightarrow F(s_1 \times f)[R(s_2 \times r_1, s_3 \times r_2, s_4 \times r_3)F(s_5 \times f)G(s_6 \times f, s_7 \times r_1, s_8 \times r_2, s_9 \times r_3)]F(s_{10} \times f)G(s_{11} \times f, s_{12} \times r_1, s_{13} \times r_2, s_{14} \times r_3),$$

the resulting parameter tuple is as follows:  $\langle f, r_1, r_2, r_3, s_1, s_2, s_3, s_4, s_5, s_6, s_7, s_8, s_9, s_{10}, s_{11}, s_{12}, s_{13}, s_{14} \rangle$ . Note that in forming the tuple for a rule, the predecessor (or left-hand side) for the rule is not considered. Further, in processing the right-hand side, only the symbol which is not in the predecessor is considered. Hence, in the example above, only the variables in bold are considered. The basic idea is simply to gather the variables within the L-System into a tuple without repetition.

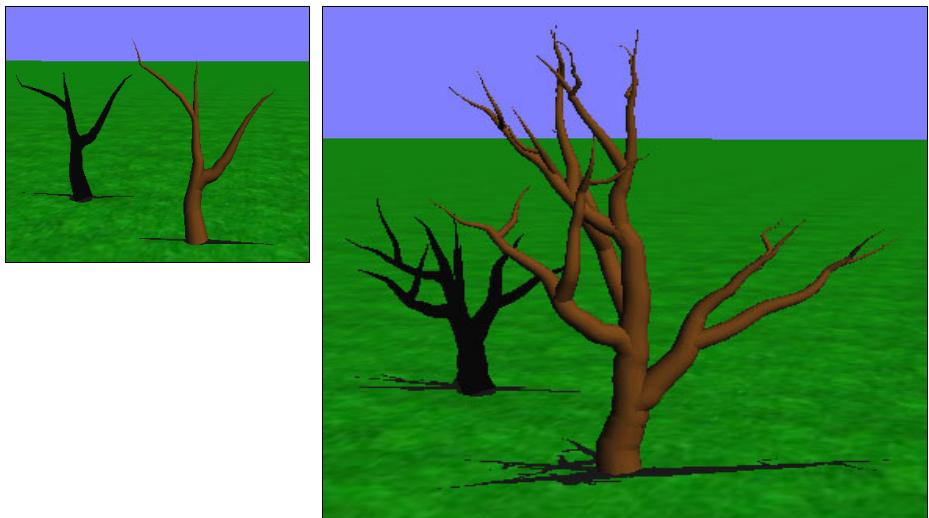
Given a parameter tuple, the task of an optimizer is then to find the sequence of values that when mapped to the parameter tuple result in an L-System that best represents the user input sketch. We shall call the sequence of values to be assigned or already stored in a parameter tuple, *a value tuple*. To measure how well an L-System represents a user input sketch, the user's input sketch is first converted to an L-String,  $S_{\text{user}}$ . The L-System is then recursively applied, each application using the next index from the stored instruction list, to obtain a string  $S_{\text{sys}}$ . The fitness of the

L-System is then measured based on the squared distance between the value tuple from  $S_{\text{sys}}$  and that from  $S_{\text{user}}$ . To search for the optimal value tuple, due to the nonlinear nature of the problem, we deploy a genetic algorithm [4]. In our implementation, we use a generational genetic algorithm coupled with Fitness Uniform Selection Scheme or FUSS [5] in order to avoid premature convergence of the search process.

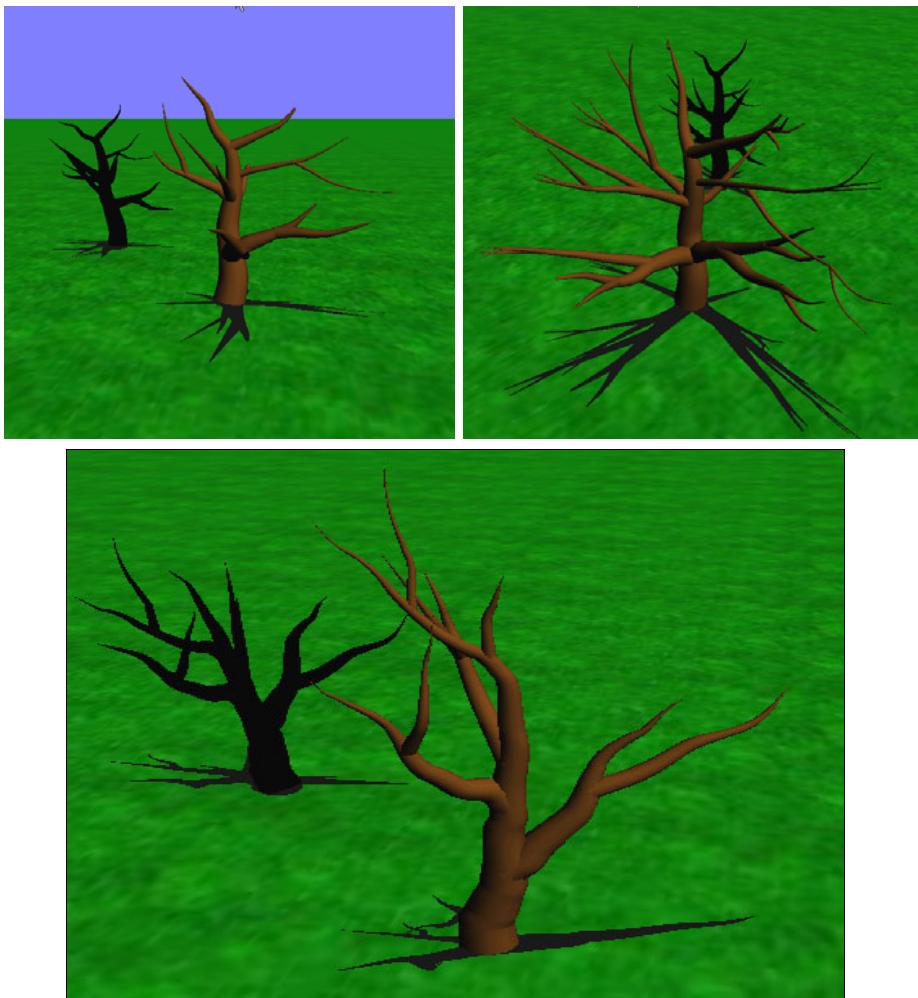
### 3.3 Results

As shown in Figure 3, the algorithms described for the sketch-and-grow interface are able to handle simple input sketch as well as moderately more complex sketch. More results are shown in Figure 4 at the end of the paper. As also shown in the accompanying video clip, the growth stroke can be made as long and complex as desired or as allowed by the user's graphics resource. To obtain results as visually close to the user's input as possible, we directly map the curviness in the original branches to the branches created by the corresponding L-system. A more sophisticated approach would be that in [1]; we have not however found the need to do this.

We note that while L-Systems are well-suited for fractal structures, users' input drawings are, in general, imprecise in nature. In fitting an L-System to a user's input drawing, we are in fact attempting to infer the fractal nature of the structure drawn. As shown in Figure 3 and 4, when the age specified by the user (using a stroke) is the same as that for the input tree, the L-System generated output, though perceptually close to the input, will, in general, be not exactly the same. Better matching results can be obtained by determining the rule index in the instruction list that leads to the biggest error in the matching between the value tuple from  $S_{\text{sys}}$  and that from  $S_{\text{user}}$ . The corresponding rule is then replicated and the optimization process repeats.



**Fig. 3.** Simple test to the left, and a more involved example to the right



**Fig. 4.** More *Sketch-and-Grow* examples

## 4 Discussion and Future Work

We have discussed an approach for modeling tree growth that relies on extracting the L-system structure of a sketched 3D tree structure. The method offers a new way of producing variants of the same tree. The L-System structure is derived using a deterministic approach, while its parametric details using a genetic algorithm. The machine-generated L-System has proven to be sufficient in simulating tree growths.

We note a number of limitations with the proposed approach. First of all, the genetic algorithm is not real-time. The user has to wait for the completion of the evolutionary iteration. The user should be properly conveyed the completion status, and the process should probably be in the background. Secondly, the approach is

purely geometric. It does not consider the physics or physiology of tree growth. Consideration of such factors are evident in a number of related work, notably [16,17]. Incorporating the factors would result in a more realistic animation of tree growth. Thirdly, the genetic algorithm is not guaranteed to find an L-System rule that can reproduced a structure close to that which has been drawn by the user. The structure drawn by the user may simply be too complex or too large such that it is simply not much hope for the genetic algorithm implementation that we currently have. Therefore, more work needs to be done in the future on a more robust L-System parameter search algorithm.

Nevertheless, we believe that the work presented in this paper represents a fresh take on tree growth simulation, and provides yet another example of a sketch-based tool useful for tree modeling.

## References

1. Hertzmann, A., Oliver, N., Curless, B., Seitz, S.M.: Curve analogies. In: Proceedings of the 13th Eurographics workshop on Rendering, Pisa, Italy, June 26-28 (2002)
2. Anastacio, F., Prusinkiewicz, P., Sousa, M.: Sketch-based parameterization of l-systems using illustration inspired construction lines. In: Proceedings of 5th Eurographics Workshop on Sketch-based Interfaces and Modeling, SBIM 2008 (2008)
3. Ferraro, P., Godin, C., Prusinkiewicz, P.: Towards a Quantification of Self-Similarity in Plants. *Fractals* 13(2), 91–109 (2005)
4. Goldberg, D.E.: *Genetic Algorithms in Search, Optimization and Machine Learning*. Kluwer Academic Publishers, Boston (1989)
5. Hutter, M.: Fitness Uniform Selection to Preserve Genetic Diversity. In: Proc. 2002 Congress on Evolutionary Computation (CEC 2002), May 2002, pp. 783–788. IEEE, Los Alamitos (2002)
6. Igarashi T., Matsuoka S., Tanaka H., Teddy: A sketching interface for 3d freeform design. In: Proc. of SIGGRAPH 1999, pp. 409–416 (1999)
7. Ijiri, T., Owada, S., Igarashi, T.: The Sketch L-System: Global Control of Tree Modeling Using Free-form Strokes. In: Butz, A., Fisher, B., Krüger, A., Olivier, P. (eds.) SG 2006. LNCS, vol. 4073, pp. 138–146. Springer, Heidelberg (2006)
8. Jacob, C.: Genetic L-System Programming. In: Davidor, Y., Männer, R., Schwefel, H.-P. (eds.) PPSN 1994. LNCS, Part III. vol. 866, Springer, Heidelberg (1994)
9. Kokai, G., Vanyi, R., Toth, Z.: Parametric L-System Description of the Retina with Combined Evolutionary Operators. In: GECCO, Orlando, Florida, USA, vol. 2, pp. 1588–1595 (1999)
10. Lintermann, B., Deussen, O.: Interactive Modeling of Plants. *IEEE Computer Graphics and Applications* 19(1), 56–65 (1999)
11. McCormack, J.: Evolutionary L-systems. In: Design by Evolution: Advances in Evolutionary Design, pp. 168–196. Springer, Berlin (2008)
12. Neubert, B., Franken, T., Deussen, O.: Approximate Image-Based Tree-Modeling using Particle Flows. *ACM Trans. Graph.* 26(3), 88 (2007)
13. Nevill-Manning, C.G., Witten, I.H.: Identifying Hierarchical Structure in Sequence: A Linear-Time Algorithm. *Journal of Artificial Intelligence Research* 7, 67–82 (1997)
14. Okabe, M., Owada, S., Igarashi, T.: Interactive Design of Botanical Trees Using Freehand Sketches and Example-based Editing, Computer Graphics Forum. In: Eurographics 2005, Trinity College, Dublin, Ireland, August 29 - September 02, vol. 24(3) (2005)

15. Prusinkiewicz, P., Lindenmayer, A.: *The Algorithmic Beauty of Plants*. Springer, Heidelberg (1996)
16. Reffye, P.d., Edelin, C., Francon, J., Jaeger, M., Puech, C.: Plant Models Faithful to Botanical Structure and Development. *Computer Graphics* 22(4) (August 1988)
17. Streit, L., Federl, P., Sousa, M.C.: Modelling Plant Variation Through Growth. *Computer Graphics Forum* 24(3), 497–506 (2005)
18. Weber, J., Penn, J.: Creation and Rendering of Realistic Trees. In: SIGGRAPH 1995 Proceedings, August 1995, pp. 119–128 (1995)
19. Zakaria, M., Shukri, S.R.M.: A Sketch-and-Spray Interface for Modeling Trees. In: Butz, A., Fisher, B., Krüger, A., Olivier, P., Owada, S. (eds.) SG 2007. LNCS, vol. 4569, pp. 23–35. Springer, Heidelberg (2007)

# Animated Volume Texture Mapping for Complex Scene Generation and Editing

Rui Shen

Department of Computing Science, University of Alberta  
Edmonton, Alberta, Canada T6G 2E8  
[rshen@cs.ualberta.ca](mailto:rshen@cs.ualberta.ca)

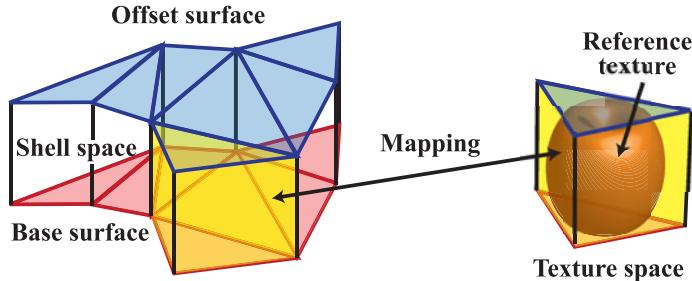
**Abstract.** Volume texture techniques can generate fine-scale geometric details on a surface. Therefore, these techniques have been broadly applied to render complex scenes such as forests and fur. Previous researchers mainly focus on the resulting geometry from volume texture mapping and the parameters for mapping the textures need to be predefined. In this paper, we visualize the mapping process in an animated way. During the animated mapping process, the parameters of the textures can be adjusted by the user for different regions on the surface in real time. The visualization and the real-time feedback facilitate the editing of the volume textures. In addition, the concept of animated mapping may also be applied to generating scenes such as the growth of forests.

## 1 Introduction

Generating fine-scale geometric details on a surface is a considerably tough task for conventional surface representation techniques: modeling a surface with complex geometry is quite difficult; two-dimensional (2D) textures fail to provide high-level reality of scenes; and generating surfaces covered by repetitive geometries is tedious. These problems can be solved by *volume texture* techniques, which were initially proposed by Kajiya and Kay for fur rendering [1]. Unlike 2D textures, a volume texture is a texture that has data in three dimensions. Volume texture techniques align the repetitive geometries on a *base surface* that has relatively low complexity. Hence, using volume textures to generate scenes with repetitive geometries that may be quite complex can be both efficient and accurate.

As shown in Figure 1, surface representation using volume textures usually involves three steps: shell construction, mapping and rendering. First, shell is constructed on the base surface for holding the volume textures that are instances of a *reference texture*. The space that the shell occupies is called *shell space* and the space that contains the reference texture is called *texture space*. Second, mapping is established between the points in the texture space and those in the shell space. In other words, the position of a volume texture on the base surface is defined by this mapping. Finally, the surface with volume textures is rendered.

In this paper, we focus on the mapping process. In previous work, this process is static and it merely serves as a preliminary step for rendering. The resulting surface is visible only after the rendering step. In our work, the mapping process



**Fig. 1.** Volume texture mapping

is visualized in an animated way, *i.e.*, every volume texture gradually moves from a certain position to its final position in the shell space. During the animated mapping process, a user can adjust the parameters of the textures for different regions in the shell space. The result of the adjustment is reflected in the scene simultaneously. The visualization and the real-time feedback facilitate the generation and editing of the volume-textured scenes. In addition, this animated mapping process can be applied to scenes such as the growth of forests.

The remainder of this paper is organized as follows. In Section 2 related work in the field of volume texture is discussed. The proposed method is introduced in Section 3. A prototype system implementing the proposed method and some results are presented in Section 4. The paper is concluded in Section 5.

## 2 Related Work

Volume texture techniques have received increasing attention in recent years. Many algorithms were proposed by previous researchers to solve the problems related to shell construction, mapping and rendering.

### 2.1 Shell Construction

Since shell space is between the base surface and an offset surface, the problem of shell construction is equivalent to the problem of offset surface generation. Due to the requirement of the mapping step, this offset surface needs to have the same connectivity as its base surface and no self-intersection. Simple normal displacement method works well for convex surfaces but is prone to self-intersections for non-convex surfaces. To address the problem with non-convex surfaces, Cohen *et al.* [2] proposed an algorithm named *simplification envelopes*. Each vertex on the base surface displaces away along its normal from its original position. An adaptive step size is assigned to each vertex to avoid self-intersections. The displacement ends when all the vertices have moved a predefined number of steps. However, a shell generated by envelopes tends to have the same thickness all over the shell. In contrast, Peng *et al.* [3] proposed an algorithm to generate elastic shells. They use an averaged distance function, which denotes the distance of a point to a surface, to calculate the direction and the desired moving distance.

## 2.2 Volume Texture Mapping

During the mapping process, positions for placing the textures in the shell space are calculated. Porumbescu *et al.* [4] introduced a bijective mapping, named *shell maps*, which establishes a mapping between shell space and texture space. Prisms are constructed both spaces, and then each prism is split into three tetrahedra using the prism tessellation algorithm [5], which is originally developed for thin shell tetrahedral mesh construction. After prism tessellation, mapping is established using barycentric coordinates. In their method, the texture space is constructed for the whole scene, *i.e.*, the mapping works like directly wrapping the base surface by the texture space. Although accuracy is guaranteed quite well using shell maps, there is no need to store repetitive textures. An alternative is to use volume *texels*, instances of a reference volume texture, which is originally introduced in the work of Kajiya and Kay [1]. In this algorithm, texels are tiled over the base surface. Neyret [6] extends this algorithm to complex scene construction such as forests. Another algorithm for volume texture mapping introduced by Lengyel *et al.* [7] for real-time 3D fur rendering is based on an approach named *lapped textures*, which is originally proposed by Praun *et al.* [8] for 2D texture mapping. In the work of Lengyel *et al.* only the local parametrization of the base surface needs to be taken into account. The reference texture is parameterized according to the distribution of the triangular facets within the current selected mapping region in the shell space.

## 2.3 Rendering of Volume-Textured Scenes

Rendering is the final step that makes the resulting surface visible. In [4], the points along a ray's trajectory are mapped into texture space for ray-tracing with photon mapping. Peng *et al.* [3] introduced a slice-based method, in which the scene is split into slices and each slice is rendered separately. Unlike this slice-based method, in which the whole scene is split into layers (slices) for rendering, Lengyel *et al.* [7] proposed a run-time rendering scheme for 3D fur, in which only textures are split into layers.

# 3 Animated Volume Texture Mapping

The proposed method, *animated volume texture mapping*, mainly consists of two algorithms: the mapping algorithm and the animation algorithm. The mapping algorithm, which is developed based on shell maps [4] and lapped textures [7], locates the position for each texel in the shell space. Arbitrary volume textures can be mapped onto regions of a base surface. According to the shape and size of the prismatic region allocated to a texel in the shell space, the animation algorithm transforms each texel gradually from its initial position to its ultimate position in the shell space. The parameters of the prismatic region can be adjusted on the fly, which results in different mapping effects. Combining the two algorithms together, we can produce the animated volume texture mapping process.

### 3.1 Mapping Algorithm Using Barycentric Coordinates

In the mapping process, we adopt the concept of texels. Texels derived from a reference texture are mapped onto every facet of the base surface. The workflow of our mapping algorithm is illustrated in Algorithm 1.

---

#### Algorithm 1. The mapping algorithm

---

- 1: Construct the offset surface  $S^o$  to the base surface  $S^b$  using an algorithm similar to simplification envelops by surface displacement along vertex normals (the displacement amount can be adjusted by the user on the fly)
  - 2: Construct the bounding prism  $P^t$  that encloses the reference texture  $R$
  - 3: Split  $P^t$  into three tetrahedra  $T_0^t$  to  $T_2^t$
  - 4: **for** Each unmapped facet  $F^b$  on  $S^b$  **do**
  - 5:     Construct a prism  $P^s$  in the shell space by connecting vertices on  $F^b$  and the corresponding vertices on  $S^o$
  - 6:     Split  $P^s$  into three tetrahedra  $T_0^s$  to  $T_2^s$
  - 7:     Map one instance of  $R$  into  $P^s$  using barycentric coordinates
  - 8: **end for**
- 

In the tessellation of prisms in the shell space, we do not require a global consistency, which is required in shell maps [4], because the reference texture is not constructed for the whole scene. However, to obtain the correct mapping, the tessellations of  $P^t$  and  $P^s$  need to have the same pattern, as shown in Figure 2. Therefore, every tetrahedron  $T^t$  in the texture space has a corresponding tetrahedron  $T^s$  in the shell space. The corresponding tetrahedra are highlighted using the same color. For example, the corresponding tetrahedron of the tetrahedron defined by vertices  $v_0^t$  to  $v_3^t$  is the one defined by vertices  $v_0^s$  to  $v_3^s$ .

**Mapping of Points.** Mapping an instance of the reference texture  $R$  into the shell space is achieved by mapping every point of  $R$  to the shell space. Barycentric coordinates are employed in the mapping process of points. Let  $p^t$

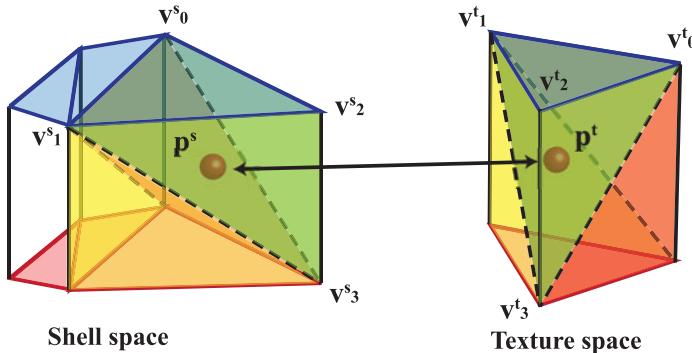


Fig. 2. Bounding prism tessellation and point-to-point mapping

denote a point/vertex inside a tetrahedron  $T^t$  in the texture space, then  $p^t$  can be represented as a linear combination of all the vertices  $v_i^t$ 's of  $T^t$ :

$$p^t = \sum_{i=0}^3 \alpha_i v_i^t, \quad (1)$$

where  $\sum_{i=0}^3 \alpha_i = 1$ . Since  $p^t$  and  $v_i^t$ 's are 3D points and  $\alpha_i$ 's need to satisfy the unit-sum constraint,  $\alpha_i$ 's can be calculated using four linear equations. After  $\alpha_i$ 's are calculated,  $p^t$ 's corresponding point  $p^s$  in the tetrahedron  $T^s$  in the shell space can be located using Equation (2). Hence, a mapping is established between  $p^t$  and  $p^s$ , as shown in Figure 2.

$$p^s = \sum_{i=0}^3 \alpha_i v_i^s. \quad (2)$$

### 3.2 Animation Algorithm Using Coordinate System-Independent Transformations

The animation algorithm makes a polyhedron (texel) to transform smoothly until it reaches its destination. In other words, this algorithm interpolates between the initial state of a texel and its final state, including position, scale and shape. The initial state of a texel can be defined by the user or automatically defined by the program. The final state of a texel is defined using the mapping algorithm introduced in Section 3.1. The transformation is independent of the reference texture's coordinate systems. Note that we cannot simply interpolate the coordinates between the corresponding vertices, because the geometry of a texel may be twisted if rotation is involved. This algorithm works as long as the pair of texels ( $X^0$  and  $X^K$ ) at initial and final states has the same vertex connectivity. The animation speed can be adjusted by the user interactively. The workflow of our animation algorithm is illustrated in Algorithm 2 assuming the texel pair  $X^0$  and  $X^K$  are already defined.

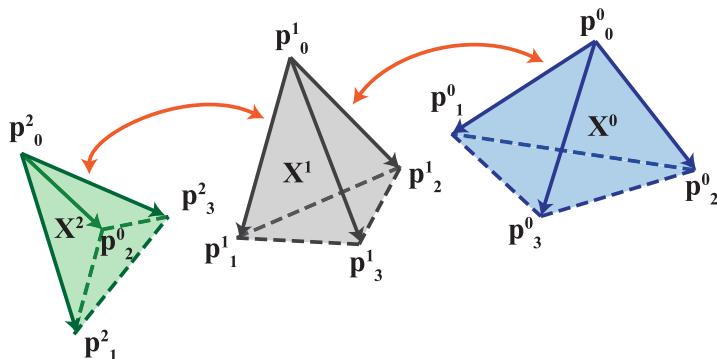
---

#### Algorithm 2. The animation algorithm

---

- 1: Randomly define a point/vertex  $p^0$  of  $X^0$  as the reference point and then calculate the corresponding point  $p^K$  of  $X^K$  using the method introduced in Section 3.1
  - 2: Calculate the relative coordinates of the other points in  $X^0$  and  $X^K$  to  $p^0$  and  $p^K$ , respectively
  - 3: Calculate the transformation parameters between corresponding points in  $X^0$  and  $X^K$  using the relative coordinates
  - 4: **for**  $k = 0$  to  $K - 1$  **do**
  - 5:   Interpolate the translation vector and the rotation angles between the corresponding points in  $X^k$  and  $X^{k+1}$  at the  $k$ th frame
  - 6: **end for**
- 

An example is shown in Figure 3. First, the tetrahedron on the right is transformed to the one in the middle using  $p_0^0$  as the reference point. Then, the

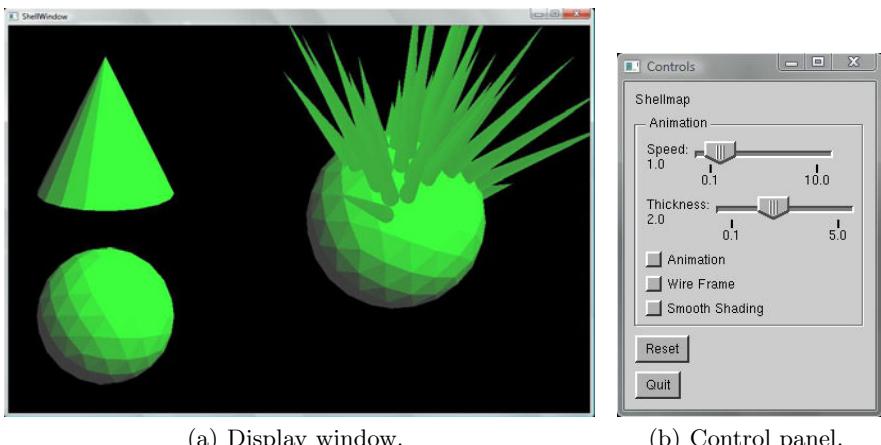


**Fig. 3.** Illustration of the animated mapping process

interpolation parameters are obtained according to the relative position of the tetrahedron in the middle to the one on the left. Finally, the tetrahedron in the middle gradually transforms to the one on the left.

## 4 Prototype System and Results

We have implemented a prototype system for animated volume texture mapping. The prototype system is built on CGAL<sup>1</sup>, a computational geometry algorithms library and GLOW<sup>2</sup>, a cross-platform graphical user interface (GUI) toolkit. OpenGL<sup>3</sup> is used to support basic rendering functions. Two user-input



(a) Display window.

(b) Control panel.

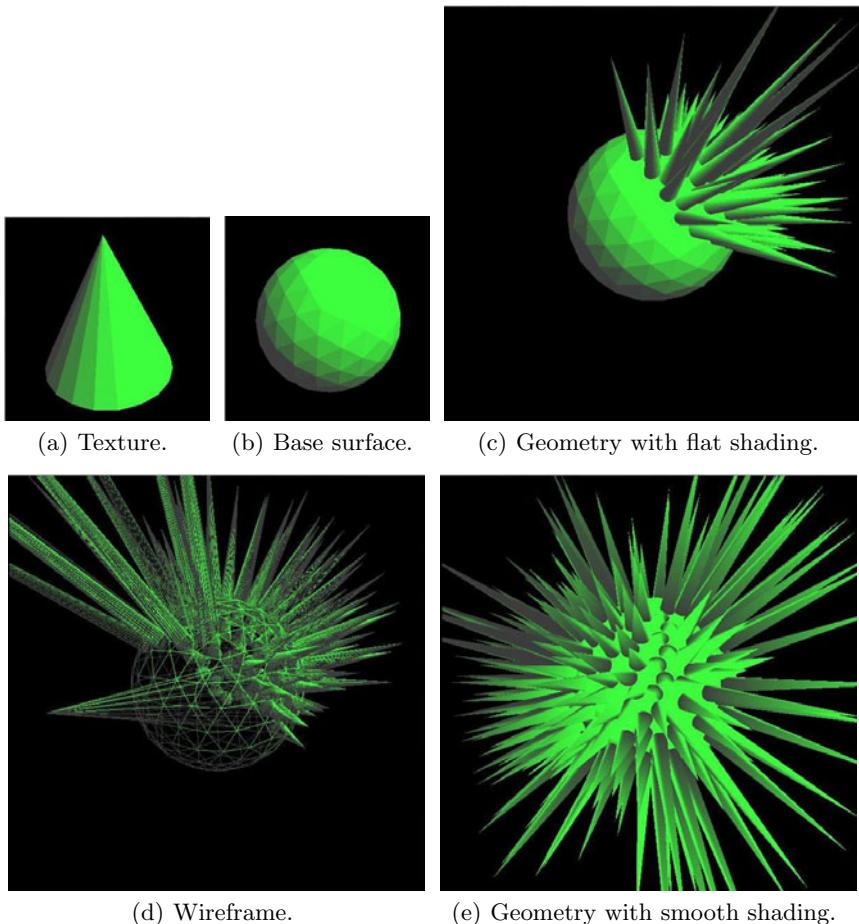
**Fig. 4.** The user interface of the prototype system

<sup>1</sup> <http://www.cgal.org/>.

<sup>2</sup> <http://glow.sourceforge.net/>.

<sup>3</sup> <http://www.opengl.org/>.

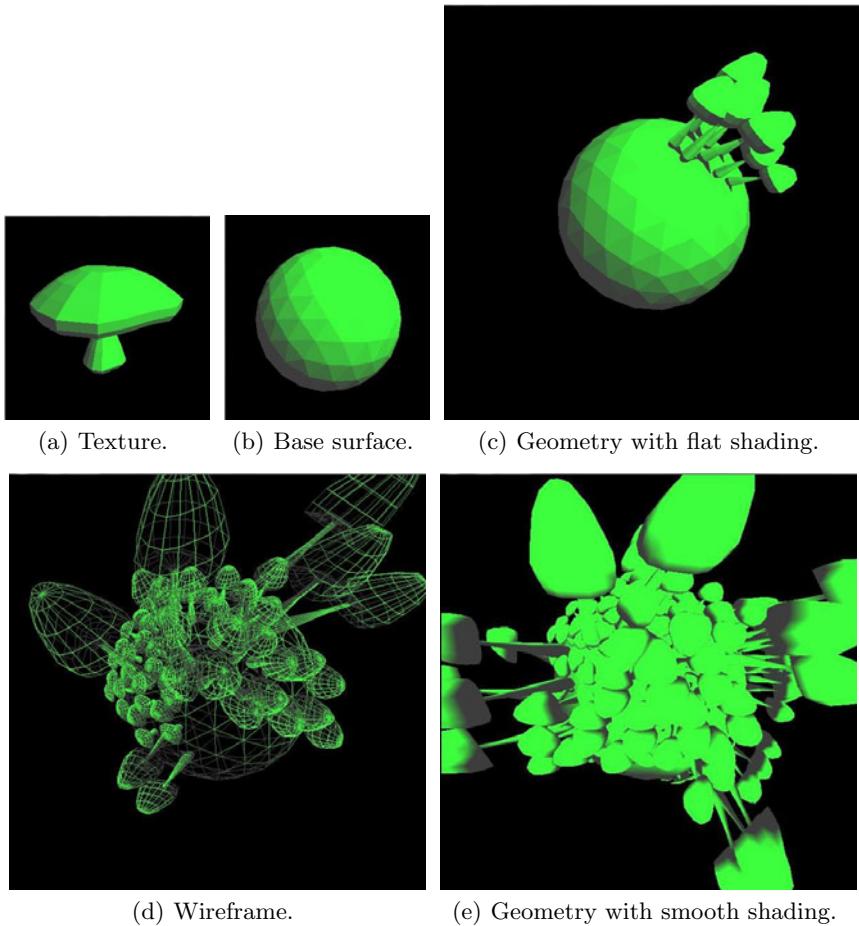
3D models serve as the base surface and the reference texture, respectively. Then the animated mapping process is executed. This system was tested under both Windows and Linux operating systems. Figure 4 gives the GUI of the system under Windows Vista. The base surface and the reference texture are displayed in two separate subwindows on the bottom left and top left of the display window (Figure 4(a)), respectively. The animated mapping process and the resulting surface are displayed in a third subwindow, which is on the right of the display window. The control panel is shown in Figure 4(b). The animation speed, shell space thickness, and rendering schemes (*i.e.*, wireframe or geometry, flat shading or smooth shading) can be adjusted in real time. The animation can be paused at anytime for examining the intermediate geometry. If the user is not satisfied with the textured scene, he/she can use the *Reset* button to clear the scene and restart the mapping process.



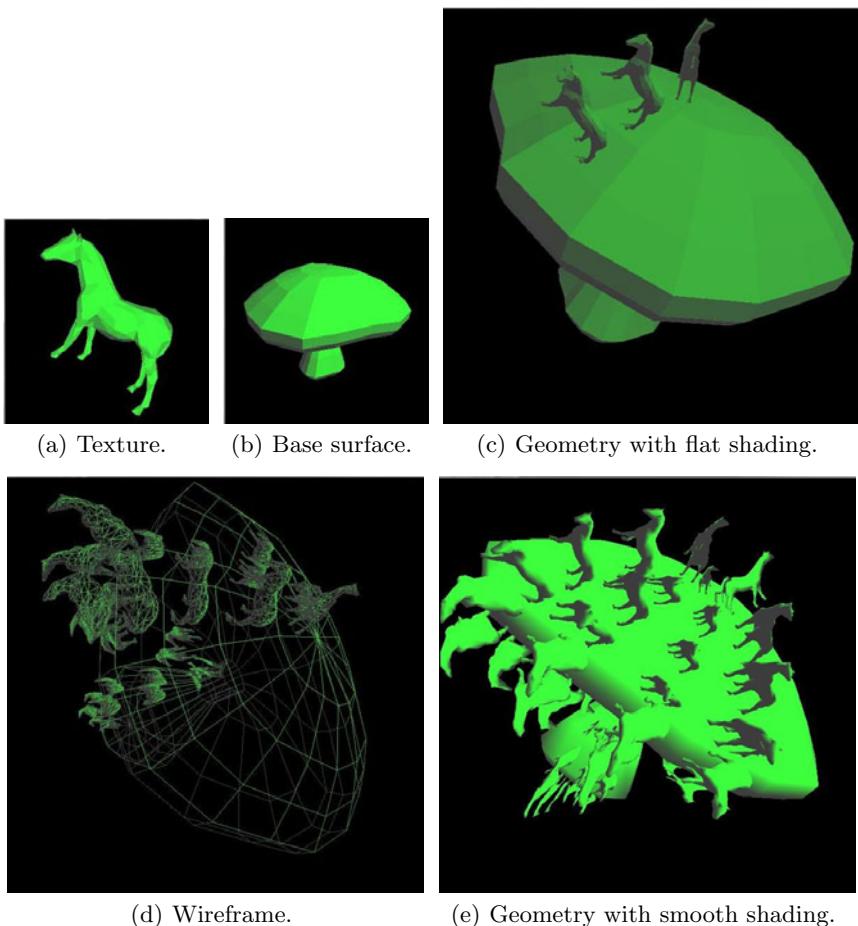
**Fig. 5.** Mapping a cone-shaped texture onto a sphere

Figure 5 shows the process of mapping a cone-shaped texture onto a sphere. The reference texture and the base surface are shown in Figure 5(a) and 5(b), respectively. A frame shortly after the start of the process is shown in Figure 5(c) rendered as geometry with flat shading. A frame in the middle of the process is shown in Figure 5(d) rendered as wireframe. The wireframe view is good for examining the transformation of the texels. A frame at the end of the process is shown in Figure 5(e) rendered as geometry with smooth/gourand shading. Two more mapping results are shown in Figure 6 and 7. Arbitrary volume textures can be mapped onto various base surfaces to create complex scenes.

In our current implementation, the initial state of a texel is defined by the program. The animation of a texel starts from the center of the scene. Figure 8 shows five frames from an animation of the growth of a mushroom by mapping a mushroom-shaped texture onto a surface. More complicated animation, such as

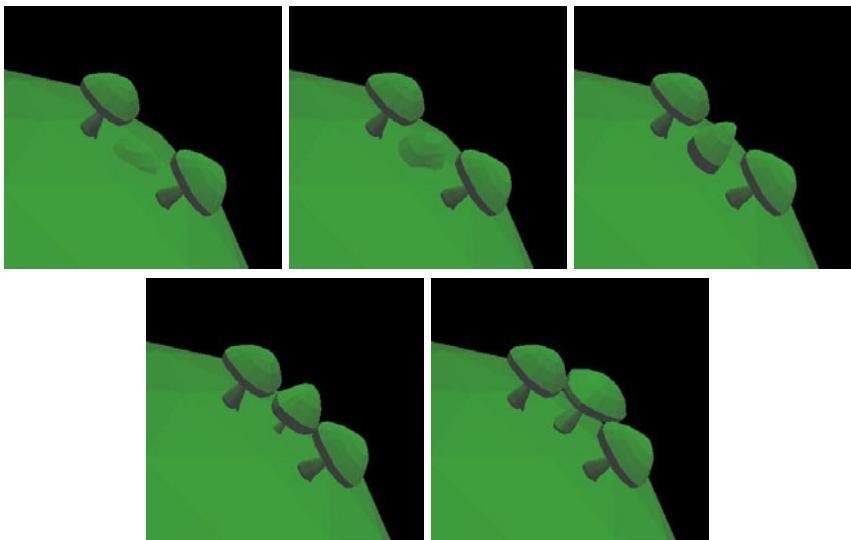


**Fig. 6.** Mapping a mushroom texture onto a sphere



**Fig. 7.** Mapping a horse texture onto a mushroom

the growth of a forest and a meteorite rain, may need simultaneous animation of multiple texels and the initial state of a texel to be calculated according to its final state. We are working on incorporating functions for this simultaneous animation. In order to give the user more control over the mapping process, we are also working on a function for the user to define the initial states of texels. In addition, bounding prisms in the shell space are currently only constructed for individual facets on the base surface. We are working on mapping volume textures onto arbitrary regions of the base surface, which requires to construct a set of composite prisms that encloses a texel in the shell and texture spaces.



**Fig. 8.** Animation of the growth of a mushroom by mapping a mushroom-shaped texture onto a surface

## 5 Conclusion

In this paper, we proposed a method for generating and editing volume-textured scenes via visualizing the mapping process. This animated mapping may also be applied to the animation of time-varying scenes. The proposed method involves two major algorithms, *i.e.*, the mapping algorithm and the animation algorithm. The mapping algorithm locates the ultimate position of a texel in the shell space using barycentric coordinates, while the animation algorithm interpolates smoothly between the initial position of a texel and its destination. The user can control the mapping process on the fly to achieve the desired scene geometry. In the future, we will perform a user study to evaluate the effectiveness of the proposed method. We will also investigate more functions to facilitate the generation and editing of complex scene geometries.

## Acknowledgement

The support of Killam Trusts, iCORE, and Alberta Advanced Education and Technology is gratefully acknowledged.

## References

1. Kajiya, J.T., Kay, T.L.: Rendering fur with three dimensional textures. In: SIGGRAPH 1989: Proceedings of the 16th Annual Conference on Computer Graphics and Interactive Techniques, pp. 271–280 (1989)

2. Cohen, J., Varshney, A., Manocha, D., Turk, G., Weber, H., Agarwal, P., Brooks, F., Wright, W.: Simplification envelopes. In: SIGGRAPH 1996: Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques, pp. 119–128 (1996)
3. Peng, J., Kristjansson, D., Zorin, D.: Interactive modeling of topologically complex geometric detail. ACM Transactions on Graphics 23(3), 635–643 (2004)
4. Porumbescu, S.D., Budge, B., Feng, L., Joy, K.I.: Shell maps. ACM Transactions on Graphics 24(3), 626–633 (2005)
5. Erleben, K., Dohlmann, H., Sporrung, J.: The adaptive thin shell tetrahedral mesh. Journal of WSCG 13, 17–24 (2005)
6. Neyret, F.: Modeling, animating, and rendering complex scenes using volumetric textures. IEEE Transactions on Visualization and Computer Graphics 4(1), 55–70 (1998)
7. Lengyel, J., Praun, E., Finkelstein, A., Hoppe, H.: Real-time fur over arbitrary surfaces. In: SI3D 2001: Proceedings of the 2001 Symposium on Interactive 3D Graphics, pp. 227–232 (2001)
8. Praun, E., Finkelstein, A., Hoppe, H.: Lapped textures. In: SIGGRAPH 2000: Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques, pp. 465–470 (2000)

# A Novel Three-Dimensional Collaborative Online Platform for Bio-molecular Modeling

Kugamoorthy Gajananan<sup>1</sup>, Arturo Nakasone<sup>1</sup>, Andreas Hildebrandt<sup>2</sup>,  
and Helmut Prendinger<sup>1</sup>

<sup>1</sup> National Institute of Informatics

2-1-2 Hitotsubashi, Chiyoda-ku, Tokyo 101-8430, Japan

{k-gajananan, arturonakasone, helmut}@nii.ac.jp

<sup>2</sup> Junior Research Group for Protein-Protein Interactions and Computational Proteomics  
Center for Bioinformatics, Saarland University  
Building E 1.1 , P.O. Box 15 11 50, 66041 Saarbrücken, Germany  
anhi@bioinf.uni-sb.de

**Abstract.** Standalone bio-molecular modeling applications with rich functionalities are already abundant. Since they are developed as single-user applications, those tools cannot be used for remote collaboration, where users want to perform bio-molecular structural modeling tasks without being in the same place physically. Even though other means of collaboration such as emails or chat messaging exist, they are or not real-time (e.g. e-mail) or do not support a shared experience of graphical objects (e.g. Instant Messaging). On the other hand, the three-dimensional Internet (3D Internet) is an emerging technology for building world-like environments for real-time collaboration. Therefore, in this paper, we will integrate a standalone bio-molecular application (BALL-View) with an online collaborative virtual world application (OpenMol) to extend the benefits both applications.

**Keywords:** Virtual Worlds, OpenSimulator, Molecular Modeling, Participatory Science, 3D Internet.

## 1 Introduction

Collaboration is an indispensable part of scientific endeavor. Online collaborative working environments can support a group of individuals to work together regardless of their geographical locations. Among the online collaborative working environments, the 3D Internet [21] refers to an online three-dimensional (3D) world-like environment like “Second Life” [24] or OpenSim [18] that provides an effective platform for building real-time collaborative working spaces for social or scientific purposes. A platform based on the 3D Internet can be used to discuss and analyze scientific data in real-time, thus enabling effective collaboration and learning. Users of such an environment can easily participate in any scientific endeavors that are simulated in the environment, by just using a computer with Internet facility. This leads to a novel form of ‘participatory science’ in virtual worlds [20].

We have built such an online collaborative three-dimensional world-like platform called OpenScienceSim [17], based on the OpenSim [18] world simulator, an open source alternative to the popular Second Life [24] platform, for scientific collaborative visualization in 3D virtual worlds. The OpenScienceSim platform supports an immersive experience of science, real-time collaboration, and participatory science. In this platform, we aim to develop applications in several scientific areas, including astrophysics [14], agriculture [19], bio-safety training, developmental biology, and others. One main application area considered in OpenScienceSim is bio-molecular science. Bio-molecular science includes molecular visualization, molecular modeling and molecular dynamics. Molecular visualization is the process of interpreting the graphical representation of molecules. Molecular modeling is a general term that refers to theoretical methods and computational techniques to model molecules, to compute their properties, or to mimic their behavior. Molecular dynamics involves simulations that are used to simulate the behavior of molecules under certain conditions for a certain period of time.

As part of the OpenScienceSim platform, we have already developed an online 3D virtual world application, called OpenMol [16, 4] for bio-molecular science. This application was developed by integrating a state of art molecular dynamics computation library called GROMACS [1]. The main focus of this application was molecular visualization and molecular dynamics simulation in a virtual world. OpenMol supports molecular visualization for users (visitors) of the OpenScienceSim virtual world, who enter the world in the form of avatars (graphical self-representations of users). OpenMol uses ball-and-stick and back-bone graphical representations for molecular visualization. In OpenMol, molecular dynamics simulation can be computed by delegating the task to the integrated molecular dynamics computation library GROMACS and can be visualized within the virtual world by using the computed results from GROMACS.

However the interface of the OpenMol application is limited and it has little features with regard to molecular modeling. On the other hand, there exist a large number of standalone bio-molecular modeling applications with rich modeling functionalities. These applications are developed mostly as single-user applications. However, researchers often have to collaborate remotely with other experts to perform and discuss task in bio-molecular modeling. Current bio-molecular modeling software has insufficient functionality to handle online (real-time) collaboration, and has to rely on e-mail exchanges with image attachments or Instant Messaging for collaboration. Those tools do not provide real-time collaboration with shared sources, such as the (shared) visualization of molecular structures.

Therefore, we extend the benefits of each application – single-user application and virtual world application – by integrating a standalone application with the OpenMol application. In particular, our approach to building a collaborative bio-molecular modeling tool integrates an existing powerful standalone tool for bio-molecular modeling called BALLView [12], which has rich molecular modeling functionalities, with OpenMol. In this way, users can collaboratively engage in bio-molecular modeling activities as avatars in the virtual world.

In particular, we enable the functionalities of BALLView to be operable from virtual world by synchronizing the molecular structural data in both applications. The integration extends BALLView to an online collaborative modeling tool, while at

same time increasing the molecular modeling features available for the OpenMol application in the virtual world. The integration of BALLView (especially the molecular modeling functionality) with OpenMol may lead to effective collaboration among the users who involve in a bio-molecular science task. The tasks cover molecular modeling, visualization, and molecular dynamics. In this paper, we will focus on molecular modeling. The resulting solution is addressed to experts in bioinformatics but also to lay people with interest in bio-molecular science.

The rest of the paper is organized as follows. In Sect. 2 we will review related work. In Sect. 3, we will present the overall architecture of the OpenMol application and details of how we integrated the other tools into OpenMol. Section 4 details the collaborative molecular functionalities of OpenMol. In Sect. 5 we compare our application to existing work (mentioned in Sect. 2) in more detail. Finally, we will summarize the contributions of the work, describe its limitations, and point at future enhancements.

## 2 Related Work

Molecular modeling applications for single users already exist and collaborative molecular modeling systems for multiple users also available. For example, BALLView [12] is an open source tool for molecular modeling and viewing mainly used for research in structural bioinformatics. It provides state-of-the-art visualization capabilities and powerful modeling functionalities including implementations of force field methods and continuum electrostatics models.

In the following, we will report on some available systems for collaboration in molecular science. After we have described our own approach in Sections 3 and 4, we will provide a more detailed comparison to existing approaches in Sect. 5.

The SnB Visualizer [25] offers synchronous collaborative viewing and modifying of protein structures. Further, it provides rich molecule modeling capabilities and visualization of different graphical representations of molecules. An interesting aspect of SnB Visualizer is that users can have a synchronous or asynchronous 3D view of the same molecule.

AMMP-Vis [6] is a collaborative multi-view virtual environment for molecular visualization and modeling that contains an integrated molecular dynamics simulator. This application also addresses the problems of awareness of presence, attention and action that are important to effectively collaborate in a virtual environment. The awareness of presence is realized by representing the users with “hand-type avatars”. The awareness of attention is achieved by using bounding boxes, which can be seen by all the other users, to indicate the area of interest to focus on a certain part of molecular structure.

Molecular visualizations already exist in virtual worlds based on Second Life. For example, the Monolith [13] application can display molecular structures within Second Life, whereby molecular data is obtained automatically from the RCSB Protein Data Bank [2]. With the Second Life built-in real-time voice communication and Instant Messaging, this application allows students and researchers to share knowledge in a collaborative environment.

Another tool for molecule visualization in Second Life is the open source Molecule Rezzer Scripts [11] which is described on the SciLands [26] website, a mini-continent and user community in Second Life related to science and technology. These scripts allow loading 3D models of molecules using balls, sticks, or both. IBM made some modifications to these scripts and used them to display a huge molecule (Rhodopsin) with thousands of atoms. The first step towards visualization of protein docking in virtual worlds is evaluated in the Bradley lab at Drexel University [10].

The External Simulator Bridge (ExtSim) [7] is a project that presents a plug-in for OpenSim. With this plug-in, objects within OpenSim can be controlled from external data feeds. According to the website a visualization of proteins is already implemented. Planned features are in-world controls for changing scale and position of atoms and improvements of the visualization.

A virtual worlds application that has molecular dynamics features could not be found except in the virtual world Project Wonderland [22] from SUN Microsystems, which was used as a platform for a project [8] in the area of molecular dynamics. The simulation was also calculated with GROMACS but the focus of that work was on the visualization of molecular dynamics. The approach of this project is very promising and gave already early insights to use virtual worlds for visualization of molecular dynamics.

### 3 Collaborative Platform for Bio-Molecular Science

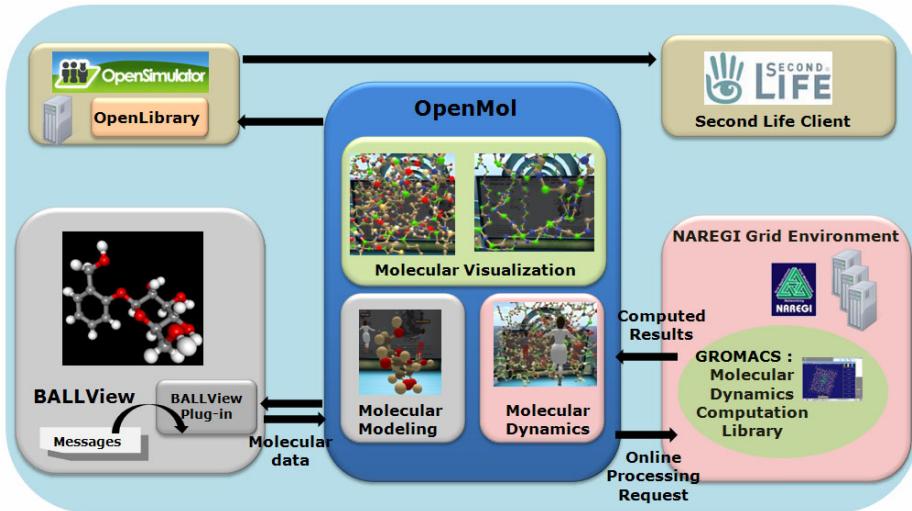
#### 3.1 Architecture of OpenMol

In order to develop a collaborative molecular modeling tool for bio-molecular science in the virtual world, we integrate the BALLView application with the OpenMol application. As previously mentioned, OpenMol was already developed with molecular dynamics functions, by integrating GROMACS to enable molecular dynamics features, and is available in the OpenScienceSim virtual world. GROMACS provides the computed results of molecular dynamics for input data which is in the form of standard molecular data, such as the RCSB Protein Data Bank (PDB) [2].

As shown in Fig. 1, we use the OpenSim 3D application server that allows us to develop the OpenScienceSim virtual world. OpenScienceSim can be accessed through a variety of clients on multiple protocols [18]. For example, we use the Second Life client to visualize simulations in the virtual world. We have already developed a framework called OpenLibrary [17], which is a simple, flexible, and fully functional interface to create and manipulate avatars and objects in the virtual world based on the OpenSim server.

The OpenMol application was developed on top of the OpenLibrary framework so that OpenMol can easily interface with the OpenSim platform. The molecular visualization component of OpenMol uses the available interface functions of OpenLibrary to create a 3D molecular structure in the OpenSim server based on the PDB molecular data structures. These 3D molecular structural representations are then visualized in the OpenScienceSim virtual world using the Second Life client.

The molecular dynamics component of OpenMol is responsible for generating the animation of molecular dynamics based on the computed results from GROMACS and the internal molecular data structures. Since the computational cost of calculating molecular dynamics is high, we run GROMACS on the NAREGI Grid middleware,



**Fig. 1.** Architecture of the OpenMol application

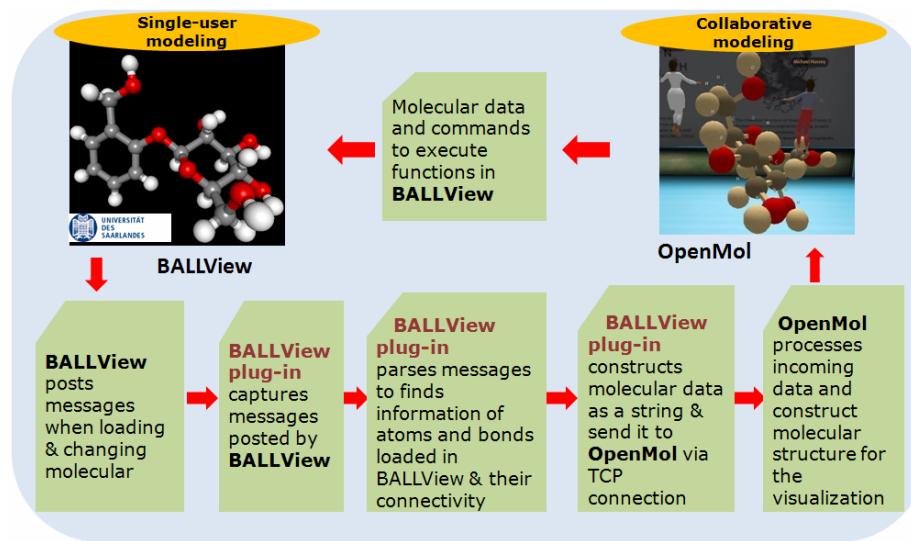
the National Research Grid Initiative in Japan [15]. These functions were already developed as part of initial phase of the OpenMol development.

For the work described in this paper, we developed a molecular modeling component which is in charge of processing incoming molecular data from the BALLView plug-in (rather than PDB) and converts it into the internal molecular data structures of OpenMol. In the next section, we will describe how we integrated BALLView with OpenMol.

### 3.2 The Integration of OpenMol with BALLView

BALLView [12] is a standalone molecular visualization and modeling tool that uses the functionalities of BALL [3]. BALLView is an object oriented application framework designed for rapid software prototyping in computational molecular Biology, molecular modeling and structural bioinformatics [12]. Since BALLView has a plug-in feature, we developed a new plug-in for BALLView. The main steps of the integration are depicted in Fig. 2.

The core function of the plug-in is to capture the messages posted by BALLView software. These messages are posted when there is a new molecular structure loaded or when there is change in the molecular structure which is already loaded. Then, the plug-in parses the captured messages into molecular data as atoms, bonds and their properties, and connectivity. The plug-in communicates with the OpenMol application that runs on OpenSim platform using a simple and non-optimized string based protocol. Using this communication mechanism, the plug-in handles the transfer of data from BALLView to OpenMol; in the other direction, OpenMol transfers molecular data and issues commands to execute functions in BALLView, so that changes in the molecular structure displayed in virtual world will be reflected in BALLView. Therefore, this integration leads to a bi-directional communication which helps synchronizing the changes in molecular structures in BALLView and OpenMol.

**Fig. 2.** The integration approach

From BALLView to OpenMol & From OpenMol to BALLView

- ADD\_ATOM = 0  
0; atom\_index, Element, position, positiony, positionz, radius, colorR, colorG, colorB
- ADD\_BOND = 1,  
1; bond\_index, atom\_one\_index; atom\_two\_index, bond\_order
- REMOVE\_ATOM = 2,  
2; atom\_index
- REMOVE\_BOND = 3,  
3; bond\_index, atom\_one\_index, atom\_two\_index
- UPDATE\_ATOM = 4  
4; atom\_index, Element, position, positiony, positionz,
- UPDATE\_BOND = 5  
5; bond\_index, atom\_one\_index, atom\_two\_index, bond\_order

Only from OpenMol to BALLView

- RUN\_MINIMIZATION = 6  
6;
- SATURATE\_FULL\_WITH\_HYDROGENS = 7  
7;
- MD\_SIMULATION = 8  
8;

**Fig. 3.** The communication protocol

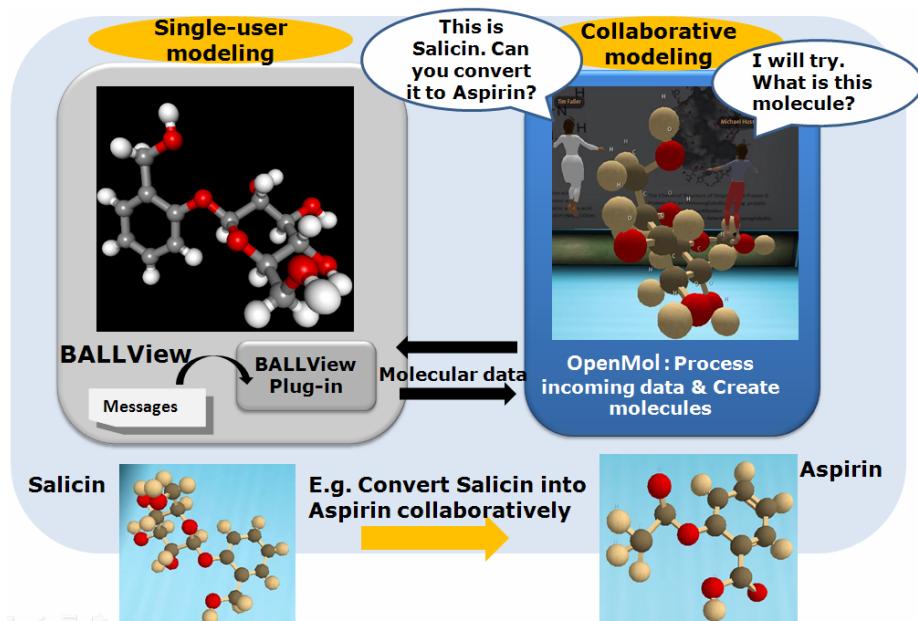
We designed and developed a protocol to communicate information back and forth between TCP Server components in the BALLView plug-in and OpenMol. The protocol shown in Fig. 3 refers to the communication link from the TCP server in the BALLView plug-in to the TCP server in the OpenMol application. The same idea is applied for the other direction of communication. In addition to the above protocol definition, we also developed another protocol definition which refers to the communication link from only the TCP server in OpenMol application to the TCP server in the BALLView plug-in. This enables us to execute molecular dynamics functions in BALLView.

## 4 Functionalities for Collaborative Molecular Modeling

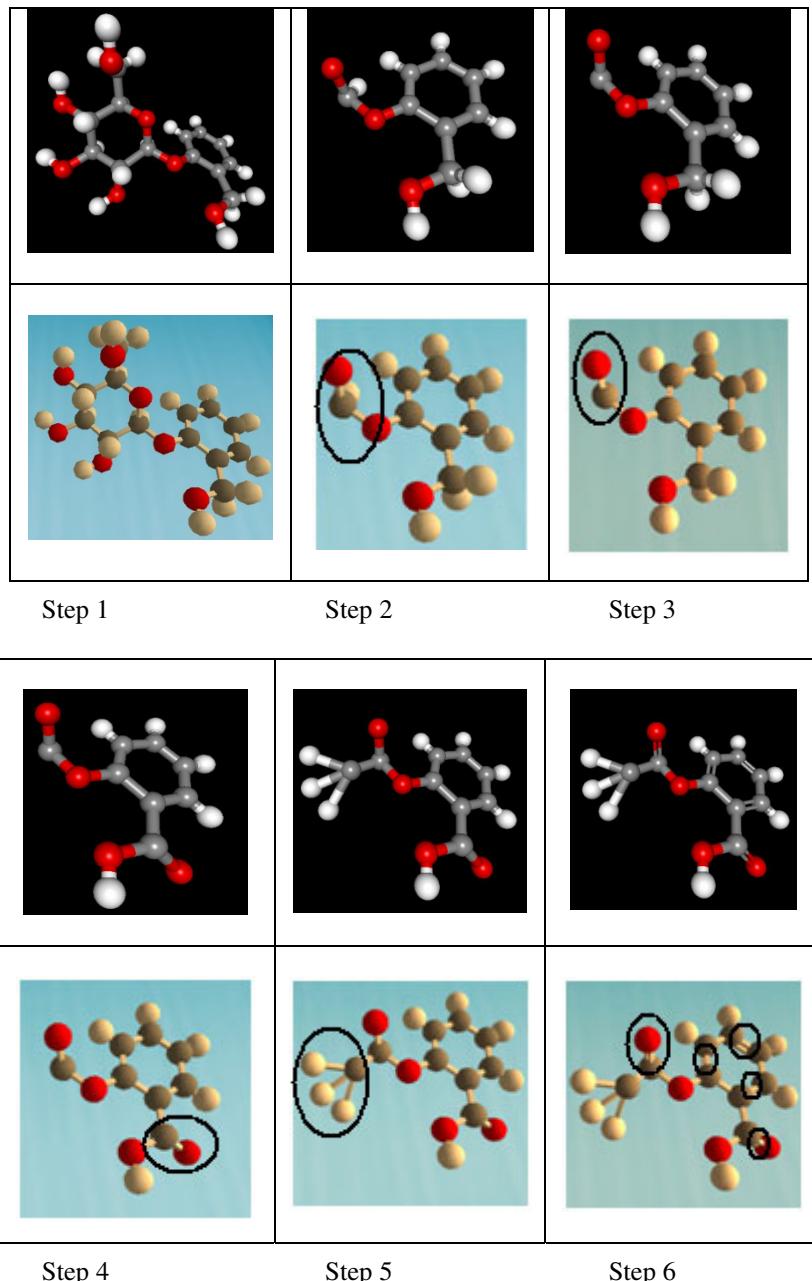
OpenMol features three sets of functionalities, including collaborative molecular viewing, collaborative molecular modeling, and collaborative molecular dynamics. In this paper, we focus on collaborative molecular modeling based on the integration of BALLView and OpenMol.

Currently, OpenMol only supports the ball-and-stick graphical representation for molecules. However, we provide two levels of representation. The first one represents the molecular structure as a full (“all-atom”) structure with all atoms and bonds. Another level represents the back-bone structure only.

Figure 4 gives an overview of the integration of single-user modeling with BALLView and collaborative modeling with OpenMol based on our plug-in architecture.



**Fig. 4.** Main functionality: collaborative bio-molecular modeling in virtual world



**Fig. 5.** Converting Salicin to Aspirin as an example of a collaborative molecular modeling task

In Fig. 5, we illustrate the steps required in order to convert the molecular structure of Salicin (an anti-inflammatory agent) into the molecular structure of Aspirin (a popular pain killer). In each step, the users, i.e. avatars, perform the molecular modeling task in the virtual world (shown in the lower picture of each step), which will be reflected in BALLView (shown in the upper picture of each step).

- Step 1: Load Salicin molecule from a source file
- Step 2: Identify the cyclic glucose and delete all atoms of the glucopyranose except for the ring oxygen and the carbon that is part of the bridge to the other ring
- Step 3: Delete the hydrogen that is bonded to the carbon between the two oxygens, where the glucopyranose was just deleted.
- Step 4: Find the carbon that is bonded to the hydroxyl group, which has two hydrogen atoms. Delete one and change the other one into oxygen.
- Step 5: Now, go back to carbon atom between the two oxygens, add a methyl group (a carbon and three hydrogens)
- Step 6: Assign alternating single/double bonds to the benzol ring, Set the bonds that connects single oxygens to the molecule, to bond order double and optimize the structure.

## 5 Comparison to Other Collaborative Systems

The main differences of OpenMol compared to these molecular viewers are (1) the real-time collaborative aspect and (2) the immersive virtual world that can be easily accessed by multiple users at anytime from anywhere using the common SecondLife viewer. In OpenMol users are represented in an intuitive way, as human-like avatars, and can profit from various communication methods like text messages, voice chat for private or public communication, or even non-verbal communication within an immersive environment with a sense of co-presence [5]. Furthermore, OpenMol is a persistent application in the virtual world, which is active even when no user is logged in the virtual world. Therefore, the users can log in and out and the last state of OpenMol is always saved.

With regard to collaboration, the SnB Visualizer [25] and OpenMol are quite similar. Both provide a real-time collaborative environment where changes performed by a user are immediately reflected in the other client viewers. By contrast to the SnB Visualizer, which uses Instant Messaging for communication among users, OpenMol is avatar-based and thus offers a more immersive environment and sense of co-presence [5]. Since OpenMol exist in the virtual world which has several ways for communication, that even more emphasize the co-presence of other users.

OpenMol is similar to AMMP-Vis [6] as both provide a virtual online collaborative environment. While AMMP-Vis represents users with “hand-only” avatars and bounding boxes, OpenMol employs human-like avatars and thus offers more natural communication. Moreover, the main communication means of AMMP-Vis is voice using Skype and chat communication [9]. In OpenMol, on the other hand, users can additionally profit from non-verbal communication, such as communicative and deictic gestures, in a shared experience space.

Although there are many molecular science initiatives in the Second Life virtual world, there is no solution which can support molecular visualization, modeling and dynamics.

## 6 Discussion and Conclusions

In this paper, we have introduced a collaborative molecular modeling application in a 3D virtual world based on OpenSim technology. The application is an integration of the popular standalone molecular modeling application BALLView and the virtual world molecular science application OpenMol. We have described the architecture of the integrated system and its main functionality of the application.

Current state-of-the-art stand-alone molecular visualizations tools provide richer visualization capabilities than OpenMol. These molecular viewers are similar to OpenMol as all of them have the ability to visualize molecules; however, OpenMol is based on the OpenSim world simulator and the Second Life viewer (or some functionally similar viewer like Hippo), which limits the freedom of displaying objects to primitive graphical objects rather than meshes. On the other hand, these molecular viewers support various formats of input files, such as PDB, MOL, MOL2, HIN and SDF; whereas OpenMol supports only the PDB format and other similar restricted formats.

Our system has some limitations in the visualization of different molecular structures as only the ball-and-stick type representations are supported by most virtual world viewers like the Second Life viewer. The realXtend viewer [23] for OpenSim can also display meshes, but we have not yet connected this specialized viewer in our environment.

The synchronization of the data between OpenMol and BALLView is currently affected by the lagging communication between OpenSim server and client. It is therefore required to perform the modeling task slowly so that all the data are correctly synchronized. Furthermore, not all the functions in BALLView can be executed from OpenMol, because not all the functions in BALLView can be invoked from the BALLView plug-in, hence not from OpenMol either. In the future, it will also be desirable if the parameters (e.g. force field of atom) can be set up from virtual world before performing the molecular modeling task. Currently, default parameters are used.

The example in Sect. 4 indicates the usefulness of extending the benefits of the standalone molecular modeling software BALLView by a collaborative platform. These days, we are performing an empirical study with the aim of investigating the value of a collaborative space in addition to BALLView. In the study, we define an experimental condition where students perform a molecular modeling task in BALLView integrated with OpenMol. Students receive instructions from an instructor in the collaborative space of the virtual world. The collaborators can also use an Instant Messaging (IM) tool (e.g. Skype) for communication. Further, we defined a control condition where students perform a molecular modeling task in a standalone BALLView while communicating with the instructor using only IM. This study is expected to demonstrate the usefulness of the OpenMol collaborative platform for molecular modeling. We will report the results in the future.

## Acknowledgments

The research was partly supported by a Grand Challenge Grant from the National Institute of Informatics (NII), Tokyo, and by the NII National Internship Programme.

## References

1. Berendsen, H.J.C., Van Der Spoel, D., Van Drunen, R.: Gromacs: A message-passing parallel molecular dynamics implementation. *Comp. Phys. Comm.* 91, 43–56 (1995)
2. Berman, H., Battistuz, T., Bhat, T., Bluhm, W., Bourne, P., Burkhardt, K., Feng, Z., Gilliland, G., Iype, L., Jain, S., Fagan, P., Marvin, J., Padilla, D., Ravichandran, V., Schneider, B., Thanki, N., Weissig, H., Westbrook, J., Zardecki, C.: The Protein Data Bank. *Acta Crystallographica Section D* 58(6), Part 1, 899–907 (2002)
3. Boghossian, N., Kohlbacher, O., Lenhof, H.: BALL: Bio-chemical Algorithms Library. *Algorithm Engineering*, 331–345 (1999)
4. Budde, A.: Conception and implementation of a virtual worlds framework for participatory science and evaluation based on a bioinformatics application. Master's Thesis, University of Augsburg and National Institute of Informatics (2009)
5. Casanueva, J., Blake, E.: The Effects of Avatars on Co-presence in a Collaborative Virtual Environment. Tech. report, Dept. of Computer Science, Univ.of Cape Town, South Africa (2001)
6. Chastine, J., Zhu, Y., Brooks, J., Owen, S., Harrison, R., Weber, I.: A Collaborative Multi-View Virtual Environment for Molecular Visualization and Modeling. In: Proceedings of the Coordinated and Multiple Views in Exploratory Visualization, CMV 2005, Washington, DC, USA, pp. 77–84. IEEE Computer Society, Los Alamitos (2005)
7. ExtSim: External Simulator Bridge for OpenSimulator, <http://www.sciencesim.com/wiki/doku.php/extsim> (accessed, July 2009)
8. Kamper, U.: Bachelor's thesis: 3D-Visualisierung und Simulation von Biomolekülen innerhalb der 3D-Umgebung Project Wonderland (accessed, July 2009), <http://proteomics-berlin.de/154>.
9. Ma, W., Zhu, Y., Harrison, R.W., Owen, G.S.: AMMP-EXTN: Managing User Privacy and Cooperation Demand in a Collaborative Molecule Modeling Virtual System. In: Virtual Reality Conference, VR 2007, March 2007, pp. 301–302. IEEE, Los Alamitos (2007)
10. Molecular Docking in Second Life, <http://usefulchem.blogspot.com/2007/06/molecule-docking-in-second-life.html> (accessed, July 2009)
11. Molecule Rezzer Scripts for SecondLife, <http://www.scilands.org/2007/04/01/molecules> (accessed, July 2009)
12. Moll, A., Hildebrandt, A., Lenhof, H., Kohlbacher, O.: BALLView: An object-oriented molecular visualization and modeling framework. *Journal of Computer-Aided Molecular Design* 19(11), 791–800 (2005)
13. Monolith: Molecular Visualization for Second Life, <http://www.scilands.org/2007/04/01/molecules> (accessed, July 2009)
14. Nakasone, A., Prendinger, H., Holland, S., Hut, P., Makino, J., Miura, K.: AstroSim: Collaborative visualization of an astrophysics simulation in Second Life. *IEEE Computer Graphics and Applications* 29(5), 69–81 (2009)
15. NAREGI, [http://www.naregi.org/index\\_e.html](http://www.naregi.org/index_e.html) (accessed, July 2009)
16. OpenMol, <http://www.youtube.com/watch?v=xTxgqBvHEuw> (accessed, January 2010)

17. OpenScienceSim, <http://www.globallabproject.net/> (accessed, January 2010)
18. OpenSimulator, <http://www.opensimulator.org/> (accessed, November 2009)
19. Prada, R., Dias, D., Prendinger, H., Nakasone, A.: Fostering agricultural environmental awareness. In: Proc. 2nd Int'l. Conf. on Games and Virtual Worlds for Serious Applications (VS GAMES-10), Braga, Portugal, March 2010. IEEE Press, Los Alamitos (2010)
20. Prendinger, H.: The Global Lab: Towards a virtual mobility platform for an eco-friendly society. Trans. of the Virtual Reality Society of Japan 14(2), 163–170 (2009)
21. Prendinger, H., Ullrich, S., Nakasone, A., Ishizuka, M.: MPML3D: Scripting Agents for the 3D Internet. IEEE Trans. on Visualization and Computer Graphics (2010)
22. Project Wonderland: Toolkit for Building 3D Virtual Worlds, <https://lg3d-wonderland.dev.java.net> (accessed, July 2009)
23. RealXtend, <http://www.realxtend.org/page.php?pg=main> (accessed, January 2010)
24. Second Life, <http://secondlife.com/> (accessed, November 2009)
25. Shake-and-Bake Scientific Visualization Software,  
<http://www.cse.buffalo.edu/ag28/snbvis> (accessed, June 2009)
26. The SciLands: A mini-continent and user community in the virtual world platform Second Life devoted exclusively to science and technology, <http://www.scilands.org> (accessed, July 2009)

# The Effects of Finger-Walking in Place (FWIP) for Spatial Knowledge Acquisition in Virtual Environments

Ji-Sun Kim<sup>1,\*</sup>, Denis Gračanin<sup>1,\*\*</sup>, Krešimir Matković<sup>2,\*\*\*</sup>, and Francis Quek<sup>1,†</sup>

<sup>1</sup> Virginia Tech, Blacksburg, VA 24060, USA

<sup>2</sup> VRVis Research Center, Vienna, Austria

**Abstract.** Virtual environments (VEs) can be used to study issues related to human navigation, such as spatial knowledge acquisition. In our prior work, we introduced a new locomotion technique (LT), named “Finger-Walking-in-Place (FWIP)”, for navigation tasks in immersive virtual environments (IVEs). The FWIP was designed to map human’s embodied ability for real navigation to finger-based LT. A two-hand based implementation on a multi-touch device (i.e., Lemur) was evaluated. In this paper, we introduce the one-handed FWIP refined from the original design, and its implementation on a Lemur and an iPhone/iPod Touch. We present a comparative study of FWIP versus the joystick’s flying LT to investigate the effect of the mapping of the human’s embodied ability to the finger-based LT on spatial knowledge acquisition. This study results show that FWIP allows the subjects to replicate the route more accurately, compared to the joystick LT. There is no significant difference in survey knowledge acquisition between two LTs. However, the results are useful, especially given that the FWIP requires small physical movements, compared to walking-like physical LTs, and has positive effect on route knowledge acquisition, compared to the joystick LT.

## 1 Introduction

Interaction techniques for navigation in virtual environments (VEs) are called “locomotion techniques” or “traveling techniques”. Since our study is focused on the user’s action/activity (i.e., locomotion in VEs), rather than the task (i.e., traveling in VEs), we use the term “locomotion techniques”. LTs can be divided into natural LTs and abstract LTs based on the mapping method between users’ input actions and locomotion control in VEs although there are several classifications of LTs.

Natural LTs are usually designed by directly using natural locomotion methods with least modification as much as possible. Examples include walking-like physical LTs and simulator-based LTs.

Abstract LTs are designed by mapping users’ input actions abstractly to locomotion (output) in VEs. For example, when you want to move forward in VEs, an abstract LT can be designed by mapping the action, “pressing a button”, to the “moving forward”

---

\* hideaway@vt.edu

\*\* gracanin@vt.edu

\*\*\* matkovic@vrvs.at

† quek@vt.edu

control. Abstract LTs are usually realized with a keyboard or mouse for desktop VEs, or a joystick/wand device for immersive VEs (IVEs). A flying interaction technique with a joystick [3] is a type of abstract LTs, and commonly used to navigate in IVEs because of its simplicity and familiarity. Compared to natural LTs, most abstract LTs can be quickly designed and evaluated for a desired VE. In addition, there is much less body fatigue.

Walking-like physical LTs are a type of natural LTs, which are generally believed to support more accurate spatial knowledge acquisition by allowing users to use body-based senses (e.g., proprioception and vestibular cue) [15][20] with little or no cognitive mapping from the user. Walking in place (WIP) is a close match to the natural walking metaphor. There are various systems and interaction schemes to provide WIP-like LTs, e.g. WIP [17][18], the extensions (e.g., “Seven League Boots” [8]) and the use of treadmills [4][10]. These studies reported that users experienced higher levels of presence when using WIP. Thus, WIP is well-suited for VEs in which natural locomotion and a high sense of presence are required. However, these walking-like LTs still present several issues in terms of cost and usability [7]. The cheaper, simpler and more convenient LTs (i.e., abstract LTs) are preferred for most VE applications, while walking-like LTs are only used for special purposes, such as realistic training and rehabilitation. Such abstract LTs are often paired with navigation aids such as maps to give the user greater awareness of her spatial orientation in the virtual world [3]. Providing navigation aids requires for designers or researchers to make additional efforts in addition to developing an LT. Using those aids demands for users to spend extra cognitive load in addition to performing an LT.

Walking-like physical LTs are useful for spatial knowledge acquisition because they are based on our embodied resources [5][6] and over-learned body-based sensory and cognitive abilities, which are not available in abstract LTs. Can we leverage these abilities by using an alternative LT, rather than walking-like physical LT, for virtual navigation? To answer the question, we introduced an alternative LT, named “Finger-Walking-in-Place (FWIP)”, in our prior work [12].

Finger-based interaction techniques can be realized by using several approaches. A sensing-glove can be used to control animation characters [13]. This approach is more suitable to the cases that need more detailed information from the joints of the hand and fingers. As a different approach, touch-based devices can be used to select and manipulate virtual objects [2]. Even though it is hardly found that touch-based devices are used for navigation in VEs, we chose to implement our FWIP on a multi-touch device [12], by observing how treadmill-supported WIP works.

The implementation of FWIP on a Lemur [11] was evaluated in our previous study [12] that showed the similar action to treadmill-supported WIP, performed by fingers, can be used as robust interaction for virtual locomotion in an IVE (e.g., CAVE [19]). In this paper, we introduce the one-handed FWIP modified from the two-handed FWIP, and describe its implementation on a Lemur and iPhone/iPod Touch devices. We also present a comparative study of the introduced FWIP on a Lemur and an iPhone/iPod Touch versus the joystick LT to investigate whether our abilities learned for real navigation can be transformed to the alternate frame of finger-based LT.

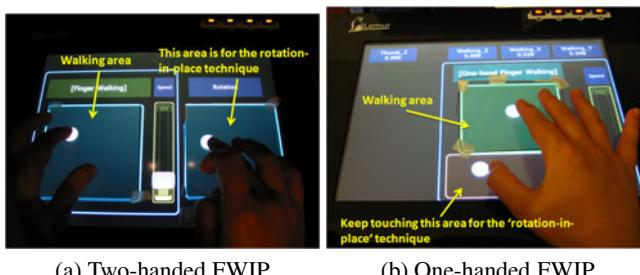
## 2 Two Locomotion Techniques

### 2.1 Finger-Walking-in-Place (FWIP) Technique

FWIP enables a user to navigate in a VE by translating and rotating a viewpoint as the user slides unadorned fingers on a multi-touch sensitive surface. In our prior work [12], three different viewpoint-rotation techniques ('walking', 'dragging' and 'jog-dialing') were introduced and separately operated from the viewpoint-translation technique. Since FWIP was designed to be separately operated on a multi-touch surface for viewpoint-translation and viewpoint-rotation, most participants used two hands to rotate and translate the viewpoint. We decided that two-handed operations are unnecessary because each technique for viewpoint-translation and viewpoint-rotation is touch-based. In addition, we observed that some of participants were confused with two separate operations, one assigned to each hand. Hence, we modified our original two-handed FWIP to one-handed FWIP combined with the 'dragging' viewpoint-rotation technique. 'Walking' and 'jog-dialing' techniques for viewpoint-rotation are excluded for one-handed FWIP because it is difficult that these two techniques are operated distinguishably from the 'walking' for viewpoint-translation. Figure 1 shows different user interface (UI) designs for two-handed FWIP (Figure 1(a)) and one-handed FWIP (Figure 1(b)) for implementation on the Lemur device.

Another implementation of the one-handed FWIP has been tested on iPhone/iPod Touch [11]. The smaller size of the touch-screen was considered to refine the one-handed FWIP by merging the walking area and the touching area (Figure 1(b)). We used two modes for usability tests, the control mode for evaluators and the walking mode for test subjects. Evaluators can control the system setup specific to the experiment (Figure 2(a)). A test subject can perform finger-walking on the multi-touch screen for virtual navigation (Figure 2(b)). In the pilot study, we observed that most subjects accidentally touched the 'back' button or touch the non-detectable area while they are walking without looking at the screen. We attached the rubber bands to limit the walking-area on the iPhone screen (Figure 2(b)). Thus, FWIP can be applied to multi-touch devices with different sizes.

Figure 3 illustrates the final design of FWIP. For viewpoint-translation, FWIP traces the trajectory of one-finger movement on the surface, as shown (Figure 3(a)). Until the touch ends, a user's viewpoint is continuously translated in a virtual world using the



**Fig. 1.** Interface design for the two-handed and the one-handed FWIP LTs

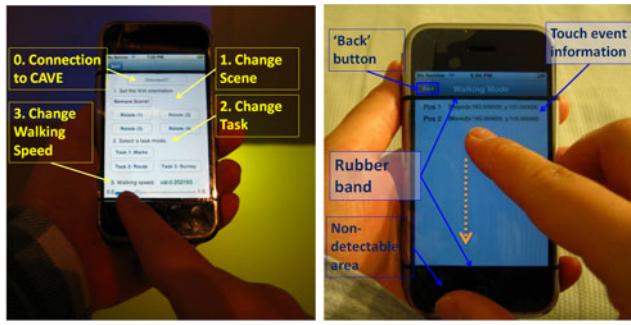


Fig. 2. User interfaces for iPhone/iPod Touch

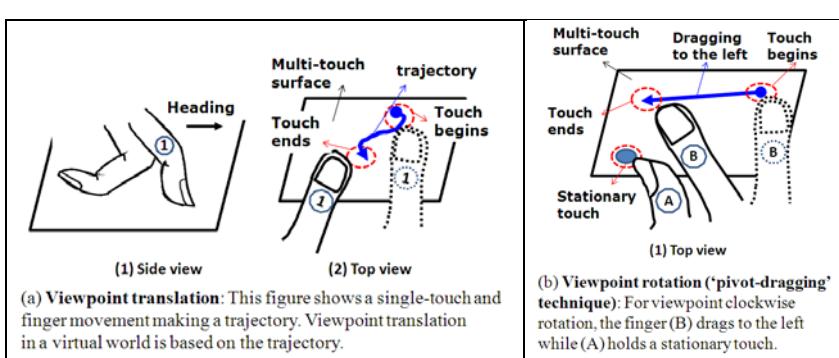


Fig. 3. The final design of Finger-Walking-in-Place (FWIP) technique

trajectory. The virtual locomotion speed can be controlled by the speed and frequency of finger movement. This speed control mechanism is the same one used when walking by controlling leg-swinging.

FWIP requires multi-touch for viewpoint-rotation. For viewpoint-rotation, FWIP is designed considering the hand-constraint that cannot be fully rotated in place, so that it should be discretely realized. This is also found in the real world for rotation-in-place. When we rotate in place in the real world, we do not usually have a full rotation at a step. The full rotation is discretely realized with several steps. While one finger (A) holds a stationary touch (i.e., ‘pivot’ touch), another finger (B) can drag to the right or to the left (Figure 3(b)). The dragging distance is used to determine the rotation angle. The faster dragging movement is the faster rotation changes. For the full rotation of a viewpoint, a user needs to repeatedly drag in the same direction. Since this rotation technique needs a ‘pivot’ touch by a finger and ‘dragging’ by another finger, we call this technique ‘pivot-dragging’ technique. Thus, the action of FWIP is very similar to that of treadmill-supported WIP in terms of relative position and spatial direction of the executing body parts.

## 2.2 Joystick-Based Flying

Joystick-based flying is a common LT in VEs. It is usually based on the direction of a (hand-manipulated) wand or based on the head orientation. For the comparative study, we used a technique based on the wand orientation to determine the traveling direction. In other words, the joystick's direction is decoupled from the direction where a user's head is aiming to give more freedom to the user to look around during navigation. The joystick on the wand is used to translate and rotate the viewpoint. The buttons on the wand are used to control the flying speed.

While the action of FWIP is repetitively executed for movement in VEs as that of WIP is, the action of the flying is relatively stationary because users keep pushing the little stick and they only have force feedback.

## 3 Comparative Study

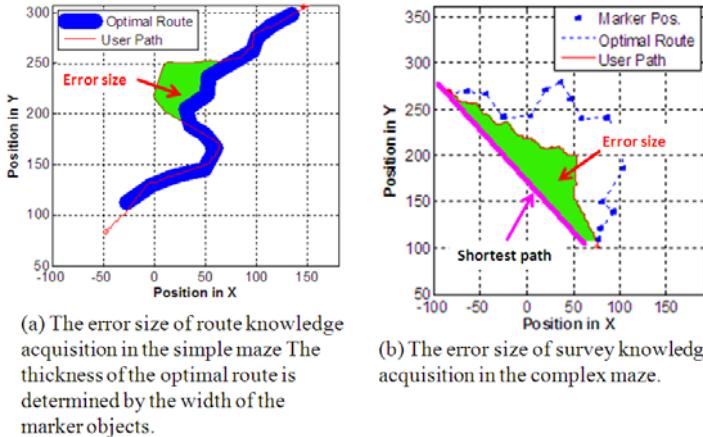
### 3.1 Methodology

We used the same experiment tasks and procedure presented by Peterson et al. [14], because their study was focused on the spatial knowledge acquisition. Their study was about the comparison of VMC and Joystick's flying techniques. In our study, the VMC is replaced with our FWIP and a multi-touch device.

Peterson et al. used maze-traveling which is generally used to investigate navigation performance in IVEs. The study showed that the experiment design is appropriate for a between-subjects study, in terms of the temporal size and the spatial size, considered of the exposure time inducing sickness symptoms in IVEs. The participants maneuvered in two virtual mazes with different complexities. The investigation was based on the Landmark-Route-Survey (LRS) model [16] that describes the process of how spatial knowledge is acquired and represented. Even though there are some arguments about the developmental sequence of LRS knowledge acquisition [9], the experiment design in [14] is reasonable to test whether or not subjects acquire spatial knowledge about a certain route, and the orientation from the entrance to the exit about the space. The results include maneuvering performance, route knowledge (RK) acquisition, and survey knowledge (SK) acquisition. Maneuvering performance is measured by the control precision; RK acquisition is measured by subjective confidence and the route replication result; SK acquisition is measured by subjective estimation of direction to the exit and a straight path length to the exit [14].

### 3.2 Performance Metrics

In order to investigate the effect of each LT on spatial knowledge acquisition in VEs, we decided to use two metrics, route knowledge acquisition accuracy and survey knowledge acquisition accuracy. These two metrics are measured by using the quantitative errors produced by subjects in two tasks, route replication task and spatial orientation estimation (the deviation from a straight-line traversal from the entrance to the exit) task.



**Fig. 4.** Two examples of the error sizes of route knowledge (RK) and survey knowledge (SK) acquisition

**RK acquisition error size:** The route error is measured as area between the optimal route (from the first marker to the last marker) and the path taken by the subject in the route replication tasks (Figure 4(a)). We denote this measure  $\mathbb{E}_{RK}$ .

**Survey knowledge acquisition error size:** The orientation estimation error is measured as area between the straight-line path (from the entrance to the exit) and the direct path taken by the subject (Figure 4(b)). We denote this measure  $\mathbb{E}_{SK}$ .

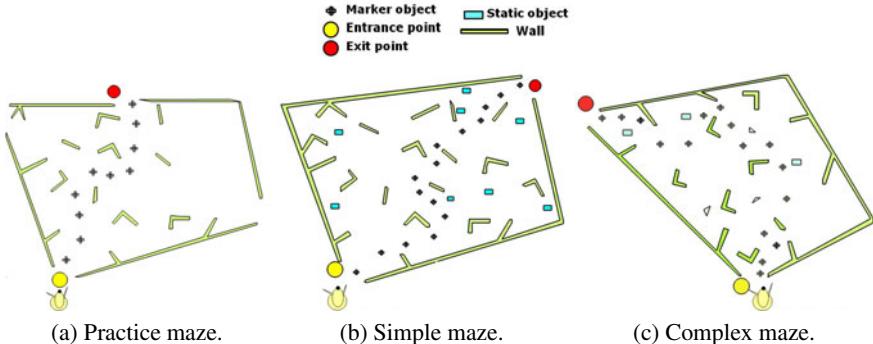
We chose the area between the optimal path and the one taken by a test subject to measure the performance as this is the cumulative deviation between the two loci. We excluded the distance traveled, which is used in [14] to evaluate SK acquisition, because it can be biased in some cases. For example, consider two users trying to find the shortest path. One of the users wanders a lot in a certain area close to the optimal path. The other user chooses a wrong direction, and travels far from the optimal path. If the first user had traveled longer distance than the second user, the distance traveled would not be an appropriate metric to evaluate their task performance.

### 3.3 Design

We used three mazes, including a practice maze, with the different complexities (Figure 5). These mazes are based on [14]. As the complexity of the mazes increases, more turns are required (Table I). The practice maze is used to familiarize the subjects with the experiment procedure used in the simple maze and the complex maze. We tried to eliminate any unnecessary head movement, such as looking down to find the markers. Consequently, the markers were taller than the subjects' height in a CAVE.

The experiment was performed in a CAVE [19] with a 10' by 10' Fakespace 4-wall CAVE display with shutter glasses, and an Intersense IS-900 VET-based head tracker.

Joystick subjects hold a wand device with a dominant hand and navigate by physically pointing the wand to indicate the forward direction of travel and employing the

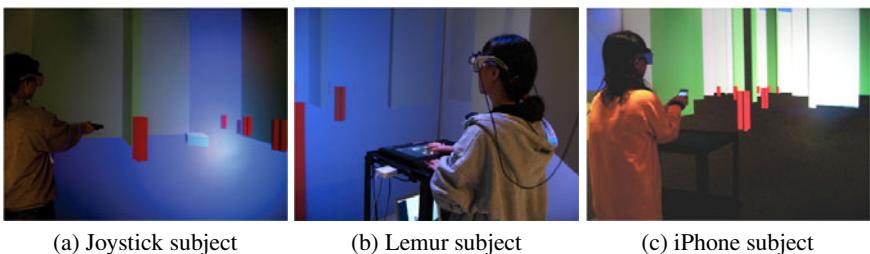


**Fig. 5.** Top views of three virtual mazes. Each maze includes marker objects and walls. The simple/complex mazes include static objects that can be used to remember their traveling paths

**Table 1.** The characteristics of the virtual mazes

	Practice Maze	Simple Maze	Complex Maze
Number of markers	10	15	16
Size (units)	265 × 185	245 × 195	230 × 230
Path length (units)	213	310	377
Cumulative angle to turn (degrees)	337	479	699
Fog effect	Yes	Yes	Yes

joystick on the wand to specify specific movements with-respect-to that forward vector (Figure 6(a)). Lemur and iPhone devices are used for the purpose of a finger-walking surface and touch/position detection, and the navigation direction is only determined by finger-movements. In order to conduct the experiment with the constraint that the FWIP subjects would not physically move in the CAVE immersive space, we placed the Lemur and the iPhone/iPod Touch on a table to provide a persistent spatial reference. The FWIP subjects would stand on a floor next to the table. While the Lemur subjects would use only one-hand (Figure 6(b)), iPhone/iPod Touch subjects would hold the device with the non-dominant hand, align it with the vertical line of the front wall in the CAVE space, and move their fingers with the dominant-hand on the screen surface (Figure 6(c)).



**Fig. 6.** Experiment setup in VT-CAVE

**Table 2.** Demographic data of the subjects

Data	JS Group	Lemur Group	iPhone Group
Mean age (years)	24.5 (Std=5.797)	20.563 (Std=1.999)	19.13 (Std=1.0888)
Gender	Female:8, Male:8	Female:8, Male:8	Female:8, Male:8
VE Experience	Novice N=10, Experienced N=6	Novice N=10, Experienced N=6	Novice N=15, Experienced N=1

### 3.4 Procedure

48 college students participated in this experiment. They were assigned to three different interaction groups: the joystick LT group (JS group), the Lemur-based FWIP group (Lemur group), and iPhone-based FWIP group (iPhone group). The subjects were asked to fill out the pre-experiment questionnaire including demographic questions, such as age, gender, and VE experience level (Table 2). The instructions were:

1. Travel along the pre-defined route with marker objects five times (to have the experience of the maze environment): During five trials, the subjects were asked to pass right through every marker object until they reach the exit. After each trial, the subjects were automatically moved back to the entrance point.
2. Estimation: After each trial, the subjects were asked how confidently they can estimate the direction to the exit and how confidently they can replicate the same route without marker objects.
3. Route replication (for RK acquisition): After five trials, the subjects had two trials to replicate the same route without visible marker objects.
4. Travel along the shortest path (for SK acquisition): After the route replication, the subjects had two trials to find the shortest path. When finding the shortest path, the subjects were allowed to walk through internal walls (no collision detection).

After all the tasks in three mazes, the post-experiment questionnaire obtained subjective responses to the experiment and free-form comments. The subjects were asked to describe the strategies employed to replicate the route and to find the shortest path to the exit. They were required to take a break after completing the tasks in each maze.

## 4 Analysis and Discussion

### 4.1 Results

We normalized each of our  $\bar{E}_{RK}$  and  $\bar{E}_{SK}$  using the largest error score, such that  $\bar{E}_{RK} = \bar{E}_{RK}/\max(\bar{E}_{RK})$  and  $\bar{E}_{SK} = \bar{E}_{SK}/\max(\bar{E}_{SK})$  in our data analysis.

**RK Acquisition:** Table 3 presents the mean error of each group and the results of our  $\bar{E}_{RK}$  analysis are shown in Figure 7.

**Simple Maze:** Since there were two outliers (one subject wandered too much and the other one was lost) in the JS group in the simple maze, we compared fourteen-subjects data for each interaction technique group. The mean error of the JS group is a little bigger compared to the other groups. Because three-group samples failed the normality test

**Table 3.** The  $\bar{E}_{RK}$  of the three groups

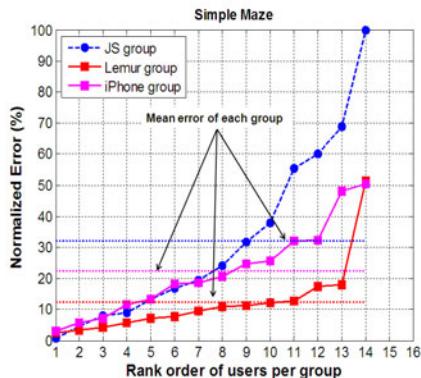
	<b>JS Group</b>	<b>Lemur Group</b>	<b>iPhone Group</b>
<b>Simple maze</b>	22.3(%)	8.63(%)	15.44(%)
<b>Complex maze</b>	50.1(%)	30.31(%)	31.45(%)

(Ryan-Joiner=0.789, 0.801, and 0.863, respectively,  $p < 0.05$ ), we used the Kruskal-Wallis non-parametric test ( $H$  statistic). This test shows significant difference among three groups' means ( $H=7.34$ ,  $p < 0.05$ ).

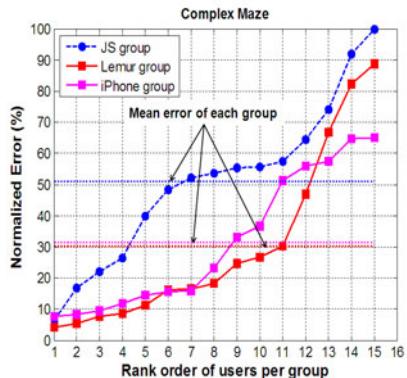
Figure 7(a) shows that nine subjects in the JS group rank below the mean error (22.3 %), while 11 subjects in the Lemur group rank below the mean error (8.63 %). In addition, 12-th and 13-th subjects rank very close to the mean error, which is not the case in the JS group. Figure 7(a) implies that the Lemur group performed evenly well against the JS group in the simple maze. On the other hand, the iPhone group is placed between the JS group and the Lemur group. Since the iPhone device should be held in non-dominant hand, its alignment may be sometimes off the vertical line of the front wall in the CAVE space. We conjecture that it may affect the task performance.

**Complex Maze:** The mean error of the JS group is a little bigger compared to the other groups. Since the Lemur group samples failed the normality test (Ryan-Joiner=0.911,  $p < 0.05$ ), we used the Kruskal-Wallis non-parametric test ( $H$  statistic). The statistical test showed no significant difference among the three groups. Figure 7(b) shows that the performance of RK acquisition was affected by the maze complexity.

Since we focus on the comparison of FWIP and joystick LTs, we are more interested in the two-group based results. When we compare the JS group vs. the Lemur group and the JS group vs. iPhone group using Mann-Whitney non-parametric test, the statistical tests show interesting results (Table 4). The table shows that the error size of the JS group is statistically greater than the error size of Lemur group in both mazes, while the



(a) Result in the simple maze



(b) Result in the complex maze

**Fig. 7.** The comparison of  $\bar{E}_{RK}$  across our three groups. We added the line between data points to easily compare the results of three groups (neither interpolation nor extrapolation)

**Table 4.** Statistical test results for the comparison of RK acquisition

	$ \mathbb{E}_{RK}(JS) - \mathbb{E}_{RK}(Lemur) $	$ \mathbb{E}_{RK}(JS) - \mathbb{E}_{RK}(iPhone) $
Simple maze	W=281, $p < 0.05$	W=245, $p > 0.05$
Complex maze	W=284, $p < 0.05$	W=277, $p < 0.05$

**Table 5.** The  $\bar{\mathbb{E}}_{SK}$  of the three groups

	JS Group	Lemur Group	iPhone Group
Simple maze	20.81(%)	21.85(%)	20.16(%)
Complex maze	54.27(%)	55.24(%)	57.0(%)

error size of the JS group is statistically greater than the error size of iPhone group in the complex maze.

**SK Acquisition:** We analyzed our  $\bar{\mathbb{E}}_{SK}$  dataset in the same way we did the  $\bar{\mathbb{E}}_{RK}$  evaluation. The statistical test showed no significant difference in JS vs. Lemur groups and JS vs. iPhone groups. Table 5 shows that the SK acquisition is not much affected by the interaction technique but the maze complexity.

## 4.2 User Behaviors

We observed that most users are focused on trying to find their better strategies to complete route replication and shortest path finding tasks. They showed a common strategy which they tried to use landmarks knowledge developed during the first five trials with marker objects. Some joystick users tried to change their physical postures/movement (e.g., physical body-rotation by fixing the wand's position on the body-chest, only use of the device without physical body-rotation, and horizontal swing of the arm as well as physical body-rotation). Some FWIP users tried to memorize the number of steps and turns at specific positions (e.g., original positions of the marker objects) in relation to some static objects (e.g., box objects or walls). Thus, we may assume that using the same action of that of walking may help the users to recall some wayfinding strategies that might already be learned from the real world.

## 4.3 Discussion

This experiment showed that the Lemur group's users acquired more accurate route knowledge in both mazes and the iPhone group's users acquired more accurate route knowledge in the complex maze, compared to what the joystick group's users did. This result implies that there are some benefits to remember and recall the subjects' route knowledge when using FWIP for navigation in VEs. In other words, our embodied resources related to spatial knowledge acquisition can be utilized by FWIP. This result is also useful, especially given how far FWIP is removed from actual walking and turning.

However, there is no significant difference for survey knowledge acquisition in the Lemur versus the joystick groups and the iPhone versus and joystick groups. Regarding this, we realized that survey knowledge is usually acquired from more exploration in

an environment. In our experiment we provided insufficient exploration opportunity to users for survey knowledge acquisition in each maze. We need further experiment to measure survey knowledge acquisition with some methodological improvements, such as providing more exploration opportunity of the maze (e.g., traveling several different routes or searching several objects placed in the maze).

We also found that our experiment design included some confounding factors as follows,

- The rotation technique of the iPhone-based FWIP is different from that of the Lemur-based FWIP due to some time-constraint at the time when we performed the experiment with the iPhone-based FWIP.
- Since the iPhone subjects held the device in non-dominant hand, its heading direction may be sometimes off the vertical line of the front wall in the CAVE space.
- While the users in the JS group kept holding a device, the users in the Lemur group used a table to place the Lemur device.
- For the wand device (to which the joystick is attached), the absolute angle of rotation is determined by a tracking system. Hence, hand rotation and body rotation cannot be distinguished. Body rotation has concomitant direction implication which hand rotation does not. We allowed only joystick-users to physically rotate in place because we understand that this conflation is typical for wand/joystick users without adequate understanding of how this conflation influences 3D interaction. On the other hand, the action of the FWIP for rotation has some constraints, compared to WIP and the joystick's flying because we cannot fully rotate our wrist on which the fingers depend.

In order to thoroughly investigate the effect of FWIP on spatial knowledge acquisition, we need to remove these factors in the next experiment.

## 5 Conclusion and Future Work

We described a touch-based, one-handed FWIP and its implementation on a Lemur and an iPhone/iPod Touch. We conducted a comparative study of FWIP versus the joystick's flying LT to investigate the effect of the mapping of the human's embodied ability to the finger-based LT on spatial knowledge acquisition.

The basic finding of this study is that FWIP designed by the similar action to that of walking helped the subjects to acquire more accurate route knowledge in virtual mazes with different complexity, showing that this mapping may provide positive effect on human spatial knowledge acquisition in VEs. In order to support this observation, we will find some theoretical foundations as well as to perform further experiment.

In Introduction Section, we raise a question, “can we leverage these abilities by using an alternative LT, rather than walking-like physical LT, for virtual navigation?”. Even though the study result shows some positive effect, we cannot fully answer to this question. In order to show that the effect of FWIP on spatial knowledge acquisition would not be significantly different from that of walking-like physical LTs for spatial knowledge acquisition, we will perform another type of comparative study, i.e., FWIP versus walking-like LTs (e.g., WIP or walking).

## References

1. Apple Computer, Inc. iPhone User's Guide For iPhone and iPhone 3G (2008)
2. Benko, H., Wilson, A.D., Baudisch, P.: Precise selection techniques for multi-touch screens. In: Proceedings of the SIGCHI conference on Human Factors in computing systems (CHI 2006), pp. 1263–1272. ACM, New York (2006)
3. Bowman, D.A., Kruijff, E., LaViola Jr., J.J., Poupyrev, I.: 3D User Interfaces: Theory and Practice. Addison-Wesley, Boston (2004)
4. Darken, R.P., Cockayne, W.R., Carmein, D.: The omni-directional treadmill: A locomotion device for virtual worlds. In: UIST 1997: Proceedings of the 10th annual ACM symposium on User interface software and technology, pp. 213–221. ACM Press, New York (1997)
5. Dourish, P.: Where the Action Is: The Foundations of Embodied Interaction. The MIT Press, Cambridge (2001)
6. Fishkin, K.P.: A taxonomy for and analysis of tangible interfaces. Personal Ubiquitous Computing 8(5), 347–358 (2004)
7. Gabbard, J.L.: Taxonomy of Usability Characteristics in Virtual Environments. MS's thesis, Virginia Polytechnic Institute and State University (1997)
8. Interrante, V., Ries, B., Anderson, L.: Seven league boots: A new metaphor for augmented locomotion through moderately large scale immersive virtual environments. In: Proc. of IEEE Symposium on 3DUI, pp. 167–170 (2007)
9. Montello, D.R., Hegarty, M., Richardson, A.E., Waller, D.: Spatial Memory of Real Environments, Virtual Environments, and Maps. In: Allen, G. (ed.) Human Spatial Memory: Remembering where, pp. 251–285. Lawrence Erlbaum, Mahwah (2004)
10. Iwata, H., Yoshida, Y.: Path reproduction tests using a torus treadmill. Presence: Teleoperators & Virtual Env. 8(6), 587–597 (1999)
11. JazzMutant. Lemur User Manual 1.6. JazzMutant (January 20, 2007), [http://www.jazzmutant.com/download/Lemur\\_v1.6\\_Manual.pdf](http://www.jazzmutant.com/download/Lemur_v1.6_Manual.pdf)
12. Kim, J., Gračanin, D., Matkovic, K., Quek, F.: Finger Walking in Place (FWIP): A Traveling Technique in Virtual Environments. In: 2008 International Symposium on Smart Graphics, pp. 58–69. Springer, Heidelberg (2008)
13. Komura, T., Lam, W.-C.: Real-time locomotion control by sensing gloves. Journal of Visualization and Computer Animation 17(5), 513–525 (2006)
14. Peterson, B., Wells, M., Furness III, T.A., Hunt, E.: The effects of the interface on navigation in virtual environments. In: Proceedings of the Human Factors and Ergonomics Society Annual Meeting, pp. 1496–1505 (1998)
15. Ruddle, R.A., Lessels, S.: The benefits of using a walking interface to navigate virtual environments. ACM TOCHI 16(1), 5:1–5:18 (2009)
16. Siegel, A.W., White, S.H.: The development of spatial representations of large-scale environments. In: Reese, H.W. (ed.) Advances in Child Development and Behavior, vol. 10, pp. 9–55. Academic, New York (1975)
17. Slater, M., Usoh, M., Steed, A.: Taking steps: the influence of a walking technique on presence in virtual reality. ACM Transactions on Computer-Human Interaction (TOCHI) 2(3), 201–219 (1995)
18. Templeman, J.N., Denbrook, P.S., Sibert, L.E.: Virtual locomotion: Walking in place through virtual environments. Presence: Teleoperators & Virtual Env. 8(6), 598–617 (1999)
19. Virginia Tech. VT-CAVE, <http://www.cave.vt.edu/> (last accessed, February 2010)
20. Waller, D., Loomis, J.M., Haun, D.B.M.: Body-based senses enhance knowledge of directions in large-scale environments. Psychonomic Bulletin and Review 11, 157–163 (2004)

# Interactive Design and Simulation of Net Sculptures

Grigore D. Pintilie<sup>1</sup>, Peter Heppel<sup>2</sup>, and Janet Echelman<sup>3</sup>

<sup>1</sup> Massachusetts Institute of Technology, Cambridge, MA

<sup>2</sup> Peter Heppel Associates

<sup>3</sup> Echelman Inc.

**Abstract.** We present a graphical user interface that allows an artist to virtually design and visualize net sculptures. Net sculptures consist of net pieces that are seamlessly connected to each other and to fixed rails. They are flexible and hence dynamic under external forces such as gravity and wind. The interface that we describe allows an artist to create net sculptures made up of multiple net pieces. Simple operations such as clicking on points and click-and-drag gestures are used to create and modify individual net pieces, and drag-and-drop gestures are used to connect net pieces to multiple rails. The effect of gravity on the net sculpture is simulated, allowing the artist to simultaneously design and visualize net sculptures as they would appear once installed in a real setting.

**Keywords:** 3D graphical user interfaces, net sculptures, mass-spring particle systems.

## 1 Introduction

Net sculptures are composed of multiple net-pieces that are joined together and hung on suspended rails, producing aesthetic and dynamic 3D sculptures that move under the influence of wind and gravity [1]. The net pieces that make up a net sculpture must have dimensions that adhere to fabrication protocols, so that they can be produced by available equipment. Net pieces are connected to each other seamlessly, either by having the same number of knots along the connecting edge, or numbers of knots that are multiples of each other (which are known as gear changes). Creating and working with net pieces in the real world can be time-consuming and expensive, and thus experimenting with complex designs is prohibitive, motivating the need for computer-aided design methods which would allow an artist to virtually design and visualize net sculptures.

## 2 Overview and Related Work

Net sculptures can be modeled using 3-dimensional surfaces. Hence, an artist could use one of the many available computer-aided design tools to design a net sculpture, for example using parametric surfaces [2]. However, such surfaces, which

could have any shape imaginable, cannot easily be decomposed into individual net pieces that can be fabricated in order to produce the net sculpture. Moreover, gravity plays a very significant role in the final shape of the net sculpture, and so without simulation to take this effect into account, a parametric surface designed by the artist may not have the same shape when produced and hung. A more effective interface would thus have to combine the particular characteristics of net pieces and realistically simulate the net sculpture as a physical object.

Since net sculptures have 3D geometries, the interface must also display and allow the artist to interact with 3D objects. There are many challenges in the development of effective 3D user interfaces, such as ensuring proper spatial orientation and interaction with objects [3,4]. Commonly available 3D design tools give a large degree of freedom over the results; however they can have steep learning curves, and an artist can easily get lost in the complexity. By combining predictable physically-based object behaviour and simple gestures, easy to use and fun 3D interfaces have been achieved [5]. In the same spirit, we present an interface that allows an artist to create and modify net sculptures while at the same time simulating their physical behaviour under the effect of gravity.

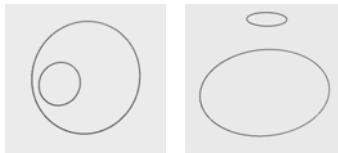
The interface does not require the user to explicitly position objects in 3D, which can be a frustrating experience when using a 2D input device such as a mouse [6]. Instead, net pieces are automatically connected either to pre-defined rails or to other net pieces, and their geometry is automatically generated. For other operations, with which the user adjusts dimensions of a net piece for example, we use 2D mechanisms. More specifically, such a mechanism involves clicking on a point with the mouse, followed by a dragging gesture (moving the mouse while the button is still pressed). During the click-and-drag gesture, the movement of the mouse is not related to positioning in 3D space, but rather to increasing or decreasing a parameter value, the effect of which is visualized in real-time.

In connecting a free-hanging side of a net piece to a rail, the user does actually move a point through 3D space. To make this movement correspond to the movement of the mouse, the point moves on a plane that is perpendicular to the viewing direction [7]. However this plane on which the point is being moved does not actually matter. Instead, if the point being moved is dropped while the mouse pointer is on top of a rail to which the net can be attached, the attachment is made. *On top of* is taken loosely to mean that the 2D position of the mouse is roughly the same as the projected 2D position of a point on the rail. This operation thus follows the typical drag-and-drop gesture that many 2D interfaces already use. The only inherently 3D operation that the artist has to perform while using the interface is to position the camera in 3D space, so as to view the net sculpture from different angles.

In the rest of the paper, we first describe the geometry of rails that net pieces can be attached to, the operations that an artist can perform to create and modify net sculptures, and finally the simulation of net sculptures to take into account the effects of gravity.

### 3 Rails

Rails can be either created by the artist, or pre-specified for a particular installation by the architectural team. There are no particular constraints in the creation of rails; they are commonly fabricated as metal tubes, and can take the form of any open or closed curve. The geometry of such rails can be created for example using B-splines by many CAD programs. These curves can be read into the net design interface either as B-splines or as continuous line segments. Two example rails that are imported into the interface are shown in Figure 1.

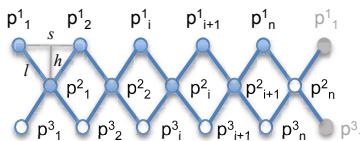


**Fig. 1.** Two rails are shown, each rail having the form of a closed curve. The two rails are shown from top view (left), and side view (right).

### 4 Net Piece Geometry

The geometry of a net piece is illustrated in Figure 2. It is made up of knots arranged in rows and columns, with each row staggered with respect to the previous row. Each row has the same number of knots as all other rows, and similarly, each column has the same number of knots as all other columns. As shown in Figure 2, each knot is connected by links to two knots in the row above (except for the knots in the first row), and to two knots in the row below (except for knots in the last row). These links are commonly called *bars*, and we will use this term through the rest of the paper.

Topologically a net piece is rectangular, with the same number of knots in each column, and the same number of knots in each row. However the bar lengths can



**Fig. 2.** A net piece is composed of knots connected by bars. The position of a knot is denoted by  $p_a^b$ , where  $a$  is the row-index and  $b$  is the column-index of the knot. The scalar  $s$  is the distance between two adjacent knots in the same row,  $h$  is the vertical distance between knots in adjacent rows, and  $l$  is the length of a bar attaching two knots. The right-most knots of this net piece link back to the left-most knots, forming a closed tube in this example.

be varied from row to row or from column to column (but not both), so that the net piece can also have a trapezoidal geometry. The number rows and columns in a net piece, along with the bar lengths, are the only parameters required to produce it using a net fabrication machine. Hence, these are also the parameters that an artist can control for each net piece when designing a net sculpture.

In determining knot positions, the Pythagorean relationship is used:

$$l^2 = (s/2)^2 + h^2 \quad (1)$$

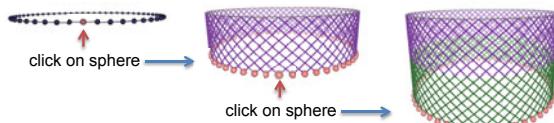
In Equation (1),  $l$  is the bar length,  $s$  is the spacing between adjacent knots in a row and  $h$  is the vertical distance between knots in adjacent rows. The user will be allowed to adjust  $s$  and  $l$ , from which  $h$  will be calculated. While  $s$  will be kept the same when computing the initial geometry of a net piece, the bar length  $l$  can vary smoothly either row-wise or column-wise, but not both, and thus  $h$  may also vary. The artist will be allowed to control how the bar length  $l$  varies throughout a net piece, as will be described in Section 6.

## 5 Creation of New Net Pieces

New net pieces can be created in two ways: by directly attaching them to a rail, or by attaching them to an existing net piece. Both ways are illustrated in Figure 3. To create a new net piece, the user puts the interface in the *new net* mode. In this mode, spheres are placed along rails or sides of existing net pieces, where new net pieces can be attached. The user then simply clicks on one of these spheres to create a new net piece.

### 5.1 Attaching a New Net Piece to a Rail

In the *new net* mode, points on a rail are generated at equidistant intervals, with distance measured in parameter space along the curve. These represent points that the nodes in the first row of a new piece can be attached to. In Figure 3, in the left-most image, these points are shown with solid spheres. To create a new net piece that attaches to these points, the user simply moves the mouse over any one of the spheres, and clicks. As the user moves the mouse over a



**Fig. 3.** Creation of net pieces by attachment to rails (left and middle images), or to existing net pieces (middle and right images). On the rail, the nodes on the top edge of the new net piece are attached to equally spaced points along the rail. On the existing net piece, the nodes on the top edge of the new net piece are attached to nodes on the bottom edge of the existing net piece.

sphere, a transparent red sphere is drawn on top of the original sphere, to let the user know that that sphere is selected. If the user clicks while a sphere is selected, a new net piece is created and attached to the rail, as shown in the middle image in Figure 3. The new net piece has the same number of knots in the width direction as points generated on the rail. Because the rail is a closed curve in this case, the net piece itself is also closed, with the first and last column attached to each other.

When creating a net piece by attaching it to a rail, the positions of the knots in the first row are set to the positions of the points generated at equal intervals along the rail. The positions of the knots in subsequent rows are computed based on these positions and the scalars  $s$ ,  $h$ , and  $l$ , which are related as given in Equation (1). In particular, we assume that  $s$  and  $l$  are given (and controlled by the user, as will be described in Section 6), and  $h$  will be calculated. Moreover, we assume that the rails are given on the  $x$ - $y$  plane, and the net will hang down in the  $z$  direction. Thus, the position of a node in a net piece is:

$$\vec{p}_b^a = \begin{cases} \frac{\vec{p}_b^{a-1} + \vec{p}_{b+1}^{a-1}}{2} - h\hat{z}, & \text{if } b \text{ is odd} \\ \frac{\vec{p}_{b-1}^{a-1} + \vec{p}_b^{a-1}}{2} - h\hat{z}, & \text{if } b \text{ is even} \end{cases} \quad (2)$$

In Equation (2),  $a$  is the column-index and  $b$  is the row-index of the knot at position  $\vec{p}_b^a$ . (The indexes are automatically wrapped around the first and last point in each column).

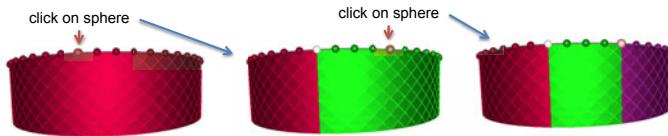
The positions that are computed for the knots in a new net piece using Equation (2) are not necessarily the final positions of the knots after taking into account the force of gravity. The final hanging shape of all the net pieces in a net sculpture will be computed using the dynamical process described in Section 7. Equation (2) is however very useful in attaining a reasonable starting geometry for the net.

## 5.2 Attaching a New Net Piece to an Existing Net Piece

In the *new net* mode, solid spheres are also drawn on the knots on the last row of a net piece to which a new net piece can be attached. For example, the middle image in Figure 3 shows such red spheres at the bottom of the net piece. Similar to the spheres shown on a rail, the user can click on these spheres to attach a new net piece to the last row of the existing net piece. When creating a new net piece by attaching it to an existing net piece, the positions of the knots in the first row of the new net piece are simply set to the positions of the knots in the last row of the existing net piece. The positions of all the other knots are then calculated via Equation (2).

## 5.3 Splitting a Net Piece into Multiple Net Pieces

A net piece can be divided into multiple net pieces, as shown in Figure 4. The resulting net pieces will have the same number of rows as the original net piece,



**Fig. 4.** Splitting a net piece into multiple net pieces. In this drawing, the net piece is drawn using both lines and triangles, for a more clear visualization of each net piece. In the image on the left, the user clicks on the sphere at the top of the column along which the net piece should be split, resulting in the net piece being divided into two at this column (middle image). The two resulting net pieces are now attached to each other column-wise, i.e. the knots on the last column of the net piece on the left are attached to the knots in the first column on the piece on the right. Another split is created by clicking on another sphere in the middle image (right-most image).

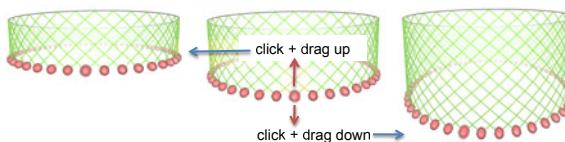
but the number of columns in each net piece may be different. To divide a net piece, the user first puts the interface in the *divide net piece* mode. In this mode, spheres are placed on the top nodes of every net piece. When the user clicks on one of these spheres, the net is divided at that column. This operation is useful for various reasons. One reason is that the artist may want to use different bar lengths in each net piece, or may want to use net pieces with different colors, material properties, or bar thickness.

## 6 Changing Net-Piece Dimensions

The interface allows the artist to modify the number of rows and columns of each net piece, as well as bar lengths, as described in the following sections.

### 6.1 Changing the Number of Rows in a Net Piece

To change the number of rows of knots in a net piece, the user first puts the interface in the *change number of rows* mode. In this mode, spheres are shown at knots in the last row of all net pieces, as shown in Figure 5. The user can then click on one of these spheres and drag it up to decrease the number of rows or down to increase the number of rows. The reason for this mapping is that when



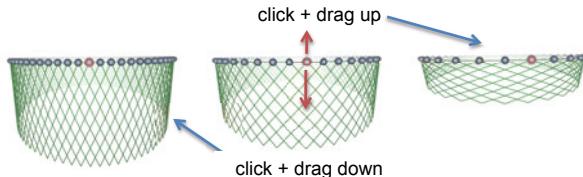
**Fig. 5.** The number of rows in the illustrated net piece is changed by clicking and dragging one of the spheres placed on the last row of the net piece. Dragging up decreases the number of rows, while dragging down increases the number of rows.

dragging up, the net piece appears to shorten, and thus the bottom row of the net seems to follow the direction of the mouse gesture.

## 6.2 Changing the Number of Columns in a Net Piece

To change the number of columns in a net piece, the user first puts the interface in the *change number of columns* mode. In this mode, for a net piece attached to a rail, as shown in Figure 6, spheres are drawn at all the points where the knots in the first row of a net piece are attached to the rail. The user can then click on one of these spheres and drag up or down to change the number of columns in the net piece.

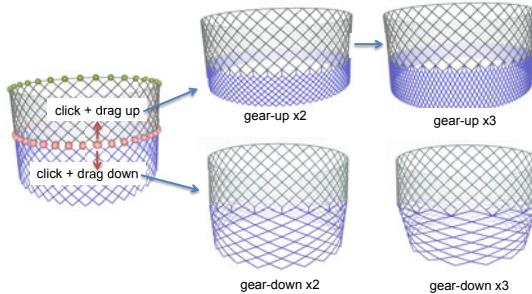
When the user drags up, the spacing between the attachment points is decreased, meaning that the number of columns increases, as more points are required to circumnavigate the entire rail. If the user drags down, the spacing between the attachment points is increased, meaning the number of columns decreases as fewer points are required to cover the entire rail. Note that the number of rows in the net piece stays constant during this operation; however, because the bar length also stays constant, the distance between knots in adjacent rows changes, and therefore the actual length of the net piece may appear to increase and decrease, as can be seen in Figure 6.



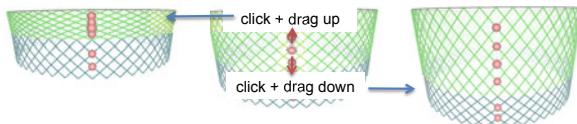
**Fig. 6.** Changing the number of columns in a net piece attached to a rail. The user clicks on a sphere and drags up to increase the spacing between attachment points, or drags down to decrease the spacing. Because the number of attachment points changes, the number of columns in the net piece also changes. Because the bar lengths and number of rows stay the same, the physical length of the net piece also increases or decreases as shown.

## 6.3 Gear Changes

To change the number of columns in a net piece connected to another net piece, the same mode and gestures are used. However, the number of columns in such a net piece is constrained, because it must seamlessly connect to the net piece above. This operation is also commonly called *gear changing*. The number of columns in a net piece connected to another net piece can be either double (gear-up of 2), triple (gear-up of 3), half (gear-down of 2), a third (gear-down of 3) of the number of columns in the net piece it is connected to. These gear changes are illustrated in Figure 7.



**Fig. 7.** Changing the number of columns in a net piece which is attached to another net piece, through gear changes. Gear-up changes are obtained by clicking on a sphere and dragging up, which doubles or triples the number of columns in the net piece (top row, right). Gear-down changes are obtained by clicking on a sphere and dragging down, which halves or thirds the number of columns in the net piece (bottom row, right).

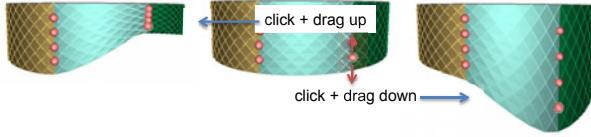


**Fig. 8.** Changing the bar length of a single net piece. The user clicks a sphere which is drawn at knots in the first column of a net piece, and drags up/down to decrease/increase the bar length. In this example, the net piece is not connected to other net pieces column-wise, so the bar length is the same in every column. A net piece attached to the last row of the net piece is updated as well to maintain that connection.

#### 6.4 Changing the Bar Length of a Net Piece

The bar length of a net piece can also be adjusted by the artist. To do so, the artist first puts the interface in *change bar length* mode. In this mode, spheres are drawn on the knots in the first column of a net piece, as shown in Figure 8. The artist can click on one of these spheres and drag up to decrease the bar length, or down to increase it. For a single net piece, the bar length is the same in all of its columns. Note that the number of rows and columns in the net pieces are not changing during this operation, even though the actual physical length is changing, due to smaller or larger bar lengths.

When two or more net pieces are attached at their first and last columns, the bar lengths are the same at each column where net pieces are connected. However, the bar lengths are linearly interpolated between the first and last column in the net-piece, as shown in Figure 9. A varying bar length results in a varying physical length of the net piece at each column, as shown in Figure 9.



**Fig. 9.** The bar length at a column of a net piece is modified by clicking on a sphere drawn on this column, and dragging up to decrease the bar length or down to increase it. For each net piece, the bar length is linearly interpolated between its values at the first and last column. The bar lengths at the columns where two net pieces are attached are the same for both net pieces.

## 7 Dynamics of Net Sculptures

The geometry created for each net piece, as described in previous sections, serves only as an initial configuration for a net sculpture. It is also desirable to the artist to see how the entire net sculpture hangs under the pull of gravity. In order to simulate this hanging shape, we model each net piece in the sculpture as a mass-spring system, where the knots are masses and the bars between them are springs. Such dynamical model has been used in various applications, such as cloth modeling [9,10] and surgery simulation [11]. Cloth has different material properties, and here we try to adapt the force equations to better capture the properties of nets. The force acting on each knot can be written as:

$$\vec{F}_j = \begin{cases} \vec{g} + \sum_{i=1}^n k_{i,j}^s (l_{i,j} - l_{i,j}^{eq}) \frac{\vec{p}_i - \vec{p}_j}{|\vec{p}_i - \vec{p}_j|} & \text{if } l_{i,j} > l_{i,j}^{eq} \\ \vec{g} + \sum_{i=1}^n k_{i,j}^c (l_{i,j} - l_{i,j}^{eq}) \frac{\vec{p}_i - \vec{p}_j}{|\vec{p}_i - \vec{p}_j|} & \text{if } l_{i,j} \leq l_{i,j}^{eq} \end{cases} \quad (3)$$

Equation (3) expresses the force on a knot  $j$ ,  $\vec{F}_j$ , as a combination of the force of gravity,  $\vec{g}$ , which points down, and the springs attaching it to  $n$  other knots. The scalar  $k_{i,j}^s$  is the stretching spring constant for the bar attaching knots  $i$  and  $j$ , and  $k_{i,j}^c$  is the compression spring constant for the same bar. A piece of rope is very resistant to stretching, but not very resistant to compression, since in the latter case it simply bends; thus we set  $k_{i,j}^s$  to be relatively high and  $k_{i,j}^c$  to be relatively low.

In Eqn. 3,  $l_{i,j}$  is the actual length of the bar attaching knots  $i$  and  $j$  during the simulation. The equilibrium length for the same bar is  $l_{i,j}^{eq}$ , which stays constant and is set when the positions of the knots are calculated as previously described. Lastly,  $\vec{p}_i$  and  $\vec{p}_j$  are the positions of the knots  $i$  and  $j$ , which are initialized using Eqn. (2), and which change during the simulation. To update the positions for each knot, Eqn. (2) is written into two ordinary differential equations, relating the acceleration, velocity, and position of every knot in the net sculpture:

$$\vec{a}_j = \frac{d\vec{v}_j}{dt} \quad (4)$$

$$\vec{v}_j = \frac{d\vec{x}_j}{dt} \quad (5)$$

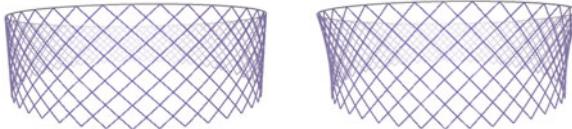
In Equations 4 and 5,  $\vec{a}_j$  is the acceleration that knot  $j$  experiences due to the forces on it,  $\vec{v}_j$  is the velocity of a knot, and  $d\vec{x}_j$  is the position of the knot. The acceleration of a knot is related to the force acting on it, as dictated by Newtons law:

$$\vec{a}_j = \frac{\vec{F}_j}{m_j} \quad (6)$$

In Eqn. 6,  $m_j$  is the mass of the knot  $j$ . For design purposes, we would simply like to find the equilibrated hanging shape of a net sculpture, ignoring any oscillatory motion that is typical of mass-spring systems. Hence we assume an infinitely damped system, free of inertia, in which the velocity on a knot is simply the force acting on it:

$$\frac{d\vec{x}_j}{dt} = \frac{\vec{F}_j}{m_j} \quad (7)$$

To find the equilibrium hanging shape of a net sculpture, Eqn. 7 is discretized and integrated using the Runge-Kutta method [12], with the initial positions of each knot being the positions computed as previously described. Net sculptures are inherently stiff systems, since the spring constants should be high and the masses relatively low, and thus stability is an issue. We use relatively low spring constants, large masses, and small time steps, making the simulation stable during the design process. Once design is complete, the spring constants are gradually increased to achieve a more accurate hanging shape. Figure 10 shows the initial and equilibrium hanging shape of a single net piece hung on a circular rail.



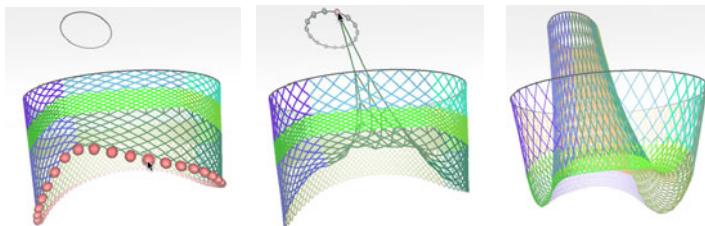
**Fig. 10.** A single net piece attached to a circular rail, with knots in the initial computed positions (left), and after the equilibrium hanging shape is reached (right)

## 8 Attaching Net Pieces to Multiple Rails

To design a more complex net sculpture, an artist often uses net pieces that connect to each other and also to more than one rail. To facilitate this within the user interface, we employ a drag-and-drop mechanism, which is illustrated in Figure 11. To perform this operation, the user first puts the interface in the *connect net* mode. In this mode, spheres are drawn on the last rows of net pieces which can be attached to a rail. The user clicks on one of these spheres and starts dragging it. While dragging, spheres are drawn on rails to which the net

piece can be attached. To complete the attachment, the user drops the sphere they are dragging on top of any of the spheres placed on the rails they wish to attach the net piece to. *On top of* means simply that on the 2D projected image, the user has moved the mouse so as to select any of the spheres drawn on the rail on which they are attaching the net piece to.

After this operation is performed, all the knots on the last row of the net piece which is being attached are automatically fixed to points on the rail to which the net piece is being attached. From this configuration, the mass-spring simulation method is used to bring the net to its hanging shape.



**Fig. 11.** The free-hanging side of a net sculpture (left), is attached to another rail (middle). A point on the free-hanging side is clicked, dragged, and dropped when the mouse pointer coincides with a point on the rail that it should be attached to. The final hanging net geometry is then shown (right).

## 9 Conclusions and Future Work

We presented a graphical user interface that allows an artist to virtually design net sculptures, which are composed of multiple net pieces that seamlessly connect to one another and to supporting rails. The artist can create and modify net pieces by clicking on points and using simple click-and-drag gestures. Moreover net pieces can be connected to other rails by dragging and dropping a point on the end of a net to the rail that it should be attached to. In preliminary trials, it was apparent that such an interface has great potential in allowing an artist to virtually design realizable net sculptures. In the future, we aim to further explore its effectiveness in allowing an artist to fully exercise their artistic and creative visions.

## Acknowledgments

We would like to thank Echelman Inc. for commissioning and funding this work. The pictures shown in the paper were created using the proprietary JNet software, ©Echelman Inc, which implements the methods presented.

## References

1. <http://www.echelman.com/>
2. Thingvold, J.A., Cohen, E.: Physical modeling with B-spline surfaces for interactive design and animation. In: Proceedings of the 1990 symposium on Interactive 3D graphics, pp. 129–137. ACM, Snowbird (1990)
3. Brooks, F.P.: Grasping reality through illusion; interactive graphics serving science. In: Proceedings of the SIGCHI conference on Human factors in computing systems, pp. 1–11. ACM, Washington (1988)
4. Coquillart, S., Frhlich, B., Hirose, M., Kitamura, Y., Kiyokawa, K., Strzlinger, W., Bowman, D.: 3D User Interfaces: New Directions and Perspectives. IEEE Computer Graphics & Applications 28, 20–36 (2008)
5. Agarawala, A., Balakrishnan, R.: Keepin' it real: pushing the desktop metaphor with physics, piles and the pen. In: Proceedings of the SIGCHI conference on Human Factors in computing systems, pp. 1283–1292. ACM, Montral (2006)
6. Hinckley, K., Pausch, R., Goble, J.C., Kassell, N.F.: A survey of design issues in spatial input. In: Proceedings of the 7th annual ACM symposium on User interface software and technology, pp. 213–222. ACM, Marina del Rey (1994)
7. Strauss, P.S., Issacs, P., Shrag, J.: The design and implementation of direct manipulation in 3D. In: SIGGRAPH 2002 Course Notes (2002)
8. Oh, J., Stuerzlinger, W.: Moving objects with 2D input devices in CAD systems and Desktop Virtual Environments. In: Proceedings of Graphics Interface 2005, pp. 195–202. Canadian Human-Computer Communications Society, Victoria, Columbia (2005)
9. Ji, F., Li, R., Qiu, Y.: Three-dimensional Garment Simulation Based on a Mass-Spring System. Textile Research Journal 76, 12–17 (2006)
10. Baraff, D., Witkin, A.: Large steps in cloth simulation. In: Proceedings of the 25th annual conference on Computer graphics and interactive techniques, pp. 43–54. ACM, New York (1998)
11. Liu, A., Tendick, F., Cleary, K., Kaufmann, C.: A Survey of Surgical Simulation: Applications, Technology, and Education. Presence: Teleoperators & Virtual Environments 12, 599–614 (2003)
12. Butcher, J.: Numerical Methods for Ordinary Differential Equations. John Wiley & Sons, Ltd., Chichester (2003)

# A Cross-Platform Framework for Physics-Based Collaborative Augmented Reality

Damon Shing-Min Liu, Chun-Hao Yung, and Cheng-Hsuan Chung

National Chung Cheng University, Chiayi, Taiwan

{damon, ycha97m, cch97m}@cs.ccu.edu.tw

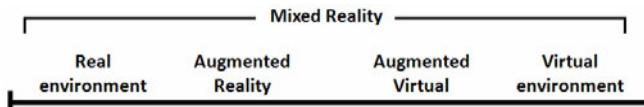
**Abstract.** Augmented Reality (AR) provides users with enhanced interaction experiences by allowing computer-generated virtual imagery to overlay physical objects. Here we aim to integrate desktop and handheld AR into a cross-platform environment in which personal-computer and mobile users can collaborate with each other in a shared scene to accomplish physically realistic experiences during the course of interaction. Particularly, users can intuitively pick up and move virtual objects using their hands in the desktop environment. In realizing the system, we exploit 1) a Client/Server architecture to connect different computing devices, where the server is responsible for maintaining and managing the virtual scene objects shared by users; 2) a marker-based tracking method that computes relationship between the camera view and markers; 3) a computer graphics API to render the scene in a Scene Graph structure; 4) an approach that combines hand tracking and occlusion-based interaction to estimate hand position in video frames.

## 1 Introduction

Augmented Reality (AR) seamlessly merges physical and virtual world through the accurate registration between the image captured by a camera and the virtual objects. Unlike AR, Virtual Reality (VR) separates users from the real world, it inevitably introduces a discontinuity between the real and virtual world. Moreover, in VR, users are fully immersed in a virtual environment; on the contrary, AR allows users to interact with the virtual images using real objects. Thus, AR brings the benefits of VR interfaces into the real world to facilitate natural collaboration. As Milgram [1] pointed out, the relationship between AR and VR can be placed along a continuum according to how much the user's environment is computer generated (Figure 1). Usually AR is considered as one part of the general area of Mixed Reality (MR).

Azuma [2] provided a distinct definition of AR as 1) a technology which combines real and virtual imagery; 2) is interactive in real time, and 3) registers the virtual imagery with the real world. Recently, an increasing number of commercial applications utilize AR technique to create attractively interactive environment.

Owing to that technique and knowledge of AR have grown to maturity [3], many collaborative applications were previously developed using desktop computers. However, the successful experience from several desktop-based AR research projects inspired researchers to port AR applications onto mobile platform as well. With the recent advances in processing power, display and memory capability, executing AR



**Fig. 1.** The relationship between AR and VR



**Fig. 2.** Mobile AR applications

applications on PDAs and cell phones has become possible (Figure 2). The widespread use of camera-equipped mobile phones also makes them a very attractive platform for deploying AR applications. Hence, AR systems nowadays are developed on unrestricted device-platforms.

Over the past decade, there was an evolution in the types of AR interface being developed; however, little research integrated desktop and handheld AR environment into one where personal-computer and mobile users can collaborate with each other in a shared scene through PCs and mobile phone, respectively. Moreover, interactive AR interfaces need to be as intuitive as possible to be accepted by users. Current design of such interfaces is considered deficient in two aspects. First, it does not fully exploit the use of free-hand interaction to manipulate the augmented virtual objects. Second, even though most AR designs allow users to naturally interact with virtual objects, few actually simulated physical circumstances as in real world during the course of interaction between users and virtual objects.

In this paper, we therefore present a cross-platform, physics-based, collaborative system for AR. We embed a physics engine into an AR application to generate more realistic virtual object movements. We describe a method for vision-based natural hand interaction tailored for desktop AR environment. In it, users can intuitively and naturally manipulate virtual objects using their hands without wearing gloves. Thus, it can provide a natural seamless interaction. Furthermore, our system allows multiple users to interact with each other through either personal computers or other mobile devices.

The rest of the paper is organized as follows. Section 2 introduces the marker-based tracking library and reviews related works in other AR research projects. Section 3 provides a hand interaction method which intuitively manipulates the virtual objects. Section 4 describes the framework of our system. Finally, we conclude the paper in Section 5.

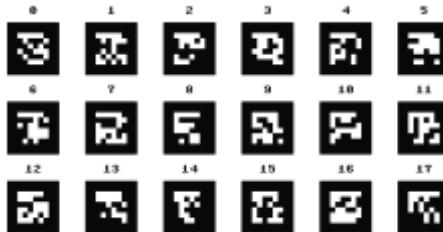
## 2 Related Works

To precisely overlay virtual images on real world, we exploited a robust technique called *marker-based tracking*. Such technique can track the pose of camera relative to physical markers and compute what viewport to draw the virtual imagery from.

## 2.1 Marker-Based Tracking Library

There are several well-known computer vision based tracking library, such as ARToolKit [4], ARToolKitPlus [5], ARTag [6], Studierstube Tracker [7] and Simplified Spatial Target Tracker (SSTT) [8]. ARToolKit was originally developed by Kato [9]. It solves the problem of tracking user's viewport using computer vision algorithm. ARToolKit first detects the square markers from an image captured by camera. Then it estimates the pose and position of the markers and displays virtual objects on top of those markers.

ARToolKit uses template matching method to recognize the position and direction of the marker. Unfortunately, this restricts the number of concurrently usable markers at run-time due to the increasing complexity of search in the image database. ARTag is an improved marker system based on ARToolKit. It uses Id numbers to encode the marker's Id with built-in forward error correction (CRC) as shown in Figure 3. Thus, ARTag can use more numbers of markers than that of ARToolKit.



**Fig. 3.** Sample Id markers templates

ARToolKitPlus is a successor to ARToolKit pose tracking library. It is optimized and extended for use on mobile devices. ARToolKitPlus provides several new features as well. It re-implemented the most important functions in ARToolKit with fixed-point arithmetic to speed up computing on mobile platform. It supported the native pixel formats for phone cameras to reduce the need for image conversion. It used the simple Id-encoded markers similar to those in ARTag to improve the marker system in ARToolKit. It does automatic thresholding by looking at the marker pattern to provide more stable tracking adjustable to changing lighting condition. It provided simple vignetting compensation to prevent that some cameras in mobile phones may exhibit strong vignetting.

Studierstube Tracker is a successor to ARToolKitPlus library. Compared to ARToolKitPlus, its memory requirement and processing power are improved. Studierstube Tracker was written from scratch for PCs as well as for mobile phones; however, it is not an open-source software. SSTT implemented a marker based approach with a minimal set of constraints. It does not require thick black borders of markers, and its tracking algorithm takes company logos or even images in books as tracking targets. It also enables the tracking of multiple touches around the edges of a tracking target.

## 2.2 Related AR Systems

To the best of our knowledge, there are few existing works that integrate personal computers and mobile devices into a cross-platform interactive AR environment. In this section, we discuss related projects using hand interaction with virtual objects as well as those research works for mobile AR applications.

Buchmann et al. [10] present a technique for fingertip-based interface, called *FingARtips*, which allows for free hand interaction with virtual objects. It can be manipulated by natural gestures such as grabbing, pressing, dragging and releasing. To track fingers, users must wear a glove with small markers attached to two fingertips and the hand. Although markers which stick on gloves help in more reliably detecting hands and fingers, users can not interact with virtual objects using their bare hands.

Lee et al. [11] develop a computer vision-based 3D hand tracking system for multimodal AR interface. Users manipulate augmented objects by finger pointing and direct touching. Furthermore, it allows users to change the color or shape of virtual objects using speech input. To get the 3D information of users' fingertip and hand center, they use stereo camera as a video input device which has two lenses on one camera body to provide stereo color image with depth map. However, this system does not provide physics simulation when users interact with virtual objects.

Fernandes and Fernandez [12] present a system that tracks the 2D position of users' hands on a tabletop surface, allowing users to move, rotate and resize the virtual objects over this surface. They use statistical models which can be obtained by analyzing a set of training images and then be used to detect the hands. Compared to other approaches, the system is less sensitive to changes in illumination. However, this system also does not provide physics simulation with virtual objects.

Seichter et al. [13] describe a touch-based interaction technique for Tangible User Interfaces (TUIs) in AR applications. This technique allows for direct access and manipulation of virtual content on a registered tracking target. It provides multiple finger touch interaction in the vicinity of markers, thus supporting spatial interactions with 3D content on desktop and mobile devices. The multiple touch input streams can be mapped in various ways to output parameters. However, this system does not provide natural hand interaction with virtual objects in 3D space.

Wagner et al. [14] present a system for multi-user interactive AR applications running on handheld devices. Their architecture is based on Studierstube [15] framework to accelerate collaborative applications. To evaluate the usability of the system, they design a multi-player game which is called *Invisible Train*. In it, users can operate the virtual train through PDAs to accomplish purpose of interactive cooperation and competition with each other.

Henrysson et al. [16] utilize mobile phones to support face-to-face collaborative AR gaming. They port ARToolKit library to Symbian phone operating system and then develop a collaborative AR tennis game to demonstrate their system. Users can use their mobile phones which act as tennis rackets to play tennis game. Unlike Invisible Train [14], AR Tennis is focus on face-to-face interaction between two users, so it does not support complicated Scene Graph architecture and Internet unit to manage consistency of the AR scene.

### 2.3 Stereo or Non-Stereo Camera Based Interface

In desktop AR interactive environment, the most natural interface for interacting with virtual objects is through *hands*. We get used to manipulating things using our hands in daily life, so the same experience can be applied to AR for more intuitive interaction. To freely manipulate the virtual object, we must know the 3D coordinates of the fingertips relative to the virtual objects in AR scene. The most notable environment setup is:

- Using a camera with one lens: It must extract the gesture information from each video image to determine whether the interaction is available or not. However, the image frames captured by camera do not have the depth information. Without correct depth information of the hand, hand interaction with virtual objects becomes unnatural to users in vision. Wearing special gloves is an alternative approach to get the depth information of the hand. However, the acquisition and the design of special gloves are not easy. Compared to bare hands, wearing a glove also increases difficulty when users interact with virtual objects.
- Using a camera with two lenses: Stereo camera can provide an image with depth information by matching the same feature in two images captured by different camera lenses. Hence, users can interact with virtual objects without wearing any equipment. However, this device is difficult to obtain.

## 3 Hand Interaction

In this section we describe a natural hand interaction method that is used in our desktop environment. This interaction method consists of three parts: Marker-based tracking method, Hand tracking method, and Occlusion-based interaction method. In it, Hand tracking step is further divided into Skin color detection and Fingertips finding.

### 3.1 Marker-Based Tracking Method

We utilized marker-based tracking library to track markers pose for correctly overlaying the virtual objects on real scenario. Because the computing power of mobile devices is limited, we chose ARToolKitPlus to handle marker tracking for our cross-platform framework. By detecting size-known square markers in each frame images, ARToolKitPlus can determine pattern inside the markers and utilize camera calibration to calculate the transformation relationship between markers coordinates and camera coordinates. This transformation relationship can be represented using a *transformation matrix*. We can simply use this transformation matrix to accurately transform any 3D virtual object from its marker coordinates to camera coordinates. Thus, the augmented objects can be seamlessly overlaid on the physical marker captured by camera in frame images.

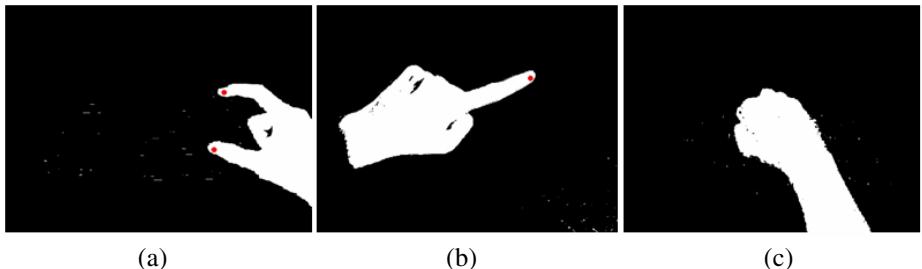
### 3.2 Hand Tracking Method

To provide natural hand interaction with virtual objects in desktop AR environment, we must know the position of the hand relative to marker coordinates. We utilize

color segmentation method to detect skin area and then exploit fingertips finding algorithm to decide the position of the fingertips in detected skin region. Because the camera has only one lens to capture frame image that do not provide depth information in it, we consider an alternative approach that uses the occlusion-based interaction method to approximately calculate the depth information of the fingertips.

To determine the position of hand in frame images, we use skin detection to separate skin color from background. Skin color segmentation is a technique which can classify each individual pixel as skin or non-skin from its neighbors. It can effectively and rapidly divide skin area from background. Color segmentation can be processed in different colorspace, such as RGB, Ycbcr, or HSV. We utilize a skin classifier which was provided by Peer et al. [17]. It operates in the RGB colorspace.

After processing the skin detection, fingertips can be found easily from hand position. We use a simple algorithm that was suggested by Hardenberg and Berard [18]. The algorithm can be executed in real-time. This algorithm explores two properties of fingertips: 1) the center of the fingertips is surrounded by a circle of skin area; 2) along a square outside the inner circle, fingertips are surrounded by a short chain. Using this algorithm, we can easily find possible fingertips in frame images in the course of interaction when users manipulate the virtual object using their hands, as shown in Figure 4. Moreover, it not only can detect fingertips and hands but also can classify fingertips that are grouped due to different hands. Thus, we can provide gesture input function extensions for users or developers to extend the commands by themselves.



**Fig. 4.** Fingertips detection: (a) two fingertips were detected. (b) one fingertip was detected. (c) no fingertips found.

### 3.3 Occlusion-Based Interaction Method

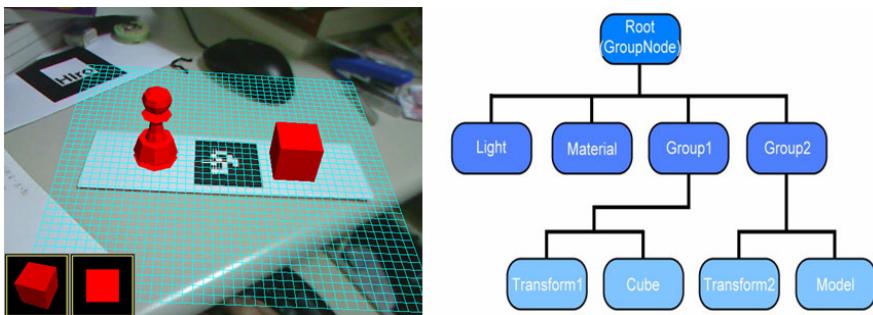
Hand tracking method can provide the 2D fingertips position, but hand interaction without depth information is unnatural to users. We therefore enhanced the work described by Lee et al. [19] to approximately calculate the fingertips position relative to marker coordinates in 3D space.

Lee et al. conformed to four situations: bad lighting conditions, motion blurs, marker deformations, or occlusion, could occasionally cause the failure of marker detection. An alternative approach is to use multiple markers to track one object by attaching a number of markers on a single object in a pre-configured spatial relationship. Hence, the poses of markers that are not visible can be estimated using the markers that are recognized.

In the previously mentioned four situations, only the marker occlusion case can partially cause tracking failure of markers in the marker set. Therefore, people use marker projection and boundary marker methods to detect the marker being out of view and being occluded, respectively. We use similar methods to check which parts of the marker set are occluded by user hands. Because the spatial relationships of all markers are known, information of the markers that are occluded by hands and information of the position of fingertips can be mutually referenced to estimate 3D coordinate positions relative to marker coordinates. However, the boundary marker method inevitably causes marker wastage. Moreover, a large number of markers also significantly degrade system performance when tracking marker poses. To overcome this problem, we enhance the tracking library to enable tracking of the arranged circle markers surrounded by fiducial markers. Consequently, we can provide natural manipulation with virtual objects using camera with only one lens.

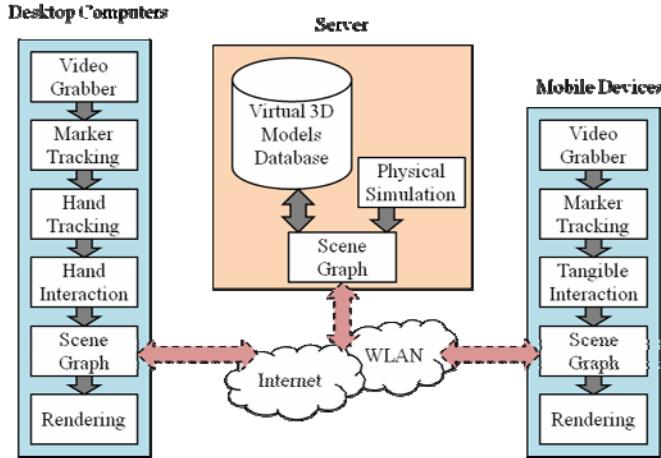
## 4 The Proposed Collaborative AR System

Here we aim to integrate personal computers and mobile devices into a cross-platform collaborative AR system. Construction of the system is handheld-device biased because its computing resource is far inferior to that of desktop computer. Once the observed performance is satisfactorily achieved on handheld device, the computing issue is no longer a concern on desktop computer. We also exploit a Scene Graph structure to enhance the ease of managing objects on mobile devices (see Figure 5). Typically, our system is based on Client/Server architecture, with a PC acting as the server, and numerous users as PC clients or mobile clients. Therefore, all collaborative AR applications are executed on Windows XP (for PC) and Windows Mobile (for handheld devices) operating systems, respectively. In addition to using ARToolKitPlus tracking library, we develop several other system functions to manipulate the scene graphs. Moreover, a physics engine is also integrated to our system to provide more realistic interaction with virtual objects.



**Fig. 5.** Example scene Graph structure on mobile devices

Overview of the system framework is shown in Figure 6. In the Client/Server architecture, server must maintain the Scene Graph structure shared by multiple clients. Server is responsible for updating and subsequently broadcasting nodes



**Fig. 6.** Overview of system framework

information in the scene graph to clients depending on different kinds of user manipulation encountered. Server also needs to calculate the physics simulation of virtual objects and manage the virtual 3D object database in the scene.

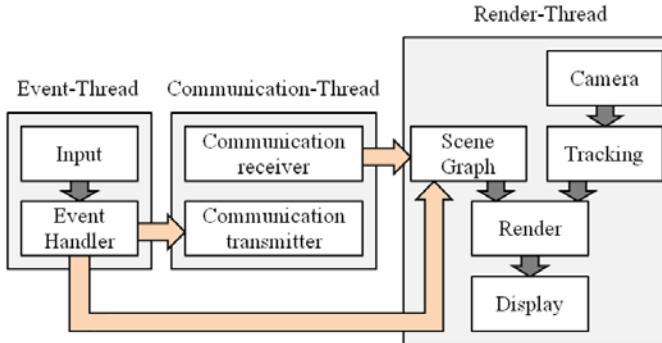
On the other hand, clients utilize marker tracking library to track physical markers from each frame image. As soon as marker is tracked, the corresponding transformation matrix can be computed to correctly overlay virtual image on the real scene. Users thus can manipulate the virtual object and collaborate with each other through their devices. In other words, clients are responsible to send to the server the node information which is modified when users interact with the virtual objects, and receive from the server the updated scene graph data in order to render the new physics-based AR scene.

#### 4.1 Auxiliary System Modules

Besides, our system provides a set of operating components to enable developers to easily create multi-user AR applications (see Figure 7). These auxiliary system modules include Event Handler, Scene Graph module, Tracking and Rendering module, and Internet Communication module. Specifically, Internet Communication model concurrently executes three threads and monitors the communication between those three threads.

- **Event-Thread.** As soon as the external device triggers the input event, the Event Handler converts the event to parameters which are used to generate scene graph data. Those parameters therefore are passed to Scene Graph module and Internet Communication module. Subsequently, the AR scene is updated by Scene Graph module according to the information stored in received parameters. On the other hand, the transmission component encapsulates the event parameters to package which is transferred to the Server.

- **Communication-Thread.** The reception component waits for the package which is sent by the Server, and converts the packet message to event parameters which are then sent to Scene Graph module. On the other hand, the transmission component waits for the parameters sent from Event Handling module, and converts the event parameters to package which is then sent to the Server.
- The **Renderer-Thread** creates the AR scene which is constructed by Scene Graph module with the transformation matrix computed using tracking library functions. This thread continuously executes through an infinite loop.



**Fig. 7.** Diagram of auxiliary system components

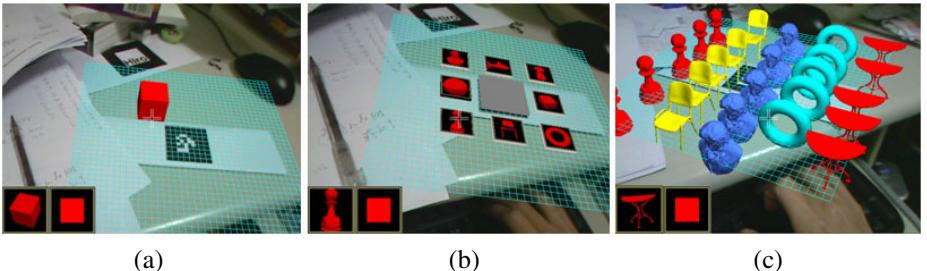
## 4.2 Physics Simulation

To maintain the virtual object consistency between each client, we build the Scene Graph data which is managed by the server to construct the AR scene shared by multiple users. The server can maintain the shared AR scene by updating the Node information which is modified when users interact with the augmented objects. Moreover, we run a physics engine for accomplishing a more realistic interaction experience.

When users manipulate the virtual objects in the AR scene, attributes of the transformation node which controls the translation and rotation information of virtual objects in Scene Graph are also updated, and sent to the server. As soon as the server receives and updates the Transformation Node in Scene Graph, the physics engine can be used to simulate the realistic objects movement using those new data. Then, the server broadcasts the new Scene Graph information to all clients to re-render their AR scenes.

## 4.3 Manipulation

Mobile phones are regarded as ideal tangible user interfaces which can be used to manipulate the virtual objects. The center of the display is marked as a cross, and users can manipulate the virtual object by targeting the cross on it (see Figure 8). Mobile device is capable of displaying the AR scene and acting as interaction tools at the same time. Moreover, because the size of the mobile phones is small, which can be easily moved and manipulated in hands, it is very suitable for users to interact with virtual objects in the AR scene.



**Fig. 8.** The proposed mobile AR environment: (a) users can create virtual object; (b) the menu of the virtual objects; (c) a case of multiple virtual objects

In our desktop AR environment, we provide the manipulation that users can interact with the virtual objects using their bare hands. There are currently three gesture input commands for hand interaction. “Two fingers are detected”: this gesture acts as the picking command for users to intuitively dabble virtual objects in AR scene. “One finger is detected”: this gesture acts as the pointing command for users to intuitively select virtual objects in AR scene. “No fingers are found”: this gesture acts as cancelling command for users to cancel the virtual object which is selected or dabbled in AR scene.

For example, users can use one finger to select one of the virtual items in option menu shown on physical marker using the *pointing* command. Then, they can use two fingers to drag the virtual objects into AR scene using the *picking* command. The virtual object is attached on hand position in AR scene until users utilize the *cancelling* command by making a fist. Although, we currently provide three commands for hand interaction, the finger finding algorithm [19] can detect all fingers image on the screen, so more gesture commands can be extended.

## 5 Conclusion

We present a cross-platform framework which integrates desktop computers and mobile devices for AR applications. Thus, users can freely choose to use PCs or mobile devices to easily interact with each other. Moreover, to provide realistic interaction, the behavior of virtual objects is simulated using a physics engine.

In the system, we developed a Scene Graph structure to maintain the AR scene shared by multiple users. Communication between each client is realized through a Client/Server architecture. Furthermore, to reconstruct the hand position relative to virtual objects in 3D space, we utilized an improved occlusion-based interaction method to calculate the depth information of hands. Thus, our system provides natural hand interaction for users.

## References

1. Milgram, P., Kishino, F.A.: Taxonomy of mixed reality visual displays. Institute of Electronics, Information and Communication Engineers (IEICE) Trans. on Information and Systems (special issue on networked reality) E77-D(12), 1321–1929 (1994)

2. Azuma, R.T.: A survey of augmented reality. *Presence: Teleoperators and Virtual Environments* 6(4), 355–385 (1997)
3. Azuma, R.T., Baillot, Y., Behringer, R., Feiner, S., Julier, S., MacIntyre, B.: Recent advances in augmented reality. *IEEE Computer Graphics and Applications* 21(6), 34–47 (2001)
4. ARToolKit, <http://www.hitl.washington.edu/artoolkit/>
5. Fiala, M.: ARTag: a fiducial marker system using digital techniques. In: *CVPR 2005* (2005)
6. Wagner, D., Schmalstieg, D.: ARToolKitPlus for pose tracking on mobile devices. In: *Proceedings of 12th Computer Vision Winter Workshop* (2007)
7. Studierstube ES, [http://studierstube.icg.tu-graz.ac.at/handheld\\_ar/stbtracker.php](http://studierstube.icg.tu-graz.ac.at/handheld_ar/stbtracker.php)
8. Simplified Spatial Target Tracker, <http://technotecture.com/content/marker-based-tracking-augmented-reality-sstt>
9. Kato, H., Billinghurst, M.: Marker tracking and HMD calibration for a video-based augmented reality conferencing system. In: *2nd IEEE and ACM International Symposium on Augmented Reality*, pp. 85–94 (1999)
10. Buchmann, V., Violich, S., Billinghurst, M., Cockburn, A.: FingARTips: gesture based direct manipulation in augmented reality. In: *2nd international conference on Computer graphics and interactive techniques in Australasia and SouthEast Asia*, pp. 212–221 (2004)
11. Lee, M., Green, R., Billinghurst, M.: 3D natural hand interaction for AR applications. In: *23rd International Conference on Image and Vision Computing, New Zealand* (2008)
12. Fernandes, B., Fernandes, J.: Bare hand interaction in tabletop augmented reality. In: *International Conference on Computer Graphics and Interactive Techniques, ACM Special Interest Group on GRAPHics and Interactive Techniques* (2009)
13. Seichter, H., Grasset, R., Looser, J., Billinghurst, M.: Multitouch interaction for tangible user interfaces. In: *IEEE 8th International Symposium on Mixed and Augmented Reality*, pp. 213–214 (2009)
14. Wagner, D., Pintaric, T., Ledermann, F., Schmalstieg, D.: Towards massively multi-user augmented reality on handheld devices. In: Gellersen, H.-W., Want, R., Schmidt, A. (eds.) *PERVASIVE 2005*. LNCS, vol. 3468, pp. 208–219. Springer, Heidelberg (2005)
15. Schmalstieg, D., Fuhrmann, A., Hesina, G., Szalavári, Z., Encarnacão, L.M., Gervautz, M., Purgathofer, W.: The Studierstube augmented reality project. *Teleoperators and Virtual Environments* 11 (2002)
16. Henrysson, A., Billinghurst, M., Ollila, M.: Face to face collaborative AR on mobile phones. In: *IEEE and ACM International Symposium on Mixed and Augmented Reality* (2005)
17. Kovac, J., Peer, P., Solina, F.: Human skin color clustering for face detection. *European Science Fiction Convention. The IEEE Region 8(2)*, 144–148 (2003)
18. Hardenberg, C.V., Berard, F.: Bare-hand human-computer interaction. In: *ACM International Conference Proceeding Series*, vol. 15 (2001)
19. Lee, G.A., Billinghurst, M., Kim, G.J.: Occlusion based interaction methods for tangible augmented reality environments. *Virtual Reality Continuum and its Applications in Industry*, 419–426 (2004)

# Accurately Measuring the Satisfaction of Visual Properties in Virtual Camera Control

Roberto Ranon<sup>1</sup>, Marc Christie<sup>2</sup>, and Tommaso Urli<sup>1</sup>

<sup>1</sup> HCI Lab, University of Udine, via delle Scienze 206, 33100, Udine, Italy

<sup>2</sup> IRISA/INRIA Rennes Bretagne Atlantique, Campus de Beaulieu, 35042, Rennes Cedex, France

**Abstract.** Declarative approaches to camera control model inputs as properties on the camera and then rely on constraint-based and/or optimization techniques to compute the camera parameters or paths that best satisfy those properties. To reach acceptable performances, such approaches often (if not always) compute properties satisfaction in an approximate way. Therefore, it is difficult to measure results in terms of accuracy, and also compare approaches that use different approximations. In this paper, we propose a simple language which can be used to express most of the properties proposed in the literature and whose semantics provide a way to accurately measure their satisfaction. The language can be used for several purposes, for example to measure how accurate a specific approach is and to compare two distinct approaches in terms of accuracy.

## 1 Introduction

Camera Control is an essential component of a large range of 3D applications, including 3D games, data exploration and visualization, virtual walk-throughs, 3D modelling and virtual storytelling [9]. In the literature, a number of camera control techniques have been proposed, ranging from interactive techniques – where the user directly or indirectly controls the camera parameters– to automated approaches in which that camera parameters and paths are computed automatically in a way that supports the user in the realisation of his tasks.

Within the category of automated approaches, *declarative* approaches focus on providing a general solution to the problem of camera control by following a three-step process: first design a general language to model camera control problems, then provide techniques to solve the problems described by the language, and finally propose means to explore the classes of solutions. Such declarative approaches generally model camera control problems as a set of properties that need to hold (*i.e.* constraints to solve) and a set of properties that should hold whenever possible (*i.e.* cost functions to optimize). The literature on declarative approaches to camera control reports three classes of properties:

- properties that directly bind camera parameters/path parameters to given values *e.g.* fix the camera up vector to ensure a horizontal view;

- properties that express geometric requirements on camera parameters/paths with respect to objects in the scene, *e.g.* requiring that the camera be at a certain distance from an object;
- properties that express requirements on the image(s) the camera will generate, *e.g.* requiring a certain object to be positioned in a given portion of the screen and not be occluded.

The last category of properties (hereinafter referred to as *screen-space properties*) is particularly interesting for several reasons. First, screen-space properties allow the user (or the application) to reason in terms of the result of the camera control process, rather than in terms of values for camera parameters or camera paths. Second, these are closer to the language a photographer/cinematographer would use to position a camera in a real world. For example, the approach described by Christie and Normand [8] uses properties such as *Close-up* and *Long shot* to express requirements on the size of objects in images. Third, they represent an expressive groundwork on which to build more abstract properties for example derived from cinematography and photography (*e.g.* rules of the thirds; Gooch *et al.* [11] use this rule to improve the composition of shots through local modifications of the camera parameters).

An overlooked issue in approaches to declarative camera control is accuracy: how exact is the evaluation of a shot, with regard to a set of properties. To reach acceptable computing times, declarative approaches often (if not always) compute satisfaction of screen-space properties in an approximate way or using approximate models, therefore leading to situations where the layout of objects on screen does not fully satisfy the given specification (*e.g.* an object may be evaluated as fully occluded while it is only partially occluded). Approximate models and approximate evaluations lead to both *false-good* and *false-bad* cases. A contribution may then consist in designing exact methods to accurately evaluate camera configurations with regard to screen-space properties, thereby offering means to compare the quality of different approaches, and to precisely evaluate the loss in accuracy of approximate models and approximate evaluators. The difficulty of reasoning about the efficiency/accuracy tradeoff is, in our view, one of the factors that limits the development of better declarative camera control approaches.

This paper deals with the above issues in two ways:

- it proposes a simple language which can be used to write expressions that capture most of the screen-space properties proposed in the literature, with the aim of providing a way to clearly express the meaning of screen-space properties;
- the language semantics provides a way to accurately measure if a camera satisfies an expression written in the language, thus allowing one to reason about the accuracy of a solution provided by existing declarative camera control approaches, or compare solutions together. However, while in principle our evaluation method could be used in place of approximate techniques in declarative camera control approaches developed so far, it is currently too computational costly for that purpose (see discussion at the end of the paper).

In this paper, we restrict our proposal to static camera control problems, i.e. Virtual Camera Composition problems, where the task is to compute the optimal configuration of camera parameters given a set of properties that must hold in a given 3D scene at a certain instant in time. Extensions to dynamic situations (where the task is to find camera paths, or more generally how camera parameters must change in time) represents a far more complex task and will be considered in future work.

The paper is organized as follows: Section 2 provides an overview of screen-space properties proposed by approaches in the literature; Section 3 presents our language for camera control, while in Section 4 we show how to express screen-space properties from the literature using our language. In Section 5 we present an example where we evaluate the accuracy of an existing approach in the literature in a specific situation. Finally, in Section 6 we conclude the paper and mention future work.

## 2 Screen-Space Properties in the Literature

Declarative approaches to camera control have been constantly exploring the balance between two orthogonal aspects: expressiveness and computational cost [9]. Expressiveness encompasses the range, nature and accuracy of constraints with which properties can be expressed over camera shots. For example, the classical **Framing** property constrains a target object to project inside a user-defined frame on the screen. The need for improving the expressiveness *i.e.* designing a cinematographic language that is both able to express a large range of properties, and that offers means to compose these properties, is of increasing importance with regard to the evolution of viewpoint computation and cinematography as demonstrated in most recent computer games [14]. On the other hand, the computational cost allotted to any camera control process directly drives the implementation of declarative approaches, generally at the cost of reducing the expressiveness. More importantly, the computational cost drives the level of abstraction of the geometry (how precisely should objects be represented and which geometric abstractions should be considered for the different properties). For example, a number of contributions rely on bounding representations for computing the visibility of targets, by casting rays from the camera to the vertices of the axis aligned bounding box [4, 5]. Furthermore limitations in the solving techniques restrict the range of possible abstractions (e.g. gradient-based techniques would require the evaluation of properties to be a differentiable function, or at least a smooth one [10]).

To provide the readers with an overview of screen-space properties used in declarative languages for camera control, we identified and gathered most common properties from the literature and display them in Table 1.

For a thorough overview of solving techniques in virtual camera control, we refer the readers to the overview proposed in [9].

**Table 1.** Main screen-space properties from the literature

Property	Description
Occlusion	Checks if target $T$ is occluded on the screen
	Checks if target $T_2$ occludes $T_1$
	Checks if target $T$ is not occluded on the screen
	Evaluation of occlusion is performed using projected bounding spheres.
Framing	Checks if no more than fraction $\min$ of the target $T$ is occluded
	Checks whether a target $T$ projects into a rectangular frame $F$ on the screen
	Checks whether target $T$ projects into intervals $X$ and $Y$
	Evaluation performed with nine-rays ray-casting to the bounding volume or off-screen rendering.
Screen Separation	Checks whether a target $T$ is out of view or occluded
	Checks whether target $T$ is at a distance $d$ to target $T_2$ on the screen
	Checks whether a target $T$ is projected in the field of view
	Evaluation not described in paper.
In Field Of View	Checks if a target $T$ is out of view or Occluded
	Checks if a target $T$ is excluded from the field of view
	Checks the projected size of a target $T$ is in a given range $v = \{\text{close-up}, \text{medium close-up}, \text{long-shot}\ldots\}$
	Evaluation not described in paper.
Size	Checks the area of a projected target $T$ is equal to a value $v$
	Evaluation uses the projected radius of the bounding sphere, the longest edge of the bounding box, the convex-hull of the projected bounding box or off-screen rendering.
Relative Spatial Location	Checks whether projected target $T_1$ is target $T_2$ ( $\text{l}(\text{eft})/\text{r}(\text{ight})/\text{a}(\text{bove})/\text{b}(\text{elow})$ )
	Evaluation done as geometric reasoning on the projected bounding volume of the objects or considering the position of the camera with respect to a plane which passes through the centers of the objects.
BetweenObjects( $T_1, T_2, T_3, XY$ ) [13]	Evaluation not explained in paper.

### 3 The Camera Evaluation Language

In this Section, we propose the *Camera Evaluation Language* (hereinafter, *CEL*). Its design is guided by two motivations: first, to propose a simple language that can be used by present and future declarative approaches to express their screen-space properties, and then to provide a way to reason about accuracy and the approximations that eventually need to be included. As we will see, all screen-space properties in table I can be expressed with a few simple CEL primitives, namely the *Rendering* and *Pixel Set* operators, and the common mathematical, logical and set operators and relations.

**Table 2.** Operators and relations in CEL. In the table,  $PS, PS_1, PS_2$  denote pixel sets.

Operator or Relation	Returns
$R(subscene)$	$PS =$ the set of pixels $p$ that results from rendering <i>subscene</i> from <i>camera</i> , with $p_{side} = 0$ for each $p$
$CR(subscene)$	$PS =$ the set of pixels $p$ that results from rendering <i>subscene</i> from the position of <i>camera</i> , using 90 degrees FOV, perspective projection and view direction towards any face of a cube centered in the camera, with $p_{side} = 0, \dots, 5$ depending on which side of the cube $p$ belongs to
$Max_x(PS, side)$	$\max(\{p_x   p \in PS \wedge p_{side} = side\})$
$Min_x, Max_y, Min_y$	...
$Max_z, Min_z$	...
$Avg_x(PS)$	$\text{avg}(\{p_x   p \in PS\})$
$Avg_y, Avg_z$	...
$Overlap(PS_1, PS_2)$	set of pixels in $PS_1$ that have the same $x, y, side$ coordinates of some pixels in $PS_2$
$CoveredBy(PS_1, PS_2)$	set of pixels of $PS_1$ that would be covered by pixels of $PS_2$ if we rendered together the subscenes that produced $PS_1$ and $PS_2$
$Left(PS_1, PS_2)$	set of pixels of $PS_1$ that are left of any pixel in $PS_2$ , considering only pixels with $p_{side} = 0$
$Right, Above, Below$	...
$Distance(PS_1, PS_2)$	$\min(distance(p, p'))$ where $p \in PS_1, p' \in PS_2$ , , considering only pixels with $p_{side} = 0$

#### 3.1 Rendering Operators

Rendering operators take any subpart of a 3D scene (e.g., an object, a group of objects, or the entire scene), render it into an image with size *imageWidth*, *imageHeight* (or a cube map), and return a set containing all pixels that, in the image, refer to the part of the 3D scene given as argument. Rendering operators are the primitive components of the language on which all operations are performed.

Rendering operators assume the existence of a current *camera*, from which rendering is performed. This is the camera we want to evaluate with respect to a set of properties. CEL defines two rendering operators, *R* and *CR* (see table 2, first two rows). The first one just returns the set of pixels resulting from rendering its argument into a 2D texture. The second one returns the set of pixel resulting from rendering its argument into a cube map from the *camera* position, using six orthogonal views (i.e., the operator arguments are rendered six times, one for each face of the cube map, starting from the current camera view direction).

Pixel sets that are returned by rendering operators are defined as follows: each pixel  $p$  is defined by its coordinates  $p_x, p_y, p_z, p_{side}$  where  $p_x, p_y$  are the coordinates of the pixel in the rendered image (or side of the cube map),  $p_z$  is its distance from *camera* and  $p_{side} = 0, 1, \dots, 5$  denotes one of the sides of the cube in case a cube map has been rendered, and is 0 in case we have rendered a 2D image. The concept of pixel set is similar to the notion of depth sprite or nailboard in image-based rendering [15].

At a practical level, the resulting pixel set can be easily computed by rendering the specified subscene part, and then take the resulting pixels (*i.e.* where color is different from the background, or the corresponding value of the Z-buffer is less than the maximum z value of the depth buffer).

### 3.2 Pixel Set Operators and Relations

Once the rendering is performed into pixel sets, a number of comparisons can be performed by applying simple operators over the pixel sets (*e.g.* computing the overlapping regions or relative spatial location). Such Pixel Set operators (see table 2, third to eighth row) and relations act on sets of pixels, perform calculations, and return numbers or pixel sets. In the following,  $p(x, y, z, side) \in PS$  is true if there exists a pixel  $p \in PS$  with such coordinates (similarly, we define also  $p(x, y, side)$ ).

Besides the operators that perform basic calculations on one set (*Max*, *Avg* and *distance*, see Table 2, third and fourth row), we define the *Overlap* operator ( $Overlap(PS_1, PS_2)$ ) that returns those pixels in  $PS_1$  that have the same  $x, y, side$  coordinates as pixels in  $PS_2$  (see figure 11), *i.e.*:

$$Overlap(PS_1, PS_2) = \{p(x, y, side) \in PS_1 | p'(x, y, side) \in PS_2\}$$

Since we consider also the *side* coordinate in the operator, when two pixel sets resulting from *CR* are used, the comparison is done on each of the six generated images.

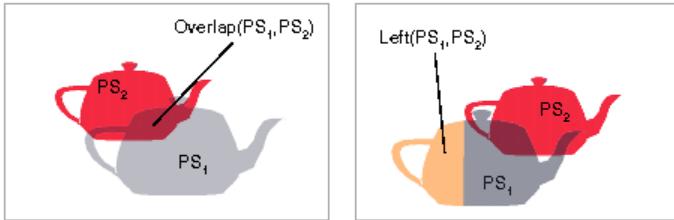
The operator *CoveredBy*( $PS_1, PS_2$ ) returns the pixels in  $PS_1$  that would be discarded by z-test if we rendered together the subscenes that produced  $PS_1$  and  $PS_2$ , *i.e.* it takes into account the overlapping region, and perform comparison on the *z* coordinate:

$$\text{CoveredBy}(PS_1, PS_2) = \{p(x, y, side) \in PS_1 | p'(x, y, side) \in PS_2 \wedge p_z > p'_z\}$$

The following operators act only on pixels with  $p_{side} = 0$ . More specifically,  $\text{Left}(PS_1, PS_2)$  returns the pixels in  $PS_1$  that are left of any pixel in  $PS_2$ , i.e.

$$\text{Left}(PS_1, PS_2) = \{p \in PS_1 | p_x < \text{Min}_x(PS_2)\}$$

Similarly, we define also *Right*, *Above*, and *Below* (these last two perform the comparison using the  $y$  coordinate).



**Fig. 1.** The *overlap* operator (left) and the *Left* operator (right) in the case of pixel sets obtained by using the *R* operator

## 4 Using CEL to Build and Evaluate Screen-Space Properties

In this Section, we propose some CEL expressions to measure values related to the screen-space properties defined in the literature.

The following expressions express and evaluate the size of a target  $T$  relative to the viewport size, *i.e.* its height, width or area on the viewport:

$$\begin{aligned}\text{Height}(T) &= \frac{\text{Max}_y(R(T)) - \text{Min}_y(R(T))}{\text{imageHeight}} \\ \text{Width}(T) &= \frac{\text{Max}_x(R(T)) - \text{Min}_x(R(T))}{\text{imageWidth}} \\ \text{Area}(T) &= \frac{|R(T)|}{\text{imageWidth} \times \text{imageHeight}}\end{aligned}$$

The following expressions express and evaluate the relative position on the viewport of two targets  $T_1, T_2$ :

$$\text{ScreenSeparation}(T_1, T_2) = \frac{\text{Distance}(R(T_1), R(T_2))}{\text{imageWidth}}$$

$$\begin{aligned}
\textbf{LeftOf}(T_1, T_2) &= \frac{|Left(R(T_1), R(T_2))|}{|R(T_1)|} \\
\textbf{RightOf}(T_1, T_2) &= \frac{|Right(R(T_1), R(T_2))|}{|R(T_1)|} \\
\textbf{AboveOf}(T_1, T_2) &= \frac{|Above(R(T_1), R(T_2))|}{|R(T_1)|} \\
\textbf{BelowOf}(T_1, T_2) &= \frac{|Below(R(T_1), R(T_2))|}{|R(T_1)|} \\
\textbf{InFrontOf}(T_1, T_2) &= \frac{|CoveredBy(Overlap(R(T_1), R(T_2)), R(T_1))|}{|Overlap(R(T_1), R(T_2))|} \\
\textbf{InsideOf}(T_1, T_2) &= \frac{|CoveredBy(R(T_2), R(T_1))|}{|R(T_1)|}
\end{aligned}$$

To express and evaluate inclusion of a target in the viewport, we use an additional geometry *CVV* (*Camera View Volume*) whose shape, position and orientation corresponds to the Camera View Volume:

$$\textbf{InViewVolume}(T) = \frac{|CoveredBy(CR(T), CR(CVV))|}{|CR(T)|}$$

i.e., we compute the fraction of pixels of the target that are covered by pixels of the view volume. By using *CR*, we also take into account the pixels that are out of the viewport.

To express and evaluate framing, we use a similar idea, i.e. we render an additional geometry *SAS* (*Screen Aligned Shape*) which is a 2D shape positioned just after the camera near plane):

$$\textbf{Framing}(T, SAS) = \frac{|CoveredBy(CR(T), CR(SAS))|}{|CR(T)|}$$

i.e., we compute the fraction of pixels of the targeted that are covered by pixels of the view volume. This approach allows to use any shape as the frame, and also, since we use *CR*, to set the frame (partly) outside the viewport.

Finally, occlusion can be expressed and measured by the following expressions (*Scene* denotes the entire scene):

$$\begin{aligned}
\textbf{Occluded}(T) &= \frac{|CoveredBy(R(T), R(Scene - T))|}{|R(T)|} \\
\textbf{OccludedBy}(T_1, T_2) &= \frac{|CoveredBy(R(T_1), R(T_2))|}{|R(T_1)|}
\end{aligned}$$

In cases where, for some argument *T* of an operator,  $R(T)=\emptyset$  (i.e. the argument is not in the camera view volume), there might be two problems:

- the expression computes a division by zero (e.g. *Occluded*), or
- the expression does not know how to compute a result (e.g. *ScreenSeparation*).

In those cases, we define the expression to return -1 as a result. Note that this is not the case of *Height*, *Width* and *Area*, where if the target is not in the camera view volume, the computed result (i.e. zero) is correct.

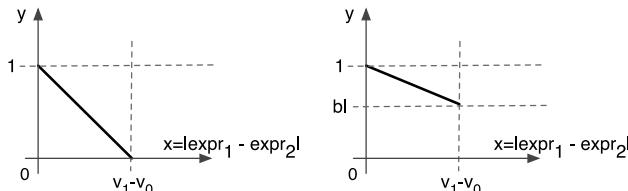
The accuracy of evaluating an expression depends on the size of the image to which rendering operators compute their results. Of course, since these operators are based on rasterization, they can never be perfectly correct. However, we can safely assume the evaluation as accurate when rendering to images that are at least the same size as the intended application window, since using a greater accuracy would not be appreciable by the user.

Since screen-space properties in the literature typically are expressed with respect to some desired value (e.g. height of an object equal to ...), in order to express screen-space properties we define also a comparison function:

$$\text{Equal}(\text{expr}_1, \text{expr}_2, v_0, v_1, bl) = \begin{cases} 0 & \text{if } \text{expr}_1 \notin [v_0, v_1] \vee \text{expr}_2 \notin [v_0, v_1] \\ 1 + |\text{expr}_1 - \text{expr}_2| \frac{bl - 1}{v_1 - v_0} & \text{otherwise} \end{cases}$$

where  $\text{expr}_1$ ,  $\text{expr}_2$  are two CEL expressions (or possibly, just a numeric value),  $0 \leq v_0 < v_1$  define an interval inside which the comparison is at least partly satisfied, and  $0 \geq bl \geq 1$  defines the minimum value of satisfaction when both  $\text{expr}_1$ ,  $\text{expr}_2$  are in  $[v_0, v_1]$ . *Equal* returns a value in  $[0,1]$  indicating how close the actual values computed for the expressions are, provided that they are inside the acceptable range (if not, the returned value is zero):

Figure 2 shows the *Equal* function with different *baseline* settings.



**Fig. 2.** The *Equal* function with  $bl=0$  (left),  $bl=0.6$  (right). To simplify the graph, we have represented  $|\text{expr}_1 - \text{expr}_2|$  in the horizontal axis.

For example,  $\text{Equal}(\text{OccludedBy}(T_1, T_2), 0.7, 0.5, 1, 0)$  means that we would like  $T_1$  occluded by  $T_2$  for the 70% of its rendered area. Satisfaction will be therefore 1 for that value. If the occluded fraction is less than 0.5, satisfaction will be 0. For values between 0.5 and 0.7, satisfaction will vary in  $[0,1]$  and be proportional to how close the value is to the desired 0.7 value.

## 5 Example: Using CEL to Measure the Accuracy of Existing Approaches

In this Section, we show how to use CEL to measure the accuracy of the approaches proposed by Burelli et al. [5] and Christie et al. [8] for a specific situation, i.e. the



**Fig. 3.** The image computed by the camera we are evaluating with respect to the properties in Table 3

camera from which the picture in figure 3 has been rendered. The set of screen-space properties we consider in this example are (in the formulation adopted in [5]) listed in the first column of Table 3 ( $t$  is the transporter in figure 3, which shows also its bounding box); their CEL equivalent expressions are shown in the third column of the same table.

The first property requests the transporter to be entirely in the camera view volume. In [5], the paper abstracts the transporter using an Axis Aligned Bounding Box (AABB) representation, it returns 1 if all the corners of its AABB are in the camera view volume, 0 if no corner is in the camera view volume, and 0.5 otherwise. Thus the value computed by [5] is 0.5 (partially in view volume) whereas the target is fully included in the view volume. The second property requires the transporter to be fully unoccluded. Since [5] measures that by ray casts towards the corners and center of the transporter AABB, it reports the object to be fully unoccluded which is obviously false. The third property requires the transporter projected size to be 30% of the image area. Since [5] measures size by evaluating the area of the projected bounding sphere, it returns the value 0.88 (i.e. we are 88% close to the desired value). Now, the fourth column of Table 3 reports the

**Table 3.** Properties and their measured values for the viewpoint displayed in Figure 3;  $t$  is the transporter

Property (as in [5])	value (as in [5])	CEL expression	CEL value	Error
$objInFOV(t, 1.0, 1.0)$	0.5	<code>InViewVolume(<math>t</math>) = 1</code>	1.0	50%
$objOcclusion(t, 0.0, 1.0)$	1.0	<code>Occluded(<math>t</math>) = 0</code>	0.87	13%
$objProjSize(t, 0.3, 1.0)$	0.88	<code>Area(<math>t</math>) = 0.3</code>	0.18	70%
Property (as in [8])	value (as in [8])	CEL expression	CEL value	Error
$Framing(t, -1, 1, -1, 1)$	0.37	<code>InViewVolume(<math>t</math>) = 1</code>	1.0	63%
$NoOcclusion(t)$	0.1	<code>Occluded(<math>t</math>) = 0</code>	0.87	90%
$LongShot(t)$	0.7	<code>Area(<math>t</math>) = 0.3</code>	0.18	52%

values obtained by accurate evaluation using the equivalent CEL expressions. The total error accumulated by [5] is 1.33, *i.e.* the approach is off by an average of 44% compared to accurate evaluation, and the worst approximation is given by the *objProjSize* property. Now, we compare the results with Christie & Normand [8]. The authors rely on a spherical representation to abstract targets and since the transporter is a long shaped object, the *Framing* property returns a very approximate result (only 37% is in the frame). The *NoOcclusion* studies the overlap of the projecting spheres of the target and of the occluders and clearly states that the target is occluded by a large value (the bounding of the lamp is very large). Since the projection size is not modeled in [8], we replace it by a *Long Shot* property. Average mistake in the case of paper [8] is 68%.

## 6 Discussion and Conclusions

In this paper, we have presented a simple but expressive language for specifying screen-space properties in Virtual Camera Control problems. The language enables a clear specification of most properties in the literature and furthermore enables the accurate measuring of their satisfaction. We hope that the language can also be a valuable tool in measuring accuracy when designing and implementing new camera control approaches where complex geometries need to be abstracted and approximations performed for the sake of performance. To this purpose, an implementation of the language, as well as more examples of using it are available at <http://www.cameracontrol.org/language>.

Future work will consist in extending the language to deal with dynamic camera control, and use it in the direction of establishing a benchmarking environment to compare models, techniques and algorithms in Virtual Camera Control. Moreover, we are also exploring the possibility of computing CEL expressions in the GPU, together with low resolution pixel sets, and see if we can get performances that are acceptable to be used inside a declarative camera control approach.

**Acknowledgments.** Authors acknowledge the financial support of the Italian Ministry of Education, University and Research (MIUR) within the FIRB project number RBIN04M8S8, as well as European Grant number 231824.

## References

- [1] Bares, W.H., Lester, J.C.: Intelligent multi-shot visualization interfaces for dynamic 3d worlds. In: IUI 1999: Proceedings of the 4th international conference on Intelligent user interfaces, pp. 119–126. ACM, New York (1999)
- [2] Bares, W.H., McDermott, S., Boudreux, C., Thainimit, S.: Virtual 3d camera composition from frame constraints. In: MULTIMEDIA 2000: Proceedings of the eighth ACM international conference on Multimedia, pp. 177–186. ACM, New York (2000)

- [3] Barres, W.H., Thainimit, S., McDermott, S.: A model for constraint-based camera planning. In: Proceedings of AAAI Spring Symposium on Smart Graphics, pp. 84–91 (2000)
- [4] Bourne, O., Sattar, A., Goodwin, S.: A constraint-based autonomous 3d camera system. *Constraints* 13(1-2), 180–205 (2008), ISSN 1383-7133, <http://dx.doi.org/10.1007/s10601-007-9026-8>
- [5] Burelli, P., Di Gaspero, L., Ermetici, A., Ranon, R.: Virtual camera composition with particle swarm optimization. In: Butz, A., Fisher, B., Krüger, A., Olivier, P., Christie, M. (eds.) SG 2008. LNCS, vol. 5166, pp. 130–141. Springer, Heidelberg (2008)
- [6] Burelli, P., Jhala, A.: Dynamic artificial potential fields for autonomous camera control in 3d environments. In: Proceedings of Artificial Intelligence and Interactive Digital Entertainment (AIIDE 2009). AAAI Press, Menlo Park (2009)
- [7] Christie, M., Languénou, E.: A constraint-based approach to camera path planning. In: Butz, A., krüger, A., Olivier, P. (eds.) SG 2003. LNCS, vol. 2733, pp. 172–181. Springer, Heidelberg (2003)
- [8] Christie, M., Normand, J.-M.: A semantic space partitionning approach to virtual camera control. In: Proceedings of the Annual Eurographics Conference, pp. 247–256 (2005)
- [9] Christie, M., Olivier, P., Normand, J.-M.: Camera control in computer graphics. *Comput. Graph. Forum* 27(8), 2197–2218 (2008)
- [10] Drucker, S.M., Zeltzer, D.: Intelligent camera control in a virtual environment. In: Proceedings of Graphics Interface 1994, pp. 190–199 (1994)
- [11] Gooch, B., Reinhard, E., Moulding, C., Shirley, P.: Artistic composition for image creation. In: Proceedings of the 12th Eurographics Workshop on Rendering Techniques, pp. 83–88. Springer, London (2001) ISBN 3-211-83709-4
- [12] Halper, N., Helbing, R., Strothotte, T.: A camera engine for computer games: Managing the trade-off between constraint satisfaction and frame coherence. *Comput. Graph. Forum* 20(3) (2001)
- [13] Halper, N., Olivier, P.: CAMPLAN: A camera planning agent. In: Smart Graphics 2000 AAAI Spring Symposium, pp. 92–100. AAAI Press, Menlo Park (2000)
- [14] Hawkins, B.: Real-Time Cinematography for Games (Game Development Series). Charles River Media, Inc., Rockland (2004) ISBN 1584503084
- [15] Schaufler, G.: Nailboards: A rendering primitive for image caching in dynamic scenes. In: Rendering Techniques 1997: Proceedings of the Eurographics Rendering Workshop, pp. 151–162. Springer, Heidelberg (1997)

# VEX-CMS: A Tool to Design Virtual Exhibitions and Walkthroughs That Integrates Automatic Camera Control Capabilities

Luca Chittaro, Lucio Ieronutti, and Roberto Ranon

HCI Lab, University of Udine, via delle Scienze, 206,  
33100 Udine, Italy  
<http://hcilab.uniud.it>

**Abstract.** This paper presents VEX-CMS, a tool to build 3D virtual museums and exhibitions, architectural walkthroughs and, more generally, applications where 3D and 2D content is presented to the user inside a 3D Virtual Environment. In this paper, we concentrate on the task of designing tours of the virtual environment to be followed by visitors, either through manual or automatic navigation, and show how a recent automatic camera control approach [3], coupled with path planning, can be exploited to accomplish the task with minimal knowledge and manual effort on the curator's side.

**Keywords:** 3D virtual museums, automatic camera control, 3D authoring tools.

## 1 Introduction

A number of graphics applications, such as 3D virtual museums and exhibitions, or architectural walkthroughs, are often built around the idea of having the user enter a 3D Virtual Environment (hereinafter, VE) and walk around (or follow pre-computed paths), see objects and places in the environment and possibly get information on them.

Although this sort of applications has been built since the 80s, they are still difficult to produce without the help of 3D graphics and computer professionals. A number of projects has focused on this issue, concentrating on problems such as facilitating the acquisition of 3D models of existing artifacts [1] or organizing collections of artifacts into digital libraries with metadata [2], from which 3D virtual exhibitions can then be assembled.

In this paper, we take a different approach and focus on museum or exhibition curators who, starting from available 3D models of a building and objects of interest, want to build a 3D virtual exhibition (or architects who want to build an interactive walkthrough of a virtual building). More specifically, we concentrate on the task of designing tours of the VE to be followed by a visitor, either with manual or automatic navigation, and show how a recent automatic camera control approach [3] can be exploited to accomplish the task with minimal knowledge and manual effort on the curator's side.

The paper is organized as follows. Section 2 reviews related work on tools for the construction of virtual exhibitions. Section 3 briefly presents the application we have

developed, called *VEX-CMS* (*Virtual EXhibition Content Management System*). Section 4 concentrates on tour and camera control aspects. Section 5 briefly presents examples of applications built with VEX-CMS. Finally, Section 6 concludes the paper and outlines future work.

## 2 Related Work

Virtual museums and exhibitions have been studied in different areas (e.g. human-computer interaction, 3D graphics and virtual reality, database systems) and research has been carried out on many aspects involved in their construction. For example, Patel et al. [1] reports the main outcomes of the European Project ARCO in developing a complete solution for digitizing artworks, storing them into multimedia databases together with relevant metadata [2], and presenting them in the context of 3D or augmented-reality exhibitions. However, there has not been much work on the specific aspect of making the design of the interactive 3D exhibition easy for non technical users such as curators and architects. One possibility that is often suggested is to exploit game engine tools. However, as noted by one of the proponents of this approach [4], programmers are still needed, and work is required to remove game-specific features and ultimately adapt the logic of the tool to a different purpose.

Some research for simplifying the creation of VEs for non technical users has been carried out considering the special case of children. For example, Kids Movie Creator (KMC) [5] is a tool that allows young children (aged 7 to 12) to create interactive VEs. KMC introduces many solutions to simplify interaction both at the conceptual and practical levels, from adopting an avatar-based interaction (i.e., the author of the VE controls an avatar that walks through the environment) to limiting modeling capabilities and object arrangement with discrete steps.

Alice [6] is a 3D programming environment designed for undergraduates with no 3D graphics and programming experience that allows to build VEs and program their behavior. Our work shares some of the assumptions and goals behind Alice, e.g. the fact that VE creation could be interesting for a much broader audience that will not necessarily have a mathematical and programming background, the importance of designing a VE creation tool with a target audience in mind, and providing users with simple methods to specify VE logic.

Assisted or semi-automatic camera control has been experimented in the context of virtual museums, but typically for helping the visitor find its way and easily navigate through the museum. For example, the research presented in [7] develops an intelligent camera control system and applies it to the task of navigating a virtual museum, using constraints to specify camera requirements. We also exploit a constraint-based approach but focus on the largely unexplored topic of using automatic camera control in the authoring process of a VE. The approach proposed by Elmquist et al. [8] automatically generates a tour of a VE that includes all landmarks; in this way, however, the curator has no control on how the visitor will experience the virtual exhibition, and so this approach might not be well suited to virtual exhibitions or architectural walkthroughs.

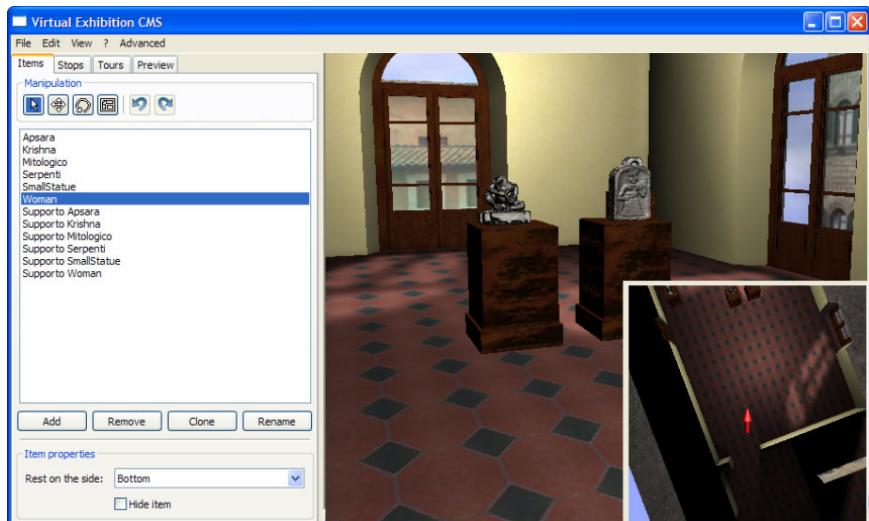
To introduce automatic camera control in the authoring process, Bares et al. [9] adopt a visual interface to specify constraints and compute static cameras, with the

goal of simplifying the process of camera placement for the non-technical user. While this kind of interface could be useful in a virtual exhibition context, it would still require the curator to concentrate on aspects such as the position, size and angle of objects in the derived camera. In our approach, we instead choose to provide a simpler and higher-level interface to the curator, only focusing on which objects should be shown by a camera and their relative importance.

### 3 The Virtual Exhibition Content Management System

VEX-CMS organizes the design of a virtual exhibition in two phases: arrangement of objects of interest in the VE, and design of virtual tours of the VE. In the following, we use the terms *curator* to indicate the creator of the VE, and *visitor* to indicate the user of the VE.

The main interface (see Figure 1) of VEX-CMS is organized in two parts: (i) a first-person view of the VE, where the curator's avatar can walk (or fly) and arrange objects in space, (ii) a series of tabbed panes, that describe the *objects of interest (items)*, *views of interest (stops)*, and *tours* that make up the virtual exhibition (the last two features will be described in detail in Section 4). The system is not meant for object modeling purposes, in the sense of creating or modifying 3D models, since that is typically outside the abilities of a curator and can be done better with existing 3D modeling tools such as Maya, 3DS or Blender. To build a virtual exhibition with VEX-CMS, a curator needs an existing 3D model of the VE and 3D models of all the objects she wants to arrange in the VE (by means of translation, rotation and scale operations).



**Fig. 1.** The main interface of VEX-CMS. On the right, a first person perspective of the VE (with the overlaid interactive map) where the curator can navigate (by walking or flying) and arrange objects; on the left, the list of currently inserted objects, and buttons to manage and arrange objects.

The first-person view of the VE can be augmented with an overlaid interactive map in a layout that is significantly more efficient than using just the first-person perspective, as we determined in an experiment with non-technical users [10]. Moreover, we added the possibility of introducing *snap-to* constraints when positioning objects, by specifying the side of the object (e.g. its bottom) that should stay attached to adjacent geometry (e.g., the floor or an artwork support). In this way, whenever the object is not attached to anything on the specified side, it will “fall” in a direction perpendicular to that side (e.g. down, if bottom is used as the chosen side) until it finds a geometry on which to stand.

The idea we followed in designing the application was to closely reproduce the experience of designing a real exhibition, where the curator visits the actual exhibition space and decides where to position objects and explanations. However, one important part of real exhibitions, i.e. lighting, cannot be directly controlled inside the application, but has to be incorporated into the 3D modeling process.

## 4 Designing Virtual Tours

In our approach, virtual tours are based on the notion of *View of Interest* (VOI), which extends the more typical notion of *Point of Interest*. A VOI is a point in the VE to which we associate: (i) a camera directed at something interesting in the context of the exhibition (e.g., to show one or more artworks), and (ii) related information, represented in HTML files that can be displayed on a 2D overlay or applied as a texture to some geometry in the VE. Therefore, the concept of VOI encapsulates both a point of interest in the VE and an interesting view direction from that point, or, in other words, a set of objects that can be appreciated from that point of view.

We support the definition of VOIs in two ways: manual and automatic. In the first case, the curator simply defines a VOI by “taking a picture” from her current viewpoint. In this way, defining a VOI requires one to navigate the environment until the desired view is reached. In the automatic case, the curator chooses which objects need to be included in the VOI and their relative importance (see Figure 2), and then a VOI which best satisfies these requirements is computed. The VOI can then be used as it is, or refined with manual navigation (e.g. slightly changing the position or orientation to get a better view). In this last case, the system will store the manual VOI instead of the automatically computed one.

VOIs can be visually connected to form one or more tours of the VE (see Figure 3). Tours are directed graphs of VOIs, with conditions for activating transitions (from a VOI to another) and information on how the transition will be performed (manual navigation with navigation aid, automatic navigation, or teleporting). In the following, we describe these features in more detail.

### 4.1 Automatic Computation of VOIs

Automatic computation of VOIs is performed by exploiting the technique described in [3], which uses a declarative approach to camera composition problems. The user (or the application, in this case) sets a number of requirements for the desired camera, and then the position and orientation of the camera that best satisfies those requirements is computed through a constrained optimization process.

More specifically, the requirements that can be expressed are in the form of properties that can either directly bind camera parameters (e.g. to avoid upside-down cameras), or express geometric requirements on camera parameters with respect to objects in the scene (e.g. to look at a certain side of an object), or express requirements on the image the camera will generate, e.g. requiring a certain object to be unoccluded and of a certain size. The constrained optimization process then treats some of the requirements as constraints that cut the 6DOF (camera position and orientation) search space by defining so-called *feasible volumes* (i.e., portions of the scene in which the camera can be positioned to satisfy the constraints), and search inside the feasible volumes is carried out using a global optimization strategy called *Particle Swarm Optimization (PSO)* [11].

To use automatic computation of a VOI, the curator chooses a set of objects to be included in the VOI, possibly filtering the list of all available objects by selecting one of the rooms in the VE (this also constrains the search space considered by PSO, and helps focusing the optimization process). The curator can also order the objects by importance (see Figure 2), and the application uses that information to generate a set of camera control requirements that are provided as input to the automatic computation.

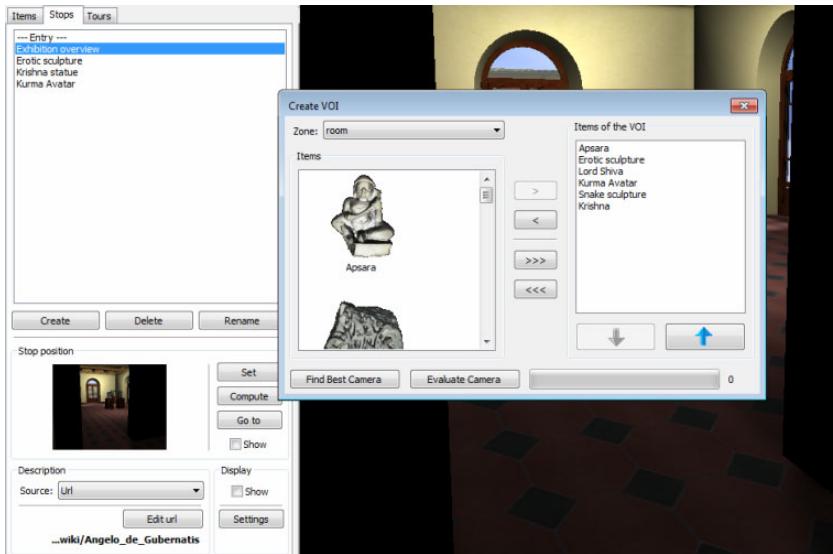
More specifically, for each object *obj* that has to be included in the VOI, the following requirements are generated (see [3] for more details on the semantics of the requirements):

$$\begin{aligned} & \textit{objInFOV}(\textit{obj}, 1.0, \textit{weight}) \\ & \textit{objHAngle}(\textit{obj}, 0, 40, \textit{weight}) \\ & \textit{objOcclusion}(\textit{obj}, 0, \textit{weight}) \\ & \textit{objProjSize}(\textit{obj}, \textit{size}, \textit{weight}) \end{aligned}$$

The first requirement (*objInFOV*) states that the object should be entirely included into the camera view volume, i.e. uncut in the image produced by the VOI camera (the 1.0 value is the required percentage of the object that must be in the camera view volume). The second requirement (*ObjHAngle*) states that the object should be seen from its front direction, with 40 degrees of tolerated deviation (the 0 value is the deviation with respect to the object front vector). The third requirement (*ObjOcclusion*) states that the object should not be occluded by other objects in the image produced by the camera (the 0 value is the required percentage of occlusion). The final requirement (*objProjSize*) states that the projected size of the object in the image produced by the camera should be equal to *size*, which is a function of the number of objects that are included in the VOI (i.e. *size* is proportional to  $1/(\text{number of objects})$ ), considering the image having an area equal to 1). Finally, all statements include a *weight* parameter, which defines how much a requirement counts in the objective function that is maximized with PSO (see [3] for more details) and is greater for more important objects.

Moreover, we add a number of requirements directly related to the camera, i.e. *camBindY* (*avatarHeight*, *avatarHeight*), *camBindRoll*(0,0), which set minimum and maximum values for the camera height (both set at the height of the visitor's avatar) and roll (which is then set as zero).

A possible issue with this approach might occur when the curator includes a set of objects which produce an over-constrained situation, i.e. a set of requirements that



**Fig. 2.** The interface for defining a VOI. In the bottom left part, one can see a preview of the VOI, and buttons to set a VOI from the current viewpoint, compute the VOI automatically, or go to a VOI. The second option opens the dialog window displayed in the center of the screen, which allows the curator to choose the objects to be included in the VOI (left part of the dialog window) and set their relative order of importance (right part of the dialog window).

cannot be all satisfied by a single camera. The requirements we generate are all used just in the optimization process, with the exception of *objHAngle*, which is used also as a constraint to cut the search space. This means that, except for *objHAngle* properties, the computation will return, even in over-constrained situations, a result which tries to satisfy as much requirements as possible, giving precedence to those with higher weight (i.e. to the most important objects). If the problem is over-constrained because of *objHAngle* properties (i.e. the search space becomes empty), we remove those constraints in the search space cutting process, and repeat the computation just using them in the optimization function, with the only drawback of taking more time to compute the result.

When the computation of the VOI ends, the result is shown to the curator together with a bar that graphically shows how much the requirements have been satisfied. The curator can accept the result (and possibly refine it manually) or edit the requirements to fine-tune them or add other requirements (e.g. framing).

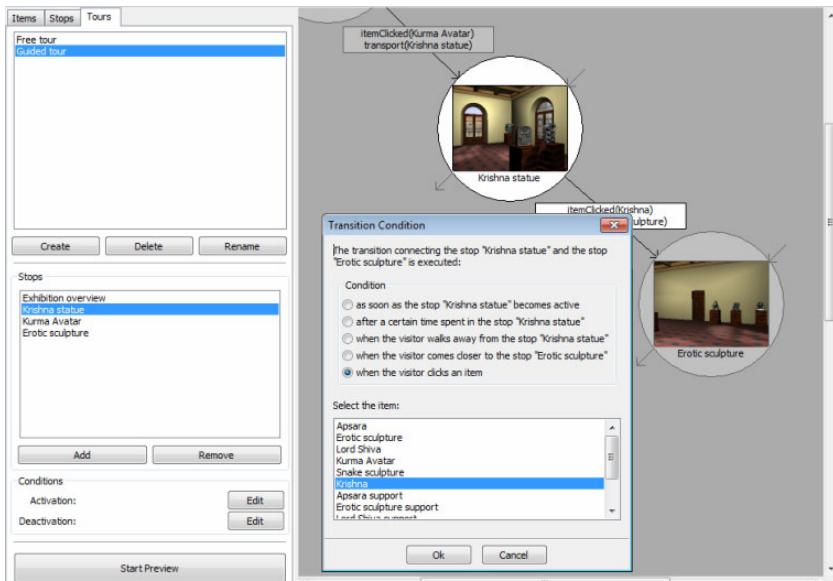
## 4.2 Connecting VOIs to Create Tours

Once a set of VOIs has been defined, the curator can connect them to create tours that a visitor can follow. Tours are displayed in the interface as graphs where VOIs are nodes, and possible transitions are represented as directed arcs that connect them (see Figure 3). The curator graphically lays out VOIs on a plane, and draws transitions as lines between them. Moreover, she can define multiple tours, which will then be proposed as alternatives when the visitor enters the VE.

From a formal point of view, the creation of a tour defines a finite state automaton, where states correspond to VOIs, and therefore, at each moment of a visit to the VE, one VOI will be the active one. Transitions between states are defined by choosing a condition for starting the transition and specifying how the transition has to be executed.

Starting conditions include events such as time elapsed since the starting VOI became active or actions done by the visitor, such as clicking on a certain item or moving away from the VOI. The transition can then be executed in three possible ways: by teleporting the visitor, by automatically navigating her avatar to the destination VOI (in this case, an animation will lead the visitor to the VOI, simulating a walk), or by drawing a path to the destination VOI (in this case, the visitor has to manually navigate the VE following the visual help provided by the drawn path).

This approach for specifying tours allows one to define different types of visits: from completely assisted ones (the visitor does not need to perform any navigation activity and is guided through the VE as if she were inside a virtual vehicle) to more interactive ones (the visitor can actively navigate the VE possibly with the help of visual paths). To define completely free visits (allowing the visitor to choose any order in which objects are seen), there is also the possibility of simply specifying enter and exit conditions for a VOI, without indicating a transition. The enter condition specifies the cases in which the automaton switches from a default state to a specific VOI state (e.g. when the visitor is sufficiently near to the VOI), while the exit condition handles the opposite situation (e.g. when a certain amount of time has elapsed since entering the VOI, we can go back to the default state).



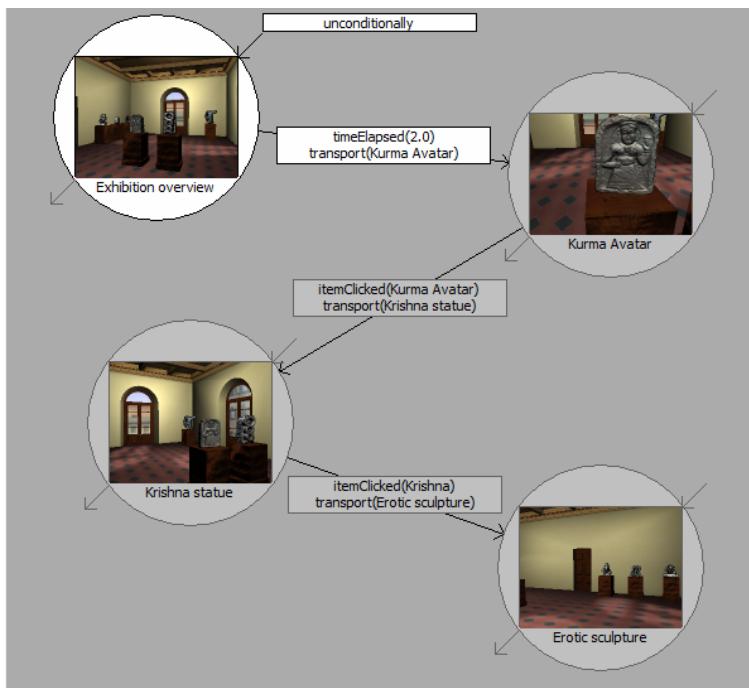
**Fig. 3.** The interface for defining a tour. VOIs are shown as icons in the right part of the interface, they can be arranged spatially and then connected by drawing arcs among them.

For path planning, we adopt a cell decomposition approach. In particular, using an automatic approach similar to the one described in [12], for each room of the VE (which is defined by bounding volumes) VEX-CMS derives a raster image in which each pixel corresponds to an area of the virtual room and the color of the pixel indicates whether the corresponding area contains a geometry (i.e., cannot be navigated) or not.

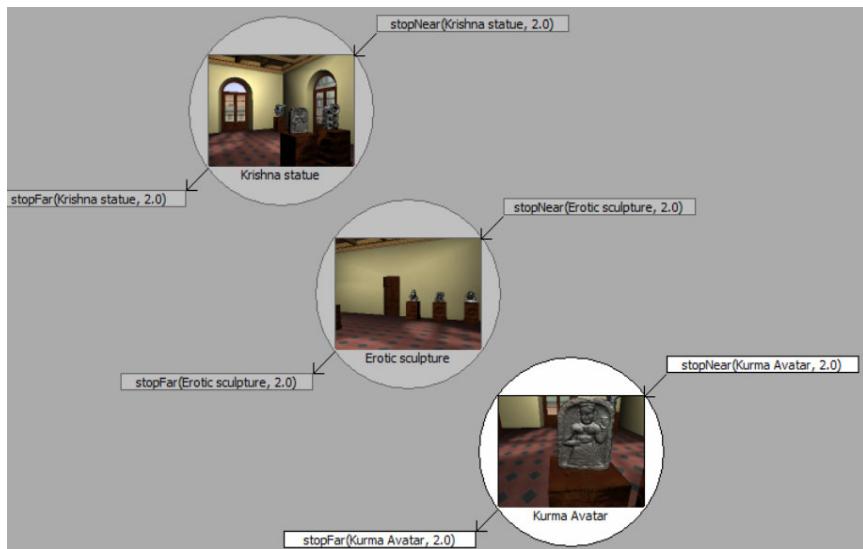
The path planner uses such images as well as connectivity information between rooms for building an adjacency graph, i.e. an undirected graph in which nodes corresponds to navigable areas while edges indicate if two areas are adjacent. Then, we compute collision-free paths in two steps, as in [13]. First, we derive the sequence of nodes that connect the nodes representing the start and end position by using Dijkstra's algorithm. Second, we compute a collision-free path connecting the centers of the cells corresponding to the sequence of nodes obtained from the previous phase.

## 5 Example Application

VEX-CMS is currently being used by a team of Indian art historians and experts to rebuild an Indian museum that was hosted in Florence, Italy at the beginning of the 19th century. 3D reconstructions of artworks have been obtained through laser scanning, while a 3D model of the original environment has been built from pictures of the original exposition.



**Fig. 4.** Automated tour for a room of the Indian exhibition



**Fig. 5.** Free tour for a room of the Indian exhibition



**Fig. 6.** An automatically computed camera showing all the artworks in a room of the virtual exhibition

Figure 4 shows a fully automatic tour that was designed for one of the rooms of the virtual exhibition. As soon as the visitor enters the room, the automaton goes into the first VOI state (top left VOI in Figure 4), which corresponds to an overview of the room (shown in Figure 6). After a few seconds, the visitor is transported to the first artwork, and associated information is overlaid on the screen. After clicking an artwork, the visitor is transported to the following one in the tour.

Figure 5 shows a free tour that was designed using the same VOIs. In this case, the visitor is free to navigate and visit artworks in any order, and a VOI is activated (or deactivated) every time the visitor is sufficiently near to (far from) it.

Finally, Figure 6 shows an example of a camera computed from requirements for all the six artworks in the room, which is an over-constrained situation (for example, not all objects can be seen from their front side).

## 6 Discussion and Conclusions

In this paper, we have presented an authoring tool for 3D virtual exhibitions and walkthroughs that uses intelligent camera control and path planning to make the design of virtual tours easier, and allows a curator to manage visitor's navigation in a new way, instead of simply relying on pre-computed paths or navigation aids such as arrows or signs. This flexibility opens up the possibility of mimicking and supporting typical visiting styles of exhibitions that have been observed in real museums. For example, Veron and Levasseur [14] have conducted ethnographic studies of museum visitors and have identified four categories of visitors, called *ant*, *fish*, *grasshopper* and *butterfly*. The ant visitor spends quite a long time to observe all exhibits, stops frequently and usually moves close to walls and exhibits. The fish visitor moves preferably in the center of rooms, and does not look at details of exhibits, making just a few or no stops. The grasshopper visitor sees only some exhibits, each for quite a long time, choosing them on the basis of personal interests and pre-existing knowledge about the contents of the exhibition. The butterfly visitor changes frequently the direction of visit, and sees almost all the exhibits, stopping frequently, but times vary for each exhibit. While the last two styles are only suited to free visits, the first two (ant and fish) can be respectively implemented by creating a tour composed by a VOI for each exhibit (the ant style) or less VOIs, but including more exhibits located in the same room (the fish style).

A limitation of our current approach is in the kind of exhibitions that can be designed. In particular, contemporary art exhibitions often include non-physical works (videos, interactive computer-based visualizations, ...), or exploit complex interactions of dynamic lights with the exhibitions space (e.g. lasers). However, while some of these features could be quite easily incorporated (e.g. videos), others are generally difficult to reproduce in a VE.

Directions for future work include experimenting with alternative ways of specifying VOIs from high-level requests (e.g. interfaces to easily specify framing properties) as well as providing more sophisticated results from the automatic VOI computation. For example, to handle over-constrained situations, instead of producing a single VOI, we could generate an establishing shot followed by several shots of smaller groups of items.

Another important direction for expanding the possibilities in the authoring phase is the incorporation of semantics into the artworks, as also stressed by the 3D-COFORM project [15]. This would allow the automatic camera control process (and therefore the curator) to target also specific parts or features of artworks.

Finally, we are currently evaluating the usability and effectiveness of VEX-CMS with several curators.

## Acknowledgements

The authors acknowledge the financial support of the Italian Ministry of Education, University and Research (MIUR) within the FIRB project number RBIN04M8S8.

## References

1. Patel, M., White, M., Walczak, K., Sayd, P.: Digitisation to Presentation - Building Virtual Museum Exhibitions. In: Proceedings of International Conference on Vision, Video and Graphics, pp. 189–196 (2003)
2. Mourkoussis, N., White, M., Patel, M., Chmielewski, J., Walczak, K.: AMS: metadata for cultural exhibitions using virtual reality. In: Proceedings of International Conference on Dublin Core and Metadata Applications, Dublin Core Metadata Initiative, pp. 1–10 (2003)
3. Burelli, P., Di Gaspero, L., Ermetici, E., Ranon, R.: Virtual Camera Composition with Particle Swarm Optimization. In: Butz, A., Fisher, B., Krüger, A., Olivier, P., Christie, M. (eds.) SG 2008. LNCS, vol. 5166, pp. 130–141. Springer, Heidelberg (2008)
4. Lepouras, G., Vassilakis, C.: Virtual museums for all: employing game technology for edutainment. *Virtual Reality* 8(2), 96–106 (2004)
5. Wang, T., Li, X., Shi, J.: An Avatar-Based Approach to 3D User Interface Design for Children. In: Proceeding of the Symposium on 3D User Interfaces, pp. 155–162. IEEE Computer Society Press, Los Alamitos (2007)
6. Conway, M., Audia, S., Burnette, T., Cosgrove, D., Christiansen, K., et al.: Alice: lessons learned from building a 3D system for novices. In: Proceedings of the Conference on Human Factors in Computing Systems, pp. 486–493. ACM Press, New York (2000)
7. Drucker, S.M., Zeltzer, D.: Intelligent camera control in a virtual environment. In: Proceedings of Graphics Interface, pp. 190–199 (1994)
8. Elmquist, N., Tudoreanu, M., Tsigas, P.: Tour Generation for Exploration of 3D Virtual Environments. In: Proceedings of VRST 2007: 14th ACM Symposium on Virtual Reality Software and Technology, pp. 207–210 (2007)
9. Bares, W., McDermott, S., Boudreaux, C., Thainimit, S.: Virtual 3D camera composition from frame constraints. In: Proceedings of MULTIMEDIA 2000: 8th ACM international Conference on Multimedia, pp. 177–186 (2000)
10. Chittaro, L., Ranon, R., Ieronutti, L.: 3D Object Arrangement for Novice Users: the Effectiveness of Combining a First-Person and a Map View. In: Proceedings of VRST 2009: 16th ACM Symposium on Virtual Reality Software & Technology, pp. 171–178. ACM Press, New York (2009)
11. Eberhart, R.C., Kennedy, J.: Particle swarm optimization. In: Proceedings of the IEEE International Conference on Neural Networks, vol. 4, pp. 1942–1948. IEEE Press, Piscataway (1995)

12. Ieronutti, L., Ranon, R., Chittaro, L.: Automatic Derivation of Electronic Maps from X3D/VRML Worlds. In: Proceedings of Web3D 2004: 9th International Conference on 3D Web Technology, pp. 61–70. ACM Press, New York (2004)
13. Kuffner, J.: Autonomous agents for real-time animation. Ph.D dissertation, Stanford University (1999)
14. Veron, E., Levasseur, M.: Ethnographie de l'Exposition. Bibliothque publique d'Information, Centre Georges Pompidou, Paris (1983)
15. Arnold, D.: 3D-COFORM: Tools and Expertise for 3D Collection Formation. In: Proceedings of EVA 2009, Florence, pp. 94–99 (2009)

# Automatic Speed Graph Generation for Predefined Camera Paths

Ferran Argelaguet and Carlos Andujar

MOVING Group, Universitat Politècnica de Catalunya, Barcelona, Spain  
`{fargelag, andujar}@lsi.upc.edu`

**Abstract.** Predefined camera paths are a valuable tool for the exploration of complex virtual environments. The speed at which the virtual camera travels along different path segments is key for allowing users to perceive and understand the scene while maintaining their attention. Current tools for speed adjustment of camera motion along predefined paths, such as keyframing, interpolation types and speed curve editors provide the animators with a great deal of flexibility but offer little support for the animator to decide which speed is better for each point along the path. In this paper we address the problem of computing a suitable speed curve for a predefined camera path through an arbitrary scene. We strive at adapting speed along the path to provide non-fatiguing, informative, interestingness and concise animations. Key elements of our approach include a new metric based on optical flow for quantifying the amount of change between two consecutive frames, the use of perceptual metrics to disregard optical flow in areas with low image saliency, and the incorporation of habituation metrics to keep the user attention. We also present the results of a preliminary user-study comparing user response with alternative approaches for computing speed curves.

## 1 Introduction

Computer graphics applications in areas such as urban management, factory planning, and mechanical design, have to deal with increasingly complex scenes with a large number of objects. Animations along predefined camera paths have proven to be a valuable tool for the inspection of complex scenes, being particularly useful for helping users to perceive and understand densely-occluded models [1]. Camera animations require specifying two components: the *camera path* (the values taken by the extrinsic camera parameters along the path) and a *speed curve* (a mapping of time/frame units to points along the path). Assuming constant intrinsic parameters, the camera path completely defines the sequence of viewpoints and thus the part of the scene that will be shown to the user, whereas the speed curve determines when and for how long the user sees the different scene elements. Although both components have a large impact on the user's mental representation and understanding of the environment, camera control literature has focused mainly on the camera path component (see [2] for a survey), whereas criteria for defining speed curves have received little attention from the research community.

Traditional tools provided by animation packages for camera animation, such as keyframing, interpolation types and speed curve graphical editors provide the animators with a great deal of flexibility but offer little support for the animator to decide which speed is better for each point along the path. Indeed, an adaptive speed approach offers multiple advantages over alternative approaches such as constant speed or simple acceleration curves. Some arguments supporting slow-motion include (a) giving more time to the user to perceive and understand the part of the scene being shown, and (b) less risk of producing visually-induced motion sickness. On the contrary, fast-motion might help to (a) maintain user attention while traversing uninteresting parts, (b) reduce animation length, and (c) save storage space in case the animation is saved as a movie. Our paper starts from the premise that deciding the *best speed* for each point along a path is a hard task for a normal user, particularly when the camera path includes multiple turns on a complex scene. Non-expert animators might create speed curves containing fast-paced segments prone to produce motion sickness to some viewers, and slow-paced segments which might prove tiresome for a part of the audience. And because many individual factors (such as age, gender and mental rotation ability) might influence user's perception and experience, the above problems might go unnoticed during animation authoring.

In this paper we address the problem of computing a suitable speed curve for a predefined camera path through an arbitrarily-complex scene. We strive at adapting speed along the path to provide non-fatiguing, informative, interestingness and concise animations. At the heart of our approach is a metric for quantifying the amount of scene change between consecutive frames which combines psychophysical results on optical flow, image saliency and habituation. Based upon this metric, we present an algorithm that takes as input a predefined camera path and outputs a suitable speed curve. Despite the large body of literature on the effects of optical flow on motion sickness and human locomotion, to the best of our knowledge this is the first work using optical flow, image saliency and habituation measures to automatically compute a suitable speed curve for a predefined camera path.

We focus on camera paths through complex scenes in fields such as mechanical engineering and shipbuilding design where semantics are roughly equally distributed among scene objects. Our approach moves away from a cinematographic point of view in that we focus on smoothness, non-fatiguing, informative and interestingness animations rather than on aesthetics, audience's emotional reactions, and other narrative aspects which play a key role in cinematography.

The rest of the paper is organized as follows. Section 2 discusses previous work on camera control and summarizes relevant psychophysical results on optical flow, motion sickness and image saliency. We give an overview of our approach in Section 3. Our metric for quantifying scene movement is described in Section 4, and the algorithm for computing speed curves is presented in Section 5. Section 6 discusses our results with two densely-occluded models, and presents a preliminary user study comparing alternative approaches. We give concluding remarks in Section 7.

## 2 Previous Work

**Camera control in real-time graphics.** A review of the state-of-art in automatic camera control can be found in [2]. In contrast to the camera path component, adaptive speed control has received very little attention from the research community. Mackinlay et al. [3] propose to move the camera towards a target at a speed relative to the distance to the target (the further away the target, the faster the movement towards it). Ware and Fleet [4] modulate velocity for the next frame based on the depth information of the current frame. The Z buffer is sampled along a few equally-spaced horizontal lines, and the minimum depth (or a geometric average) is used to modify the forward and sideways motion. Nieuwenhuisen et al. [5] adapt the speed of the camera along the path to the radius of curvature (the smaller the radius, the lower the camera speed). They also set maximum acceleration and deceleration limits for the camera in order to prevent abrupt speed changes.

**Optical flow and self-motion illusion.** The optical flow can be defined as the projection of velocities of 3D surface points onto the imaging plane of a visual sensor. The optical flow plays a key role in how camera motion is perceived by the audience. Psychologists have studied the effect of optical flow patterns in locomotion control [6], self-motion perception [7], spatial updating [8], and motion sickness [9]. Visually-induced self-motion illusion is called *vection*. Vection occurs, for example, when someone in a stationary vehicle sees how an adjacent vehicle starts to move. A number of works demonstrate that optical flow can inducevection [7]. Visually-induced motion sickness is explained in terms of cue conflicts between visual and vestibular senses [10]. Virtual reality researchers have identified a number of potential factors associated with motion sickness, including individual factors, display factors and speed-related factors (optical flow, rate of linear or rotational acceleration, duration of visual stimuli) [11]. Unfortunately, previous works have focused on flight simulators and thus make two important assumptions: (a) the ground is assumed to be the major source of optical flow, and (b) the movement is rectilinear. Resulting optical flow rate measures (such as flight velocity w.r.t. altitude above ground level) are specific to flight simulation and do not apply to camera animations through complex indoor scenes where optical flow comes from all viewing directions, nor to paths involving a large number of turns. So et al. [12] propose a metric which considers all 6-DOF motion components (translational and rotational motion) and measures scene complexity by a more generic unit of spatial frequency (SF). However, their work still assumes that the scene complexity is roughly constant during the animation, and computes a single SF value for all the scene. Although this assumption is reasonable for simple outdoor scenes, it does not fit well with complex outdoor scenes with large variability of detail among different areas. Our metric also incorporates scene complexity and optical flow measures, but it has been designed to be evaluated every frame so that it enables to solve the reverse problem of adjusting camera speed so as to guarantee a roughly-constant flow rate.

**Image saliency.** An area in an image is called visually salient when it stands out relative to its surrounding neighborhood. Saliency maps have been used in computer graphics for accelerating global illumination by selective rendering, i.e. adapting render quality on an image region according to its saliency. A few works deal with the computation of saliency maps for both static [13] and dynamic scenes [14]. Longhurst et al. [15] present a GPU-based algorithm for real-time computation of saliency maps. Our image saliency metric is strongly based on [15], but we compute image saliency to disregard optical flow in those image areas with low image saliency, rather than for selective rendering. Note that Longhurst et al.’s metric for motion saliency takes the opposite approach: moving objects are assigned lower saliency values so that they can be rendered with lower quality.

### 3 Overview

Given a camera path, we strive to compute a suitable speed curve for it. Without loss of generality, we assume the camera path  $\mathcal{P}$  is parameterized by a suitable parameter  $u$  (e.g. arc length or local time units) so that  $\mathcal{P}(u)$  evaluates to the camera extrinsic parameters at parameter value  $u$ . Camera paths created with conventional 3D modelers can be easily converted into the above representation. The output of our algorithm is a sequence of parameter values  $\mathcal{U} = \{u_i\}_0^n$  and a sequence of speeds  $\mathcal{V} = \{v_i\}_0^{n-1}$  where  $v_i$  is the average speed at the segment defined by  $u_i, u_{i+1}$ . We claim that achieving a good trade-off between the following (potentially opposing) qualities tends to produce engaging, visually-pleasant animations (see Table II):

*Smoothness.* Classic guidelines on good camera motion practices include the avoidance of abrupt speed changes; the speed of the camera should be lowered when making a sharp turn to avoid objects to move too fast through the view.

*Non-fatiguing.* Considering the variety of symptoms that can occur due to motion sickness, including eye strain, headache, pallor, sweating, vertigo, disorientation, ataxia, and nausea [9][16], it seems reasonable to attempt to minimize the risk for the animation to produce motion sickness.

*Informative.* We want the animation to convey as much information as possible about the scene. Taking into account the limitations of human perception, this accounts for giving the user enough time to gather knowledge about the scene objects shown at each path segment.

*Interesting.* Since we only control a single DoF (time), maintaining the user attention during the whole animation means adapting the speed so that the camera moves faster when the view contains uninteresting or already-seen objects.

*Conciseness.* We want to generate concise animations whose length is adapted to the scene and path characteristics. Short animations save time and resulting movies require less storage space and transmission times.

**Table 1.** Summary of strategies adopted in our algorithm and how they contribute to each particular goal

	Smoothness	Non-fatiguing	Informative	Interesting	Conciseness
Limit amount of optical flow	x	x	x		
Avoid abrupt speed oscillations	x	x			
Adapt speed to number of salient features		x	x	x	
Use habituation measures to disregard already-known objects		x	x		

## 4 Quantifying Perceived Scene Motion

Given two parameter values  $u_{i-1}$ ,  $u_i$  and a time period  $\Delta t$ , we strive to quantify the amount of change a user will perceive when traversing the path segment  $(u_{i-1}, u_i)$  in  $\Delta t$  units of time. Our metric combines an optical flow map  $M_F(x, y, \Delta t)$  (x and y are pixel coordinates), an image saliency map  $M_I(x, y)$  and an habituation map  $M_H(x, y)$ , all of them computed from a render from  $\mathcal{P}(u_i)$  at pixel-level. We first describe how these individual maps are computed, and then we define the combined map  $M(x, y, \Delta t)$  and the final value  $F(u_{i-1}, u_i, \Delta t)$ .

**Optical flow map.** A camera moving at constant speed does not result in perception of smooth motion, particularly when the scene contains multiple levels of scale. Consider for example a camera moving towards a monument from out space. When approaching the Earth the speed must be obviously much faster than when approaching the monument. We could simply modulate speed according to the target distance [3] or to depth buffer contents [4], but these approaches do not take into account rotational components of camera motion and hence do not apply to paths involving camera turns. An alternative approach is to modulate speed according to the rotational component of the movement [5], but this would disregard scene complexity. Instead, we based our metric on the optical flow. We have already discussed the effects of the optical flow in self-motion perception [7] and motion sickness [16]. Furthermore, the optical flow elegantly integrates the translational and rotational components of camera motion as perceived according to scene structure.

We compute the optical flow map using the magnitude of velocities of 3D surface points projected onto the image plane. Velocities are computed per-fragment considering the motion experimented by the fragment when the camera moves from  $\mathcal{P}(u_{i-1})$  to  $\mathcal{P}(u_i)$ . We render the scene from  $\mathcal{P}(u_i)$  using a vertex shader to compute two transformed vertices, one using the modelview-projection matrices from  $\mathcal{P}(u_i)$  and the other using the matrices from  $\mathcal{P}(u_{i-1})$ . This allows the fragment shader to compute the screen-space distance ( $\Delta x, \Delta y$ ) each fragment

has moved. The retinal speed  $\omega$  of a fragment is computed by using a constant  $k$  representing the retinal size of a pixel (in degrees):

$$\omega = k \frac{\sqrt{\Delta x^2 + \Delta y^2}}{\Delta t} \quad (1)$$

Resulting values are normalized using the exponential decay proposed in [15]. Note that the higher the retinal speed of a fragment, the higher its  $M_F$  value.

**Image saliency map.** The optical flow map defined above does not take into account luminance and chrominance variations across the image. It makes more sense to disregard optical flow in those image regions with low or null contrast, and to give more importance to contrasted edges. This is supported by previous psychological studies [12]. Initially we considered to apply an edge detector to identify contrasted edges in the image, but this would disregard results on human visual perception and visual attention models [13]. Therefore our image saliency map combines an edge detector with a center-surround difference map, which together identify salient features in the image. Our image saliency map  $M_I$  is computed by averaging a center-surround difference map of each color channel, and an edge saliency map. The computation of these individual maps is based on the work by Longhurst, Debattista and Chalmers [15]. The center-surround saliency [13] is computed by subtracting images from consecutive levels of a Gaussian image pyramid which represent the image at a variety of resolutions. Edge saliency is computed using a Canny edge detector. Details on how these maps can be computed in real-time in the GPU can be found in [15].

**Habituation map.** Habituation refers to the fact that objects shown on the screen become familiar over time. Since we aim at maintaining user attention during the animation, it makes sense to incorporate a measure of object habituation so that the speed is modulated according to the degree of novelty of the on-screen objects. We found the habituation map for selective rendering described in [15] to be suitable for our purposes.

**Combined flow map.** We compute a combined flow map as follows:

$$M(x, y, \Delta t) = M_F(x, y, \Delta t)[(1 - \alpha)M_I(x, y) + \alpha M_H(x, y)] \quad (2)$$

Note that we modulate the optical flow map  $M_F$  by the factor between square brackets, which measures visual saliency, contrast and novelty at pixel level. The weight constant  $\alpha$  can be used to trade-off novelty vs. saliency, its effect being more apparent around unstable views where a large number of previously-unseen objects become visible. Experimentally we have found that with  $\alpha = 0.2$  the habituation effect can still be perceived without slowing too much camera motion around unstable views. Finally, we quantify the amount of change a user perceives when traversing a path segment  $(u_{i-1}, u_i)$  in  $\Delta t$  units of time as the average value of  $M(x, y, \Delta t)$  among pixels that pass the z-test pixels, i.e.  $F(u_{i-1}, u_i, \Delta t) = \text{avg } \{M(x, y, \Delta t)\}$ . Although our metric shares some basic

building blocks with selective rendering literature [15], the way we combine these blocks for speed modulation is completely opposing. This is particularly apparent in the use of motion saliency. Whereas selecting rendering literature uses flow speed to *decrease* the saliency and hence the rendering quality of fast-moving objects, flow speed *increases* the amount of change measured by our metric. The different maps required for our metric can be computed entirely in GLSL [17]. Using 512x512 maps, the generation time for the test scenes was about twice that of rendering a frame.

## 5 Speed Curve Computation

We now discuss how to compute a suitable speed curve for a predefined camera path  $\mathcal{P}(u)$ . For the sake of simplicity, we assume that the path is parameterized by arc length, so that the distance traversed by the camera from  $u_{i-1}$  to  $u_i$  is given by  $u_i - u_{i-1}$  (we assume  $C_0$  continuity). The algorithm (see Algorithm 1) solves a reverse flow problem: given a user-defined flow threshold  $\zeta$ , and a time step  $\Delta t$ , the algorithm iteratively computes the speed value  $v$  for which the amount of scene change quantified by our metric equals approximately  $\zeta$ . In other words, the algorithm finds  $v$  such that  $F(u, u + v \cdot \Delta t, \Delta t) \approx \zeta$ . At each iteration, the algorithm checks whether the current speed  $v$  fulfills the requirement above. If so, the pair  $(u + v \cdot \Delta t, v)$  is written to the output. Otherwise, the speed  $v$  is incremented (resp. decremented) depending on the sign of  $F - \zeta$ , using an extended binary search. We assume that  $g(v) = F(u, u + v \cdot \Delta t, \Delta t)$  is continuous and monotonically increasing for any fixed  $u$ . This assumption holds for all reasonable camera paths through static scenes. A pathological example where this assumption fails is the camera traversing a wall. In this situation there is obviously no way to ensure flow smoothness around the crossing point, regardless of the value of  $v$ . Our algorithm can be trivially extended to handle these rather unfortunate cases by leaving the speed unchanged for the pair  $(u_i, u_i + v \cdot \Delta t)$  if convergence fails after a fixed number of iterations.

The time step  $\Delta t$  can be chosen according to the intended frame rate. For a 50 fps animation, we can set  $\Delta t = 1/50$  s. The initial speed  $v_0$  is used just as a hint for the first speed value to check in the first iteration. The output of the algorithm is almost insensitive to this value, as the speed will be increased/decreased automatically depending on the evaluation of the metric. The only obvious requirement is that  $v_0 < l/\Delta t$ , where  $l$  is the path length. A reasonable initial guess for  $v_0$  is the expected average speed. The flow threshold  $\zeta$  allows to trade-off animation goals such as smoothness, non-fatiguing and concision. The slower the threshold, the slower the camera will move and thus the longer the animation. Experimentally we have found a threshold  $\zeta = 0.08$  to produce visually-pleasant animations. Despite the strong non-linearity of our metric, experimentally we have found that speed curves computed by our algorithm with decreasing threshold values are basically scaled down, low-pass filtered copies of each other. This allows the user to run the algorithm with an initial threshold and test different threshold values with no additional computational overhead.

---

**Algorithm 1.** Speed curve computation

---

```

Algorithm computeSpeedCurve
input:
    camera path  $\mathcal{P}$ , time step  $\Delta t$ , initial speed hint  $v_0$ , flow threshold  $\zeta$ 
output:
    sequence of  $(u, v)$  pairs
begin
     $u = 0; v = v_0; \varepsilon = 1e - 3$ 
    while  $u < \text{length}(\mathcal{P})$  do
         $u' = u + v \cdot \Delta t$ 
         $f = F(u, u', \Delta t)$ 
        if  $|f - \zeta| < \varepsilon$  then
            output( $u', v$ )
             $u = u'$ 
        else if  $f > \zeta$  then
            decreaseSpeed( $v$ )
        else
            increaseSpeed( $v$ )
        end if
    end while
    gaussianFilter(output)
end

```

---

In our tests, convergence was reached after three iterations, on average. Since the computational cost of evaluating our metric is roughly twice the cost of rendering a normal frame, the algorithm takes about  $6 \times$  the animation length. This preprocessing overhead is comparable to that of high-quality video encoders. Our algorithm modulates velocity at each segment disregarding velocity values computed for previous segments. This might result in abrupt speed changes, which are handled by applying a Gaussian smoothing filter to the speed values. We used a kernel of 2 seconds for the experiments.

## 6 Results

We tested our algorithm with two densely-occluded models: an office building and an oil tanker. We created a camera path for each model by defining the control points of a spline curve. The viewing direction was kept tangential to the curve, and the up vector was fixed to a vertical direction. We compared our speed curve computation algorithm with three alternative approaches: (1) speed modulation by scene depth as proposed in [4], (2) speed modulated solely by optical flow, and (3) speed modulated by image saliency.

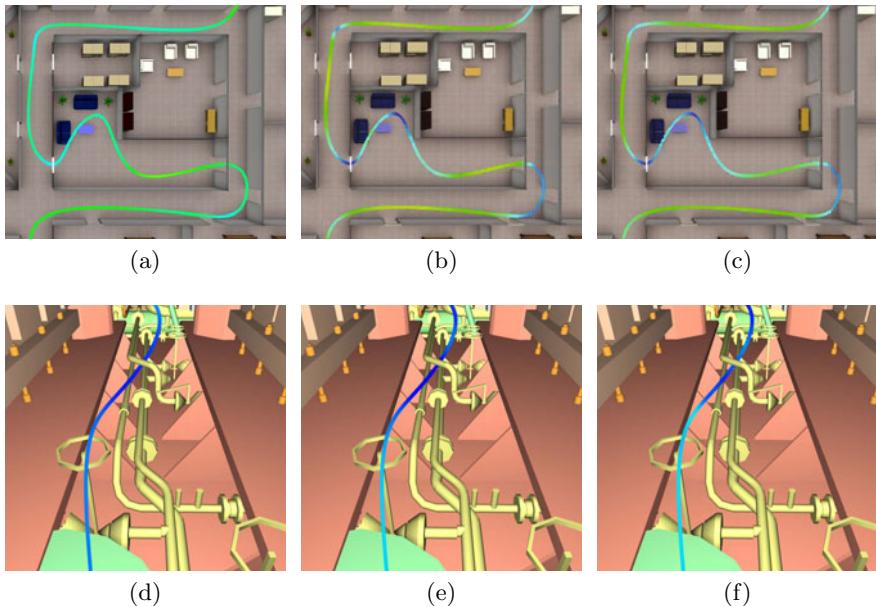
For option 1 we had to provide a maximum speed value which was modulated by the average scene depth; we set this value to the maximum speed of the path computed by our algorithm. For computing the speed curve in option 2 we

applied our algorithm but replacing our metric  $M$  by the average optical flow  $M_F$ . For option 3 we applied a similar approach as in option 1 but instead of modulating the speed by the average scene depth, we used the average image saliency  $M_I$ . In order to compare the different approaches, resulting speed curves were scaled so that the duration of all animations matched with that computed with our metric. This allows us to compare the speed at each point along the path, and see how the different approaches balance the speed at each point.

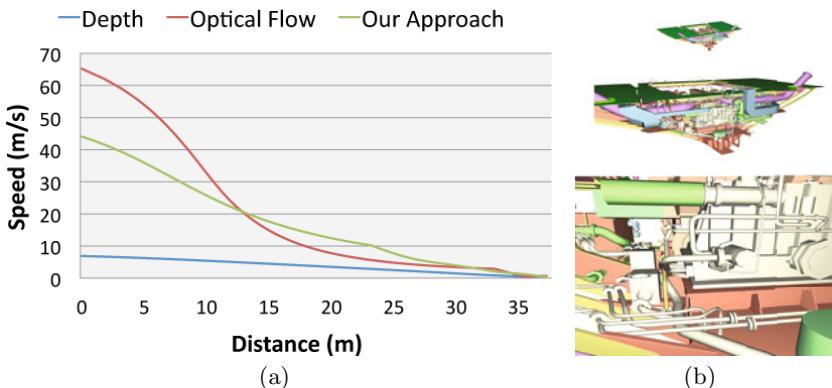
Option 3 produced speed graphs with almost constant speed, which were completely inappropriate for scenes involving multiple levels of scale (see the accompanying videos [17]). A major flaw of the depth-based approach (option 1) is that it disregards camera turns, leading to faster motion along turns, see Figure 1(a). Furthermore, the camera shows down too much when getting close to an object, regardless of its saliency, see Figure 1(d). Speed graphs obtained with our approach and option 2 were quite similar, see Figure 1(b) and (c). The main differences arose when the camera got too close to an object, Figure 1(e) and (f), option 2 computing slower speed values than our approach.

We also compared these techniques in terms of response to changes in the level of scale. In the oil tanker path, the camera started outside the model and moved towards it. Figure 2(a) shows how the speed is modulated as the viewpoint approaches the oil tanker. The depth-based approach required almost 20 s to get close to the model, while option 2 and our approach provided a much faster approximation. The complete video sequences for all the examples can be found in [17].

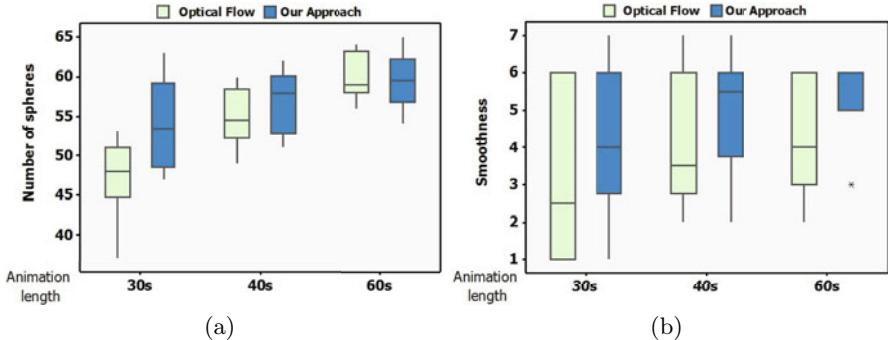
We conducted a preliminary user study to evaluate potential benefits of our approach in terms of informativeness, conciseness and smoothness. Given the previous results we only considered our approach and using only optical flow. Users were requested to count the number of targets (spheres) appearing during the animation through the ship model by pressing a button each time a sphere was spotted. We used a within-groups design, the independent variables being the method used to compute the speed graph (only optical flow or our combined approach) and the animation length (30 s, 40 s and 60 s). This resulted in two blocks, one per technique, of three trials each. The order of the blocks was randomized to counter-balance learning effects. The dependent variable was the number of spheres detected. The test was conducted on a desktop PC with a 22' LCD screen, providing a horizontal field-of-view of 45 degrees at viewer's distance. Our rendering engine ensured that the frame rate was fixed to 60 fps. Eight healthy subjects aged  $25 \pm 5$  participated in the experiment. Results are shown in Figure 3. Concerning the number of detected targets, the ANOVA showed both factors to have a significant effect ( $p < 0,01$  for both animation length and technique), our metric allowing users to count more spheres in less time. This seems to confirm our hypothesis that incorporating image saliency and habituation measures for modulating the speed allows users to gather more knowledge about the scene. Note for example that the 30 s animation computed with our combined metric allowed users to detect, on average, the same number of targets than with the 40 s animation using only optical flow, Figure 3(a),



**Fig. 1.** Results using different methods to compute the speed curve: depth-based (left), optical flow (center) and our combined approach (right). The color temperature along the path indicates speed (the coolest the color, the higher the speed). Top: the depth-based approach (a) disregards camera turns; notice than for (b) and (c) the speed decreases when turning. Bottom: when the viewpoint gets close to an object, the speed can decrease too much if saliency information is ignored.



**Fig. 2.** Speed curves for the oil tanker model when the camera is approaching from the outside (a). Deceleration times were 20 s, 6 s and 11 s resp. The frames at the start, middle and end of the path segment are shown in (b).



**Fig. 3.** Box plots for the number of detected spheres (a) and smoothness score(b)

demonstrating the benefits of our approach to produce concise animations. Users were requested also to fill a short questionnaire after each trial about how much speed changes did they notice during the task. The ANOVA found a significant effect for animation length  $p < 0.05$  (this was obviously expected) and also for technique  $p < 0.05$ , being our approach significantly better than using only optical flow, Figure 3(b). This result was also expected as the optical flow measure alone disregards edge contrast, which plays a key role in self-motion perception.

## 7 Conclusions and Future Work

The main contributions of this paper are (a) a perceptual metric for quantifying the amount of change between consecutive frames of an animation, which combines psychophysical results on optical flow, image saliency and habituation, and (b) an algorithm that, given a camera path, computes a speed curve so that speed is modulated to keep our flow metric approximately constant. Despite the large body of literature on the effects of optical flow on motion perception, the best of our knowledge this is the first work using optical flow, image saliency and habituation measures to automatically compute a suitable speed curve for a predefined camera path. We have shown that our algorithm produces smooth, non-fatiguing, informative and concise animations, the benefits of speed adaptiveness being particularly useful for large, densely-occluded scenes. As future work we plan to develop a real-time version of the algorithm based on low-resolution saliency maps so that speed can be modulated in runtime to correct user input during navigation.

**Acknowledgements.** This work has been partially funded by the Spanish Ministry of Science and Technology under grant TIN2007-67982-C02.

## References

1. Elmquist, N., Tsigas, P.: A taxonomy of 3D occlusion management techniques. In: Proceedings of the IEEE conference on virtual reality, pp. 51–58 (2007)
2. Christie, M., Olivier, P., Normand, J.: Camera Control in Computer Graphics. Computer Graphics Forum 27(8), 2197–2218 (2008)
3. Mackinlay, J.D., Card, S.K., Robertson, G.G.: Rapid controlled movement through a virtual 3d workspace. In: ACM SIGGRAPH 1990, pp. 171–176 (1990)
4. Ware, C., Fleet, D.: Context sensitive flying interface. In: Proceedings of the 1997 symposium on Interactive 3D graphics. ACM, New York (1997)
5. Nieuwenhuisen, D., Kamphuis, A., Overmars, M.: High quality navigation in computer games. Science of Computer Programming 67(1), 91–104 (2007)
6. Prokop, T., Schubert, M., Berger, W.: Visual influence on human locomotion. Experimental Brain Research 114(1), 63–70 (1997)
7. Dichgans, J., Brandt, T.: Visual-vestibular interaction and motion perception. Cerebral control of eye movements and motion perception, 327–338 (1972)
8. Riecke, B., Cunningham, D., Bulthoff, H.: Spatial updating in virtual reality: the sufficiency of visual information. Psychological Research 71(3), 298–313 (2007)
9. Kennedy, R., Lane, N., Berbaum, K., Lilienthal, M.: Simulator sickness questionnaire: An enhanced method for quantifying simulator sickness. The International Journal of Aviation Psychology 3(3), 203–220 (1993)
10. Reason, J.: Motion sickness adaptation: a neural mismatch model. Journal of the Royal Society of Medicine 71(11), 819 (1978)
11. Kolasinski, E.: Simulator Sickness in Virtual Environments. US Army Research Institute for the Behavioral and Social Sciences (1995)
12. So, R., Ho, A., Lo, W.: A metric to quantify virtual scene movement for the study of cybersickness: Definition, implementation, and verification. Presence: Teleoperators & Virtual Environments 10(2), 193–215 (2001)
13. Itti, L., Koch, C., Niebur, E., et al.: A model of saliency-based visual attention for rapid scene analysis. IEEE Transactions on pattern analysis and machine intelligence 20(11), 1254–1259 (1998)
14. Yee, H., Pattanaik, S., Greenberg, D.: Spatiotemporal sensitivity and visual attention for efficient rendering of dynamic environments. ACM Transactions on Graphics (TOG) 20(1), 39–65 (2001)
15. Longhurst, P., Debattista, K., Chalmers, A.: A GPU based saliency map for high-fidelity selective rendering. In: Proc. of the ACM Conference on Computer graphics, virtual reality, visualisation and interaction, Africa, p. 29 (2006)
16. LaViola Jr., J.J.: A discussion of cybersickness in virtual environments. SIGCHI Bulletin 32(1), 47–56 (2000)
17. <http://www.lsi.upc.edu/~virtual/SG2010>

# Example-Based Automatic Font Generation

Rapee Suveeranont and Takeo Igarashi

Department of Computer Science, The University of Tokyo, Japan

[iiieyes@gmail.com](mailto:iiieyes@gmail.com), [takeo@acm.org](mailto:takeo@acm.org)

**Abstract.** It is difficult and time-consuming to create and maintain a consistent style in all characters in manual font design process. Thus, we present a method for automatically generating a new font from a user-defined example. From each character outline, a skeleton is derived to be used as a topological structure. The system can then represent an arbitrary font using a weighted blend of outlines and skeletons from template fonts. The system takes the outline of a single character drawn by the user and computes blending weights from a database to reproduce the given outline. The weights are then applied to all characters to synthesize the new font. In this paper, the algorithm is described in detail and its quality evaluated with some examples.

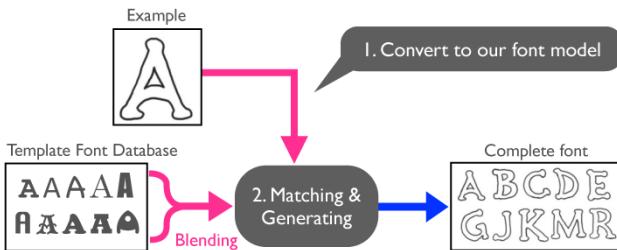
## 1 Introduction

The digital font design process begins with the sketching of all characters on paper. The designer then converts the characters into vector outlines and manually edits each character using a font-editing application. This is very time consuming and requires professional skill, artistic sense, hands-on experience, and expertise in typography, a combination of talents that few people other than professional font designers possess. Therefore, an automated process for generating fonts from a few examples is desirable.

Xu et al. [13] developed a system that automatically generates Chinese calligraphy. Their system is well suited to calligraphic languages thanks to its internal stroke-based font representation. Unfortunately, their method is not appropriate for characters from other language families, such as those that use a Latin alphabet.

Our research uses the work of Xu et al. as a basis but focuses on Latin alphabets and scalable outline fonts, which, compared to stroke-based fonts, lend themselves better to detailed design. Our goal is to create a system that facilitates the font design process by automating character creation with minimal user intervention. We take a single character outline as an input to generate a complete character set. The contributions of our research include a two-step font generation and a style preservation technique. Based on a skin and skeleton model, we introduce a morphable font model that is parameterized by blending weights. The style of the example character can be approximated by searching for the most similar blended style in a template database. Based on the premise that a few key characters hold most of the important features, our system enables the user to infer all other characters and generate them from the weights.

Because only one example is required, our system enables both amateurs and professional font designers to rapidly and effortlessly create new fonts. We demonstrate the effectiveness and capability of our method here.



**Fig. 1.** Overview of our automatic font generation system

## 2 Related Work

**Font Representation.** Alternative font representations to outline-based include feature-based and stroke-based. Shamir and Rappoport [1] proposed a feature-based approach that views fonts as high-level features such as serifs, bars, arcs and joints, rather than as low-level objects such as points and splines. This approach is strengthened by the component-based approach [2], which constructs fonts by using building block of parameterizable template shapes. Stroke-based representation of typographic character has been extensively studied because of its intuitiveness in characterizing calligraphic alphabets such as Chinese and Japanese. This approach provides complete topological information but is deficient in complex outline presentation. Many researchers have attempted to enhance the technique using concepts such as stylized stroke fonts [3] and skeletal strokes [4].

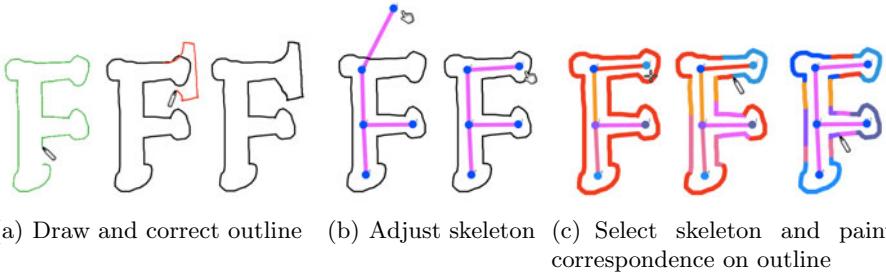
**Shape-Preserving Deformation.** One of our goals was to preserve the original style of the example character. Applications in 2D computer graphics usually involve object transformation or deformation. The original appearances of an object is usually depreciated by linear operations, which is undesirable. A number of studies has been undertaken to overcome this [4][5][6][9][10][11], many of which have tried to preserve a certain aspect of a 2D shape, such as area [5] or the original shape [6]. Sorkine et al. [10], used the Laplacian of a mesh to achieve rotation and scale-invariant surface editing. Nealen et al. [11] extended the idea and presented more general procedure using the Laplacian and positional constraints. Their approach motivated us to develop a Laplacian method for 2D outline that preserves the detailed features of a font (such as serifs, corners and junctions) and is suitable for text fonts.

**Example-based Generation.** Blanz and Vetter's [12] technique for creating a 3D human face from a 2D example facial picture is particularly noteworthy.

They presented the idea of a morphable face model that is searchable to match the input. Xu et al. [13] developed algorithms to automatically generate Chinese calligraphy. Their method works well for calligraphic scripts but cannot capture the complex designs of fonts used in printing and displaying.

### 3 Overview

We developed an easy-to-use graphical user interface that allows the user to draw an example outline (Figure 2). To edit the outline, the user can oversketch to replace existing outline. Our system automatically computes the skeleton and the correspondence between the skeleton and the outline. If the computed results are unsatisfactory, the user can modify them manually. When the user clicks a button, our system computes and generates a complete font.



**Fig. 2.** Our system’s simple user interface

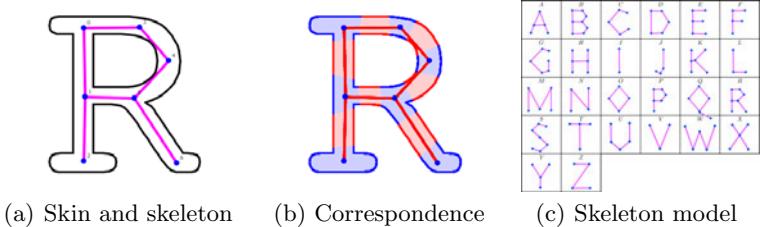
## 4 Morphable Font Model

### 4.1 Glyph Representation

The system uses a hybrid representation consisting of two components: a detailed profile skin and a structural skeleton. These two components permit the user to independently change the fine decorative style and the overall topological of the figure. For example, consider the K glyph with its original skin and skeleton. One can freely move, resize, or rotate the skeleton, and the skin is effectively deformed according to the skeleton but maintains its characteristic style. We call this *skin transfer*.

### 4.2 Correspondence of Skin and Skeleton

As a prerequisite for blending, the system requires a compatible skin correspondence for all template glyphs. An inappropriate correspondence produces badly blended outlines. Because there is no trivial universal skin correspondence-matching algorithm, a customized algorithm is required for each application. The



**Fig. 3.** Glyph components. The skin and skeleton are rendered as black outline, pink edges, and blue vertices (a). Their correspondence: main features (blue) and internal features (red). (b) Note that the main features (rather than the internal features) are usually those that hold the decorative outline of the font. (c) Skeleton model.

system avoids conflicting topology matching between various font styles of the same alphabet by using one standard skeleton model.

In our initial observations of numerous fonts, we noticed that stylized portions of the skin usually enclose some skeleton vertices, which may be terminals, junctions or corner of the topology. Accordingly, outlines around skeleton vertices can be thought as more important (or *main features*), whereas outlines around skeleton edges, which are mostly straight lines and curves, can be thought of as less important (or *internal features*), as shown in Figure 3(b). Hence, the system matches the skin to either the vertices or the edges of the skeleton in accordance with its topological structure. Together with the skeleton model, we have a constructed compatible correspondence for all characters that conform to the model.

### 4.3 Morphable Model

We present a *morphable font model* that defines arbitrary fonts in terms of two types of blending weights: skeleton weights and skin weights. Let  $F = \{f_1, f_2, \dots, f_n\}$  be a set of template fonts  $f_i$  from a database. We define an arbitrary blending style  $S$  and a skeleton  $T$  using the following linear blending functions:

$$S = \text{BlendStyle}_F(\Omega_S) = \sum_i^F \omega_{Si} \cdot \text{Skin}_{f_i} \quad (1)$$

$$T = \text{BlendSkeleton}_F(\Omega_T) = \sum_i^F \omega_{Ti} \cdot \text{Skeleton}_{f_i} \quad (2)$$

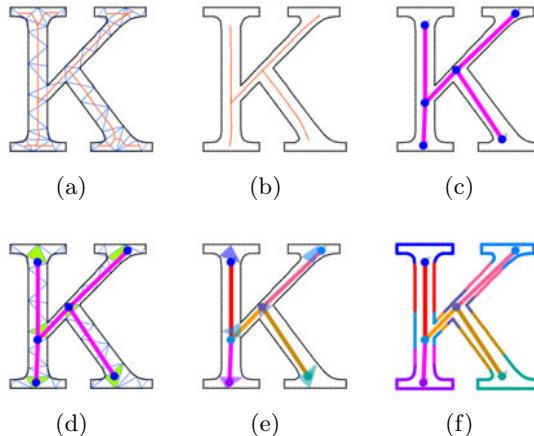
$\Omega_S = \{\omega_{S1}, \omega_{S2}, \dots, \omega_{Sn}\}$ , is a set of blending weights for skin, and  $\Omega_T = \{\omega_{T1}, \omega_{T2}, \dots, \omega_{Tn}\}$  is for skeleton. For skin, blending is performed by first sampling corresponding skin outlines with the same number of vertices and then by linearly blending the vertex positions. For skeleton, this operation is performed only on skeleton vertex positions.

## 5 Algorithm

Font generation is accomplished in two steps: conversion of the example character to a morphable model and estimation of its blending weights.

### 5.1 Conversion to Morphable Model

The system first converts example character outline to our model, using a semi-automatic method. When automatic conversion yields unsatisfactory results, the user can manually modify them.



**Fig. 4.** Skeleton extraction and correspondences estimation. (a) Chordal axis. (b) Remove small branches. (c) Embed skeleton. (d) Nearest internal triangles. (e) Colorize skeleton features. (f) Fill correspondence.

**Skeleton Extraction.** We first extract a skeleton from a given outline based on a predefined skeleton model. The goal is to adjust the skeleton model so that it fits inside the outline and approximates the overall figure as accurately as possible. We compute the chordal axis transform of triangulation of the outline (Figure 4(a)), as introduced in [78], and then remove small branches and make it smooth (Figure 4(b)). We first exhaustively search for terminal and junction nodes first. Nodes of degree two can then be computed by following along the shortest path.

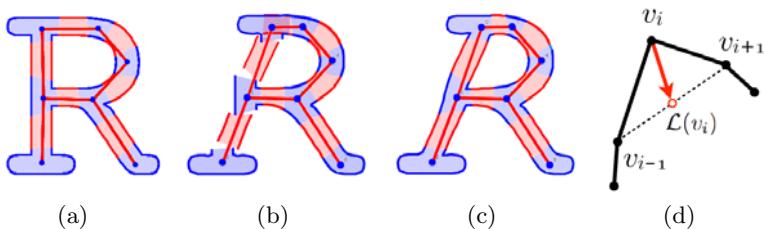
**Skin–Skeleton Correspondence.** Next, we establish a correspondence between the skin and the skeleton based on the triangulation of the previous step. The goal is to assign a certain portion of the skin outline to either a vertex or edge of the skeleton. Our motivation was the use of influence on attracted mesh vertices in the skeleton extraction technique proposed by Au et al. [14]. In our case, the mesh has the desirable property that its faces expand to full stroke

breadth. Consequently, each triangle or edge is surrounded by skin points on every side so that they effectively group portions of the outline. For each skeleton vertex, we find the nearest internal triangle within a fixed range (Figure 4(d)). If there is no such triangle, an edge is used instead. We begin by establishing the corresponding skin at all points on such triangles or edges (Figure 4(e)). Finally, we flood-fill the remaining portion of the skin sequentially between each pair of established points (Figure 4(f)).

**Feature-Preserving Skin Transfer.** We blend the skin and skeleton separately (Section 5.2) and then combine the two to generate the final outline of a character. Applying simple linear transformation locally to the skeleton edges yields undesirable scaled and rotated effects, whereas fixing only the main features produces a broken outline (Figure 5(b)). We use a feature-preserving stitching technique to handle these problems [10]. We stitch the main and internal features and recover to the original style by performing a Laplacian-based editing operation on the skin (Figure 5(c)). The Laplacian coordinates are relative coordinates defined by the difference between a vertex and the average of its neighbors. Let  $V = \{v_i\}$  be the set of original vertices on the skin. We can define the Laplacian coordinates of the vertex  $v_i$  using uniform weights (Figure 5(d)); specifically,  $\mathcal{L}(v_i) = \frac{1}{2}(v_{i+1} + v_{i-1}) - v_i$ . The feature-preserving stitching is then expressed by:

$$\begin{bmatrix} W_L L \\ W_P \end{bmatrix} V' = \begin{bmatrix} \mathcal{L} \\ W_P V \end{bmatrix} \quad (3)$$

Solving Equation 3 in a least-square sense yields a new skin that retains the original style. There are two types of constraints here:  $W_L L V' = \mathcal{L}$  is the Laplacian constraint, and  $W_P V' = W_P V$  is the positional constraint.  $W_L$  and  $W_P$  are diagonal weighting matrices. We set  $W_{Pii} = 1$ , whereas  $W_{Lii}$  ranges from 1 to 10 depending on whether the vertex  $v_i$  is in the middle of a main feature ( $W_{Lii}=1$ ) or inside an internal feature ( $W_{Lii}=10$ ). These weights keep the main features fixed and the internal features smoothly connected.



**Fig. 5.** (a) Effects of different transformations on the original glyph. (b) Simply fixing main features generates a broken outline. (c) Feature preservation using our method. (d) Laplacian coordinates.

## 5.2 Estimation of Blending Weights

**Database.** We prepare a database by converting all template fonts to our model using the method described in Section 5.1. If there are poor skeletons or correspondences, we fix them manually. When interpolating a font in the database, one can save memory by loading only the required characters from each font one at a time.

**Common Coordinate.** Before we can match a target style, we need to ensure that the matching is performed on the same font size. We use the cap height of the font (the distance from the base line to the top of a capital letter) as a common coordinate because it is consistent in every glyph, unlike width and line height.

**Style Matching.** In this step, the skin–skeleton representation of the example outline is taken as an input and the blending weights  $\Omega_T$  for the structure are found, followed by the blending weights  $\Omega_S$  for the style. Because it is impossible to find a globally optimal choice of weights (because of the exponentially large search space involved), we try to find an approximately optimal solution. We use the gradient descent method to find the solution that yields the minimum value of the penalty functions  $I(\Omega_T)$  for the skeleton, and  $J(\Omega_S)$  for the style, where  $J(\Omega_S) = \alpha \cdot D(\Omega_S) + \beta \cdot A(\Omega_S)$ ,  $\alpha, \beta$  are constants,  $D(\Omega_S)$  is the total Euclidean distance, and  $A(\Omega_S)$  is the total angular difference between corresponding points on the skin of the blended fonts and target font. The algorithm is simple: We start with a set of random weight parameters  $\Omega_{S_1} \dots \Omega_{S_M}$ , based on the assumption that only some fonts contribute significantly to the optimal solution. Accordingly, we first determine the  $M$  template fonts that produce the smallest penalties. Each startup parameter occupies different portions of the blending space. We add these parameters to the queue. Picking the weights with smallest penalty  $\Omega'$  from the queue, we expand it to  $N * 2$  new candidates, where  $N$  is the number of template fonts. The new weights are obtained by increasing or decreasing contributing weight from each template font by a small value  $\rho$ :

$$\Omega_i^* = \{\omega'_1, \dots, \omega'_i \pm \rho, \dots, \omega'_N\} \quad (4)$$

$$Queue \leftarrow \arg \max_{\Omega_i^*} (J(\Omega_i^*) - J(\Omega')) \quad (5)$$

We add only weights for which the penalty value decreases by more than some threshold—that is, weights that seem to follow the fastest route to the solution. The algorithm repeats these steps until time runs out or the queue becomes empty. When the algorithm terminates, the resulting local optimal blending weights represent the style of the example. For the skeleton,  $I(\Omega_T)$  is defined similarly.

## 6 Results

Our system was implemented in JAVA and all experiments were conducted on a computer with a 2GHz CPU and 1.5 GB of RAM. We set up the template font

database beforehand, which consisted of 32 template fonts selected to represent various styles, such as serif, san serif, comic, and fancy, to examine the versatility of the system.

### 6.1 Automatic Font Generation

We tested our system by providing sample characters and observing the outputs. All of the example designs were created with our system. The fonts generated are shown in Figure 6.

**Table 1.** Time used for automatic font generation for each experiment

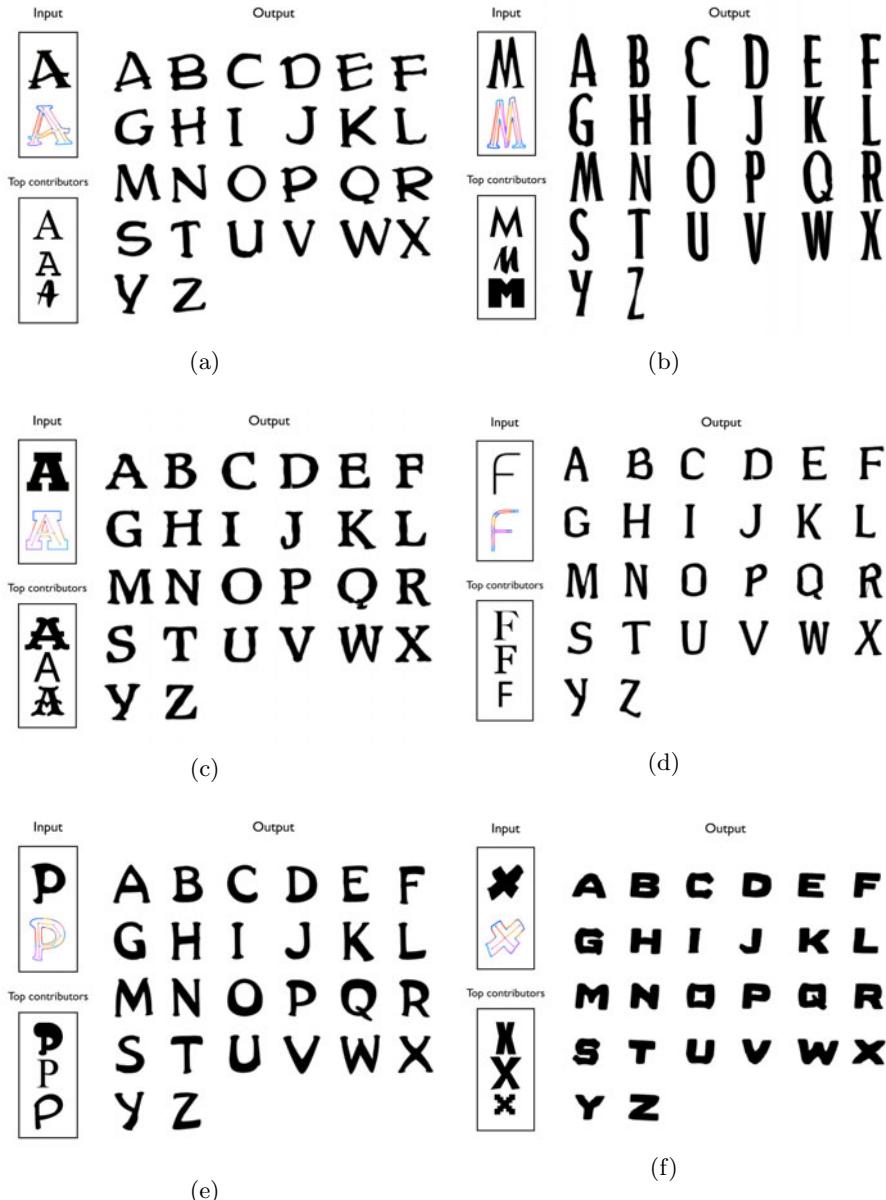
Experiment character	Results		
	Matching time (s)		Generating time (s)
	Skeleton	Style	
A	0.98	5.63	63.48
M	1.42	9.52	69.93
F	1.03	6.20	64.27
K	1.20	6.98	69.09

The generated glyphs in Figure 6(a) and 6(b) consistently appear to have the same style. The design of every characters represents the original glyph well. The overall dimensions—width, height, and slant—are consistent through out the fonts. This suggests that the skeleton estimation is reliable in most scenarios. Note also the consistent thickness of each font.

However, there is an obvious problem with linear blending in creating a novel design. The failed case in Figure 6(c) suggests that even when the top contributing fonts are promising (the blending of the first and second should be close to the example), the results can be far from perfect. If the template database is small, it cannot provide a precise estimate of the example style, and thus the system was unable to approximate the curvy corner of F in Figure 6(d). Paradoxically, this remains a problem even as the database gets larger. Specifically, contributions from most of the templates are rendered less significant because their results are averaged out by too many styles. One solution is to use only a few significant styles. Another might be to find different blending techniques that accumulate the contributions without deteriorative effects.

### 6.2 User Test

We conducted three informal experiments, each involving a single user. All of the users were interested in font design but had little or no practical experience. After a brief introduction, we asked each user to design three characters with different styles and to generate fonts from them. There was no time limit. After the test, the users were asked to evaluate our system on a scale of 1 to 5. We present the results in Figure 6(e), Figure 6(f), and Table 2.



**Fig. 6.** For each experiment, the example outlines and correspondences (top) and the generated fonts (right) are shown. The three template fonts that contribute the most to blending weight of skin are listed in the bottom left box. (a)(b) Our generated fonts. (c)(d) Failed cases. (e)(f) Fonts generated by users.

The average time required to finish a drawing was less than 3 min, and even the longest experiment took less than 5 min. This suggests that our system is very fast compared to traditional process. One user commented that the system is very simple and straightforward because it requires only one outline. We also observe that after the users generated one font, they attempted to edit other characters without being given any instructions in this regard. This suggests that in an interactive font generation system, the user should be able to repeatedly revise his or her design for as many characters as he or she wants.

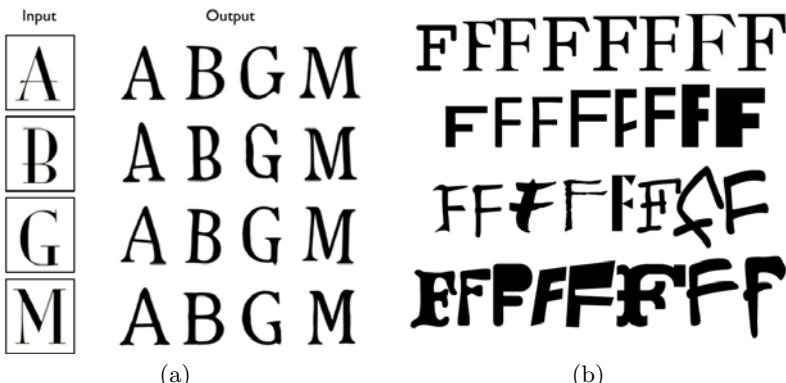
**Table 2.** Results of the user tests

User	Time to finish drawing (s)				Evaluation
	1	2	3	Average	
A	210	121	105	145.3	3
B	285	240	213	246.0	3
C	130	78	81	96.3	3
	Total average			162.5	3

### 6.3 Consistency

We conducted a test to check the consistency of our method. To do this, we prepared a professional font that was not included in our template database. We then input each character into our system one at a time, and recorded the glyphs that were generated. If the generated glyphs from each input character had the same outlines, we could imply that our system produces consistent and reliable results.

The samples used in the tests are shown in Figure 7(a). In general, all test cases produced comparable styles. The structural aspects were also consistent despite small differences in the angles in each output.



**Fig. 7.** (a) Experiment on the consistency of multiple examples of the same style. Given the input examples (leftmost column), the output characters are generated (right columns). (b) Template fonts in our database.

## 7 Limitations and Future Work

Our current system is suitable for creating body text fonts, but not for fancy or stylized ones. Topological differences present one of the most difficult problems. The model should be more flexible and should permit correspondence matching even when such topological differences are present. In certain cases, such as small *g* and *g* (which have different topologies), it is debatable whether there should even be a correspondence.

Our example-based method lacks the ability to create a completely novel design. To generate such fonts with higher reliability, one must use a better blending technique that does not average out main-feature skin. Even though our system is capable of generating many styles, it cannot faithfully produce every style a user can conceive.

In addition to single example, we would like to extend the applicability of our system for multiple example inputs. We intend to redesign our morphable model and matching algorithm to reflect this concept. Such an approach would lead to a more accurate solution for each iteration. Our current implementation of the target style optimization step can be further improved in terms of speed and memory footprint. Using a better data structure or a faster algorithm would improve the overall performance. In addition, a practical optimization technique that moves toward a global optimum (such as stimulated annealing) should be tested. Finally, we wish to extend the database to include a greater variety of styles so that our system can generate fonts more accurately.

## References

- Shamir, A., Rappoport, A.: Feature-based design of fonts using constraints. In: Hersch, R.D., André, J., Brown, H. (eds.) RIDT 1998 and EPub 1998. LNCS, vol. 1375, pp. 93–108. Springer, Heidelberg (1998)
- Hu, C., Hersch, R.D.: Parameterizable fonts based on shape components. *IEEE Comput. Graph. Appl.* 21(3), 70–85 (2001)
- Jakubiak, E.J., Perry, R.N., Frisken, S.F.: An improved representation for stroke-based fonts. In: SIGGRAPH 2006: ACM SIGGRAPH 2006 Sketches, p. 137. ACM, New York (2006)
- Hsu, S.C., Lee, I.H.H., Wiseman, N.E.: Skeletal strokes. In: UIST 1993: Proceedings of the 6th annual ACM symposium on User interface software and technology, pp. 197–206. ACM, New York (1993)
- Alexa, M., Cohen-Or, D., Levin, D.: As-rigid-as-possible shape interpolation. In: SIGGRAPH 2000: Proceedings of the 27th annual conference on Computer graphics and interactive techniques, pp. 157–164. ACM Press/Addison-Wesley Publishing Co., New York (2000)
- Igarashi, T., Moscovich, T., Hughes, J.F.: As-rigid-as-possible shape manipulation. In: SIGGRAPH 2005: ACM SIGGRAPH 2005 Papers, pp. 1134–1141. ACM, New York (2005)
- Igarashi, T., Matsuoka, S., Tanaka, H.: Teddy: a sketching interface for 3d freeform design. In: SIGGRAPH 1999: Proceedings of the 26th annual conference on Computer graphics and interactive techniques, pp. 409–416. ACM Press/Addison-Wesley Publishing Co., New York (1999)

8. Levet, F., Granier, X.: Improved skeleton extraction and surface generation for sketch-based modeling. In: GI 2007: Proceedings of Graphics Interface 2007, pp. 27–33. ACM, New York (2007)
9. Shamir, A., Rappoport, A.: Compacting oriental fonts by optimizing parametric elements. *The Visual Computer* 15(6), 302–318 (1999)
10. Sorkine, O., Cohen-Or, D., Lipman, Y., Alexa, M., Rössl, C., Seidel, H.-P.: Laplacian surface editing. In: Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing, pp. 175–184. ACM, New York (2004)
11. Nealen, A., Igarashi, T., Sorkine, O., Alexa, M.: Laplacian mesh optimization. In: GRAPHITE 2006: Proceedings of the 4th international conference on Computer graphics and interactive techniques in Australasia and Southeast Asia, pp. 381–389. ACM, New York (2006)
12. Blanz, V., Vetter, T.: A morphable model for the synthesis of 3d faces. In: SIGGRAPH 1999: Proceedings of the 26th annual conference on Computer graphics and interactive techniques, pp. 187–194. ACM Press/Addison-Wesley Publishing Co., New York (1999)
13. Xu, S., Lau, F.C.M., Cheung, K.-W., Pan, Y.: Automatic generation of artistic Chinese calligraphy. In: IAAI 2004: Proceedings of the 16th conference on Innovative applications of artificial intelligence, pp. 937–942. AAAI Press/The MIT Press (2004)
14. Au, O.K.-C., Tai, C.-L., Chu, H.-K., Cohen-Or, D., Lee, T.-Y.: Skeleton extraction by mesh contraction. In: SIGGRAPH 2008: ACM SIGGRAPH 2008 papers, pp. 1–10. ACM, New York (2008)

# Sketch-Based Interfaces: Exploiting Spatio-temporal Context for Automatic Stroke Grouping

Lutz Dickmann<sup>1</sup>, Tobias Lensing<sup>1</sup>,  
Robert Porzel<sup>1</sup>, Rainer Malaka<sup>1</sup>, and Christoph Lischka<sup>2</sup>

<sup>1</sup> University of Bremen,  
Bibliothekstr. 1, D-28359 Bremen, Germany  
`{dickmann,tlensing,porzel,malaka}@tzi.de`  
<sup>2</sup> s.l.  
`c.lischka@khora.de`

**Abstract.** In this paper, we investigate how discourse context in the form of short-term memory can be exploited to automatically group consecutive strokes in digital freehand sketching. With this machine learning approach, no database of explicit object representations is used for template matching on a complete scene—instead, grouping decisions are based on limited spatio-temporal context. We employ two different classifier formalisms for this time series analysis task, namely Echo State Networks (ESNs) and Support Vector Machines (SVMs). ESNs present internal-state classifiers with inherent memory capabilities. For the conventional static SVM, short-term memory is supplied externally via fixed-length feature vector expansion. We compare the respective setup heuristics and conduct experiments with two exemplary problems. Promising results are achieved with both formalisms. Yet, our experiments indicate that using ESNs for variable-length memory tasks alleviates the risk of overfitting due to non-expressive features or improperly determined temporal embedding dimensions.

**Keywords:** Sketch-based Interfaces, Stroke Grouping, Contextual Computing, Reservoir Computing.

## 1 Introduction

In freehand sketch drawing, sketched objects often consist of multiple strokes. Manipulating these objects within a sketch-based interface—e.g., during agile collaborative meetings or presentations—requires that all their constituent strokes be selected and grouped. As of today, this step is usually performed manually with lasso or box selection tools. This can be time consuming, especially when strokes are distributed sparsely over a large canvas or placed densely within a small area. In the sketch understanding domain, automatic stroke grouping is typically addressed as part of the overall goal to recognize sketched objects. A common consensus here is to perform stroke grouping by matching template objects from a prerequisite database. Such an approach requires explicit *a priori*

knowledge of all the objects that may occur. What is more, it can pose a combinatorial problem.

In this work we present a possible alternative in the form of an intelligent assistant that automatically groups strokes without matching explicitly represented objects. We show that when using predictions that pertain to consecutive strokes in a limited spatio-temporal context of a sketched scene’s creation history, no explicit object recognition is required to form meaningful stroke groups in an experimental setting. The view that the spatio-temporal patterns in a sketch creation process represent an unfolding discourse context in a certain domain context—using the terminology originally suggested by Gurevych et al. [4] for natural language processing (NLP) tasks—governs our approach. In the case at hand, we seek to capture discourse context via short-term memory (STM).

In the following section, we will shortly outline related work in the handwriting and sketch processing domain. After that, we state our fundamental ideas and continue with methodical and technical details. Then, experimental results are reported and consequently discussed.

## 2 Related Work

In the sketch understanding research domain, scene segmentation is often a necessary prerequisite of sketch interpretation. Typically, this is done by performing template matching algorithms. Sezgin & Davis, for instance, first perform primitive fitting to approximate freehand strokes as a series of distinct line and arc segments, then build Hidden Markov Models (HMMs) for all possible objects to be recognized in a scene, and align these on the temporal vector of a scene’s creation history [15]. This approach implies pre-computing log-likelihood scores for all possible object classes, all possible per-class stroke counts, and all possible positions on the observation vector, then at the analysis stage to solve the problem of determining the combination with the highest score.

In the field of mathematical handwriting recognition, Wan & Watt employ both temporal and spatial proximity thresholds in order to determine when an object segmentation decision is appropriate [17]. This is seconded by Xie et al. who use an object grouping parser based on bounding box proximity thresholds in the circuit diagramming domain [21]. Both techniques rely on fixed heuristics or parser rules and capture only a specific sketching domain. Nataneli & Faloutsos [12] and Zhou et al. [22] present SVM-based workflows for stroke classification and grouping and letter segmentation in handwritten Japanese text, respectively. Except for the spatial structural approach of Nataneli & Faloutsos [12], all of the aforementioned works employ the creation history of a sketched scene for data organization and analysis. Sezgin & Davis provide empirical evidence for per-user predictability of stroke orderings for recurring known objects [15]. The *part-by-part drawing principle* as postulated by Avola et al. supports the exploitability of the creation history for grouping decisions by stating that “[d]uring the hand-drawing process of one or more than one objects whose parts are clearly identifiable, the user generally ends one part or object before drawing another” [1].

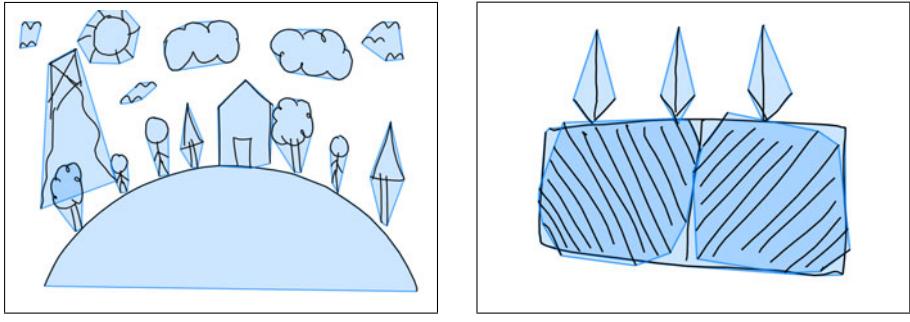
### 3 Concept, Methods, and Implementation

#### 3.1 Rationale and Overview

We present an intelligent stroke grouping assistant which uses only few spatio-temporal characteristics of strokes and does not employ feature selection algorithms. With reference to the part-by-part drawing principle [151], we assume that stroke interspersion does not occur between consecutively drawn objects. Grouping decisions are irrevocable and pertain to a current event, taking into account a series of consecutive past events, but neither is there an anticipation of future events, nor will new insights have an effect on past decisions. In order to analyze to what extent meaningful grouping decisions can be made based on STM, we utilize and compare two different state-of-the-art machine learning techniques that appear sufficient to cope with the given problem. The first one is a support vector machine (SVM) [2] that serves as our baseline and operates on explicitly represented contextual features which pertain to a fixed number of past strokes. The second is a relatively recent approach to recurrent neural networks, namely the echo state network (ESN) technique as introduced by Jaeger [7]. In contrast to the SVM approach, this classifier has an internal state, i.e., it does not only statically operate on the currently supplied input. Both techniques work with subsymbolic numerical input—as opposed to HMMs, for instance, which require prior vector quantization and codebook creation. We follow established recommendations for commonly used basic setups wherever applicable, cf. Hsu et al. [6] and Jaeger [7]. Both concurrently used assistant variants are supplied with input from two exemplary “toy problems”. We assess the suitability of both candidate techniques for the given problem and compare the results in terms of precision and recall measures.

#### 3.2 Sketching “Toy Problems”

**Naïve Landscape Scenes (NLS).** The first problem—naïve landscape sketches—is designed to reflect a rather general problem in freehand sketching. There are numerous perceived objects in each scene. There is left room for arbitrary drawing strategies in terms of stroke dynamics, per-object stroke ordering, object detail, and object creation order. NLS as we consider them contain outline style objects as follows, drawn in arbitrary order: 1 hill, 1 house with 1 door, 1 sun with rays, at least 2 trees, at least 1 cloud, 2 stick figures, at least 1 flock of birds, 1 kite with a cord that one of the stick figures holds. Figure 1 shows an example from the NLS test set. The overall number of strokes in the train set is 1061, with 813 group decisions (76.63%) and 248 segmentation decisions (23.37%). The number of strokes in the test set is 131, with 99 group decisions (75.57%) and 32 segmentation decisions (24.43%). Of the total 1192 decisions in the overall dataset, 912 are group decisions (76.51%) and 280 are segmentation decisions (23.49%), therefore the prior segmentation probability is  $\omega = 0.2349$ .



**Fig. 1.** Excerpt scenes from the employed toy problems. (a): Naïve landscape sketch scene. (b): Hatchings and Arrows Scene. Correct groups (ground truth) are highlighted with transparent blue convex hulls.

**Hatchings and Arrows (H+A).** The second problem—a hatching scene with additional arrows—is much more constrained and further specialized. It is designed to pose a specific problem to the classifier, which can only be disambiguated using discourse context in the form of short term memory. Here, strict constraints as to a spatio-temporal drawing order, object constitution, and direction are employed. In the particular case exhibited in cf. Fig. II (b), the local ambiguous measurement is implied by the similarity of the arrow heads and the transition between the two different kinds of hatchings. Both are composed of exactly the same strokes. However, the arrows’ strokes should be grouped while the last stroke of the first hatching and the first stroke of the second hatching should be segmented. In the following, we describe the creation of a H+A scene by using compass-style pointers to coarsely indicate stroke directions. The first shape drawn in the scene is a rectangle that is constructed with one single stroke, counter-clockwise, starting in the NW quadrant of the canvas: N–S; W–E; S–N; E–W. The side aspect ratio is approximately 2:1. Next, a division element is added (approximately in the center of the already present box) by drawing a straight downward stroke N–S. This virtually segments the box into two parts which subsequently are both filled with different hatching patterns. The left hatching fills the box segment SW–NE, the filling direction on the right is NW–SE. The constituting individual strokes are directed NW–SE for the left hatching pattern, SW–NE for the right one. Three arrows are finally drawn above the box with loosely equidistant spacing so they point down towards the box. Each arrow is constructed from three separate strokes, the first one in N–S direction. The second (NW–SE) and third (SW–NE) one form the arrow head. The overall number of strokes in the train set is 660, with 549 group decisions (83.18%) and 111 segmentation decisions (16.82%). The number of strokes in the test set is 77, with 65 group decisions (84.42%) and 12 segmentation decisions (15.58%). Of the total 737 decisions in the overall dataset, 621 are group decisions (83.31%) and 123 are segmentation decisions (16.69%), therefore the prior segmentation probability is  $\omega = 0.1669$ .

**Table 1.** Symbol look-up table w.r.t. the feature sets employed for the NLS task and for the H+A task. Note that for the event-coded cases, stroke-level mean  $\bar{\sigma}_{vel}$  and  $\sigma_{vel}$  are used instead of vertex-level horizontal and vertical velocity.

Symbol	$\Delta_P(pause)$	$\Delta_S(duration)$	$d_{AABB}(distance)$	$vel_x(hor.)$	$vel_y(vert.)$
¶ (NLS)	•	•	•	○	○
† (NLS)	•	•	•	•	•
£ (NLS, H+A)	○	•	○	•	•
\\$ (H+A)	○	•	•	•	•

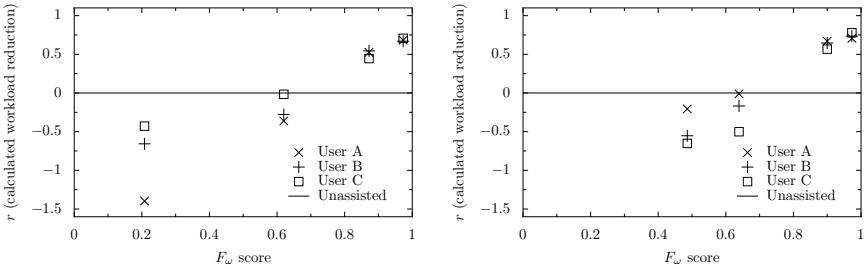
### 3.3 Unfolding the Scene History

**Feature Extraction.** Among the feature-oriented taxonomic approaches to contextual computing in the machine learning domain, we highlight the works of Katz et al., Turney, or Widdows [10,16,19,18]. With our focus on STM exploitation, we differentiate between discourse-contextual features that already encode (spatio-)temporal inter-stroke measurements and stroke-centric features that only contain local intra-stroke characteristics measured on the vertex level. We extract horizontal stroke velocity ( $vel_x$ ), vertical stroke velocity ( $vel_y$ ), proximity as the absolute axis aligned bounding box (AABB) distance between a stroke and its predecessor stroke ( $d_{AABB}$ ), absolute constrained time delay—i.e., pause—between a stroke and its predecessor stroke ( $\Delta_P$ ) with a timeout of 3000 milliseconds, and the absolute time duration of a stroke ( $\Delta_S$ ).

For event-timed scene encoding, inter-stroke features are calculated per stroke (e.g., distance between two bounding boxes or horizontal and vertical velocity mean and standard deviation). In contrast, for discrete-time approaches such inter-stroke features are calculated per vertex (e.g., distance between the current vertex of the current stroke to the bounding box of the predecessor stroke). We will use different constellations of the extracted features within the following problem-specific evaluations with labels as given in Table II.

**Data Set Construction.** The SVM is supplied with an event-timed (ET) representation of the scene history. In this representation, each stroke represents an event. That is, short-term memory is provided to the SVM externally by employing a sliding window method. As explained above, in the ET approach vertex level features are summarized to numerical values that represent these aspects only on the stroke level. Because of the transformation of vertex level dynamics to mean and standard deviation as stroke level features, potentially indicative dynamics on the vertex level are only rudimentarily accounted for.

In contrast to the ET case introduced for the SVM, we supply an ESN with a discrete time signal (DTS) sampled over the scene history so that it can capture actual vertex level dynamics. Feature vector expansion is not necessary here since the temporal dynamics of the scene history can be encoded and supplied in the temporal domain—as a signal representing a time series. We expect the discrete



**Fig. 2.** User test result plots for the experimental assessment of workload reduction based on two scenes from the NLS set (left/right).  $r$  is the workload reduction ratio, the horizontal straight line labeled ‘Unassisted’ expresses the underlying assumption that there is no workload reduction when no assistant is employed (with  $F_\omega$  unspecified)

time representation of the scene history to be most suitable for the ESN because of its nature as a classifier designed to deal with spatio-temporal dynamics.

To allow for direct commensurability of classifier setups with the SVM and ESN formalisms, we also provide the ET signal representation of the scene history to an ESN in another line of experiments. With this approach to composing an ESN input signal, each time step represents information about one stroke. The internal memory of the recurrent network can thus be seen as an alternative to the explicitly represented fixed window that is necessary in the SVM case.

### 3.4 Evaluation Metrics

**Precision & Recall Analysis.** Building on the common precision and recall analysis [14], we introduce the  $F_\omega$  score as a weighted mean of two class-specific macro-average  $F_1$  scores determined for grouping and for segmentation decisions. With  $\omega$  as the prior segmentation probability, we simply calculate  $F_\omega = \omega * F_{1\text{group}} + (1 - \omega) * F_{1\text{segment}}$ . In what follows,  $F_\omega$  denotes the biased mean with  $\omega$  pertaining to the respective prior probabilities of the used datasets.

**User Utility Value vs.  $F_\omega$ .** In order to coarsely estimate the actual utility value of the proposed assistant for different achieved  $F_\omega$  scores, we perform a small-scale experiment as a “sanity check.” Here, three test subjects have to manually repair two exemplary scenes from the NLS set, each at different levels of degradation. We consider as workload reduction the difference between the cost of manually grouping a flattened scene with standard tools as known from *Adobe Illustrator*, for instance, and the cost of establishing grouping ground truth from a scene with potentially erroneous groups as yielded by the grouping assistant. Our understanding of measuring cost here is to log the number of clicks and keystrokes and the task completion time. We normalize these quantities per scene with respect to the manual grouping results and calculate a savings ratio  $r_q$  per measured cost class  $q$  with this ground truth baseline.

The resulting  $r_q$  is negative if the repair process cost exceeds that of unassisted manual grouping. To obtain a unified value for the three measured quantities

*duration*, *clicks*, and *keystrokes*, the overall workload reduction ratio  $r$  is heuristically calculated with doubled weights for *duration*. This weighting schemes does not involve coefficients depending on the workstation or device topology, it can thus only suffice for an exemplary assessment in a constrained setting. The user test results as shown in Fig. 2 suggest that higher  $F_\omega$  scores indeed imply better assistance, and that the utility value variance between different users appears to decrease with increasing scores. However, we must also note that assistive technologies with performance levels below a certain threshold according to this metric ( $\approx 0.75$ ) can *add* substantial workload (rather than diminish it as intended). This will guide our interpretation of  $F_\omega$  scores.

### 3.5 Classifier Setups

**ESN Setup Procedure.** Echo state networks as proposed by Jaeger [7] denote an approach from the domain of reservoir computing (RC) where the hidden layer of a recurrent neural network (RNN) is treated as a dynamical reservoir (DR). Training is performed by linearly combining the desired output signal from a variety of nonlinear dynamical transformations of the input and/or output signals inside the DR. In contrast to conventional RNN training methods, the internal connection weights of the DR are not trained. The main advantages of this approach are computational cheapness and guaranteed convergence. In order for the approach to work, a DR used in an ESN should exhibit a special type of damped dynamics. A DR with such dynamics is said to have the echo state property, which basically ensures that the current network state is uniquely defined by its input and output. As a consequence of the echo state property, ESNs exhibit a form of short term memory—cf. Jaeger [9].

ESN setup parameters are estimated manually first following the experience and intuition of the experimenter as described by Jaeger [8]. Once a functioning standard configuration is found it is consequently reused and adapted only if needed. An assumption here is that the ESN should be applicable to deal with different toy problems whose solutions require different amounts of discourse contextual information. In contrast to the SVM, the ESN's performance should neither be significantly decreased by unnecessary additional contextual information nor should it be necessary to make exhaustive adaptations to the classifier's setup parameters for different toy problems from the same domain.

Our general ESN configuration for the scope of this work is as follows: a standard ESN is set up with multiple input signals  $u_i(n)$  and one output signal  $y(n)$ . Internal units use the *tanh* activation function while output units use the logistic sigmoid activation function. The number of internal units is set to  $N = 800$  for the discrete time signal case and to  $N = 80$  for the event-timed case. The DR spectral radius is set to  $\alpha = 0.999$  as to maximize short-term memory capacity [8]. Input weights are sampled from a uniform distribution within range  $[-3.0, 3.0]$ . The sampling rate is set to  $\Delta t = 0.1\text{sec}$  (10Hz). At this sampling rate input signals obtained from the discrete time signal construction approach on the NLS train set unfold to 9692 samples, of which 200 samples are initially discarded for a washout phase during training. The resulting network's

memory capacity is approximately  $MC \approx 27$  time steps in simulated time or  $\sim 2.7$  seconds in real time. Statistical data extracted from the NLS train and test set indicates that this enables the network to roughly capture two to four strokes on average, determined based on the arithmetic mean of stroke duration and stroke delay times. This estimation of memory capacity is based on similar approaches by Hertzberg et al. and by Jaeger [58].

**SVM Regularization and Kernel Parameter Estimation.** The underlying idea of the SVM approach is to find a maximum margin hyperplane that separates data points from two classes. The set of support vectors closest to this plane comprises all points with non-zero weights after the training procedure, which presents a convex optimization problem. With a soft margins SVM as proposed by Cortes & Vapnik [2], the constant  $C$  needs to be determined as a penalty parameter for regularization (it is also referred to as complexity parameter). In line with the recommendations of Hsu et al. [6] for basic SVM setups, we choose a radial basis function (RBF) kernel, for which another parameter ( $\gamma$ ) has to be estimated. Our heuristic to employ a cross-validation-based grid search procedure is directly adopted from Hsu et al. We first perform one coarse search with  $C = \{2^{-5}, 2^{-5+\Delta_C}, 2^{-5+2\Delta_C}, \dots, 2^{15}\}$  and  $\gamma = \{2^{-15}, 2^{-15+\Delta_\gamma}, 2^{-15+2\Delta_\gamma}, \dots, 2^3\}$ , where  $\Delta_{C\text{coarse}} = \Delta_{\gamma\text{coarse}} = 1.0$ . This result is then refined with a more fine-grained search (step size  $\Delta_{C\text{fine}} = \Delta_{\gamma\text{fine}} = 0.25$ ) within the respective neighborhood (within a  $\pm 2.0$  range for the exponents yielded by the coarse search).

We iteratively expand the feature vector with additional memory steps to which we will also refer as the embedding dimension with  $m = 0$  for no additional past events, cf. the terminology of Mukherjee et al. [11]. This procedure is followed in two variants concurrently: to keep all other parameters except the embedding dimension  $m$  constant, we run each of the experiments once with an “out-of-the-box” SVM setup that only uses default values  $C = 1.0$  for the complexity parameter and  $\gamma = 0.01$  for the RBF kernel parameter. Additionally, we conduct the rather time-consuming parameter estimation with the previously outlined grid search heuristic for each experiment—that is, for each expansion step of the feature vector (embedding dimension). Consequently, we refer to classifiers with estimated parameters (EP) and such with unestimated parameters (UEP) to distinguish these setups. All classifier setups use feature scaling with factors determined prior to training by normalizing the training set features to the range  $[0.0, 1.0]$ . The same scaling factors are then applied during testing. We employ the sequential minimal optimization SVM training implementation included in Weka [3,20].

## 4 Evaluation

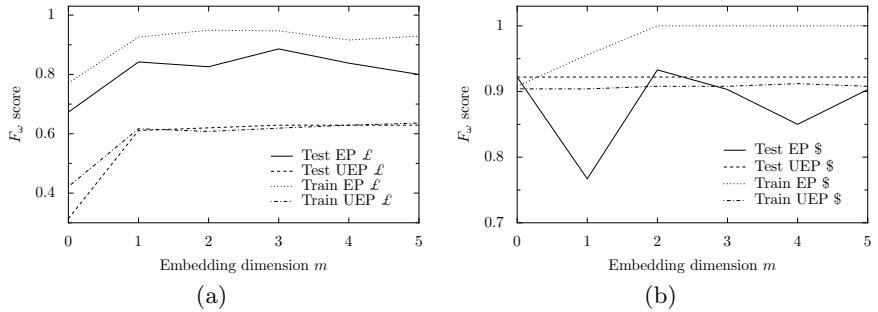
As is common practice in machine learning research, we use  $\sim 90\%$  of each set for training and the remainder for testing. Each set contains 20 sketched scenes annotated with ground truth, of which 18 are initially randomly chosen as train set, and the remaining two are used as a test set. These data sets are

**Table 2.** Comparison of classifier results. Best test results per feature constellation (Fts.) and task are highlighted.  $m$  denotes (stroke) events for the SVM cases, for the ESN setups  $m$  expresses MC in simulation steps.

Experiment series:		Naïve Landscape Scenes				Hatchings and Arrows					
Classifier	Fts.	$m$	Default setup $F_\omega$	Train	Est. parameters $F_\omega$	Train	Default setup $F_\omega$	Test	Est. parameters $F_\omega$	Train	Test
ESN DTS	¶	27	0.944	0.933	—	—	—	—	—	—	—
ESN ET	¶	14.3	0.973	0.960	—	—	—	—	—	—	—
SVM ET	¶	0	0.943	<b>0.973</b>	0.962	0.960	—	—	—	—	—
SVM ET	¶	1	0.957	<b>0.973</b>	0.985	0.960	—	—	—	—	—
SVM ET	¶	2	0.957	<b>0.973</b>	0.985	0.947	—	—	—	—	—
SVM ET	¶	3	0.957	<b>0.973</b>	0.988	0.947	—	—	—	—	—
ESN DTS	†	27	0.962	0.947	—	—	—	—	—	—	—
ESN ET	†	14.3	0.956	0.950	—	—	—	—	—	—	—
SVM ET	†	0	0.947	0.959	0.990	0.960	—	—	—	—	—
SVM ET	†	1	0.959	0.973	0.993	0.974	—	—	—	—	—
SVM ET	†	2	0.961	0.973	1.000	0.974	—	—	—	—	—
SVM ET	†	3	0.965	0.960	1.000	<b>0.987</b>	—	—	—	—	—
ESN DTS	£	27	0.875	0.774	—	—	0.980	0.856	—	—	—
ESN ET	£	14.3	0.773	0.865	—	—	0.988	<b>1.000</b>	—	—	—
SVM ET	£	0	0.422	0.314	0.771	0.673	0.679	0.657	0.826	0.743	—
SVM ET	£	1	0.617	0.611	0.926	0.842	0.819	0.828	0.948	0.790	—
SVM ET	£	2	0.608	0.620	0.949	0.826	0.907	0.922	1.000	0.903	—
SVM ET	£	3	0.619	0.629	0.947	<b>0.886</b>	0.884	0.922	1.000	0.965	—
SVM ET	£	4	0.629	0.629	0.916	0.838	0.883	0.922	1.000	0.965	—
SVM ET	£	5	0.636	0.629	0.929	0.800	0.889	0.922	1.000	0.933	—
ESN (DTS)	\$	27	—	—	—	—	0.992	0.962	—	—	—
ESN (ET)	\$	14.3	—	—	—	—	0.988	<b>1.000</b>	—	—	—
EP SVM (ET)	\$	0	—	—	—	—	0.904	0.922	0.909	0.922	—
EP SVM (ET)	\$	1	—	—	—	—	0.904	0.922	0.956	0.767	—
EP SVM (ET)	\$	2	—	—	—	—	0.909	0.922	1.000	0.933	—
EP SVM (ET)	\$	3	—	—	—	—	0.909	0.922	1.000	0.903	—
EP SVM (ET)	\$	4	—	—	—	—	0.912	0.922	1.000	0.850	—
EP SVM (ET)	\$	5	—	—	—	—	0.909	0.922	1.000	0.903	—

kept constant during all experiments to conserve the ability to conduct visual inspection of the results based on well-familiar scenes. We perform this basic sequence of experiments with each of the feature sets specified above—¶, †, and £ for the NLS example, £ and \$ for the H+A problem—in their respective encoding (event-timed or discrete-time sampled). In the SVM case, we additionally exercise iterative expansion of the embedding dimension, with  $m \in \{0, 1, 2, 3\}$  (NLS with ¶, †) and  $m \in \{0, 1, 2, 3, 4, 5\}$  (H+A, NLS with £).

The overall results obtained from all experiments conducted in this series are listed in Table 2. The ET ESN and the DTS ESN perform similarly with all feature sets considered except feature set £, for which the event-timed ESN achieves considerably better scores. In all cases, the event-timed ESN reaches  $F_\omega$  scores equal or better than those of the respective discrete time ESN setups. We therefore put emphasis on the comparison of the event-timed ESN with the event-coded SVM. For the memory-range tested with both EP and UEP SVMs, it can be observed that the ET ESN performance is always slightly below the optimum value reached by the best-performing SVM in the NLS case. This is the EP version (estimation per each classifier training) in two out of three cases. Recall that the parameters for the ESNs employed are manually adjusted only once, in contrast. For the ¶ feature set the ESN is initially *en par* with the



**Fig. 3.** SVM result plots for the NLS task with feature constellation  $\mathcal{L}$  (a) and for the H+A task using feature set  $\$$  (b)

EP SVM—however, while increasing memory can increase SVM classification performance, this may also lead to a decline as compared to smaller values of  $m$  due to overfitting. In the treatment of the H+A toy problem, only the ESN classifiers achieve an  $F_\omega$  score of 1.0 on the test set (with  $F_\omega = 0.988$  on the train set), while the EP SVM variants overfit on the train set with  $F_\omega = 1.0$  and cannot achieve test performance higher than  $F_\omega = 0.965$ . The extended feature set  $\$$  has no impact on the ET ESN and increases the performance of the DTS ESN slightly but does not aid to let the EP or UEP SVM variants achieve higher test scores than with feature set  $\mathcal{L}$  only. The result plots in Fig. 3 shed further light on the SVM sensitivity to different feature constellations and embedding dimensions.

In order to complement the results from our main evaluation based on fixed test sets, we conduct exemplary resampled paired  $t$ -tests—the applied procedure is detailed by Dietterich [3]. Such a “statistical sanity check” appears reasonable for situations where an ESN setup and an SVM setup both achieve identical  $F_\omega$  scores for the same task. Note that ESNs are initialized randomly, therefore the validity of the reported scores requires confirmation. We pick the NLS case with feature set  $\P$ , for which both the ET ESN and the EP SVM (with  $m = 0$ ) score  $F_\omega = 0.96$ . 30 trials are conducted with uniformly random partitions of the overall available NLS dataset (with a train set proportion of  $\sim 66\%$ ). In each trial, the instantiations of both classifier variants are trained and tested on the same data—the null hypothesis is that both algorithms will exhibit the same performance and error rate. The accuracy-based resampled paired  $t$ -test yields  $t = -0.18706$ , with the primary metric  $F_\omega$  as employed in this work  $t = 0.18868$ . In both cases,  $|t| < t_{29;0.975} = 2.04523$  and also  $|t| < t_{29;0.9} = 1.3114$ . Thus we cannot reject the null hypothesis: there is no significant difference in performance between the two considered classifier setups (with identical reported  $F_\omega$  scores). In a second approach, we conduct two further  $t$ -tests with 10 trials each to check whether in the H+A case the ET ESN is also significantly better in comparison to an EP SVM with  $m = 0$  and  $m = 1$  for feature set  $\$$  if we diverge from the fixed test set. Also, we use a more fine-grained parameter estimation grid search

for these re-runs, which renders slightly altered SVM  $F_\omega$  scores on the original fixed test set:  $F_{\omega_{Train}} = 0.909$  and  $F_{\omega_{Test}} = 0.922$  for  $m = 0$ ;  $F_{\omega_{Train}} = 0.959$  and  $F_{\omega_{Test}} = 0.789$  for  $m = 1$  (cf. Table 2). In all cases ( $m \in \{0, 1\}$ , accuracy-based and  $F_\omega$ -based) we find that  $|t| \approx 10 > t_{9;0.99} = 2.8214$ . Here, the null hypothesis can be rejected: it appears to hold that the ESN indeed performs significantly better in the regarded exemplary cases, no matter whether we spend more cycles on SVM parameter estimation and/or whether we vary the train and test sets for the H+A task.

## 5 Discussion

We find that a discourse-contextual paradigm for exploiting STM can help to successfully form meaningful stroke groups. Feature choice along with domain/data knowledge is a primary prerequisite to proper configuration with both used formalisms. This can be seen for the NLS case, where both classifiers do not perform notably better with more features. The temporal delay and spatial proximity features appear to be most expressive here. An important issue w.r.t. the exploitation of discourse context by an SVM is that different problems may require different memory lengths to reach the best solutions. Mukherjee et al. note w.r.t. potential SVM sensitivity to embedding dimension expansion that overfitting problems may occur with non-linear kernels [1]. With regard to tackling different sketching problems, this may pose a problem when using the SVM. The ESN apparently deals with such issues more robustly. Inferring from the comparison results, we can state that while the SVM can reach slightly better classification results when we experimentally determine and optimize it for a particular memory length, the ESN appears to be relatively robust to overfitting issues and provides a more general solution with its inherent STM at similar performance. We were able to determine one adequate network setup from the first ESN experiment conducted and then transfer that setup to other signal codings, feature sets, and toy problems with little or no modification of setup parameters. However, it is to be noted that the determination of a suitable ESN setup is rather complex and laborious. As stated by Jaeger [9], it requires profound experience and intuition to determine adequate network setups. Then, however, a solution that works for one problem is likely to work for other related problems as well. This makes the ESN a more suitable technique when considering a transfer to other sketch domains and applications.

## 6 Conclusion

Considering the rapidly developing field of surface computing and sketch-based interfaces, we have focused on the problem of automatically grouping consecutively drawn strokes. Posing that a database of explicit object representations for template matching on a complete scene is not desirable, we have investigated how grouping decisions can be made relying only on limited spatio-temporal context in the form of short-term memory. The formalisms of internal-state ESNs

(inherent memory capabilities) and conventional static RBF-SVMs (memory representation in Euclidian space via feature vectors) have been proposed for this time series analysis task and set up according to the respective requirements. An evaluation has been performed based on two different exemplary problems, “Naïve Landscape Scenes” and “Hatchings and Arrows.” We have shown that the contextual computing approach as exercised here yields promising results in terms of error rates and workload reduction for both formalisms in the considered experimental settings. Moreover, our overall results suggest that ESNs are better suited for variable-length memory tasks as their use alleviates the risk of overfitting that may otherwise occur due to non-expressive features or improperly determined temporal embedding dimensions.

**Acknowledgements.** The research presented in this paper was supported by the German Research Foundation (DFG) as part of the Collaborative Research Centre 637 “Autonomous Cooperating Logistic Processes.”

## References

1. Avola, D., Caschera, M.C., Ferri, F., Grifoni, P.: Ambiguities in sketch-based interfaces. In: Proceedings of the 40th Annual Hawaii International Conference on System Sciences (HICSS 2007), January 2007, p. 290b (2007)
2. Cortes, C., Vapnik, V.: Support vector networks. *Machine Learning*, 273–297 (1995)
3. Dietterich, T.G.: Approximate statistical tests for comparing supervised classification learning algorithms. *Neural Computation* 10, 1895–1923 (1998)
4. Gurevych, I., Porzel, R., Malaka, R.: Context: Integrating Domain- and Situation-specific Knowledge. In: Wahlster, W. (ed.) SmartKom - Foundations of Multimodal Dialogue Systems, pp. 269–284. Springer, Berlin (2006)
5. Hertzberg, J., Jaeger, H., Schönher, F.: Learning to ground fact symbols in behavior-based robots. In: ECAI, pp. 708–712 (2002)
6. Hsu, C.-W., Chang, C.-C., Lin, C.-J.: A practical guide to support vector classification. Technical report, Department of Computer Science and Information Engineering, National Taiwan University, Taipei 106, Taiwan (2003)
7. Jaeger, H.: The echo state approach to analysing and training recurrent neural networks. Technical Report 148, GMD - German National Research Institute for Computer Science (December 2001)
8. Jaeger, H.: Short term memory in echo state networks. Technical Report 152, GMD - German National Research Institute for Computer Science, May 28 (2002)
9. Jaeger, H.: A tutorial on training recurrent neural networks, covering BPPT, RTRL, EKF and the “echo state network” approach (2nd revision). Technical Report 159, Fraunhofer Institute for Autonomous Intelligent Systems, AIS (2005)
10. Katz, A.J., Gately, M.T., Collins, D.R.: Robust classifiers without robust features. *Neural Comput.* 2(4), 472–479 (1990)
11. Mukherjee, S., Osuna, E., Girosi, F.: Nonlinear prediction of chaotic time series using support vector machines. In: IEEE Workshop on Neural Networks for Signal Processing VII, pp. 511–519. IEEE Press, Los Alamitos (1997)
12. Nataneli, G., Faloutsos, P.: Robust classification of strokes with svm and grouping. In: Bebis, G., Boyle, R., Parvin, B., Koracin, D., Paragios, N., Tanveer, S.-M., Ju, T., Liu, Z., Coquillart, S., Cruz-Neira, C., Müller, T., Malzbender, T. (eds.) ISVC 2007, Part I. LNCS, vol. 4841, pp. 76–87. Springer, Heidelberg (2007)

13. Platt, J.: Sequential minimal optimization: A fast algorithm for training support vector machines. Technical report, Microsoft Research (1998)
14. Rijsbergen, C.J.v.: Information retrieval, 2nd edn. Butterworths, London (1979)
15. Sezign, T.M., Davis, R.: HMM-based efficient sketch recognition. In: Proc. 10th Intl. Conf. Intelligent User Interfaces (IUI 2005), pp. 281–283 (2005)
16. Turney, P.D.: Robust classification with context-sensitive features. In: IEA/AIE 1993: Proc. 6th Intl. Conf. Industrial and Engineering Applications of Artificial Intelligence and Expert Systems, pp. 268–276 (1993)
17. Wan, B., Watt, S.: An interactive mathematical handwriting recognizer. In: MathML Conference (2002)
18. Widdows, D.: A mathematical model of context. In: Blackburn, P., Ghidini, C., Turner, R.M., Giunchiglia, F. (eds.) CONTEXT 2003. LNCS (LNAI), vol. 2680, pp. 369–382. Springer, Heidelberg (2003)
19. Widdows, D., Dorow, B.: A graph model for unsupervised lexical acquisition. In: Proceedings of the 19th international conference on Computational linguistics, pp. 1–7. Association for Computational Linguistics, Morristown (2002)
20. Witten, I.H., Frank, E.: Data Mining: Practical machine learning tools and techniques, 2nd edn. Morgan Kaufmann, San Francisco (2005)
21. Xie, Q., Sun, Z.-X., Feng, G.-H.: Online diagramming recognition based on automatic stroke parsing and bayesian classifier. In: Proc. Machine Learning and Cybernetics 2006, August 2006, pp. 3190–3195 (2006)
22. Zhou, X.-D., Wang, D.-H., Liu, C.-L.: A robust approach to text line grouping in online handwritten japanese documents. Pattern Recogn. 42(9), 2077–2088 (2009)

# A Method for Reconstructing Sketched Polyhedral Shapes with Rounds and Fillets\*

Pedro Company, Peter Ashley, and Clifford Varley

Department of Mechanical Engineering and Construction, Universitat Jaume I, Spain  
`{pcompany, varley}@emc.uji.es`

**Abstract.** In this paper we present a method for detecting rounds and fillets in engineering sketches and drawings, and automatically generating a 3D model of the corresponding object, with rounds and fillets applied. This method is useful both as a component of computer-aided sketching tools and in determining design intent—although rounds and fillets are common in engineering parts, they often conceal design intent, which is more easily determined from the object's underlying polyhedral skeleton.

**Keywords:** Sketch-based modelling, design intent, features, rounds and fillets.

## 1 Introduction

Sketches are an important kind of graphics, as they assist product designers during the creative stages of design and help them to develop inventions. Computer-Aided Sketching (CAS) tools should provide their users with a sketching environment which allows them to make full use of their conceptual design and innovation talents, while also providing full integration with the subsequent phases of the design process [1], [2].

We seek to make our CAS tools "smarter". They should recognize *cues* (sometimes *clues* or *regularities*), common properties of sketches which reveal corresponding properties of the object. There are many such cues, and some of them have already been successfully studied [3]. Cues which convey design intent are particularly important. For example, Li et al [4], a recent contribution aimed at finding symmetry, uncovers important design elements embodied as high-level geometric relations.

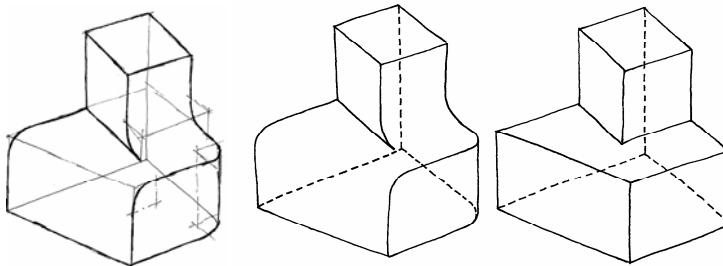
In this paper we describe a novel method for solving the problem of automatically reconstructing polyhedral shapes which include *rounds* and *fillets*. Unlike previous methods which produce free-form surfaces from sketches (see [5] for a historical survey), our goal here is to detect and detach analytical surfaces which are geometrically simple but constitute common CAD/CAM features. Apart from the obvious advantage of reducing the workload of the designer (avoiding the step of obtaining the mind's eye image of the polyhedral skeleton), the method isolates features which play

---

\* The Spanish Ministry of Science and Education and the European Union (Project DPI2007-66755-C02-01), and the Ramon y Cajal Scholarship Programme are acknowledged with gratitude.

specific roles in designed parts (such features can be efficiently managed as independent features by current geometrical engines).

Ideally, reconstruction systems should automatically reconstruct shapes such as the one depicted in Figure 1 left. However, overtracing and other similar problems are still difficult to cope with [6]. Recognizing this, we aim for the more modest goal of interpreting tidied hand-drawn line-drawings (Figure 1 middle)—this goal is still a step beyond the current strategy of drawing the skeletal shape (Figure 1 right), constructing the object, and then manually adding blends to the resulting 3D model.



**Fig. 1.** Sketch of a shape with rounds and fillets (left), its tidied line-drawing (middle), and its polyhedral wireframe skeleton (right)

## 2 Our Method

We shall initially develop our method for *wireframes* of *quasi-normalons* (wireframe sketches and line drawings show all edges of the portrayed object, visible or not, while natural sketches and line drawings show only the part of the portrayed object visible from the chosen viewpoint; quasi-normalons are those polyhedra where non-orthogonal edges may be removed without losing any vertex).

Our method will be divided into four main stages: 1) detect rounded edges and fillets (Section 2.1); 2) obtain the polyhedral skeleton (Section 2.2); 3) reconstruct the skeleton; 4) add rounded edges and fillets.

Stages 1 and 2 are specific to this work and it is these we describe in more detail. Stage 3, geometrical reconstruction, converts tidied line-drawings into 3D models [7]. This has already been partially solved in the context of sketch-based modelling—there is no general solution as yet, but we have developed an algorithm which can construct 3D models of quasi-normalons with as many as 50 faces [8]. Stage 4 is a common task in current CAD modelling environments, which encourage the users to create “skeletons” and then add rounds manually. This simplifies the creation of complex shapes. Current geometrical engines of CAD applications are quite efficient in managing rounds as separate features added on top of model trees. The theoretical problems of adding rounds and fillets while maintaining the design intent is solved, and current academic interest deals with more complex and subtle variations such as infinitely sharp and semi-sharp edges [9] and [10].

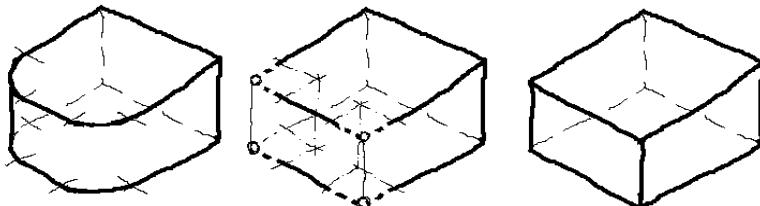
## 2.1 Detect Rounded Edges and Fillets

We shall detect rounded edges and fillets in wireframe sketches by: 1) detecting circular arcs (projected as elliptical arcs) may be easily done after segmenting the sketch strokes into simple lines, which is a solved problem for tidied hand-drawn line-drawings [11].

Detecting circular arcs (projected as elliptical arcs) may be easily done after segmenting the sketch strokes into simple lines, which is a solved problem for tidied hand-drawn line-drawings [11].

The task of forming pairs of arcs may be subdivided into the following actions:

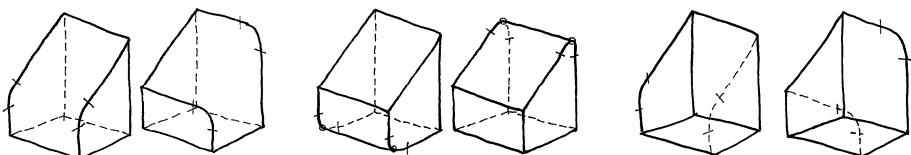
- Pair those arcs which are contained in parallel faces and share a tangent contour line (such as the pair depicted in the left part of Figure 2, which share the vertical tangent line). By this means, faces can be detected in the skeleton [7].
- For the remaining arcs, pair those arcs which are contained in parallel faces, are similar in size and orientation, and are connected to mutually parallel lines (such as the pair indicated by the additional thin lines in the middle part of Figure 2, which do not share any tangent line).



**Fig. 2.** Part with two rounds (left), arcs removed, edges extended and new vertices and edges added (middle), resulting polyhedral shape (right)

Rounds applied to single edges in quasi-normalon shapes fall in one of the three categories shown in Figure 3: a) both arcs are fully visible, and no one line connects them (since the edge has disappeared because of the rounding operation); b) one arc is fully visible, the other is partially occluded and one contour line is tangent to both, and c) one edge is fully visible, the other is fully occluded and no one line connects both.

Rounds in oblique edges of quasi-normalon shapes can be classified into the same three categories (Figure 3). It does not matter whether or not the rounded edge meets at  $90^\circ$  with the other edges connected to the same junction.



**Fig. 3.** Rounds of single edges in quasi-normalons: two arcs visible and no line (left), one and a half arc plus tangent line (centre), one arc (right)

Fillets in quasi-normalon shapes can only be classified into the second and third categories, since tangent contour lines may never appear (as fillets are concave shapes and may not belong to the contour of quasi-normalon shapes).

## 2.2 Obtain the Polyhedral Skeleton

We shall produce the polyhedral skeleton by repeating the following sequence for every pair of arcs: 1) suppress both arcs; 2) remove any contour line tangent to the suppressed pair of arcs; 3) extend the tangent lines to produce new pairs of vertices; 4) add new lines connecting the new pairs of vertices.

After suppressing the first arc, the lines connected to its ends must be extended until they intersect. The intersection point is the first new vertex, where the new edge must start. The same intersection procedure applied to the second arc produces the second new vertex, where the new edge must finish.

## References

1. Company, P., Contero, M., Varley, P.A.C., Aleixos, N., Naya, F.: Computer-Aided Sketching as a Tool to Promote Innovation in the New Product Development Process. *Computers in Industry* 60(8), 592–603 (2009)
2. Olsen, L., Samavati, F.F., Sousa, M.C., Jorge, J.: Sketch-based modeling: A survey. *Computers & Graphics* UK 31(1), 85–103 (2009)
3. Yuan, S., Tsui, L.Y., Jie, S.: Regularity selection for effective 3D object reconstruction from a single line drawing. *Pattern Recognition Letters* 29(10), 1486–1495 (2008)
4. Li, M., Langbein, F.C., Martin, R.R.: Detecting design intent in approximate CAD models using symmetry. *Computer-Aided Design* 42(3), 183–201 (2010)
5. Kara, L.B., Shimada, K.: Sketch-based 3D-shape creation for industrial styling design. *IEEE Computer Graphics and Applications* 27(1), 60–71 (2007)
6. Company, P., Varley, P.A.C.: Operating modes in actual versus virtual paper-and-pencil design scenarios. In: 2009 Intelligent User Interfaces (IUI) Workshop on Sketch Recognition, Sanibel Island, Florida, February 8 (2009)
7. Company, P., Piquer, A., Contero, M., Naya, F.: A Survey on Geometrical Reconstruction as a Core Technology to Sketch-Based Modeling. *Computers & Graphics* 29(6), 892–904 (2005)
8. Varley, P.A.C., Company, P.: A new algorithm for finding faces in wireframes. *Computer-Aided Design* 42(4), 279–309 (2010)
9. Hoppe, H., DeRose, T., Duchamp, T., Halstead, M., Jin, H., McDonald, J., Schweitzer, J., Stuetzle, W.: Piecewise smooth surface reconstruction. In: ACM Siggraph, pp. 295–302 (1994)
10. DeRose, T., Kass, M., Truong, T.: Subdivision surfaces in character animation. In: ACM Siggraph, pp. 85–94 (1998)
11. Pu, J., Gur, D.: Automated Freehand Sketch Segmentation Using Radial Basis Functions. *Computer-Aided Design* 41(12), 857–864 (2009)

# Pressure-Based 3D Curve Drawing

Chan-Yet Lai<sup>1,2</sup> and Nordin Zakaria<sup>2</sup>

<sup>1</sup> PETRONAS Research Sdn. Bhd.

<sup>2</sup> Frontier Computer Modeling Laboratory, Universiti Teknologi PETRONAS

chanyet\_lai, nordinzakaria@petronas.com.my

**Abstract.** 3D curve drawing is a fundamental operation in the creation and manipulation of surfaces and 3D models. Although techniques for specifying 3D curves using 2D input devices have been developed extensively, they are mostly indirect, with the need of multiple viewpoints or multiple lines to specify a 3D curve. We propose a simple technique to directly specify a 3D curve - using pen pressure to draw a single line of varying thickness, and mapping the thickness at each point along the curve to a depth distance. Hence, the thicker the 2D curve at a particular point, the closer the corresponding 3D point to the camera. This method is easy and intuitive, and yet surprisingly it has not been reported before. We illustrate the effectiveness of the method with numerous examples.

**Keywords:** Sketch-based user interface (UI), 3D curve drawing.

## 1 Introduction

Inference of 3D lines and curves from sketched 2D strokes is a fundamental problem in sketch-based 3D interfaces [1,4]. Sketch curve is imprecise by nature but it has been recognized by many authors as one of the important features in any modeling tools, to allow users to sketch the desired 3D curves directly and quickly without having to go through complicated curve specification procedures. Therefore a quick and easy method to generate 3D curve is highly on demand.

The technique we are proposing here uses pen pressure data to generate 3D curve. The user sketches a 2D curve with varying thickness according to the amount of pressure applied during sketching. The more pressure applied during sketching at a particular point, the thicker the 2D curve at that point. The system maps depth distance to each point along the 2D curve based on its thickness. The user can then modify the depth profile of the 3D curve generated by increasing or reducing the thickness of the drawn 2D curve.

## 2 Related Work

There has been a great amount of research work in the field of sketch-based and free-form surface modeling [7,8,9,10,11,12]. However, research work on 3D curve sketching is very few. Since surfaces can be derived from curves, curve input is the more

fundamental task and has to be explored more deeply [2]. It is difficult and tedious to create curvy and complex geometric surface using commercial 3D modeling system; users have to perform tedious and time consuming detail 3D curve specification. Bae et al [3] presented a holistic system that enables product designers to create complicated 3D curve networks while maintaining a consistent sketch-like workflow. This system is very powerful yet complicated; it is mainly for professional artist in their design work. Cohen et al [1] demonstrates a method for specifying 3D curves with 2D input from a single viewpoint. User draws a curve with its shadow on the floor plane to compute the curve's 3D shape. The constraint of this system is that it is difficult to judge how the shadow should look like for a complex 3D curve. Amicis et al [2] proposed a pencil and rubber metaphor in one tool to allow user to specify 3D curves directly within a semi-immersive environment known as *Virtual Table*. In the automobile industry, Grossman et al [5] presented a system to construct non-planar 3D curves by drawing a series of 2D curves using the 2D tape drawing technique and interaction style. In both [2] and [5], special hardware is needed for 3D curve drawing. Mizuno et al [6] illustrated the use of pressure sensitive pen to control depth of carving. In our paper, pressure is used to control the thickness of the 2D curve which serves as a visual aid in drawing and modifying the 3D curve. The thickness of the 2D curve is used to imply the depth distance of a 3D curve.

This paper presents a simple yet less costly 3D curve drawing technique. It provides direct interaction to sketch and edit the 3D curve; a novice user can sketch an approximate 3D curve without much overhead in learning the interface. Our implementation relies on a Wacom pen tablet; however it can be applied to any similar technology that provides pressure information.

### 3 Pressure-Based Curve Drawing

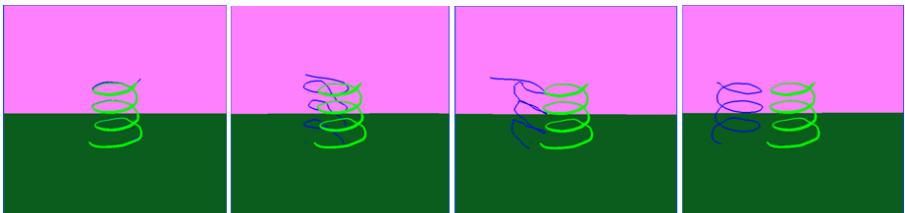
The basic operations supported by our curve-sketching system are as follows: sketching a new 2D curve with varying thickness, rotating the 3D curve generated, and increasing and decreasing the 2D curve thickness to change its depth profile. The beauty of this system is its simplicity: by making use of pen pressure, only a single viewpoint, and a single line is needed.

The 3D curve is generated with the concept ‘bigger thing appears closer’ in mind. In the editing mode, any sketch on the screen will not generate new line. To increase the line thickness, user draws more strokes on the existing line. This will then move the affected points in the 3D curve closer to the user. The user can also decrease the line thickness by applying more strokes on the existing 2D curve using the eraser tip of the Wacom pen. This process will move the affected points in the 3D curve further from the user.

#### 3.1 Drawing 2D Curve and Mapping to 3D

The 2D curve is rendered as a polygon strip. The pressure applied to the Wacom pen during the sketching process will affect the size and shape of the polygons created. The greater the pressure the bigger the polygon appears. With the concept ‘bigger thing appears closer’ in mind, user is able to estimate and control the depth difference along the 3D curve when sketching the 2D curve; hence varying depth distance.

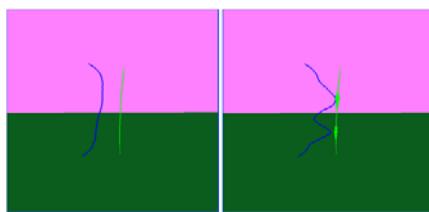
The 2D curve sketched in Windows coordinate is transformed into 3D curve in world space. The 3D curve generated has a constant thickness. Pressure data from Wacom pen is used to manipulate the depth distance. Fig. 1 demonstrates a 3D (blue) generated by controlling the drawing pressure of a 2D (green) curve.



**Fig. 1.** Generating 3D coil by controlling 2D curve thickness

### 3.2 Manipulating Thickness and Depth

To increase the 2D curve thickness, user applies more strokes on the existing 2D curve. This technique resembles the natural way of drawing with pencil and paper, where multiple overlapping strokes make a line thicker. In our system, the pressure data from every additional overlapping stroke is used to recalculate the depth distance of the affecting points in 3D curve. Fig. 2 shows the effect of increasing the 2D curve thickness and its corresponding depth distance change in the 3D curve. The 2D curve serves as visual aids in controlling the depth distance of the 3D curve; the thicker the 2D curve, the closer the 3D curve to the user. Fig. 2(left) presents the original 2D (green) and 3D (green) curve before editing, while Fig 2(right) demonstrates the increased curve thickness is mapped to the closer depth distance; the affected points in the 3D curve become closer to the user.



**Fig. 2.** 2D and 3D curve before editing (left). 3D curve depth distance change after increasing 2D curve thickness (right).

To decrease the 2D curve thickness, user simply uses the eraser tip of the Wacom pen to ‘erase’ the 2D curve. The points affected in the 3D curve are now further from the user. The mechanism of decreasing the line thickness and changing the depth distance is similar to that described above for increasing line thickness, the only differences are the polygon now appears smaller and the 3D curve appears further from the user.

## 4 Conclusion

The proposed approach allows for fast specification of approximate 3D curves. Objects in our daily life such as human hair, animal fur, brush, tree branches and etc can be modeled more quickly and easily without the need for mathematical knowledge or 3d manipulation dexterity. The incorporation of the ‘pencil and paper’ concept together with the ‘big thing appears to be closer’ concept has made this system intuitive to use.

Work is in progress to investigate on how the user can more accurately estimate the amount of pressure to be applied in order to achieve specific depth distance change. Further, a user study will be needed to validate the effectiveness of the approach.

## References

1. Cohen, J.M., Markosian, L., Zeleznik, R.C., Hughes, J.F., Barzel, R.: An interface for sketching 3D curves. In: Proceedings of the 1999 symposium on Interactive 3D graphics, Atlanta, Georgia, United States, April 26-29, pp. 17–21 (1999)
2. Amicis, R.D., Bruno, F., Stork, A., Luchi, M.L.: The Eraser Pen: A New Paradigm for Curve Sketching in 3D. In: International Design Conference – Design 2002, Dubrovnik, May 14-17 (2002)
3. Bae, S.H., Balakrishnan, R., Singh, K.: ILoveSketch: As-natural-as-possible sketching system for creating 3D curve models. In: ACM Symposium on User Interface Software and Technology 2008, Monterey, CA, USA, October 19-22 (2008)
4. Schmidt, R., Khan, A., Kurtenbach, G., Singh, K.: On Expert Performance in 3D Curve-Drawing Tasks. In: EUROGRAPHICS Symposium on Sketch-Based Interfaces and Modeling (2009)
5. Grossman, T., Balakrishnan, R., Kurtenbach, G., Fitzmaurice, G., Khan, A., Buxton, B.: Creating Principal 3D Curves with Digital Tape Drawing. In: Proceedings of the ACM CHI 2002 Conference on Human Factors in Computing Systems (2002)
6. Mizuno, S., Kobayashi, D., Okada, M., Toriwaki, J., Yamamoto, S.: Virtual sculpting with a pressure sensitive pen. In: ACM SIGGRAPH 2003, San Diego, California, USA (2003)
7. Igarashi, T., Matsuoka, S., Tanaka, H.: Teddy: A Sketching Interface for 3D Freeform Design. In: ACM SIGGRAPH 1999, Los Angeles, pp. 409–416 (1999)
8. Ijiri, T., Okabe, M., Owada, S., Igarashi, T.: Floral diagrams and inflorescences: Interactive flower modeling using botanical structural constraints. ACM Transactions on Computer Graphics 24(3), 720–726 (2005); ACM SIGGRAPH 2005, Los Angeles, USA
9. Masry, M., Lipson, H.: A Sketch-Based Interface for Iterative Design and Analysis of 3D Objects. In: 2nd Eurographic Workshop on Sketch-Based Interfaces and Modeling (2005)
10. Cook, M.T., Agah, A.: A Survey of Sketch-Based 3-D Modeling Techniques. Interacting with Computers 21, 201–211 (2009)
11. Olsen, L., Samavati, F.F., Sousa, M.C., Jorge, J.A.: Sketch-based modeling: A survey. Computer & Graphics 33, 85–103 (2009)
12. Pu, J., Lou, K., Ramani, K.: A 2D Sketch-based User Interface for 3D CAD Model Retrieval. Computer-Aided Design & Application 2(6), 717–725 (2005)

# An Interactive Design System for Water Flow Stains on Outdoor Images

Yuki Endo, Yoshihiro Kanamori, Jun Mitani, and Yukio Fukui

University of Tsukuba,  
1-1-1 Tennodai, Tsukuba, Ibaraki, 305-8573, Japan  
`{endo,kanamori,mitani,fukui}@npal.cs.tsukuba.ac.jp`

**Abstract.** Weathering and aging effects are essential for synthesizing realistic outdoor images in the field of film production and scene production. The problem here is that photographed materials do not yield desired images without editing, and manual painting requires labor-intensive work as well as professional skills. In this paper, we focus on a salient aging effect, *stains by water flows*, and present a system that allows the user to add such stains directly and easily onto outdoor images. Our system represents a water droplet with a particle and simulates the dissolution, transport and sedimentation of deposits using particles in the regions specified by the user. In the simulation, realistic and complex stains can be obtained by accounting for the roughness of the surface where water flows. Furthermore, the user can adjust the amount of deposits according to the perspective in the image. The quick feedback of the simulation enables interactive manipulation. We demonstrate the effectiveness of our system through various results and user evaluations.

**Keywords:** weathering and aging, image processing, particle system.

## 1 Introduction

Photorealistic reproduction of weathering and aging phenomena often plays an important role in the field of film production and scene production. Especially, *weathering and aging of buildings*, e.g., rusts, stains and pollution caused by rainfalls, chemical reaction, ultraviolet light and other various factors, have high demand because buildings are ubiquitous in our daily lives and thus might look unnatural without such phenomena. A usual way to seek such realism is to use photographed materials as texture images and edit them to obtain desired results. However, looking for appropriate materials itself is not easy, and editing materials requires labor-intensive work as well as professional skills.

In this paper, we focus on a salient aging effect, *stains by water flows*, and present a system that allows the user to add such stains directly and easily onto building walls in outdoor images. Our system represents a water droplet with a particle and simulates the dissolution, transport and sedimentation of deposits using particles in the regions specified by the user. This simulation scheme is based on a simple model proposed for 3D models by Dorsey et al. [2], but we

modify it to improve the performance and the usability for image editing. In the simulation, realistic and complex stains can be obtained by accounting for the surface roughness where water flows. The user can specify the initial and terminal positions of particles by drawing a few lines. Furthermore, the user can adjust the amount of deposits according to the perspective in the image; using a pair of auxiliary lines drawn by the user, our system makes water flow stains shorter and thinner as the distance from the viewpoint gets longer. The quick feedback of the simulation enables interactive manipulation. We demonstrate the effectiveness of our system through various results and user evaluations.

## 2 Related Work

Realistic representation of weathering and aging phenomena has been an important theme in the field of computer graphics. The previous methods can be broadly grouped into *image-based approaches* that obtain information from real photographs and *simulation-based methods* that handle specific targets such as rusts, cracks and dirt. Here we briefly introduce some of them, and refer to the survey paper by Merillou et al. [8] for more information.

**Image-based Approaches.** Gu et al. [6] obtained and modeled time-space varying bidirectional distribution functions (BRDFs), and then reproduced temporal variations of materials. Wang et al. [12] constructed a manifold that represents the temporal variation of a material using BRDFs measured at various points on the material in order to reproduce the transition of complicated texture patterns. While these methods target 3D models, Xue et al. [13] applied Wang et al.'s method to objects in 2D images to synthesize the weathered or de-weathered appearance. Although image-based approaches successfully reproduce various texture patterns, they suffer from handling physical properties of the target objects because they do not account for the physical law of the phenomena.

**Simulation-based Approaches.** Physically-based simulations of weathering and aging have been applied to various targets such as stones [4], paint peeling [9], cracks of clay-like materials [11] and wooden materials [14]. These methods are accompanied by geometric changes. Previous methods that do not handle geometric changes include weathering and aging by dust [7], rust [3] and moss [1].

Dorsey et al. [2] employed a particle simulation to reproduce stains caused by flows streaming on 3D models. In their method, particles dissolve and carry deposits, and the deposits are accumulated on the surface according to a set of partial differential equations. Their method can represent various patterns of realistic stains by tuning parameters, but lacks flexible control for designing stains. On the other hand, our simulation scheme is a modified version of Dorsey et al.'s method specialized for 2D image editing, and collaborates with the user interface for designing water flow stains.

**Dorsey et al.’s Model.** In Dorsey et al.’s model, a particle represents a water droplet parameterized by the mass  $m$ , position  $\mathbf{x}$ , velocity  $\mathbf{v}$ , and the amount of dissolved deposits  $S$ . Particles are initially assigned on the surface of a 3D model at random by a rainfall, and then flow downward under the influence of the gravity and frictions. Particles interact with each other due to the repulsive forces among them. Water is absorbed into the object surface, and thus the surface also has a set of parameters, namely, the surface roughness  $r$ , the amount of absorbed water  $w$ , the rate for absorption  $k_a$ , the saturated amount of absorption  $a$ , and the amount of sediment  $D$ . Regarding the deposits carried by water particles, there are attributes such as the adhesion rate constant  $k_S$ , the solubility rate constant  $k_D$ , the evaporation rate  $I_{sun}$ , and the initial deposition amount on the surface  $I_D$ . Dorsey et al.’s model uses a scalar surface roughness  $r$  or a displacement map on the surface in order to obtain the interesting movement of particles; the roughness makes particles disperse whereas the displacement map let particles move slowly across a bumpy surface, yielding more sediment along cracks and valleys. Consequently, the absorption of water is modeled as follows;

$$\frac{\partial m}{\partial t} = -k_a \frac{a - w}{a} \frac{A}{m}, \quad (1)$$

$$\frac{\partial w}{\partial t} = k_a \frac{a - w}{a} \frac{m}{A} - I_{sun}, \quad (2)$$

where  $t$  denotes the simulation time and  $A$  is the diameter of the water particle. Similarly, the solution and sedimentation of deposits are modeled as follows;

$$\frac{\partial S}{\partial t} = -k_S S + k_D D \frac{m}{A}, \quad (3)$$

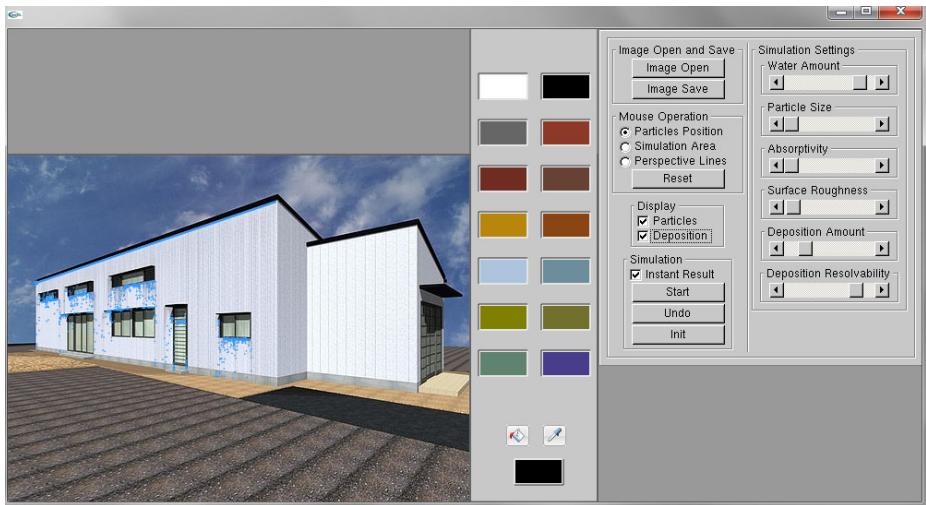
$$\frac{\partial D}{\partial t} = k_S S \frac{A}{m} - k_D D + I_D. \quad (4)$$

When the mass  $m$  becomes smaller than a certain threshold, the particle is removed from the simulation.

### 3 User Interface

This section describes our prototype system for designing images containing water flow stains. Our system adds such stains using particle simulations, and allows the user to easily specify the regions where the water particles flow on the input image.

Fig. II shows the screenshot of our prototype system. In the left window that displays the input image and flowing particles (blue), the user can directly specify the regions for simulation and the perspective of the input image, and then obtain simulated results quickly. In the right window, the user can configure the parameters and other settings of the simulation. In the middle panel, the user can select the color of deposits using either of the color pallet, color chooser or color picker. The current color is displayed at the bottom.



**Fig. 1.** Screenshot of our system<sup>1</sup>

### 3.1 Control Lines

In the left window in Fig. 1, the user specifies three types of control lines (or curves) to control the particle simulation (Fig. 2);

**Source line** (the blue line in Fig. 2(a)): to specify the initial position of particles. Particles are emitted from these lines.

**Terminal lines** (the green lines in Fig. 2(a)): to specify the positions where the simulation terminates. Particles are removed when they touch these lines.

**Perspective lines** (the two red lines in Fig. 2(b)): to specify the perspective of the input image. The amount of deposits is adjusted according to the perspective of the input image, specified by these lines. See Section 4 for more details.

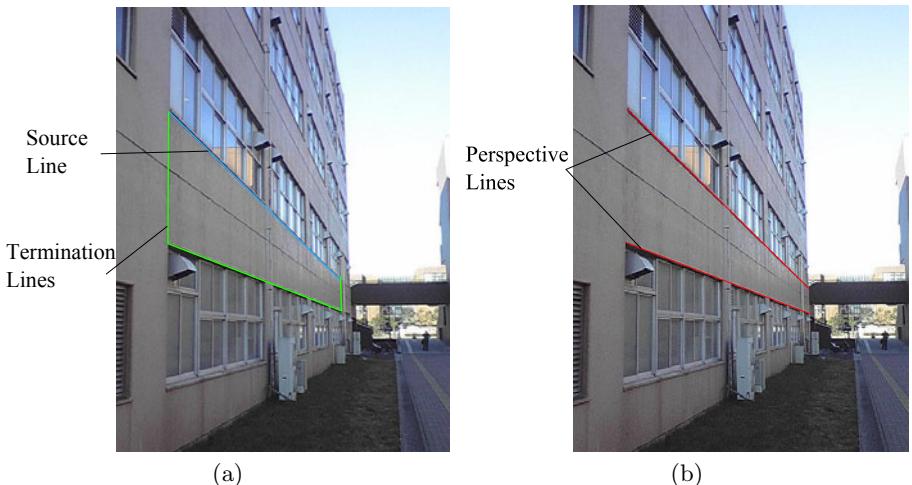
The user can draw a straight line by clicking its endpoints and a curve by dragging the cursor.

### 3.2 Simulation Parameters

The user can adjust the following six parameters to make a variety of water flow stains using scroll bars. The way each parameter influences the simulation is described in Section 4.

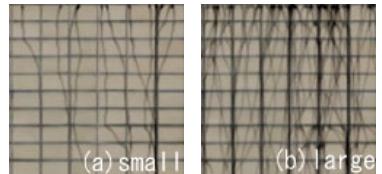
<sup>1</sup> Author of the input image: heavymoonj

URL: <http://www.flickr.com/photos/heavymoonj/261996484/>



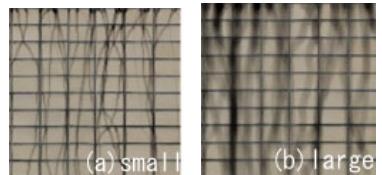
**Fig. 2.** Three types of control lines for the particle simulation. (a) The blue line specifies the source position of particles, and the green lines represent the terminal positions; particles are removed when they touch these lines. (b) The perspective of the input image is specified by the pair of the two red lines in order to adjust the amount of deposits according to the perspective.

**Water Amount.** The parameter for the water amount specifies the number of particles. By increasing this parameter, more complex stains caused by more flows can be obtained (Fig. 3).



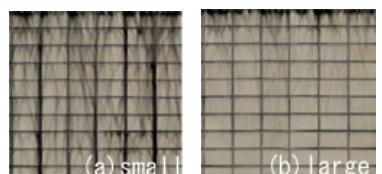
**Fig. 3.** Effect of Water Amount

**Particle Size.** The relative size of particles with regard to the input image can be adjusted using this parameter (Fig. 4).



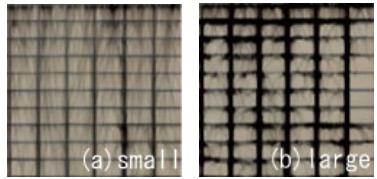
**Fig. 4.** Effect of Particle Size

**Absorptivity.** This parameter specifies the rate with which the mass of each particle decreases. The flow distance of each particle becomes short by increasing this parameter (Fig. 5).



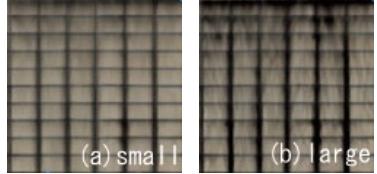
**Fig. 5.** Effect of Absorptivity

**Surface Roughness.** This parameter determines how much each particle is diffused and decelerated due to the bumps on the surface. By increasing this parameter, more deposits tend to accumulate at cracks and ditches (Fig. 6).



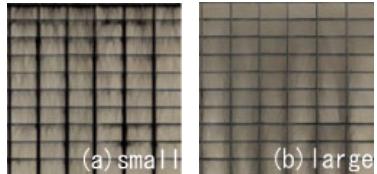
**Fig. 6.** Effect of Surface Roughness

**Deposition Amount.** This parameter specifies the amount of deposits attached on the surface. A large value yields thick stains (Fig. 7).



**Fig. 7.** Effect of Deposition Amount

**Deposition Resolvability.** This parameter determines how easily deposits dissolve in water particles. A large value yields a blurred image (Fig. 8).



**Fig. 8.** Effect of Deposition Resolvability

## 4 Simulation Scheme

This section describes the simulation scheme to control water particles. Our scheme is based on Dorsey et al.’s model [2] because of its simplicity. In their model, each particle represents a water droplet that dissolves and transports deposits which are then accumulated along the tracks of particles. While their model targets the 3D space, we simulate on the 2D domain, i.e., the input image. We simplify their model to reduce the number of parameters and to provide faster feedback to the user. Furthermore, we extend it to handle the surface roughness on which particles flow and to adjust the amount of deposits according to the perspective of the image.

**Basic Modifications to Dorsey et al.’s Model.** As described above, we modify Dorsey et al.’s model for interactive simulations on a 2D image. We define the  $xy$  coordinate system along the horizontal and vertical edges of the image. To accelerate the simulation, we ignore the interaction among particles and frictions but only consider the gravity. According to Eq. (1), (2), (3) and (4), the variations of the mass and amount of absorbed water depend on the diameter of particles, which means the “thickness” of the deposit color changes

according to the diameter. This is not desirable for our purpose because we want to control the “thickness” only by  $I_d$  (i.e., *Deposition Amount* in Section 3.2), and thus we omit  $m/A$  and  $A/m$  from the equations. In Eq. (3) and (4), we set the range of  $k_D$  and  $k_S$  [0, 1], and let  $k_S = 1 - k_D$  to reduce the number of parameters. The user can control  $k_a$  and  $k_D$  as *Absorptivity* and *Deposition Resolvability* respectively, as described in Section 3.2.

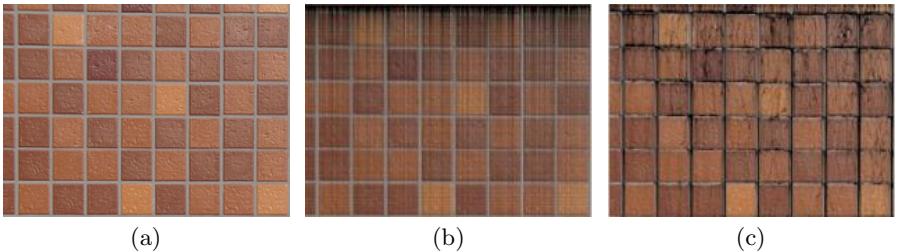
**Displacement due to Luminance Variations.** As shown in Fig. 9(a), the luminance values change greatly in general across cracks and valleys in images. Therefore, we construct the displacement map from the variations of the luminance values in the input image, and use the surface roughness  $r$  as a coefficient to amplify the influence of the map. The particle velocity  $\mathbf{v} = (v_x, v_y)$  is reduced according to the variations of the luminance values;

$$v_x^{diffuse} = b r \xi M_y(\mathbf{x}), \quad (5)$$

$$\frac{\partial v_x}{\partial t} = v_x^{diffuse} - v_x \frac{M_x(\mathbf{x})}{r_{max} - r + 1}, \quad (6)$$

$$\frac{\partial v_y}{\partial t} = -v_y \frac{M_y(\mathbf{x})}{r_{max} - r + 1}, \quad (7)$$

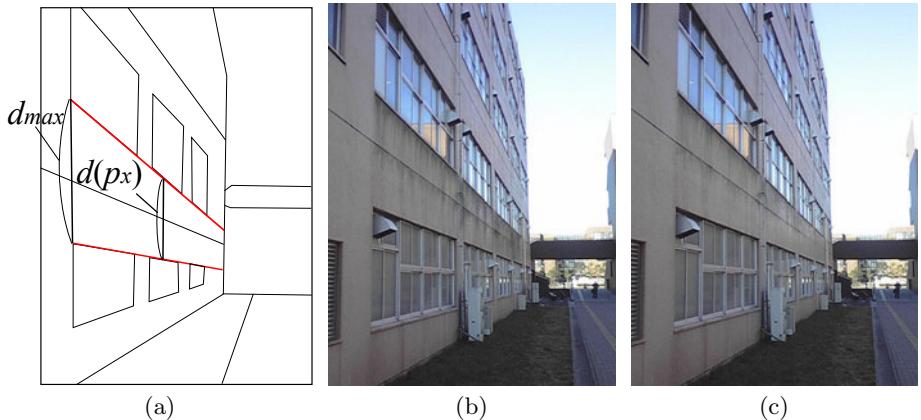
where  $M_x(\mathbf{x})$  and  $M_y(\mathbf{x})$  are the horizontal and vertical variations of luminance values at particle position  $\mathbf{x}$ ,  $b$  is a constant,  $\xi \in [-1, 1]$  is a random value, and  $r_{max}$  is the maximum of  $r$ . The user can control  $r$  as *Surface Roughness* as described in Section 3.2. Fig. 9 shows the results (c) with and (b) without accounting for the surface roughness. Compared to Fig. 9(b), Fig. 9(c) exhibits more sedimentation along the ditches, which makes the result look much more realistic.



**Fig. 9.** Simulation results (c) with and (b) without accounting for the displacement defined by the variations of luminance values in (a) the input image.

**Adjustment to the Perspective in the Image.** In a perspective view, water flow stains look shorter and thinner as the distance from the viewpoint becomes greater. However, such sense of distance is lost on an image, and thus we fail to

reproduce the perspective effect when directly applying our scheme (Fig. 10(b)). To address this issue, our system lets the user specify the perspective using a pair of auxiliary lines (i.e., *Perspective Lines* in Section 3.1). See Fig. 10(a) for an illustration. Let  $d(p_x)$  be the vertical distance between the two lines at  $x = p_x$  and  $d_{max}$  be the maximum of  $d(p_x)$ . To adjust the perspective in the input image, our system multiplies the gravity, the particle mass and the deposition amount by  $\rho = d(x)/d_{max}$ , where  $x$  is the  $x$  coordinate of the particle position. Fig. 10(c) illustrates that the adjustment successfully reproduces the perspective effect.

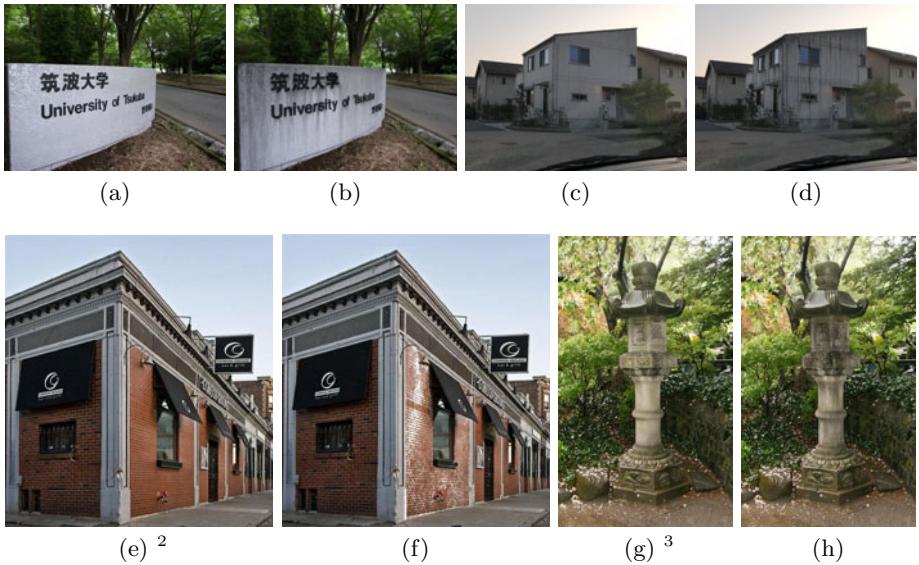


**Fig. 10.** (a) An illustration for the perspective adjustment. Simulated results (c) with and (b) without the perspective adjustment.

## 5 Results

We implemented our prototype system using C++ language, OpenGL and GLUI, and executed our system on a PC with a Xeon 2.66GHZ, 2GB RAM, and an NVIDIA Quadro FX 3450 graphics card. Fig. 11 shows the resultant images with water flow stains synthesized using our system. The user succeeded to design realistic stains within moderate time. Regarding the time for editing, the user generally spent the most of time in tuning parameters. As the image size becomes larger, the simulations takes longer as shown in Fig. 12. We believe that the time for simulations will be much shorter using minified images and GPU implementation in future work.

**User Test.** To validate the effectiveness of our system, we performed a user test to compare its performance to the *Adobe Photoshop*®. Six students, all novice users of our system and image editing software, participated in the study. After



**Fig. 11.** (b)(d)(f)(h) Synthesized images with water flow stains using our system. The image sizes of their inputs are (a)  $500 \times 375$ , (c)  $512 \times 384$ , (e)  $452 \times 491$ , and (g)  $300 \times 500$  pixels. The times for editing are about (b) 3 min, (d) 10 min, (f) 4 min, and (h) 5 min.

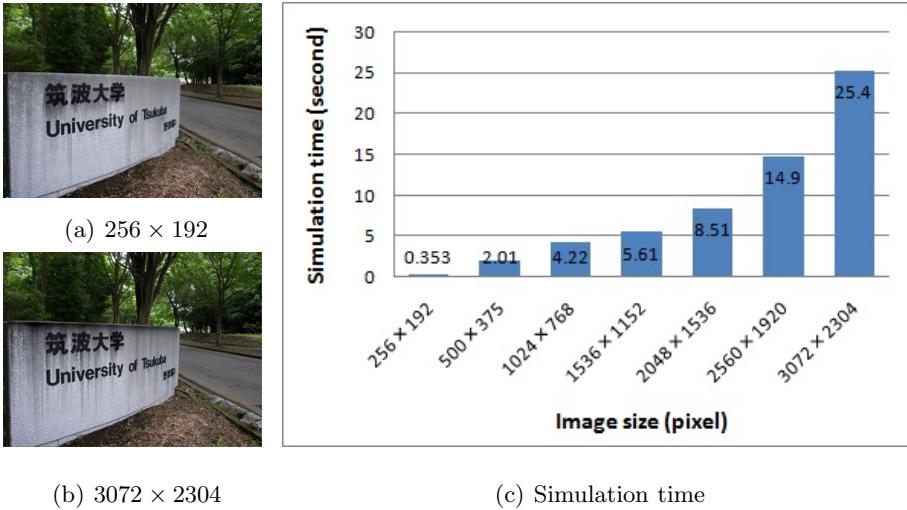
a brief tutorial, each subject was asked to synthesize water flow stains on the tile image (Fig. 14(b)) in reference to the stained tile image (a real photograph, Fig. 14(a)), using either our system or the alternative software (Photoshop). The testing order of the software was alternated between subjects; three used Photoshop first, and the other three used our prototype system first. The subjects were allowed to work on the task until satisfied, for up to 20 minutes. Fig. 10 shows some of the resulting images. Fig. 13(a) shows the time of design process across six subjects and two design tools. Subjects using our system overall took less than 70% of the time when they did using Photoshop. Fig. 13(b) shows the subjective evaluation of the produced images. Ten people voted on the resulting images, regarding how natural each image is. A binomial test of the scores shows that our system performed better than the Photoshop in making water flow stains except for the results of participant #6, who seemed to misunderstand our system and tried to use it as an ordinary paint tool. These results show that our system achieves better quality in less time than the Adobe Photoshop.

<sup>2</sup> Author of the input image: new hobby

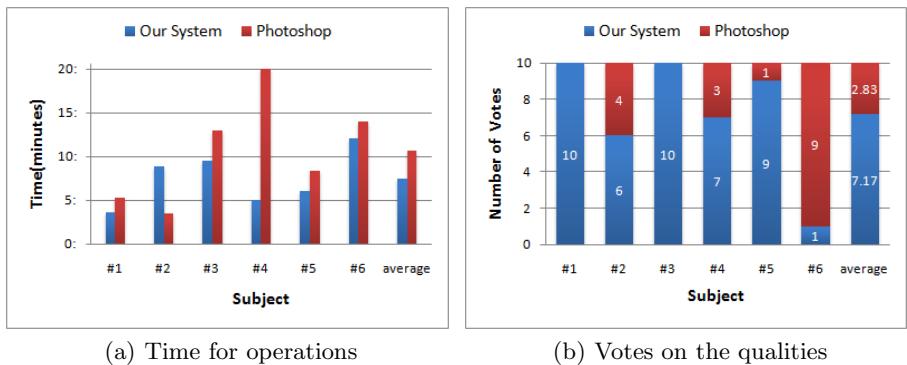
URL: <http://www.flickr.com/photos/newhobby/2427677211/>

<sup>3</sup> Author of the input image: peterjr1961

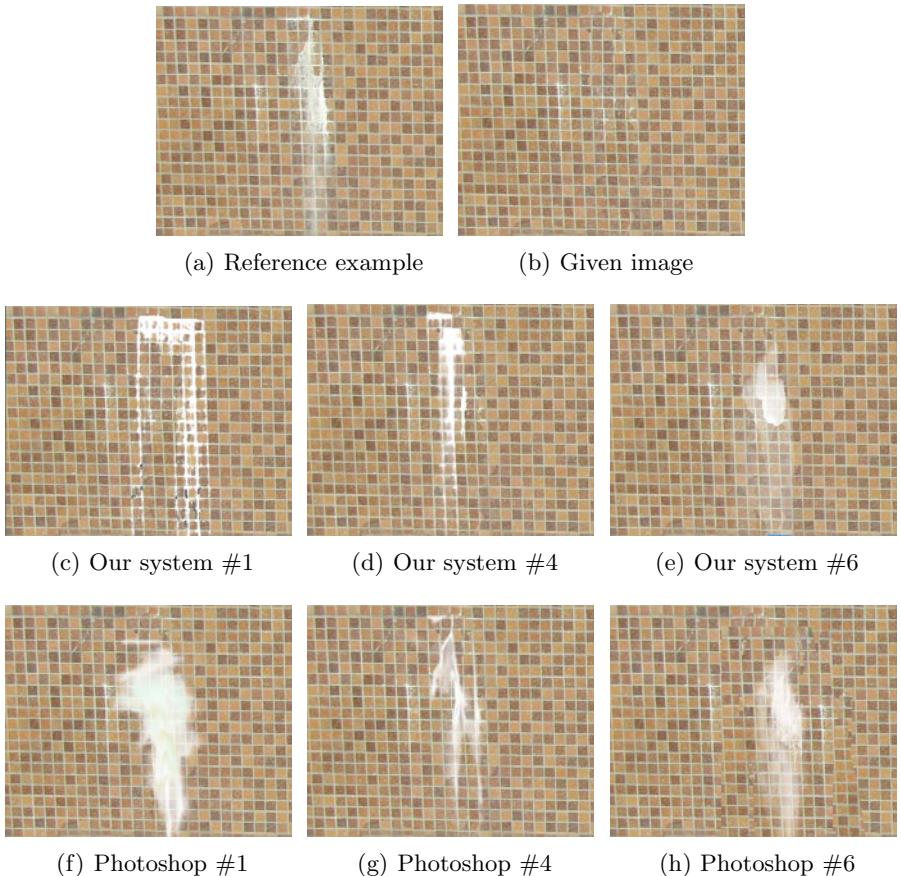
URL: <http://www.flickr.com/photos/peterjr1961/3506811726/>



**Fig. 12.** (a)(b) Resultant images with different image sizes for the same scene as Fig. 11(a). The image size is listed under each image. (c) shows the simulation time (second) for a single execution with each image size.



**Fig. 13.** Graphs summarizing the results of a user test. Each of six subjects was asked to synthesize two images, one using our system and the other using Photoshop, so that the images became satisfying for the subject. (a) shows the time of design process across six subjects and two design tools. (b) Then ten people voted on the result, regarding how natural each image is. The number represents that of votes for each system.



**Fig. 14.** Results of the user test, comparing our system to Photoshop. The users were presented (a) a reference example (a real photograph) and asked to edit (b) a given image until the user gets satisfied. The middle row (c)(d)(e) presents the results by different users using our system whereas the bottom row (f)(g)(h) shows those using Photoshop.

## 6 Conclusion and Future Work

We have proposed an interactive system that helps users design water flow stains on outdoor images. Based on a particle simulation modified from Dorsey et al.'s model, our system allows the user to specify the initial and terminal positions of particles by drawing a few control lines (Section 3.1). Additionally, the user can adjust the simulation to the perspective in the input image by drawing a pair of auxiliary lines (Section 3.1). Regarding the simulation scheme (Section 4), we reduced the parameters used in Dorsey et al.'s model to improve the usability, and ignored the interaction between particles to accelerate simulations. Our

system automatically estimates the bumpiness of the surface where particle flows, using the luminance variations in the input image. A user test demonstrated that our system yields better results more quickly than a generic paint tool in the task of synthesizing water flow stains (Section 5). A limitation of our current system is that luminance artifacts (e.g., specular highlights, shadows) in the input image might affect the simulation, which will be avoided by removing such artifacts [5] or by manually editing the displacement map (Section 4). Because our current system fixes the direction of gravity vertically downward and assumes the surface on which particles flow is flat, the flow directions are limited. The use of the control mesh [10] will improve the controllability of particle flows.

## References

1. Desbenoit, B., Galin, E., Akkouche, S.: Simulating and Modeling Lichen Growth. In: Eurographics 2004, pp. 341–350 (2004)
2. Dorsey, J., Pedersen, H.K., Hanrahan, P.: Flow and Changes in Appearance. In: Proc. of SIGGRAPH 1996, pp. 411–420 (1996)
3. Dorsey, J., Hanrahan, P.: Modeling and Rendering of Metallic Patinas. In: Proc. of SIGGRAPH 1996, pp. 387–396 (1996)
4. Dorsey, J., Edelman, A., Jensen, W.H., Legakis, J., Pedersen, H.: Modeling and Rendering of Weathered Stone. In: Proc. of SIGGRAPH 1999, pp. 225–234 (1999)
5. Finlayson, G., Hordley, S., Drew, M.: Removing shadows from images. In: Heyden, A., Sparr, G., Nielsen, M., Johansen, P. (eds.) ECCV 2002. LNCS, vol. 2353, Part IV, pp. 823–836. Springer, Heidelberg (2002)
6. Gu, J., Tu, C., Ramamoorthi, R., Belhumeur, P., Matusik, W., Nayar, S.: Time-Varying Surface Appearance. In: SIGGRAPH 2006, pp. 762–771 (2006)
7. Hsu, S., Wong, T.: Visual Simulating Dust Accumulation. IEEE Computer Graphics and Applications 15(1), 18–22 (1995)
8. Merillou, S., Ghazanfarpour, D.: A survey of aging and weathering phenomena in computer graphics. Computers & Graphics 32, 159–174 (2008)
9. Paquette, E., Poulin, P., Drettakis, D.: The Simulation of Paint Cracking and Peeling. Graphics Interface, 59–68 (2002)
10. Pavic, D., Schonefeld, V., Kobbelt, L.: Interactive image completion with perspective correction. The Visual Computer 22, 671–681 (2006)
11. Tanoue, Y., Hirota, K., Kaneko, T.: Simulation of Three-Dimensional Cracks and Animation. Information Processing Society of Japan 99(19), 13–18 (1999) (in Japanese)
12. Wang, J., Tong, X., Lin, S., Pan, M., Wang, C., Bao, H., Guo, B., Shum, H.: Appearance manifolds for modeling time-variant appearance of materials. In: SIGGRAPH 2006, pp. 754–761 (2006)
13. Xue, S., Wang, J., Tong, X., Dai, Q., Guo, B.: Image-based Material Weathering. In: EUROGRAPHICS 2008, pp. 617–626 (2008)
14. Yin, X., Fujimoto, T., Chiba, N.: CG Representation of Wood Aging with Distortion, Cracking and Erosion. The Journal of the Society for Art and Science 3(4), 216–223 (2004)

# Automated Hedcut Illustration Using Isophotes

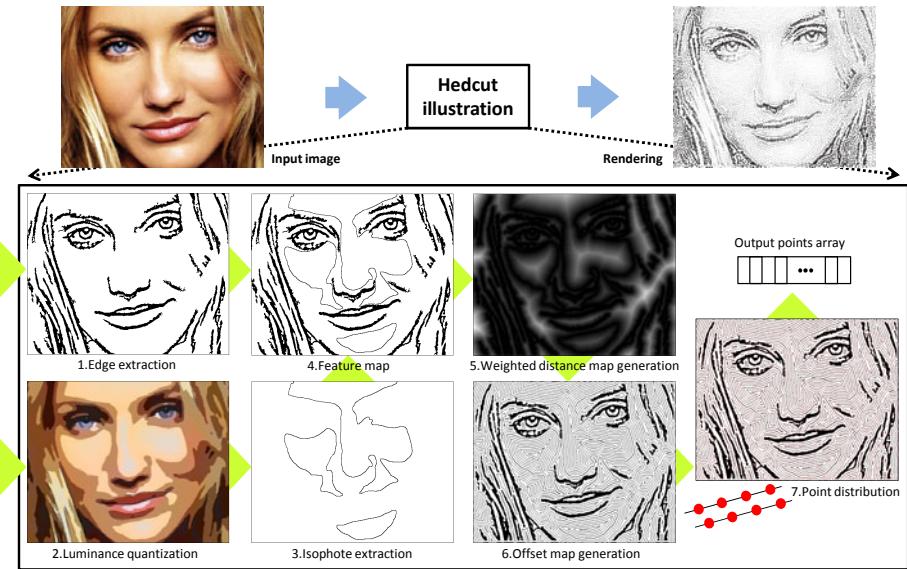
SungYe Kim, Insoo Woo, Ross Maciejewski, and David S. Ebert

Purdue University, West Lafayette IN 47907, USA

**Abstract.** In this work, we present an automated system for creating hedcut illustrations, portraits rendered using small image feature aligned dots (stipples). We utilize edge detection and shading cues from the input photograph to direct stipple placement within the image. Both image edges and isophotes are extracted as a means of describing the image feature and shading information. Edge features and isophotes are then assigned different priorities, with isophotes being assigned the highest priority to enhance the depth perception within the hedcut portrait. Priority assignment dictates the stipple alignment and spacing. Finally, stipple size is based on the number of points and intensity and the gradient magnitude of the input image.

## 1 Introduction

Hedcut is an illustration drawing style that emulates the look of engraving and woodcuts and is mainly associated with portrait illustrations in The Wall Street Journal. These drawings use a combination of hatching and oriented stippling. Stippling is a popular illustrative rendering technique used to convey local tone through the placement of a large number of dots. Since points are the simplest graphic primitives, and manual stippling is an extremely time consuming task (normally taking hours for each illustration), automating stippling is a desirable goal. Researchers have developed many techniques for creating computer-generated stipple renderings [14,15,22,23] while focusing on non-periodic distributions of stipples as utilized in traditional scientific illustrations. In contrast, the hedcut style places stipples directly along feature lines in order to emulate woodcuts and engravings. Stipples are aligned to detail edges following the highlights and contours of an image, while capturing textures and shading within a given photograph. However, little research has been done in creating such feature aligned stipple images. Recently, Kim et al. [13] introduced an edge directed image stippling technique. While their framework proves to be an effective method for generating hedcut style images, their work does not reflect the 3D structure of a surface in an image when the surface does not include any edge features. As an extension to this work, we enhance this type of computer-generated hedcut by utilizing shading information found within an input photograph. By utilizing the shading information, we are able to provide users with more perceptual cues as to the depth of the image [3,6,7]. Our extension is informed by observations found in professional hedcut drawings. Further, if one analyzes the traditional problem of sphere shading, we can begin to understand an artist's methodology.



**Fig. 1.** Conceptual diagram of our hedcut illustration system

Typically, artists often begin drawing 3D objects by first placing feature lines that demarcate the key structures of an object. In the example of a sphere, the artist would first draw a freehand circle. Next, the artist would sketch in the core shadow of the sphere in order to give the circle depth. This type of sketch should correlate to isophotes within the image. Thus, by taking into account the isophotes and using those to inform our stipple placement, we are able to improve upon the previous work by Kim et al. [13].

In this paper, we present an automated hedcut illustration system using isophotes (lines with constant illumination) from a photograph to better capture the perceptual cues of the input image. Figure 1 illustrates the conceptual diagram of our hedcut illustration system. As seen in Figure 1, we take a photograph as input to the system and perform a series of steps from edge extraction to stipple placement, thereby creating a hedcut illustration. Our contributions include: 1) the use of isophotes for representing local shape and shading in hedcut illustrations and 2) the use of a weighted constraint based on edges and isophotes for importance mapping and depth perception.

## 2 Related Work

In non-photorealistic rendering (NPR), we can find a variety of approaches focusing on different styles for creating human facial illustrations, such as caricatures and portraits. Gooch et al. [8] interactively created caricatures of human faces by producing and deforming black and white illustrations. Luo et al. [16] also used

a black and white facial portrait generator to produce NPR facial expression animation. Although it is not restricted to facial illustration, Mould and Grant [19] proposed methods to create stylized black and white images from photographs through energy minimization and adaptive thresholding, while preserving details. Moreover, there has been much research in creating human portraits in line drawing or sketches to achieve stylized caricature. For example, particularly Chen et al. [2] introduced a system for generating human portrait sketches. Their system takes a human face image and outputs a sketch that represents the drawing style (i.e., manga) provided by an artist as examples. Our research also focuses on stylizing facial photographs; however, as opposed to previous NPR research, we concentrate on representing the hedcut style.

In hedcut illustration, stippling and hatching techniques are used to render images, while drawing primitives (i.e., dots, short lines) along a certain direction. Traditionally, stippling aims to attain tone matching between an input image and an output illustration by non-periodically placing small dots. Much research [4][10][23] has concentrated on capturing the hand-drawn aesthetics of stipple drawing by optimizing the position of points using a Lloyd's relaxation algorithm based on Voronoi diagrams or other methods. For example, Balzer et al. [1] recently proposed a method for improving the density adaptation of point distribution by utilizing the concept of capacity. Further, there were efforts to analyze, as well as emulate, the distribution of points from hand-drawn stippling works [11][14][17]. However, in creating stylized hedcuts, tone matching is less important.

A key step in creating hedcut images is proper stipple alignment. Mould [18] presented a progressive stippling algorithm using graph search to align points to edge features that are weighted with an importance value computed by the intensity and gradient magnitude. Hence, his stippling results showed an effect emphasizing edges, while maintaining the blue noise property within non-edge areas. Kim et al. [13] used edge features to guide the optimization of points based on the centroidal Voronoi diagrams (CVD). Unlike Mould's work, they also constructed feature flow for the entire input image based on edge features in order to align points on non-edge areas to the feature flow. Our work extends their work by incorporating and providing enhanced perceptual depth cues and shading features by generating weighted constraints based on shading features.

### 3 Hedcut Illustration Generation

In this section, we present our hedcut illustration technique using isophotes to extend and prioritize features from a photograph as constraints for feature aligned stipple point distributions. In order to better illustrate our rendering method, we approximate a head as a shaded sphere for description purposes in this section. We then demonstrate the applications of our rendering style on a series of face photographs.

### 3.1 Feature Map Generation

The first step for our technique is to generate a feature map from an input image. The features are used to create constraints for the optimization of point distribution. In this work, we consider two types of features; edges and isophotes.

#### Edge Extraction

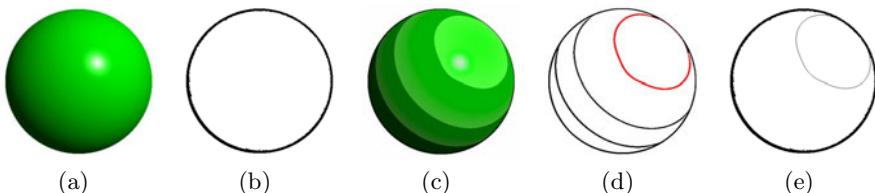
Edges are easily extracted using existing edge detection algorithms such as Sobel, Laplacian of Gaussian (LoG), and Canny operators. Luo et al. [16] compared five typical edge detection operators to extract edges on face images, and found that LoG and Canny operators were the best at extracting all detailed feature edges.

In this work, we use a Difference of Gaussian (DoG) filter as shown in Equation 1, which was used in [12, 20, 21] with variance as a reasonable compromise between accuracy and speed, as an approximation of LoG. Then, we remove edges with a length less than some predefined threshold, and employ a morphological opening operation (binary erosion followed by binary dilation) with a structuring element of a circle in order to alleviate noise and smooth the edges. Figure 2(b) shows edges extracted from the sphere image in Figure 2(a). For all results in this paper, we used  $\sigma_1=1.0$  and  $\sigma_2=1.5$  as parameters for a DoG filter, a threshold of 200 pixels for short edge removal, and a circle of radius 1.0 for morphological opening.

$$DoG \triangleq G_{\sigma_1} - G_{\sigma_2} = \frac{1}{\sqrt{2\pi}} \left[ \frac{1}{\sigma_1} e^{-(x^2+y^2)/2\sigma_1^2} - \frac{1}{\sigma_2} e^{-(x^2+y^2)/2\sigma_2^2} \right] \quad (1)$$

#### Isophote Extraction

An isophote in an image is a group of pixels with constant intensity and isophotes have often been used to analyze local shapes by recovering the normals of 3D objects in the shape from shading (SFS) research [5, 21]. In NPR, isophotes also correspond to toon shading boundaries and were used by Goodwin et al. [9] to adjust the thickness of artistic strokes in computer generated illustrations to



**Fig. 2.** Feature map generation. (a) is an input image of a shaded sphere, (b) shows an edge extracted from an input image, (c) is an image abstracted by luminance quantization, and (d) shows isophotes from (c). Finally, (e) shows a feature map that combines the edge from (b) and isophote (in red) from (d).

better represent local shape and depth relationships. By using isophotes, our work differs from Kim et al.'s work [13] in that we provide the sense of both the shape and depth of an object to the final illustration. Using isophotes to capture shading features in an image is a common way for human artists to draw 3D objects, i.e., when drawing a sphere, artists start with a circle for the boundary and a great circle (an ellipse) inside it for directional sketching, in order to provide the impression of depth.

To determine isophotes, we apply a bilateral filter to an input image to remove noise while preserving edges. Then, we convert an input image into CIELab color space and perform luminance quantization [24] to create a cartoon-like image as shown in Figure 2(c). From this result, regions that have constant intensity are segmented and ordered by their intensity. Next, we select an area with the highest intensity and extract its boundaries. Figure 2(d) presents isophotes extracted from the result in Figure 2(c) and highlights the isophote selected as the boundary of the region with the highest intensity.

Finally, a feature map for the shaded sphere image is shown in Figure 2(e), combining edges and the selected isophote. Further, in a feature map, edges and isophotes are saved with different feature ids to distinguish the feature type.

### 3.2 Weighted Constraints Generation

The next step for our technique is to generate constraints from a feature map for the feature aligned stipple point distribution.

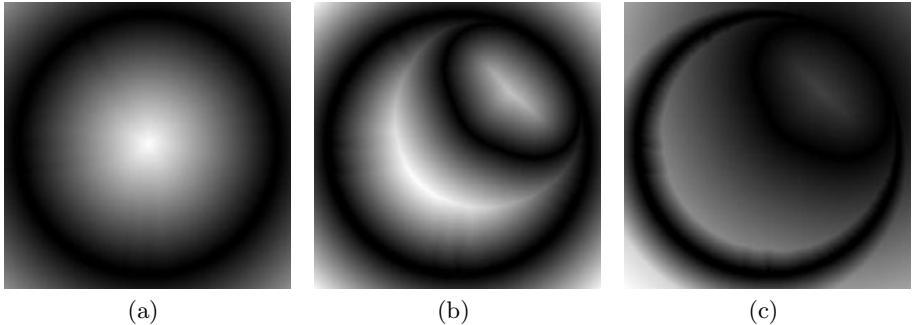
#### Weighted Distance Map Generation

Given a feature map, we apply a Euclidean distance transform, while applying different weight values according to the feature type, resulting in a weighted distance map. Our weighted distance,  $D_f(x, y)$ , is given by dividing Euclidean distances with the priority value,  $P_f$ , of each feature, as shown in Equation 2. In this work, we assign higher priority values for isophotes than that of the edges, in order to enhance the perceived depth. If a point  $(x', y')$  is on isophotes in a feature map,  $f(\cdot)$ , the distance value of a point  $(x, y)$  is weighted by the priority of isophote. Edges have a default priority. (In this work, we use a priority value of 2.0 for isophotes.) While deciding the distance of a point  $(x, y)$ , we also save the priority value applied to the point in a separate buffer, the parent priority buffer- $P_p$ ,  $P_p(x, y)$  thus includes the priority value that is used when calculating the distance of a point  $(x, y)$ . In Figure 3, we compare general distance maps to our weighted distance map.

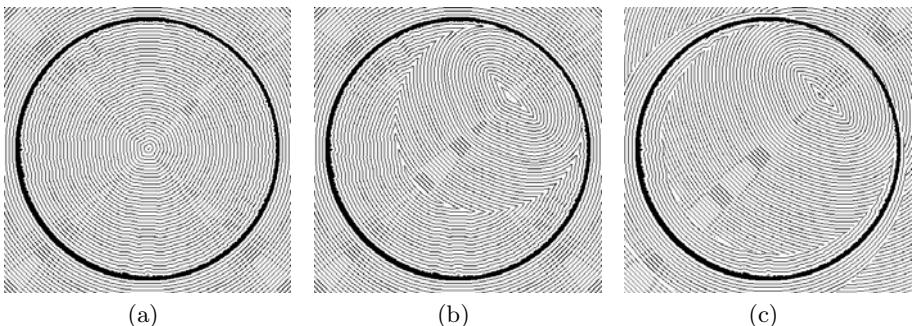
$$D_f(x, y) = \min_{x', y'} \left( \frac{1}{P_f(x', y')} \sqrt{(x - x')^2 + (y - y')^2} \right) \quad (2)$$

#### Offset Map Generation

Once a weighted distance map is generated, we create an offset map by using Equation 3, including an offset id for each pixel that is used as a constraint



**Fig. 3.** Weighted distance map. (a) shows a distance map using only edge as feature for Figure 2(a). (b) presents a distance map for our feature map (Figure 2(e)) without applying priority, whereas (c) shows our weighted distance map created with higher priority (5.0) for isophotes than that of edges.



**Fig. 4.** Offset maps. (a) shows an offset map generated from a distance map using only edges (Figure 3(a)). (b) shows an offset map from a distance map for our feature map, but without applying priority (Figure 3(b)). (c) shows an offset map created from our weighted distance map with higher priority (5.0) for isophotes (Figure 3(c)).

to guide the optimization of point distribution. To do this, we evenly space a weighted distance map, resulting in offset lines of a width  $w_o$  with an interval  $(l - w_o)$  as shown in Figure 4. Pixels located directly on a feature line comprise the first offset line ( $id_o = 0$ ), and the offset line is increased by one for such increment of  $w_o$  that a new line is placed from a feature line. In Equation 3, the range of the current offset line,  $[R_{min}, R_{max}]$  is computed with the width and interval of an offset line as well as the current offset id like  $R_{min} = ((l - w_o)id_o) - (w_o/2)$  and  $R_{max} = ((l - w_o)id_o) + (w_o/2)$ .

$$O(x, y) = (id_o + 1) \quad \text{if } (D_f(x, y)P_p(x, y) \in [R_{min}, R_{max}]) \quad (3)$$

As shown in Figure 4, when we use a small offset line width to tightly align points, relatively thick lines are created as the first offset lines if we use a distance

map generated with feature lines from a DoG filter. In this case, points may be loosely aligned to these first offset lines compared to other thin offset lines. This provides the effect of appealing emphasized feature lines in the final illustration. Alternatively, we can guarantee a tight alignment to all offset lines by using feature lines with width of 1.0. In this work, we used  $w_o=1$  and  $l=6$  for all result images.

### 3.3 Point Distribution

For stipple point distribution, we employ the constrained Lloyd relaxation algorithm proposed by Kim et al. [13] that aligns points to constrained areas (i.e., offset lines within an offset map). We begin with the initial points regularly distributed with an offset line interval, and then generate centroidal Voronoi diagrams (CVD) to distribute the initial points, i.e., the initial points become the centroids of corresponding Voronoi regions. During each iteration for optimization, we reconstruct Voronoi regions for each point while considering only pixels on offset lines, and update the centroids of corresponding Voronoi regions. This process makes the initial points (centroids) move toward the offset lines. In the case that a point does not have any pixels on offset lines within its Voronoi region, we remove the point because we do not want to position points inbetween offset lines. For details, see [13].

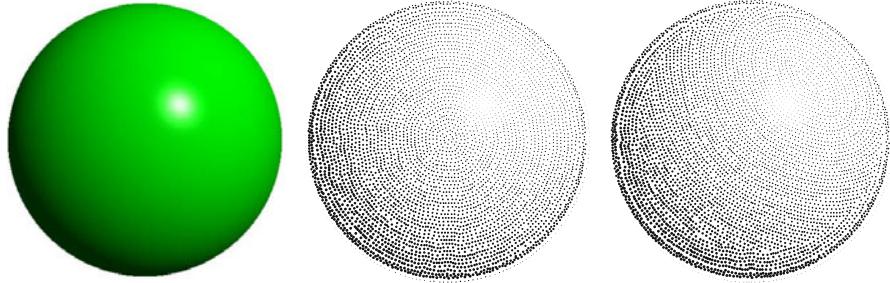
### 3.4 Rendering

Once the optimization is completed, we render the list of points optimized with circles for stippling illustration. To simulate tone changes in an input image, we slightly change the radius of circles by computing importance with the intensity  $\rho$  and gradient magnitude  $g$  of an input image, as shown in Equation 4

$$r_i = r_{max}(\alpha(1 - \rho_i) + \beta g_i)^{\frac{1}{\gamma}} \quad (4)$$

where,  $r_{max}$  is the maximum radius and  $\alpha$  and  $\beta$  are weight factors for darkness and gradient, respectively.  $\rho_i \in [0, 1]$  is an intensity value and  $g_i \in [0, 1]$  is the gradient magnitude of a pixel in an input image corresponding to the  $i^{th}$  point.  $\gamma$  is used for tone control. Figure 5 shows the results for the image of a 3D sphere. From this result, we see that the flow of stipples in Figure 5 (right) provides better sense of 3D shape compared to Figure 5 (middle).

Using the appropriate maximum size of a stipple considerably affects on the final quality. Accordingly, we use different  $r_{max}$  that depends on an input image and the number of points optimized, assuming that the average gray value of an output image,  $g_{avg}(I_{out}) \in [0, 1]$  approximates that of an input image,  $g_{avg}(I_{in}) \in [0, 1]$  with an appropriate constant  $\kappa$ , as shown in Equation 5. Based on this, we sum the areas of circles to compute the average gray value of an output image. Since an output image includes only black and white pixels, we



**Fig. 5.** Rendering a sphere image. (left) shows a 3D sphere image. (middle) is the result using constraints from only edge features, whereas (right) shows our result using both edges and isophotes as features.

obtain the average gray value by subtracting the ratio of black pixels from 1.0. Finally, as shown in Equation 6, we induce  $r_{max}$  from Equation 4 and 5

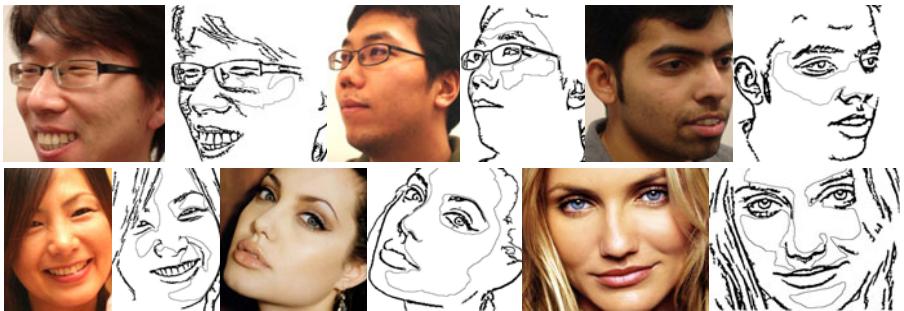
$$\begin{aligned} g_{avg}(I_{in}) &= \kappa g_{avg}(I_{out}) \\ &= \kappa \left( 1 - \frac{\sum_{i=0}^{N_p-1} \pi r_i^2}{wh} \right) \end{aligned} \quad (5)$$

$$r_{max} = \sqrt{\left( 1 - \frac{1}{\kappa} g_{avg}(I_{in}) \right) \frac{wh}{\pi \sum_{i=0}^{N_p-1} (\alpha (1 - \rho_i) + \beta g_i)^{\frac{2}{\gamma}}}} \quad (6)$$

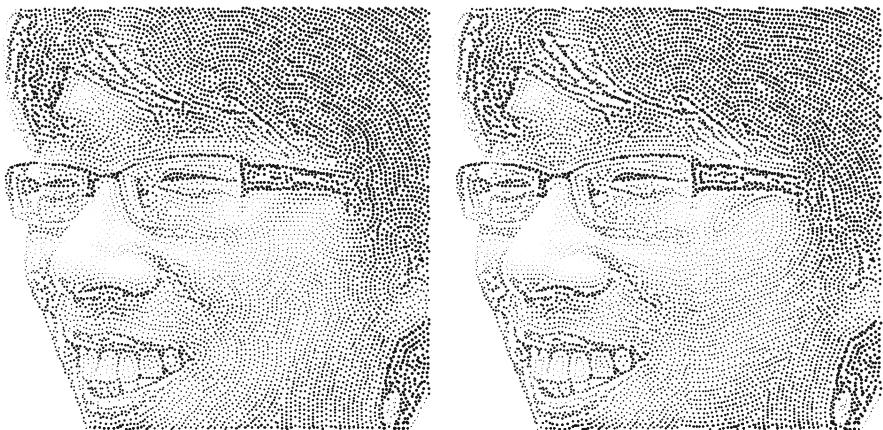
where,  $N_p$  is the number of points optimized, and  $\sum_{i=0}^{N_p-1} \pi r_i^2 \in [0, wh]$ .  $w$  and  $h$  are the width and height of an input image, respectively. In this work, we set the parameters for all the resultant images as follows:  $\kappa=1.0$ ,  $\alpha=0.6$ ,  $\beta=0.4$  and  $\gamma=1.3$ .

## 4 Results

To demonstrate our technique, we used several facial photographs as shown in Figure 6. For all results, we assigned isophotes to have higher priority (2.0 or 5.0) than that of edges (1.0) when generating our weighted distance maps. Particularly, using isophotes as features is effective when rendering large low-contrast areas where there are no edge features. For instance, see the cheek areas in Figure 6 (top row). For such areas, shading information plays an important role in perceiving a local shape. Feature maps, generated for each photograph, combining edge features and isophotes are also shown in Figure 6. Figure 7 compares the difference between results using only edge features and using both edges and isophotes. Compared to Figure 7 (left), Figure 7 (right) shows the shading flow created by isophotes on the cheek area. Figure 8 also demonstrates our results



**Fig. 6.** Original photographs and corresponding feature maps from our method, combining edges (thick) and isophotes (thin)



**Fig. 7.** Comparison between results (left) using only edge features and (right) using edges as well as isophotes with higher priority. See the differences of stipple flows on the cheek area.

with edges overlaid from original photographs in Figure 6. In these figures, stipbles are aligned to the flows that are created by edges and isophotes, and are stressed with circles of larger sizes on pixels with higher gradient magnitudes (edge features). By using a larger gradient weight factor in Equation 4, edges can be more emphasized in the final illustration.

We developed our technique on a PC with an Intel Xeon CPU 2.66GHz processor and 3GB of RAM without hardware acceleration. The performance of our work depends on the size of an input image and the number of stipple points. For example, it takes approximately 3.7 minutes with default parameters (mentioned in each section) for Figure 7 (right) of  $500 \times 477$  resolution, and the final illustration includes 9,386 stipbles. Most of the computation time is spent in

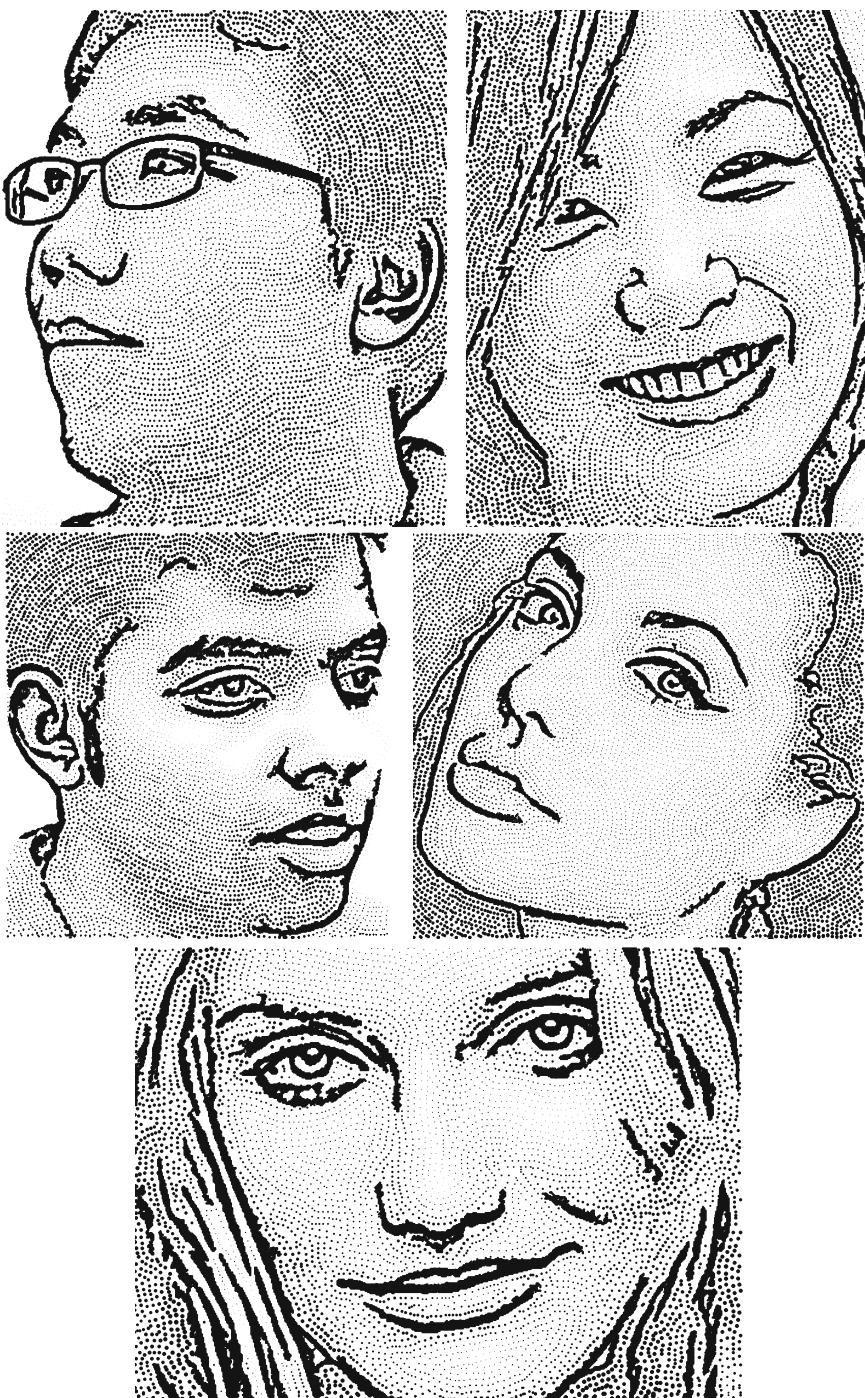


Fig. 8. Our results

creating the weighted distance map and optimizing the point distribution, while reconstructing Voronoi regions for each point (approximately 1.6 minute for creating a weighted distance map and 2.1 minutes for optimization for Figure 7 (right)). Our technique can be accelerated by using a linear time algorithm for a distance map and an improved algorithm for constructing Voronoi regions.

## 5 Conclusion and Future Work

We have presented an automated hedcut illustration technique that utilizes both edges and isophotes to create the constraints for the distribution of stippling. By utilizing isophotes and prioritizing them, our illustrations are able to present local features for the regions, particularly where edges do not exist. However, our technique still has several limitations. First, the quality of our illustrations depends on several factors such as the quality of features (e.g., details, smoothness, and connectivity), the width and interval of offset lines and the number of initial stipple points. Particularly, we need to select the appropriate width and interval of offset lines to avoid unpleasant holes in the final illustrations. Second, the priority for isophotes may be different according to objects and even positions on the same object since it represents the curvature of a surface. Even with these limitations, our method is able to improve on previous computer generated hedcut illustration work by preserving local shape through the inclusion of shading information.

## References

1. Balzer, M., Schlömer, T., Deussen, O.: Capacity-constrained point distributions: a variant of Lloyd’s method. *ACM Transactions on Graphics* 28(3), 1–8 (2009)
2. Chen, H., Liu, Z., Rose, C., Xu, Y., Shum, H., Salesin, D.: Example-based composite sketching of human portraits. In: Proc. of the 3rd International Symposium on Non-photorealistic Animation and Rendering, pp. 95–153 (2004)
3. Cole, F., Sanik, K., DeCarlo, D., Finkelstein, A., Funkhouser, T., Rusinkiewicz, S., Singh, M.: How Well Do Line Drawings Depict Shape? *ACM Transactions on Graphics* (Proc. SIGGRAPH) 28(3), 1–9 (2009)
4. Deussen, O., Hiller, S., Van Overveld, C., Strothotte, T.: Floating Points: A Method for Computing Stipple Drawings. *Computer Graphics Forum* 19(3), 40–51 (2000)
5. Dragnea, V., Angelopoulou, E.: Direct Shape from Isophotes. In: Proceedings of the ISPRS Workshop BenCOS (2005)
6. Durand, F.: Perceptual and Artistic Principles for Effective Computer Depiction. In: SIGGRAPH 2002: ACM SIGGRAPH 2002 Course Notes (2002)
7. Fleming, R.W., Singh, M.: Visual perception of 3D shape. In: SIGGRAPH 2009: ACM SIGGRAPH 2009 Courses, pp. 1–94 (2009)
8. Gooch, B., Reinhard, E., Gooch, A.A.: Human Facial Illustrations: Creation and Psychophysical Evaluation. *ACM Transactions on Graphics* 23(1), 27–44 (2004)
9. Goodwin, T., Vollick, I., Hertzmann, A.: Isophote distance: a shading approach to artistic stroke thickness. In: Proc. of the 5th International Symposium on Non-photorealistic Animation and Rendering, pp. 53–62 (2007)

10. Hiller, S., Hellwig, H., Deussen, O.: Beyond Stippling – Methods for Distributing Objects on the Plane. *Computer Graphics Forum* 22(3), 515–522 (2003)
11. Isenberg, T., Neumann, P., Carpendale, S., Sousa, M.C., Jorge, J.A.: Non-Photorealistic Rendering in Context: An Observational Study. In: Proc. of the International Symposium on Non-photorealistic Animation and Rendering, pp. 115–126 (2006)
12. Kang, H., Lee, S., Chui, C.K.: Coherent Line Drawing. In: Proc. of the 6th International Symposium on Non-photorealistic Animation and Rendering, pp. 43–50 (2007)
13. Kim, D., Son, M., Lee, Y., Kang, H., Lee, S.: Feature-guided Image Stippling. *Computer Graphics Forum* 27(4), 1029–1216 (2008)
14. Kim, S., Maciejewski, R., Isenberg, T., Andrews, W.M., Chen, W., Sousa, M.C., Ebert, D.S.: Stippling by example. In: Proc. of the 7th International Symposium on Non-Photorealistic Animation and Rendering, pp. 41–50 (2009)
15. Kopf, J., Cohen-Or, D., Deussen, O., Lischinski, D.: Recursive Wang Tiles for Real-Time Blue Noise. *ACM Transactions on Graphics* 25(3), 509–518 (2006)
16. Luo, Y., Marina, L., Sousa, M.C.: NPAR by Example: Line Drawing Facial Animation from Photographs. In: 2006 International Conference on Computer Graphics, Imaging and Visualisation, pp. 514–521 (2006)
17. Maciejewski, R., Isenberg, T., Andrews, W.M., Ebert, D.S., Sousa, M.C., Chen, W.: Measuring Stipple Aesthetics in Hand-Drawn and Computer-Generated Images. *IEEE Computer Graphics & Applications* 28(2), 62–74 (2008)
18. Mould, D.: Stipple Placement using Distance in a Weighted Graph. In: Proc. of the International Symposium on Computational Aesthetics in Graphics, Visualization, and Imaging, pp. 45–52 (2007)
19. Mould, D., Grant, K.: Stylized black and white images from photographs. In: Proc. of the 6th International Symposium on Non-photorealistic Animation and Rendering, pp. 49–58 (2008)
20. Redmond, N., Dingliana, J.: Adaptive Abstraction of 3D Scenes in Realtime. In: Eurographics 2007 Short Papers, pp. 77–80 (2007)
21. Sára, R.: Isophotes: The Key to Tractable Local Shading Analysis. In: Hlaváč, V., Šára, R. (eds.) CAIP 1995. LNCS, vol. 970, pp. 416–423. Springer, Heidelberg (1995)
22. Schlechtweg, S., Germer, T., Strothotte, T.: RenderBots—Multi Agent Systems for Direct Image Generation. *Computer Graphics Forum* 24(2), 137–148 (2005)
23. Secord, A.: Weighted Voronoi Stippling. In: Proc. of the International Symposium on Non-photorealistic Animation and Rendering, pp. 37–43 (2002)
24. Winnemöller, H., Olsen, S.C., Gooch, B.: Real-time video abstraction. In: SIGGRAPH 2006: ACM SIGGRAPH 2006 Papers, pp. 1221–1226 (2006)

# **Articulate: A Semi-automated Model for Translating Natural Language Queries into Meaningful Visualizations**

Yiwen Sun, Jason Leigh, Andrew Johnson, and Sangyoon Lee

Electronic Visualization Laboratory, University of Illinois at Chicago  
{ysun25, spiff, ajohnson, slee14}@uic.edu

**Abstract.** While many visualization tools exist that offer sophisticated functions for charting complex data, they still expect users to possess a high degree of expertise in wielding the tools to create an effective visualization. This paper presents *Articulate*, an attempt at a semi-automated visual analytic model that is guided by a conversational user interface to allow users to verbally describe and then manipulate what they want to see. We use natural language processing and machine learning methods to translate the imprecise sentences into explicit expressions, and then apply a heuristic graph generation algorithm to create a suitable visualization. The goal is to relieve the user of the burden of having to learn a complex user-interface in order to craft a visualization.

**Keywords:** visual analytics, natural language processing, conversational interface, automatic visualization.

## **1 Introduction**

Much has been investigated on the design of sophisticated visual analytic tools in a variety of disciplines. However, the effort end-users have to make to craft a meaningful visualization using these tools has been mostly overlooked. The users of such tools are usually domain experts with marginal knowledge of visualization techniques. When exploring data, they typically know what questions they want to ask, but often do not know, or do not have the time to learn, how to express these questions in a form that is suitable for a given analysis tool, such as specifying a desired graph type for a given data set, or assigning proper data fields to certain visual parameters. To facilitate the use of advanced visualization tools by domain experts, we propose a semi-automated visual analytic model: *Articulate*. The goal is to provide a streamlined experience to non-expert users, allowing them to focus on using the visualizations effectively to generate new findings.

Survey results according to search engines like Ask.com show that a third of search queries are entered as natural language questions rather than keywords [1]. Furthermore, a 2007 National Science Foundation workshop report on “Enabling Science Discoveries through Visual Exploration” [2] noted that “there is a strong desire for conversational interfaces that facilitate a more natural means of interacting with science.” Scientists frankly do not have the time or patience to learn complex visualization tools. Zue [3]

also pointed out that spoken language is attractive in interface design, because it is the most natural, efficient, flexible, and inexpensive means of communication. It is also a skill that humans have been using for over a hundred millennia as compared to computers which have only become widely available since the mid-80s. And while computing software and user-interfaces will become obsolete over time, the use of spoken dialog as the primary form of human communication is unlikely to become obsolete. This inspired us to adopt a conversational interface in the semi-automated visual analytic model. A model like this would allow an end-user to pose natural language inquiries, and then let the system assume the burden of determining the appropriate graph, and presenting the results. It is hoped that such a capability can potentially reduce the learning curve necessary for effective use of visual analytics tools, and thereby expanding the population of users who can successfully conduct visual analyses.

In this paper, we propose a two-step process to translate a user's verbal description to a representative visualization: first, parse the imprecise queries into explicit commands using natural language processing and machine learning methods; then, determine an appropriate type of visualization based on the commands and data properties automatically. In this initial work we limit the types of visualizations that can be generated to standard 2D graphs and plots. However in the future we fully intend to extend this to accommodate complex visual representations such as those commonly used in scientific visualization (such as volumetric or streamline visualizations).

The primary contributions of this research include: 1) the incorporation of a conversational interface and natural language parser to allow for natural language input rather than grammar based commands; 2) the development of an algorithm to automatically generate graphs based on the classification of visual analytic tasks; 3) the development of a Simplified Visualization Language (SimVL) as an intermediate expression for a user's specification, which is precise, and easy to convert in a representative graph.

The remainder of the paper is organized as follows. We discuss related work in Sect. 2. In Sect. 3, we introduce the natural language parser and the proposed algorithm for graph generation. The implementation details of the system are presented in Sect. 4. Then in Sect. 5, we present a preliminary user study and its findings. Finally, we conclude in Sect. 6, where we outline directions for future work.

## 2 Related Work

In the last decade several approaches for automatic visualization have been proposed. One of the first was Show Me [4], an integrated set of user interface commands and defaults that automatically generate visual presentations based on the VisQL specification language. Users place data fields into columns and rows in the interface panel to specify VisQL commands. In order to generate insightful visualizations, an understanding of the relationships between columns and rows is needed. Rezk-Salama et al. demonstrated a semantic model for automatic transfer function editing in volume rendering applications [5]. However the design of the semantic model is such that the presence of computer scientists and domain experts are needed when selecting meaningful semantic parameters and mapping parameters to low-level primitives.

Another interesting approach is VisMashup [6], which simplifies the creation of customized visualization applications with a pipeline. Once the pipeline is constructed, the application can be generated automatically. While this infrastructure enables an application designer to assemble custom applications quickly, the designer still needs some visualization background to build up the pipeline from components or templates. Although *Articulate* shares some of these same goals, our approach goes a step further by allowing the users to verbally articulate what they want to see with minimal apriori knowledge of how to use the user-interface.

A number of speech-based interfaces have been developed that help users to access information using a conversational paradigm. JUPITER [7] for example allows users to obtain worldwide weather forecast information over the phone using spoken dialog. It is a mixed initiative system [3] that requires zero user training, and accepts a large range of user inputs. This approach has been extended to similar domains where the vocabulary is sufficiently limited to support practical conversational interface, such as travel planning [8], health information access [9]. But few of the systems we surveyed targeted the problem of visual analytics.

Cox et. al.'s work [10] was the first to integrate a natural language interface into an existing information visualization system. Cox's work takes the natural language question, determines the appropriate database query, and presents the results to the user. Results from a pilot user study indicated a considerable need for natural language interface to aid users in data analysis. However, the number of questions supported by the system was small and mostly grammar dependent. More importantly, in the result generation phase, they did not take the advantage of varied plot types based on the characteristics of different analytic tasks, but instead only utilize tables and bar charts.

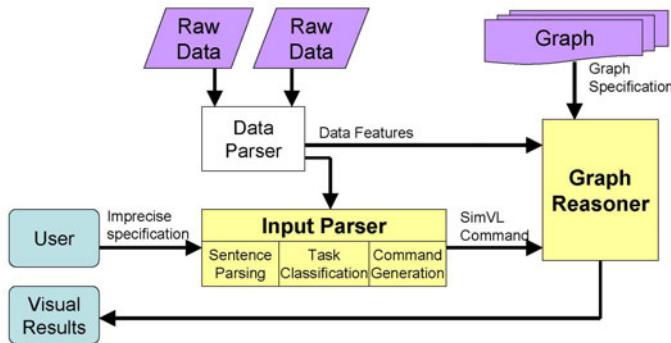
Though the idea of using a conversational interface to craft visualizations is not new, our work addresses a growing trend toward in this area, as witnessed in the recent work of Wolfram Research [11].

### 3 Design and Methodology

The purpose of the *Articulate* system is to process a user's imprecise description and generate a meaningful visualization that best answers their question. Figure 1 gives an overview of our model.

To illustrate how the model works, let us look at an example. Suppose a user loads in a dataset about hydrologic information, and makes a query: "*How was conductivity related with depth*". The data parser will first read the original data file collecting information such as attribute names, data types, and matching the words *conductivity* and *depth* to existing attributes. Meanwhile, the input parser will interpret the sentence as a request for a relationship graph with two attributes assigned as the x and y axes. This interpretation is expressed as SimVL commands and entered into the graph generator, where information about the data from the data parser and the intermediate commands are funneled. Properties of various graph types are also funneled in via the graph specification. The reasoner uses this information to determine that the most appropriate visualization will be a scatter plot.

The essential parts of our framework are two steps: input parser and graph reasoner. In Sect. 3.1 to 3.3 we describe how the imprecise query is interpreted, followed by the algorithm for graph generation in Sect. 3.4.



**Fig. 1.** Overview of the *Articulate* System

### 3.1 Parsing

The translation of user's imprecise specification is based on a natural language parser imbued with machine learning algorithms that are able to make reasoned decisions automatically. User's initial input to the system is a natural language query. The query sentence is parsed into a sequence of words tagged with part-of-speech labels using the Stanford Parser [12]. These labels mark the lexical category for each word, such as noun, verb, adjective, adverb, preposition, and etc. based on both its definition, as well as its context. For example, the sentence "*How was conductivity related with depth*" will be tagged as:

*how/WRB; was/VBD; conductivity/NN; related/VBN; with/IN; depth/NN*

Using these tags, we can distinguish the functions of each word and apply different rules based on that. In addition, the stem of each word, i.e. the root of the word, is also extracted. The stemmed result for the previous example is shown below:

*how be conductivity relate with depth*

Compared with the original sentence, the difference is all about the tense of verbs: "was" is stemmed as "be", "related" is stemmed as "relate". The stems avoid the morphological ambiguity. For example, relate, relating, related all have the same root, and should be recognized as the same keyword in the classification step.

### 3.2 Classification

Based on the parsing results, we map the query into a smaller feature space, and apply a supervised learning method in this space to predict the class of the task.

The feature space is defined as a nine-dimensional space. Each dimension describes one feature of the query, such as the existence of keywords for one class, or the number of attribute names appearing in the query. Specifically, the features are:

- comparison\_keyword [true/false]
- relationship\_keyword [true/false]
- composition\_keyword [true/false]
- distribution\_keyword [true/false]
- statistics\_keyword [true/false]
- manipulation\_keyword [true/false]
- timeseries\_keyword [true/false]
- visual\_parameter\_keyword [true/false]
- number\_of\_attributes [0, 1, 2, 3]

The first eight features all have Boolean values: “true” if there is a word in the query matching a keyword in the corresponding dictionary, “false” if not. The keywords in each dictionary are selected according to empirical knowledge. For example, *associate*, *correlate*, *link*, *relate*, *relevant* are often used in the queries intended for relationship or connection between two or more variables, so they are entered into the relationship dictionary. In addition, the entries in the dictionary are grouped by their lexical category, i.e. noun, verb or adjective. The matching between the stemmed words and the entries in a dictionary is done by a string comparison. Instead of simple word matching, the algorithm first checks the lexical category in the dictionary compared with the part-of-speech tag on the stemmed word, if they are similar then checks whether the two strings are identical. This way certain lexical ambiguity caused by words with multiple function categories can be avoided.

The last feature can have multiple values: 0 represents no attribute names appeared in the query, 1 represents only one attribute, 2 represents two attributes, 3 represents more than two. For calculation of this feature, the original words not the stems are evaluated with existing attribute names. Because in this scenario, each attribute name is regarded as a special property noun and usually appears as a column in the data file, it has to keep its surface form, but stemming may change the form of a word, for example, “women” will be stemmed as “woman”, which will bias the comparison result.

After extracting the feature vector, we use a decision tree learning method to classify the query. Inspired by Abela’s chart chooser [13], we identify seven classes: comparison, relationship, composition, distribution, statistics, manipulation, and other. The first six classes are chosen based on the characteristics of different visual analytics tasks. For example, the relationship task focuses on discovering the correlation between two or more attributes; the comparison task often aims at exploring the trend over time or categories. The decision tree is built upon a set of rules obtained by applying labeled training datasets. Upon receiving the feature vector, the tree will calculate the expected values of competing alternatives. The one with the highest probability is the predicted class of the query. If “other” is the predicted class, which means the current system does not have a solution for that, or could not decide on which class the query falls in closely, then a clarification message will be presented to the user. For example if the user asked “*which car should I buy?*” the system will response with “*I’m not sure how to answer that. Can you try asking that in a different way?*”

### 3.3 SimVL Command Generation

To pass the classification results as well as the specified attributes to graph reasoner precisely, we propose a Simplified Visualization Language (SimVL). SimVL is specified in a formal grammar, in which each command corresponds to a specific assignment. The purpose of using SimVL is to provide a standard and simplified format for user's request in order to facilitate the graph reasoning. To accommodate the different purposes of visual analytics, this language is divided into three major categories: sketch command, analysis command and manipulation command.

**Sketch commands** are the ones that describe the semantics of some general visualization task. The grammar of this command is presented below:

$$\begin{aligned} < \text{sketch} > ::= & \text{PLOT } < \text{class\_of\_task} > \\ | & \text{PLOT } < \text{class\_of\_task} > \text{ OVERTIME} \end{aligned}$$

As shown above, a statement is composed of an action with one or two parameters. The first parameter indicates the classified visual analytic task type, including RELATIONSHIP, COMPARISON, COMPOSITION and DISTRIBUTION. The second parameter indicates whether time series data is required.

**Analysis commands** are normally used when the user is interested in the statistical feature of data. For example, minimum, maximum, mean, median and etc. so the commands are defined as an action followed by two parameters: one indicates the feature, which can be MIN, MAX, MEAN, MEDIAN, RANGE; the other lists the attribute names:

$$< \text{analysis} > ::= \text{ANALYSE } < \text{feature} > \text{ OF } < \text{attribute\_list} >$$

**Manipulation commands** are used to alter some common visual metaphors, such as axis, size and color of data points.

$$\begin{aligned} < \text{manipulation} > ::= & \text{SETX } < \text{attribute\_list} > \\ | & \text{SETY } < \text{attribute\_list} > \\ | & \text{SET SIZE\_OF\_POINT TO } < \text{attribute} > \\ | & \text{SET COLOR\_OF\_POINT TO } < \text{attribute} > \end{aligned}$$

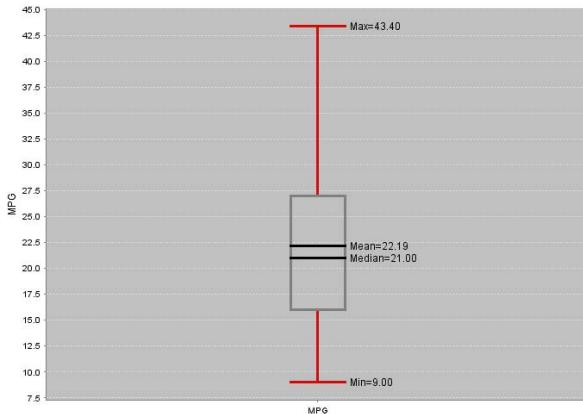
Similar to the sketch commands, they are also defined as a list of statements. However, these commands do not enforce the generation of a new graph, but focus on the mapping or assignment of visual metaphors.

### 3.4 Graph Reasoner

The final phase is the generation of the graph. Each plot type has its own advantages and disadvantages for certain types of data. For example: bar charts are often used for comparing values by category, whereas pie charts are good at illustrating relative magnitudes or frequencies. Just as a visualization expert might weigh up the pros and cons of different graphs in the determination of a desired plot type, the graph reasoner works as a smart agent carrying out a similar reasoning process autonomously. The rules

of reasoning can be divided into three categories corresponding to the three SimVL command types.

Analysis commands usually focus on the statistical features of data. So a box-and-whisker chart [14] is a convenient way of graphically depicting these features: the ends of the whisker represent the minimum and maximum of the data, the bottom and top of the box are always the 25th and 75th percentile; while, the line in the center of the box can be mean or median based on user's request. The spaces between the different parts of the box indicate the dispersion in the data. Figure 2 shows an example.



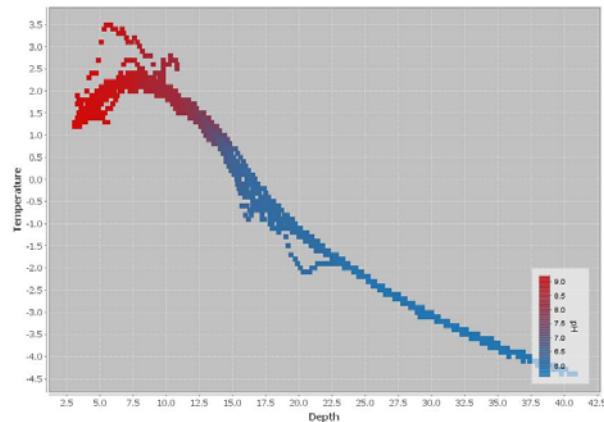
**Fig. 2.** Result for analysis commands translated from “what is the range of MPG”

Manipulation commands are typically follow-ups from a previous graph, such as switch axes, change color of point based on values of another attribute. Hence, the reasoner only needs to recognize the visual metaphor from the SimVL command, and map the specified attribute onto that. Figure 3 gives an example.

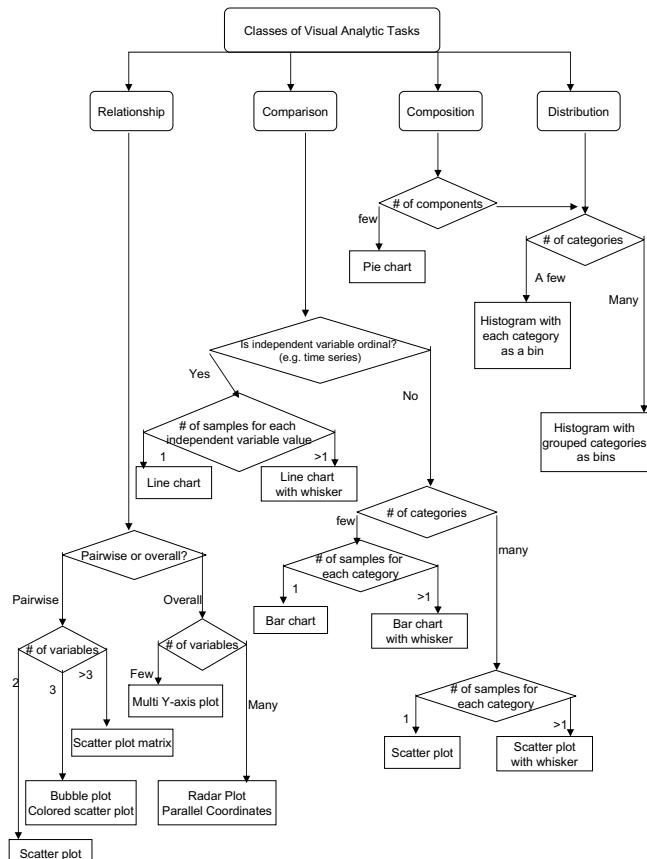
The most complicated part is the sketch commands. Figure 4 illustrates the algorithm of how the decision for the final graph is formed for these commands. There are generally four sub-algorithms for the four classes in sketch commands. Let us look at an example, given a car dataset, a query “how has MPG changed over the years” will be translated as SimVL commands:

```
PLOT COMPARISON OVERTIME
SETY MPG
```

The first command indicates the query is a comparison task, so we will follow the comparison sub-algorithm. It then checks whether the independent variable is ordinal. Since there is an OVERTIME parameter in the command, the independent variable will be year, thus the answer for this test is “Yes”. Next step, it will look into the dataset to find whether there is a unique *MPG* value on each year. Unfortunately it does not hold. So the final graph will be a line chart with whisker (as shown in Fig. 5). Similar to the box-and-whisker plot, the top and bottom ends of whiskers indicate the maximum and minimum *MPG* values for each year.



**Fig. 3.** Result for manipulation commands translated from “can you color by pH” following a query “what is the correlation between depth and temperature”



**Fig. 4.** Algorithm of the graph generation for sketch commands



**Fig. 5.** Result for sketch commands translated from “*how has MPG changed over the years?*”

## 4 System Implementation

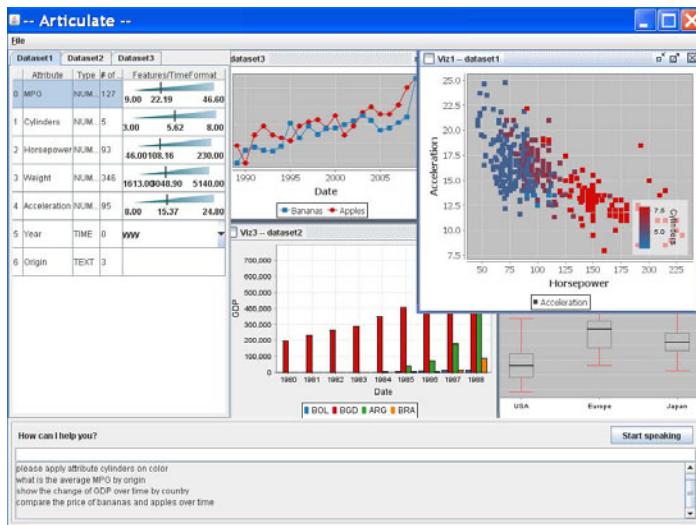
The *Articulate* system is developed in Java. Users can input their queries by speech or text. To recognize speech we use Chant Speech Kit. Based on Microsoft SAPI, this kit translates a spoken query into a plain-text sentence. The structure of the sentence is parsed using the Stanford Parser and outputted as words with part-of-speech tags. The techniques used for query classification are based on the C4.5 system [15], which is a decision tree learning algorithm. The last component of the system is the graph engine. We adopt JFreeChart, a free Java chart library that supports a wide range of chart types. In the visualization panel, traditional mouse interaction is also supported, such as zooming and brushing, tooltips on pointing, as a complement to natural language interface for those types of tasks that are easy to perform with a GUI but cumbersome to speak. In the input panel, there is a query history window which maintains all the previous inquiries allowing users to verify if the system has correctly interpreted their natural language input (Fig. 6).

## 5 Evaluation and Results

To explore the limits of the *Articulate* system, we conducted a preliminary user study with eight graduate students. All of them are computer science majors and have some experience with visualization tools. Each of them was presented with one or two of the following datasets: a hydrologic dataset which contains 10 attributes and 4000 instances; Cars dataset found from XmdvTool website, with 8 attributes for 406 different cars; and Average U.S. Food Prices dataset which contains prices for eight different kind of food over 11 years, published by U.S. Bureau of Labor Statistics.

In this study, users were asked to interact with the *Articulate* system by expressing natural language queries to finish three tasks within 20 minutes:

- Find meaningful correlations among the data attributes.
- Find as many trends as possible about the data.
- Find other interesting features about the data.



**Fig. 6.** Graphical user interface for the *Articulate* system. The bottom of the interface is where the user's spoken queries are displayed. The individual graphical windows depict the visualizations created as a result of the queries.

After each query, users were asked to identify their purpose of their query based on six categories: comparison, relationship, composition, distribution, analysis, manipulation. A comparison of their choice and the resulting classification is shown in Table 1.

**Table 1.** Results for classification. Numbers for comparison, relationship, composition and distribution categories are merged into a sketch category.

Purpose	Correctly Classified	Incorrectly Classified
Sketch	50 (81%)	12 (19%)
Analysis	20 (87%)	3 (13%)
Manipulation	9 (90%)	1 (10%)

All subjects in our study were able to use the current system to find some data features and trends. But as shown in the table, for each category of query there is an average of 14% classification error. To discover the cause of this, we looked through each incorrectly classified query, and made a couple of interesting findings:

#### Finding 1: Support for refined or follow-up queries is needed

For example, “what are the other attributes around depth of 7.5”, “what is the heaviest American car”. These queries are usually follow-ups from sketch questions. In the first example, a certain value or a value range is specified, like a filter in the database operation. In the second example, the query contains superlative adjective *heaviest* referring to the attribute *weight*. To link the comparative or superlative adjective to a data attribute properly, further knowledge about the data will be needed.

Our current conversational interface is more of a user-initiative system in which the user has freedom in what they say to the system, while the system remains relatively passive, asking only for clarification. As Zue et. al.[3] pointed out this may lead “the user to feel uncertain as to what capabilities exist, and may, as a consequence, stray quite far from the domain of competence of the system”. To avoid such situations, a more active feedback from the system will be needed, for example suggesting related queries as well as associated results to help the user find their solution.

### **Finding 2: Metadata is needed for providing context to parsing**

Take “*compare the price of meat*” for example. *Meat* actually refers to *beef* and *chicken* in the Average U.S. Food Prices dataset. Correct recognition of such terms requires knowledge about synonyms or hypernyms of the attribute names. One solution might be using WordNet[16] to find Synset for each attribute. This also suggests in the creation of a database or table, each attribute accompanied with a meta-field briefly explaining the term should be preferable.

Another field needed in the metadata will be the Units of the data. Unknown data units can confuse the query classification. For example, “*compare the price of apples with tomatoes*” versus “*compare cylinders with horsepower*”. In the first example, *apples* and *tomatoes* have the same data unit (price), so put their values both on the y axis and use *years* as the x axis is more desirable than plotting them on x, y axes respectively. But in the second example, the two attributes *cylinders* and *horsepower* have unmatched units, user would expect them plotted as x and y axes separately.

Next, we repeated the study using Microsoft Excel. Eleven graduate students participated in this study. All the subjects have used Excel before and are familiar with its basic capabilities. Two of them were familiar with Excel’s graphing features, eight had occasional experience with them, and one had no experience. Each of the subjects were given six queries, which were picked from the correctly classified queries in the first study. The subjects were asked to plot the results using Excel. After that we asked the subjects for their opinions on the resulting Excel charts. We found that one user liked Excel’s result, six of them felt the results were acceptable though not ideal, and four of them found the charts completely inappropriate. Most subjects found the steps needed to create a chart in Excel to be complex, and the means for selecting data ranges and series, confusing. Furthermore, we found that at least half of the subjects that used Excel had to create more than one chart and call additional Excel functions (such as sort, min, max) to describe a query that otherwise could have been expressed with a single sentence in *Articulate*. Lastly, subjects on average took twelve times longer to describe a query in Excel than in *Articulate*— which typically took less than a minute. These initial results were encouraging, and we intend to conduct a larger study with domain scientists in the future.

## **6 Conclusions and Future Work**

In this paper, we presented *Articulate*, a novel visual analytic approach that utilizes a natural language interface to translate imprecise verbal descriptions into meaningful visualizations. We integrated natural language processing and automated graph generation algorithm to make the implicit sentence semantics explicit in the final representation.

We believe this approach has the potential to help scientists as well as laypeople, including educators and policymakers, to quickly produce effective visualizations without extensive knowledge of visualization techniques and/or tools.

Future directions in this research will include more active reaction from the system, such as suggestion of related queries in response to unclassified questions; support for a wider range of manipulation commands, including sort, pick; and the evaluation of the system using populations that are non-computer scientists. Furthermore we hope to extend this approach to also encompass more advanced visualization techniques such as those commonly used in scientific visualization (for example, the visualization of volumetric data as isosurfaces, or vector data as streamlines).

**Acknowledgments.** This project was funded in part by National Science Foundation grants CNS-0703916 and CNS-0420477.

## References

1. Silicon Republic News, <http://www.siliconrepublic.com/news/article/14758/randd/googles-top-inventor-says-talking-computers-are-the-future>
2. Ebert, D., Gaither, K., Gilpin, C.: Enabling science discoveries through visual exploration. In: NSF Workshop report, Washington, D.C (2007)
3. Zue, V., Glass, J.R.: Conversational Interfaces: Advances and Challenges. Proceedings of the IEEE, 1166–1180 (2000)
4. Mackinlay, J.D., Hanrahan, P., Stolte, C.: Show me: Automatic presentation for visual analysis. IEEE Trans. on Visualization and Computer Graphics 13, 1137–1144 (2007)
5. Salama, C.R., Keller, M., Kohlmann, P.: High-level user interfaces for transfer function design with semantics. IEEE Trans. on Visualization and Computer Graphics 12, 1021–1028 (2006)
6. Santos, E., Lins, L., Ahrens, J., Freire, J., Silva, C.: VisMashup: Streamlining the Creation of Custom Visualization Applications. IEEE Trans. on Visualization and Computer Graphics 15, 1539–1546 (2009)
7. Zue, V., Seneff, S., Glass, J.R., Polifroni, J., Pao, C., Hazen, T.J., Hetherington, L.: JUPITER: a telephone-based conversational interface for weather information. IEEE Trans. on Speech and Audio Processing 8, 85–96 (2000)
8. Seneff, S., Polifroni, J.: Dialogue management in the Mercury flight reservation system. In: ANLP/NAACL 2000 Workshop on Conversational systems, pp. 11–16 (2000)
9. Sherwani, J., Ali, N., Tongia, R., Rosenfeld, R., Memon, Y., Karim, M., Pappas, G.: Health-Line: Towards Speech-based Access to Health Information by Semi-literate Users. In: Proc. Speech in Mobile and Pervasive Environments, Singapore (2007)
10. Cox, K., Grinter, R.E., Hibino, S.L., Jagadeesan, L.J., Mantilla, D.: A Multi-Modal Natural Language Interface to an Information Visualization Environment. J. of Speech Technology 4, 297–314 (2001)
11. Wolfram Research, <http://www.wolframalpha.com>
12. The Stanford Parser, <http://nlp.stanford.edu/software/lex-parser.shtml>
13. Abela, A.: Advanced Presentations by Design: Creating Communication that Drives Action. Pfeiffer (2008)
14. Tukey, J.W.: Exploratory Data Analysis. Addison-Wesley, Reading (1977)
15. Quinlan, J.R.: C4.5: Programs for Machine Learning. Morgan Kaufmann, San Francisco (1993)
16. WordNet, <http://wordnet.princeton.edu/>

# Visual Analysis of Time-Motion in Basketball Games

Roberto Therón and Laura Casares

University of Salamanca

**Abstract.** This work aims to facilitate the task of basketball coaches, by means of the visualization and analysis of the players' movements in the court. This is possible thanks to the use of Global Positioning System (GPS) devices that generate data of the position of the player, almost in real time. The main objective of our proposal consists on the tracking, both statistics and kinematics, of a basketball player due to the physical activity developed during a match. The comparison of the data from several players or between two teams also will improve the performance and tactical capacity of players and trainers.

On one hand, the study of time-motion in sports is largely covered in the literature. On the other hand, the use of personal GPS devices for training purposes is a common practice. However, the design of interactive visualization tools that exploit the data stored in GPS devices during a match, thus enabling to perform its visual analysis, is still an open area. The work presented in this paper identifies the relevant aspects of the basketball game that are valuable for a coach in terms of team and individual performance analysis, and discusses the design and implementation of a tool that exploits the methods and techniques of a visual analytics approach.

## 1 Introduction

Today, one of the main features of a basketball coach is the visual memory. They often have to be able to remember plays and game situations for further analysis that will structure his team's tactics. Knowing what works and what fails becomes a great mental exercise whose source is the visual memory of the coach (also supported by game and training video recordings). Typically, to make this work, and only available after the game, the coach analyses the statistics of the two teams regarding converted baskets, personal fouls, free throws and the like.

Our proposal aims to facilitate the work of coaches and advance the study of basketball, offering a chance to see and analyse the movements that the players have previously made in the field of play, and their implications. In addition, we aim to provide both statistical and kinematic analysis of the data collected for each of players to facilitate monitoring over several physical exercises. The main data sources are individual wearable Global Positioning System (GPS) devices, that enable the collection of real time data related to the position of each player in the court during the whole match or training exercise.

Comparing data from several players or two teams are also aspects that will improve the performance of athletes through training.

The rest of the paper is organized as follows: Section 2 is devoted to a brief review of the current state of the research related to time-motion analysis in sports, its common

methods and the opportunities that current technologies (GPS, visual analytics) have to offer to this field. Section 3 presents the requirements for basketball game analysis from a coach's viewpoint. Section 4 presents our visual analytics approach for basketball's time-motion analysis. Finally, Section 5 contains the conclusions derived from the work and exposes the future lines of research in which we are working now.

## 2 Background

Time-motion analysis (TMA) is a standard analytical procedure to determine the time and energy invested in an activity for a period of time [5]. Through this process the various patterns of movement involved in sport situations, such as speeds, durations or distances are collected and tallied [3]. Thus, we obtain valuable information on the use of energy systems, and on specific movement patterns of each sport [18]. In particular, TMA has been used in several disciplines, such as rugby [13] [11], football [6] [14] [10], hockey [9] [8], and basketball [12] [18].

The first aspect to take into account in TMA is the method used to acquire the data to be analysed. Once the data is stored and prepared, the analysis methodology itself has to be chosen. In the following we discuss the adequacy of GPS devices (as data source) and visual analytics (as analysis approach) that we have chosen to be used in our proposal.

### 2.1 Methods for Time-Motion Analysis

There are basically two ways to get data from TMA, video systems, which are by far the most used [3], and GPS devices. Regarding the former, using multiple cameras situated at a height and distance of variable pitch (in the case of basketball and other sports with small courts, like tennis, the height is between 1 and 5 m and the distance from the court between 2 and 10 m).

The second approach has been less used so far as accuracy was quite low (in the case of positions, several meters of error). However, thanks to the development of differential GPS (d-GPS), using one or more additional reference stations on the ground (only satellite reference stations are used in traditional GPS), the accuracy increases considerably [22].

Furthermore, there is an ongoing effort to upgrade the Global Positioning System<sup>1</sup> with new, advanced capabilities to meet growing military, civil, and commercial needs. The program is being implemented through a series of satellite acquisitions, including GPS Block IIR-M, GPS Block IIF, GPS Block III, and the Next-Generation Operational Control Segment (OCX). Three new signals have been designed for civilian use: L2C (the second civilian GPS signal, designed specifically to meet commercial needs; when combined with L1 C/A in a dual-frequency receiver, L2C enables ionospheric correction, a technique that boosts accuracy), L5 (the third civilian GPS signal, broadcast in a radio band reserved exclusively for aviation safety services), and L1C (the fourth civilian GPS signal, designed to enable interoperability between GPS and international satellite navigation systems, i.e., Galileo). This establishes a panorama in which the

---

<sup>1</sup> <http://www.space.commerce.gov/gps/modernization.shtml>

research related to the design of tools that best exploit GPS devices is much needed, which is the case of our problem at hand.

Thus, GPS technology has been adapted in recent years to serve as a training tool, improving portability and generating useful data for the athlete (distance travelled, speed, etc.). But otherwise operate as a normal GPS, generating data at 1Hz (once per second). Generally it has been used in sports' training covering long distances, such as walking or cycling. This breakthrough technology has also resulted in literature related to general human locomotion [19].

In both methods, in addition to the data that pertain to a typical TMA analysis, biological data, such as the athlete's heart rate, can also be monitored.

Currently the affordability of existing GPS wrist devices it is possible to have the 10 outfield players wearing one in a basketball game. These devices, despite having a margin of error, open the door for research into new technologies to improve the performance of top athletes, providing solutions for the particular case of basketball and for most team sports in which the precision errors are less noticeable.

## 2.2 What Does Visual Analytics Have to Offer?

One key aspect is that TMA usually generate large amounts of data. For example, the collection, every second, of the position for all the players of both teams in a basketball game would generate more than 20,000 positions, excluding data on speed, direction, heart rate, etc. The inspection and interpretation of these data using conventional techniques (basic statistics and charts, reporting, etc.) is therefore often difficult and tedious.

Visual Analytics [21] gives us a methodology that supports the cognitive process of data analysis and decision making by displaying information interactively. Based on different branches, such as the psychology of reasoning and perception, design aesthetics, data mining, human-computer interaction or information visualization, visual analysis eases the cognitive load in the process of analysing large volumes of data by incorporating perceptive intelligence to the process of abstract reasoning. The state of the art over TMA does not usually cover the aspects of visual inspection of results, focusing on obtaining and validating data, and its subsequent interpretation by conventional methods or simple representations. Visual analysis can greatly facilitate and accelerate the understanding of the data obtained through TMA. Even, and thanks also to the evolution of the technology needed to acquire data, interactive visual TMA can become useful not only as a tool for performance analysis *a posteriori*, but also as a tool for analysis and decision making in real time during the celebration of games or other sporting events.

Visual Analytics seeks to facilitate the analysis process by relying on interactive representations of data. There are several generic models of reasoning that integrate visualization and data mining interaction [7][4] and that can be tailored to specific problems such as TMA in basketball. In particular, visual analysis has been applied to fields that generate very large data volumes, such as biology, where we have data sets such as gene expression data, with tens of thousands of genes under hundreds of conditions [17][16], or that deal with space-time problems such as in [20][15]. Finally, although using synthetic data, visual analytics has been used to approach the analysis of time and motion in virtual environments [2].

However, our literature review suggests that the application of these models explicitly and systematically to TMA in sports problems has not been addressed. There are some basic applications of data visualization that are supplied with TMA technologies for different sports. For example, RealTrack ([www.realtrackfootball.com](http://www.realtrackfootball.com)) offers four modules covering 1) the basic plot of heart rate, 2) kinematic data monitoring, 3) calculation of positional relationships between players and their graphic representation and 4) video module that allows browsing the data collected over time. Especially in football, many similar systems to RealTrack have been developed in the last three or four years (for a review of them, see [1]). These systems offer visualization techniques that are becoming standards in the field of TMA, but these displays are usually isolated from each other, hampering the integration of knowledge and their interfaces are limited, which slows the flow of reasoning. An approach based on visual analysis would improve the cognitive ability of the athlete or coach to make decisions about training and tactics to follow, revealing additional knowledge about the problem.

### 3 Goal and Requirements

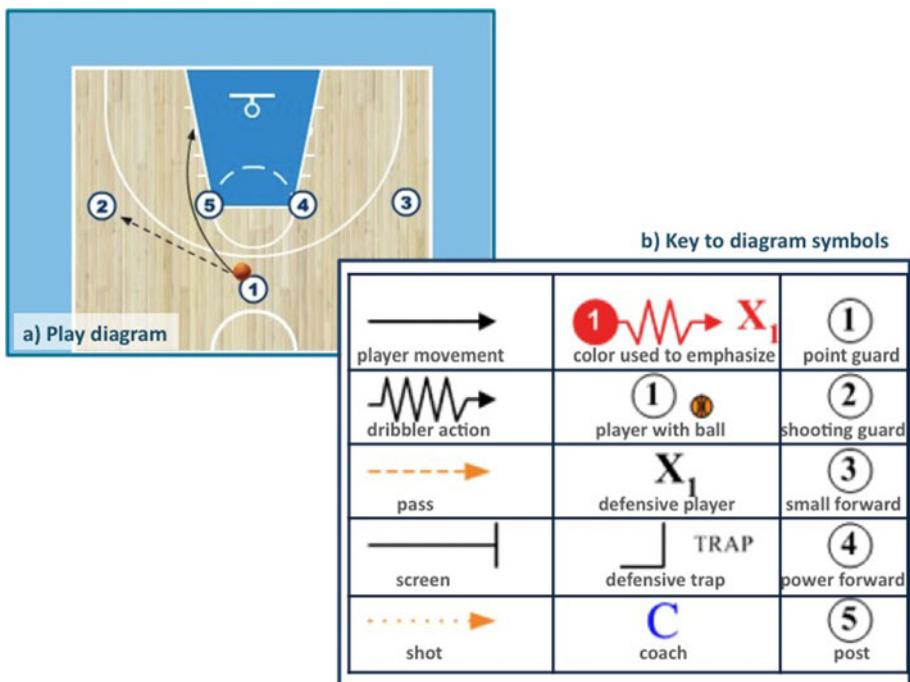
A visual analytics approach must provide the basis for an analytic discourse between the analyst and the information. This discourse is iterative, non linear and evolutive, and must facilitate the testing of different hypothesis, the addressing of the problem at different levels of abstraction, and from different points of view [21].

In the visual analytics area, presentation of results is just the last step of an iterative process that involves previous phases in which visualization (in this case, of the basketball players' performance) has a main role. In order to build a guideline of the visual analysis of the basketball game, we will identify which are the main questions that can be asked to a basketball visual TMA tool by the coach, and how to develop a model to answer them from a visual analytics point of view.

The main question a visual TMA tool for basketball answers is: ***Where was a player in the court at a given moment and how this position relates to the rest of the players?*** Specific instances of this question can generate unlimited hypotheses to be tested using an analytic discourse supported by visual analytics. For example, the hypotheses *the opponent's high score is due to a small area covered in defence by my team or half court pace changes broke the opponents' zone pressure tactics* are just instances of this main question. In addition, we can split the main question into several questions, and then combine them to build our list of requirements.

The following were the objectives of our proposal partially based on the requirements extracted from what a generic basketball's coach would want from an interactive visual TMA tool for the basketball game:

- If possible, the visualizations must be based on the current coaches' practice of diagrammatic representations (see figure 1)
- Handling of GPS devices. The application must be able to read files exported from a GPS device. Also, it will be necessary to transform the data from the GPS coordinates (degrees: latitude, longitude) to UTM (Universal Transverse Mercator, in meters) and scale them to fit on the screen.



**Fig. 1.** a) Basketball play diagrams are the universal language of basketball coaches. By using a small set of symbols, complex play actions involving positions of the players during the duration of the play are described, i.e., a static representation of time and motion (image taken from <http://coaching.fibaeurope.com>). b) In the table, the basic set of symbols are presented: offensive players are depicted as numbered circles and a pass is represented as a dashed line. Thus, the diagram in a) explains a play in which the point guard passes the ball to the shooting guard and advances towards the basket, running behind the post.

- Representation of position and trajectory. The position and trajectory of movement of each player at any given time of the game or exercise been analysed must be available to the user.
- Take into account the individual performance of each player: substitutions. In a basketball game there are numerous substitutions between players entering and leaving the court upon the coach's decision. As part of the players and team analysis, the tool must be able to deal with these changes of the players in court and represent them in a way.
- Time-motion analysis. The main objective of the tool is to be able to ask the tool about the position of the player at any time of the exercise or game. For that purpose, the tool should have an interactive timeline that allows the user to navigate through each player's movement history at will and to filter out events in order to focus on particular time intervals. The game situations and player actions in the court must be reproduced in movie mode or inspected step by step. Furthermore, regarding the TMA, the tool must be able to:

- Compute and represent areas and distances. Given that the use of spaces is very important in a sport like basketball, the tool will have to allow dynamic visualization of the area occupied between several players and the calculation and presentation of the distance between two players or a player and a point of the court at any time.
- Compute and represent kinematic variables. The system must be able to calculate and plot the kinematic variables (speed, distance travelled, time) player for statistical and analytical purposes.
- Compare players and teams. The tool should allow the visualization of the comparison of several kinematic variables of the two players or teams of users.
- Produce reports. In order to track players throughout the season, the coach should have access to a report of every player in PDF format displaying all the data and representations provided by the tool.
- Finally, the tool should incorporate automated data processing mechanisms and a highly interactive visual interface that fosters the analysis.

## 4 Interactive Visual TMA of Basketball

In this section we present and discuss several key aspects and design choices of our proposal.

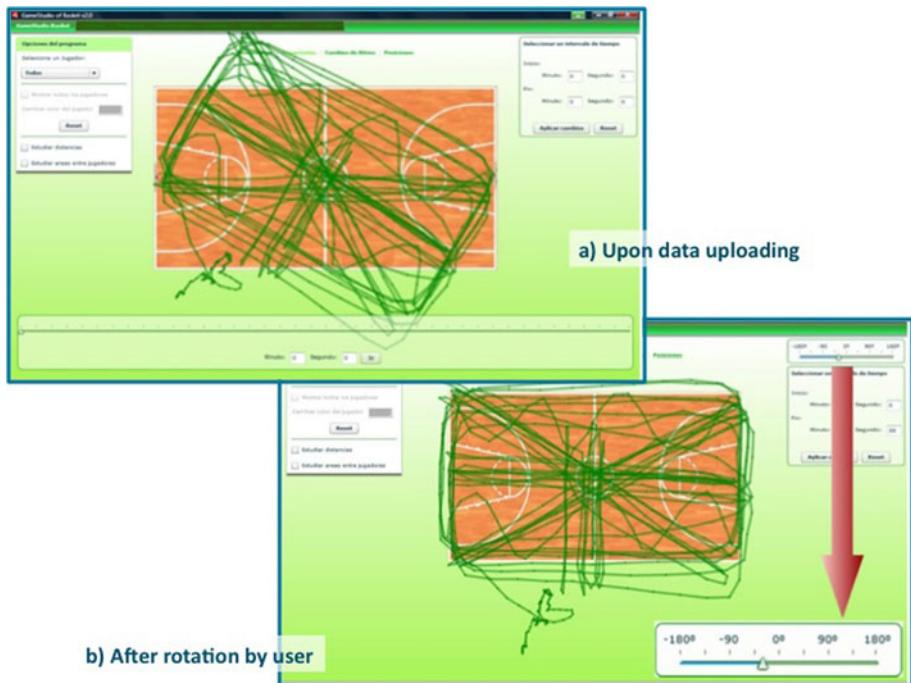
### 4.1 GPS Data Acquisition

Once received the data from our GPS devices, we noticed that the format of the exported files were TCX, a proprietary format from the manufacturer Garmin. Thus, the first step in order to be able to visualise the data was to convert these files to the XML format. There is no standard XML format for GPS devices and we had to decide upon using one of the two most common XML schemas for geographic information, GPX (GPS eXchange Format) and KML (Keyhole Markup Language). After testing both formats, we opted for GPX, because KML was too heavy and forced us to take into account too many elements, while we wanted to obtain only the latitudes, longitudes and time of each position. In addition, GPX is a format much more widespread among GPS devices.

Next step in order to fulfil our objective was to convert geographic coordinates (latitude and longitude degrees) to UTM (meters). Finally, as mentioned previously, the converted points were scaled down to fit in the screen representation.

### 4.2 Orientation of the Dataset

Once we got the coordinates of the points represented by our tool, a new problem arose that was unexpected. The court, in our representation, is oriented horizontally, however, the coordinates were obtained with the Earth globe as reference and, therefore, until their representation in the tool is shown, is not possible to comprehend the right direction to use, since a point in space, without further information, does not have an orientation by itself. In this case we implemented a control for the user to orientate appropriately the representation (see figure 2).



**Fig. 2.** a) The data acquired by the GPS devices are transformed, scaled down and represented in the visual interface. As it can be seen the representation is centred in the court thanks to the initial jump of the game; however, it is rotated in relation to the represented court. b) The user can control the orientation of the dataset to fit in the court. Once this correction is done, the coach can proceed with the rest of the analysis.

#### 4.3 Motion Visualization

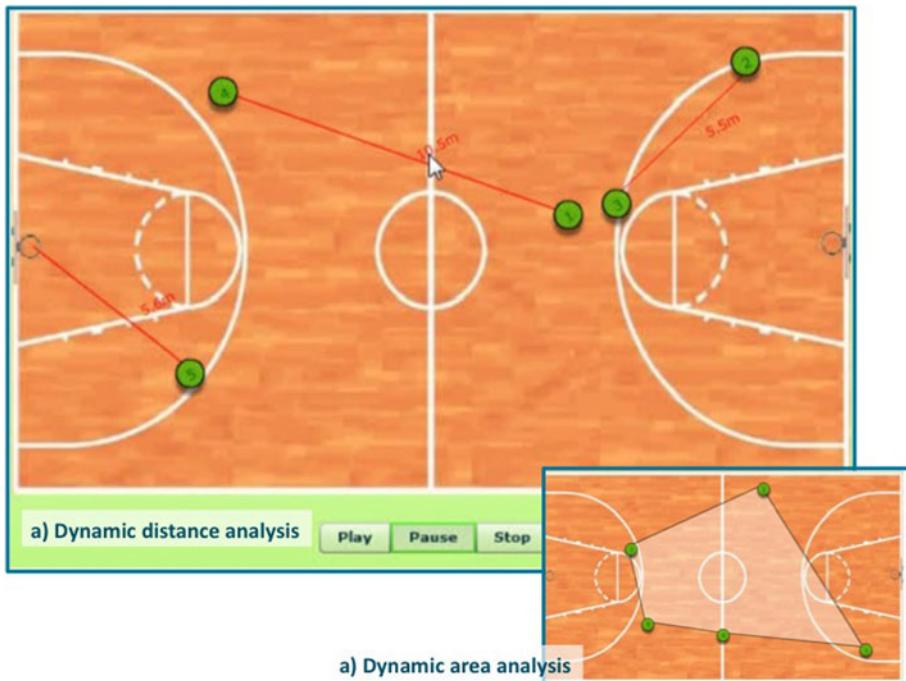
Figure 1 shows the set of symbols used for representing basketball players and their movements. Based on it, and not considering at this stage the representation of the ball, we decided to use the international standard representation for each player: a circle and a number within (see figure 3), while each player movement would be represented by a continuous line. Thus, the representation of the path followed by the player during the completion of a game or exercise action is a continuous line joining all the player's positions, from the first to the last one. One may argue that the result of such representation would be a concatenation of curved lines that simulate the movement of the player. However, according to the expert coaches, the typical movement of a basketball player is either straight or angled. Thus, we decided to join the positions of the player with straight lines (figure 2 shows in green an overview of all the positions occupied by a player during the game).

#### 4.4 Timeline

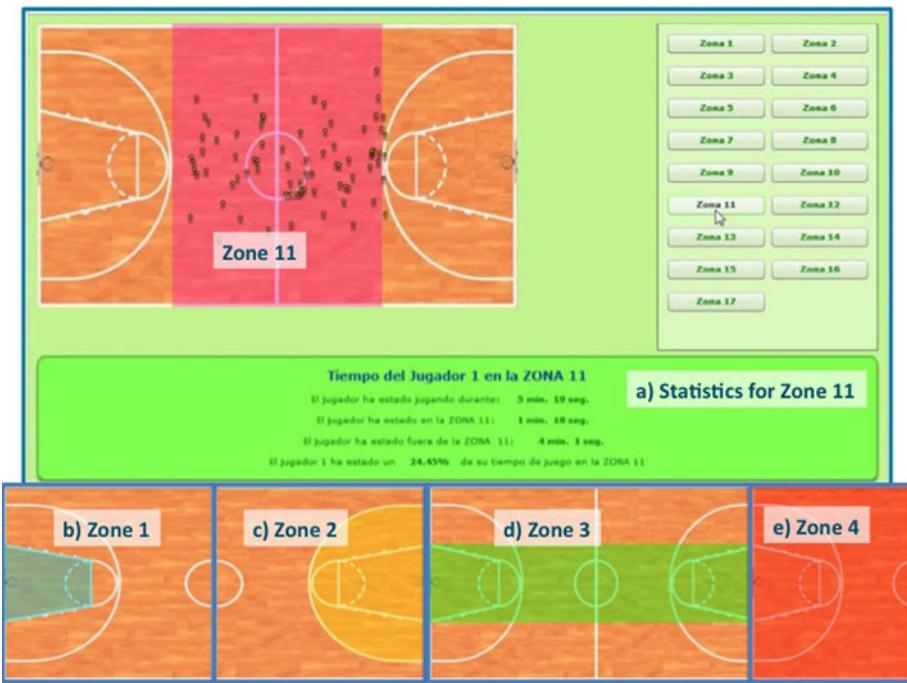
For analysis purposes, a timeline control (it can be seen in figure 2a, below the court's representation) permits the user to access the position of each player in the court at a given time by dragging a handle. We wanted to provide the tool with a simple interface, so there is also the option of moving to a particular event of the game, simply by entering the exact minute and second the coach wants to inspect in an input box.

A movie mode has also been implemented, so the coach can view the evolution of the game actions (related to a particular analysis feature, such as distances and zones, that are discussed later) for a particular period (see the movie buttons at the bottom of figure 3a, for instance).

In the timeline, all the substitutions occurred during the game are also represented. To facilitate this, the timeline has a button "Enter player substitution" that gives the possibility to change the players. These changes are reflected in the course of the game animation as the timeline progresses (the player who leaves the court will disappear from the representation, while the player who enters will be visible to the user).



**Fig. 3.** a) Dynamic distance analysis. The coach has selected the five players of the team in order to study the distances among them and the basket for a particular period of time. b) Dynamic area analysis. The same study as in a) has been carried out, in this case trying to analyse how well the team are covering areas of the court, either for attack or defence, so the coach can efficiently tune the tactics or assess the performance of the team.



**Fig. 4.** a) Analysis of zone 11 (midcourt zone) presence during the game for player 1. The numerical statistics are available in the dark green area. The tool has a set of 17 predefined zones, such as: b) free throw lane, c) inside the three-point line, d) basket-to-basket lane, or e) half court.

#### 4.5 Distances

Distances between players are of particular interest for the coach. Although current GPS technology does not provide a precision to the centimetre, we decided to provide the coach with rough evidence that would be a valuable aid to tune his or her tactics. The study of distances has been designed in a simple way: the user picks as many pair of objects (a player or court point) as he or she wishes, and the distance between each pair is computed; a line from the first object to the second one is drawn, together with a label reflecting the distance in meters (see figure 3a). Thus, 2 players (circles) or a player and a point of the court, are joined by a thin line labelled with a distance value. Once two objects have been selected for distance analysis, the lines and distance labels are updated dynamically in the movie mode.

#### 4.6 Areas

In basketball, the use of space is one of the most important aspects to take into account regarding the team's performance. Therefore, we wanted to provide the coach with an option to represent the area covered at any time by 3, 4 or even 5 players. This functionality aims at giving the user a clear means for understanding the use of space in

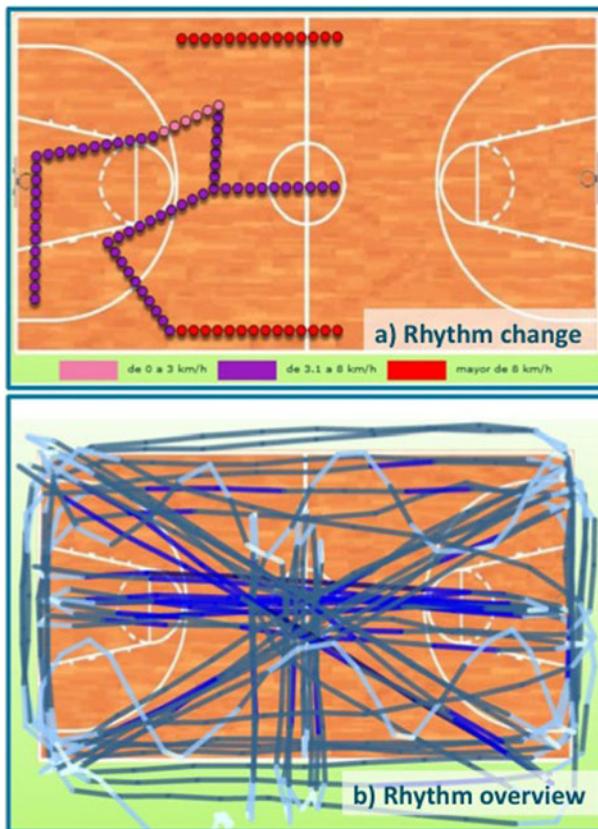
a way that allows players or the coach to draw conclusions on how to play and correct mistakes at certain times of the game (see figure 3 b).

#### 4.7 Zones

In conducting a full and interesting statistical study about a player we felt that a clear representation of the time that a player has been in a certain zone of the court was needed. This would allow us to know the role that a member of the team has played during the game, unravelling patterns such as the fact he or she tends to play more in a zone or simply highlight that a player is left or right handed.

#### 4.8 Players Comparison and Reports

Although the application is oriented towards interactive dynamic visual analysis, we have implemented a reporting option that includes the generation of a variety of



**Fig. 5.** a) Change of pace analysis for a small interval of time (red colour means faster). b) Overview of players' pace change for the whole duration of the game (dark colour means faster). This option is more efficient for the analysis large periods of time regarding both visualization cluttering and computational load (80% faster to compute).

statistical charts and a numerical summary for a player or a 'players vs. player' comparison. This information is available both within the tool and as a PDF file.

#### 4.9 Pace Changes

Another important component of the coach's tasks is pace change analysis. Since speed is measured regarding the distance travelled in an interval of time and the change of pace is a speed change in an instant, we decided to represent the speed of player throughout the analysed period using colour coded circles, and thus, the visualization of changes of pace would be depicted as 2 consecutive circles of different colour (see figure 5a). This method is useful when a small interval of time is analysed. However, in order to give an overview of the different paces during the whole game, we colour the player's trajectory lines using a light-to-dark scale (see figure 5b).

### 5 Conclusion and Further Work

The development of dynamic, interactive visual solutions is a necessity in a historic moment when the generation capacity of data is huge (the accuracy of GPS devices will improve and/or combining them with other technologies such as video systems, will allow the derivation of the players' position with negligible errors) while the means of analysis have not bloomed evenly. In this regard, the recent advent of visual analytical science is very promising.

As future improvements we have considered to adapt the tool to import a file that has the complete statistics of a basketball game, so that could be visually reflected giving way to the analysis of: steals, evolution of the score of the match, points scored by each player, and so on. Moreover, and focusing on the usefulness of the tool towards the physical training of the players, it would be interesting to capture and represent data in real time.

Currently, one important and missing information that has not been included yet is the representation of the ball during the basketball game. Since in this work we have focused on using GPS devices, this goal could be achieved by combining our method with the use of several cameras that capture every move on the court from different perspectives. This method would bring both more accuracy and the capture of the position of the ball.

### Acknowledgments

This work was supported by the MEC (project Graccie (Consolider-Ingenio, CSD 2007-00067)) and by the JCyL (project GR34).

### References

1. Carling, C., Bloomfield, J., Nelsen, L., Reilly, T.: The role of motion analysis in elite soccer. *Sports Med.* 38(10), 839–862 (2008)
2. Chittaro, L., Ranon, R., Ieronutti, L.: Vu-flow: A visualization tool for analyzing navigation in virtual environments. *IEEE Transactions on Visualization and Computer Graphics, Special Issue on Visual Analytics* 12(6), 1475–1485 (2006)

3. Dobson, B.P., Keogh, J.W.L.: Methodological issues for the application of time-motion analysis research. *Strength and Conditioning Journal* 29(2), 48–55 (2007)
4. Fry, B.: Computational Information Design. Doctoral. PhD thesis, Massachusetts Institute of Technology (2004)
5. Gross, D.R.: Time allocation: A tool for the study of cultural behavior. *Ann. Rev. Anthropol.*, 519–558 (1984)
6. Keane, S., Feilly, T., Hughes, N.: Analysis of work rates in gaelic football. *Aust. J. Med. Sci. Sport.* 25, 100–102 (1993)
7. Keim, D.A., Mansmann, F., Schneidewind, J., Ziegler, H., Thomas, J.: Visual analytics: Scope and challenges. In: Simoff, S.J., Böhnen, M.H., Mazeika, A. (eds.) *Visual Data Mining*. LNCS, vol. 4404, pp. 76–90. Springer, Heidelberg (2008)
8. Lafontaine, D., Lamontagne, M., Lockwood, K.: Time-motion analysis of ice-hockey skills during games. In: Riehle, H., Vieten, M.M. (eds.) *ISBS 1998: XVI International Symposium on Biomechanics in Sports*, pp. 481–484. International Society for Biomechanics of Sports, Konstanz (1998)
9. Lothian, F., Farrally, M.: A time-motion analysis of women's hockey. *J. Hum. Movement Stud.* 26, 255–265 (1994)
10. Mayhew, S., Wegner, H.: Time-motion analysis of professional soccer. *J. Hum. Movement Stud.* (11), 49–52 (1985)
11. Mc Lean, D.: Analysis of the physical demands of international rugby union. *J. Sports Sc.* 22, 285–296 (1992)
12. McInnes, S.E., Carslon, J.S., Jones, C.J., McKenna, M.J.: The physiological load imposed on basketball players during competitions. *J. Sports Sci.* 13, 519–528 (1995)
13. Meir, R., Arthur, D., Forrest, M.: Time and motion analysis of professional rugby league: A case study. *Str. Cond. Coach.*, 1–5 (1996)
14. Mohr, M., Krstrup, P., Bangsbo, J.: Match performance of high-standard soccer players with special reference to development of fatigue. *J. Sports Sci.* 21, 519–528 (2003)
15. Santamaria, R., Theron, R.: Treevolution: visual analysis of phylogenetic trees. *Bioinformatics* 25(15), 1970–1971 (2009)
16. Santamaria, R., Theron, R., Quintales, L.: Bicoverlapper: a tool for bicluster visualization. *Bioinformatics* 24(9), 1212–1213 (2008)
17. Santamaria, R., Theron, R., Quintales, L.: A visual analytics approach for understanding biclustering results from microarray data. *BMC Bioinformatics* 9 (2008)
18. Taylor, J.: Basketball: Applying time motion data to conditioning. *J. Strength. Cond.* 25(2), 57–64 (2003)
19. Terrier, P., Schutz, Y.: How useful is satellite positioning system (gps) to track gait parameters? a review. *J. Neuroengineering Rehabil.* 2, 28–38 (2005)
20. Theron, R.: Visual analytics of paleoceanographic conditions. In: *IEEE Symposium on Visual Analytics Science and Technology*, pp. 19–26. IEEE Press, Los Alamitos (2006)
21. Thomas, J.J., Cook, K.A.: A visual analytics agenda. *IEEE Computer Graphics and Applications* 26, 10–13 (2006)
22. Wilson, A.M., Witte, T.H.: Accuracy of non-differential gps for the determination of speed over ground. *J. Biomech.* 37, 1891–1898 (2004)

# Event Line View: Interactive Visual Analysis of Irregular Time-Dependent Data

Krešimir Matković<sup>1</sup>, Alan Lež<sup>1</sup>, Denis Gračanin<sup>2</sup>,  
Andreas Ammer<sup>1</sup>, and Werner Purgathofer<sup>1</sup>

<sup>1</sup> VRVis Research Center in Vienna, Austria

{Matkovic,Lez,Ammer,Purgathofer}@VRVis.at

<sup>2</sup> Virginia Tech

gracanin@vt.edu

**Abstract.** In this paper we present a novel approach to visualize irregularly occurring events. We introduce the *event line* view designed specifically for such events data (a subset of time dependent data). The *event line* view is integrated in a coordinated multiple views (CMV) system and linked with other conventional views to support interactive visual analysis. The main idea is to analyze events relative to two categorical attributes from a multidimensional multivariate dataset. Since we are interested in the categorical dimension we have also integrated and linked the *tag cloud* view in the CMV system. To the best of our knowledge this is the first integration of the *tag cloud* view in a CMV system. The *tag cloud* view can depict a ratio of the selected items versus the non-selected items.

The proposed approach is illustrated using the VAST Challenge 2008 Geo-Spatial data set that contains data on interdiction or landing of illegal immigrants in the USA. It is a multivariate multidimensional dataset with irregular events that illustrates the potential and capabilities of the proposed approach and the developed CMV system.

**Keywords:** Interactive Visual Analysis, Coordinated Multiple Views, Events in Time, Tag Cloud.

## 1 Introduction

Interactive visual analysis helps with data exploration and analysis in various application domains. There are numerous successful visualization approaches for various data types. Multivariate multidimensional data sets are collections of records (or rows in a table like in a spreadsheet program, e.g.) where each record contains multiple attributes. This is one of the most common data organization in information visualization.

We investigate a special case of such data sets we call *events in time*. The main characteristics is that various events (represented by single records) happen at irregular times. A good example of such a data are traffic (or any other) accidents. We do not know in advance when an accident will occur, but we are interested to find some patterns and hidden correlations in order to prevent future accidents, if possible. Another example is a publication list of a research

institution. Here we also have temporally irregular events, but some regularity might occur (larger frequency close to important conference submission deadlines, e.g.). In a business domain selling of various goods can also be considered to be an event. The same is true for events in a computer system.

Time dependent data visualization is a very active field of research [12], but we focus on a specific variant of such data sets. We present a novel view, the *event line* view, tailored specifically for the data sets containing irregular events. Since there are usually more dimensions in a data set, and we are looking for a hidden correlations, we have integrated the new view in a coordinated multiple views (CMV) system.

The correlation among the irregular events can be explored by custom grouping of the events. The events can be grouped based on the specific values of the categorical dimensions. In general, we can have a large number of categorical dimensions in a data set.

We use two categorical dimensions within a single *event line* view. In most cases, two dimensions provide sufficient grouping flexibility, e.g. considering spatial (e.g. location) and temporal (day of a week) categorical values. Linking several *event line* views within the CMV system allow us to use any number of categorical dimensions.

Once the events are selected in the *event line* view, we look for correlations among them. A correlation is indicated by the common values in other categorical dimensions. What we need is a way to show what values of a categorical dimension are the most often present in the selected events.

Categorical data is usually described with strings (or tags) that can be represented using a *tag cloud* view [2]. Therefore, we included the *tag cloud* view in the CMV system. That way, we can use the *event line* view to select events and the *tag cloud* view to identify possible correlation among the selected events. In this way a data set containing events in time can be efficiently analyzed. To the best of our knowledge we are the first to integrate the *tag cloud* view in CMV.

We explain the view design and show effectiveness of the view using the VAST Challenge 2008 Geo-spatial dataset [4]. Our entry in that challenge [11] describe the use our CMV system (ComVis tool [8]) that, in connection with some helper applications and Google Earth, allowed us to explore geo-temporal characteristics of the data set and answer the challenge questions. We build on these results in order to improve the visualization and analytical capabilities by adding new views to the CMV system.

## 1.1 Related Work

Event data visualization and analysis tool are still not very common. Suntinger et al. [14] were among the first exploring such data sets. They have introduced a system called Event Tunnel used for business applications.

Visualization of event data can be considered a part of a larger topic of high-dimensional and time-dependent data visualization. Many approaches to the interactive visual analysis of time dependent data have been already introduced. Müller and Schumann, for example, provide good overview of the visualization methods for time-dependent data [12].

The related visualization techniques can be classified into two groups based on whether or not the visual representation itself is time-dependent. Aigner et al. [1] provide an overview of visual methods for analyzing time-oriented data and discuss general aspects of time-dependent data. Multivariate data visualization techniques for multi-variate time-dependent data include the ThemeRiver [6], Spiral Graph [16] and others.

Time-dependent data often exhibit some periodic behavior. Such serial periodic data are of special interest. For example, time continues forward sequentially but includes recurring periods (weeks, months, and years) [3]. The challenge is how to simultaneously display serial and periodic attributes of the data set. Spiral arrangement of the data provides easy visual cues to both serial and periodic aspects of the data. Special visual metaphors like the Calendar View or Lexis Pencils are also used. In the Calendar View [15], van Wijk et al. use a very intuitive calendar view to display clusters of the time series data. Lexis pencils [5] display various time-dependent variables on the faces of the pencil. Pencils can depict several variables and can be positioned in space to indicate the spatial context of the data.

Extensions of traditional visualization techniques to time-dependent data include the Wormplot that provides a scatter plot for each time step. The Time Wheel leverages parallel coordinates so that the time axis is in the center and all other axes are located circularly around it. The MultiComb arranges those other axes in a star-shaped pattern.

Wegenkittl et al. [17] use extruded parallel coordinates, linking with wings, and three-dimensional parallel coordinates. Brushing the time axis to display details of the selected time frame is a very common and useful interaction technique used with static representations [8, 17]. Timebox widgets by Hochheiser et al. [7] can be used to brush both the time axis and the attribute axis of graphs.

Coordinated multiple views are also a well established methodology in data analysis. A good overview of the Coordinated and Multiple Views state of the art and what future may hold for this area is given by Roberts [13].

*Tag cloud* view is a standard way of visualizing text, especially for web. There are numerous papers dealing with tag clouds and improving of placement. Lohmann et al. [10] give an overview of various tag clouds algorithms. However, we are not aware of a *tag cloud* view integration into a coordinated multiple views system.

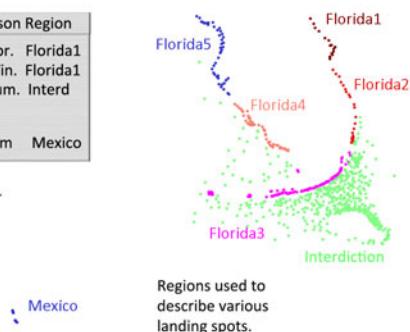
## 2 Data Model

Our main goal is to analyze data that contains time-based events. The events appear at irregular times, and there are more attributes describing event than just time. We will introduce the data model used first.

We explore the data sets that contain many records, and each record contains several attributes. One record corresponds to one event. One of the attributes is time, and others can be either categorical or numerical values. Such a data set represents a typical multidimensional multivariate data set. It can be accidents

LAT	LON	Type	Pass.	USCG	Date	Death	Vessel	Day	Month	Season	Region
-80.44	28.50	Land	24	N/A	20070510	0	Go Fast	Thu.	May	Spr.	Florida1
-80.86	29.11	Land	18	N/A	20070309	0	Rustic	Fri	Mar	Win.	Florida1
-79.96	24.98	Inter	18	Stal.	20070901	0	Rustic	Sat.	Sep	Sum.	Interd
...											
-86.74	21.26	Land	10	N/A	20070704	0	Rustic	Wed.	Jul	Sum	Mexico

Dataset used in the analysis, there are 917 records in total.



**Fig. 1.** The table on the left illustrate multidimensional multivariate dataset used. There are some numeric dimensions (latitude, longitude, number of passengers and number of deaths), and some categorical dimensions (week day, month, region, ...). There is also the time dimension. One record (row in the table) represents one event in time. The implicit map on the right, created using a scatterplot and longitude and latitude shows how we have aggregated the region dimension.

data set containing time when an accident happened, information if cars, trucks, or busses were involved, were there any injuries, etc. Another example can be phone calls. There we could have time the call was started, duration, persons involved, various attribute of persons such as age, gender, etc.

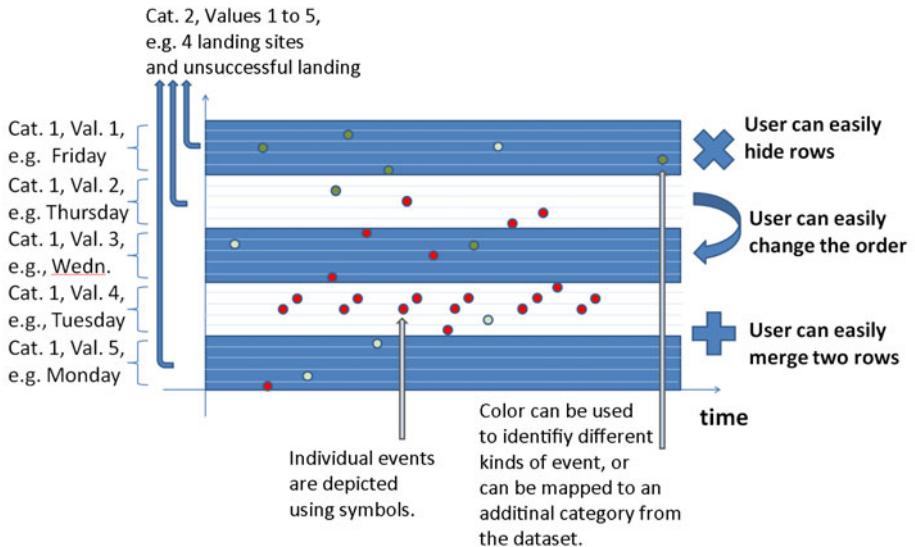
We use the data set originally issued for the VAST Challenge 2008. The data set includes the interdiction records collected by the United States Coast Guard and information from other sources about illegal landings on shore. Each record has several attributes, including longitude and latitude (either form interdiction or from the landing), date and type (interdiction or landing), type of vessel used by immigrants, name of US Coast Guard Ship in case of interdiction, number of passengers on board, and number of fatalities.

We have derived some additional data in order to easier answer questions that appear in the analysis. We have used date to compute day of week, month, and season as additional attributes. We have also grouped landing positions into seven regions, five in Florida, one in Mexico, and one for interdicted vessels.

Figure 1 illustrates the dataset on the left and shows five Florida and one Mexican regions. The seventh region represents successful interdiction spots. We have used simple scatter plot to depict latitude and longitude. The map created in this way is distorted due to the changed aspect ratio, but Florida can be easily recognized.

### 3 The Event Line View

The events line view is tailored especially for analysis of events in time with respect to two categorical attributes. We will describe the view design first, followed by interaction and integration in a coordinated multiple views setup.



**Fig. 2.** The main principle behind the *event line* view. The *x*-axis depicts time and the *y*-axis represents two categorical dimensions. The *y*-axis is divided in strips (blue and white in the image), so that there is a strip for each value in the first category (e.g. a weekday). Each strip is then subdivided into values of another category. In our case, those values can be landing regions. Each event is now represented by a symbol in the view. There can be various symbols depending on an additional criterion. The user can merge strips, hide strips, and change strips order. The *y*-axis can be re-parameterized to show different dimensions.

We use the *x*-axis as time while the *y*-axis is used for two categorical values. We compare the events with respect to the two categorical attributes, as this is often the case in the analysis. In the case of our data an analyst would like to see events and compare how e.g. day in week correlates with successful interdiction/landing during the time.

In order to do so we will divide the *y*-axis in parts. Each part represents one value for the first category. In the case of above example this would be successful interdiction in upper half of the axis and successful landing in to lower part. Now when the axis is divided, we subdivide each part in the subranges, one for each value of the second category. In our case, we will have seven subcategories, one for each day in the week. This means that number of ranges will equal to a product from number of ranges in each of the two categories, or 14 in our case: seven for landings and seven for interdiction.

The events themselves are depicted using a point (or any other symbol) in the corresponding range strip and time position. Figure 2 illustrates the main idea. The user can choose the categories at any time and change the parametrization of the view.

Since the number of ranges on the  $y$ -axis can be very large and make the view less useful, we propose several mechanisms to cope with this issue. One option is to hide less interesting ranges. That saves space, but at the same time reduces information available. Another possibility is to merge several ranges into one. This can be done for the ranges that are less interesting, or in the case when the screen space is limited. When the ranges are merged, the points or symbols used to depict events have different shape or color depending on their original ranges.

The shape and color of the symbols can always be customized dependent on range or any additional attribute from the data set. We currently support circles and capital letters as symbols. Letters are convenient when ranges are merged and we want to distinguish between original events. Of course, any other symbol could be used, but it is challenging to design a large number of easily distinguishable small symbols.

In case of a long time span, or a coarse time scale (like for example events described with month (or day) only, it is possible that more than one event share the same position in screen. We propose two techniques here, one is symbol size increase based on the number of events represented by a single symbol, and the other one is dispersion of the symbols in a close neighborhood. The overlapping symbols can also occur when several ranges are merged. The same technique is used to overcome the problem in this case, too. Additionally, a zoom of the time axis is useful for the very long time spans.

The order of ranges on  $y$ -axis can enhance the comparison possibilities significantly. The patterns in neighboring ranges can be spotted more easily. Our events line view makes it possible to rearrange ranges by moving a specific range up or down.

The events line view is integrated in a coordinated multiple views system. Whenever a subset of data is brushed (selected) in any other view, corresponding events in the *event line* view are highlighted. The view supports brushing too, the user can brush certain events in the view, and corresponding items in all other views will be highlighted. There can be several *event line* views with different parameterizations active simultaneously.

## 4 The Tag Cloud View in Coordinated Multiple Views

The *event line* view is used to analyze various events in respect to two categorical values. Categorical values themselves can be visualized in numerous ways. Parallel sets [9] have been developed especially for categorical values. A histogram, stacked bars, a pie chart or even a scatter plot can be used as well.

The *tag cloud* view is a standard view for displaying tags or keywords. It is widely used on the web and represents a well known and very efficient way of visualizing frequency of appearance of single words or short phrases. Since categorical values are often represented by words, and we focus on comparison of events in respect to categorical data, we decided to include the *tag cloud* view in the CMV system.



**Fig. 3.** The *tag cloud* view integrated in the CMV system. The top left *tag cloud* view shows months as contained in our data set. The user selected only the interdiction region (in a histogram view not showed here) and the selection is shown with context in the lower row (horizontal and vertical), and without the context in the top right *tag cloud* view.

There are many ways how to optimize layout of a *tag cloud* view [10]. Our goal is not to improve the *tag cloud* view itself, but to include it in the CMV system and use it to depict categorical data. To the best of our knowledge, this is the first attempt to use the *tag cloud* view as a part of the CMV system.

If the user brushes a subset of the data in any view, the selection will be highlighted in the *tag cloud* view as well. In order to show the ratio of the selected and unselected items for each category value (or a word in the *tag cloud* view), we show the original *tag cloud* view in gray (or any user selected context color), and highlight a part of the word depending on the selection using focus color (red in our case). The selection can be depicted horizontally or vertically.

Figure 3 illustrates the idea. The *tag cloud* view shows distribution of months in our dataset. Just as expected, most of the events occurred during warm, summer months.

The top left *tag cloud* view shows the overall distribution. We have brushed interdictions now (selection was done using a simple histogram not shown in Figure 3). The bottom row shows two versions (horizontal and vertical) of the selection.

The user can optionally choose to show brushed values only, as depicted in the top right *tag cloud* view. In this case only words that belong to the selection are shown (only focus, without context), and resized accordingly. The context is not shown, but comparison within selection is easier. The user can easily switch between the two modes in order to gain better insight into data.



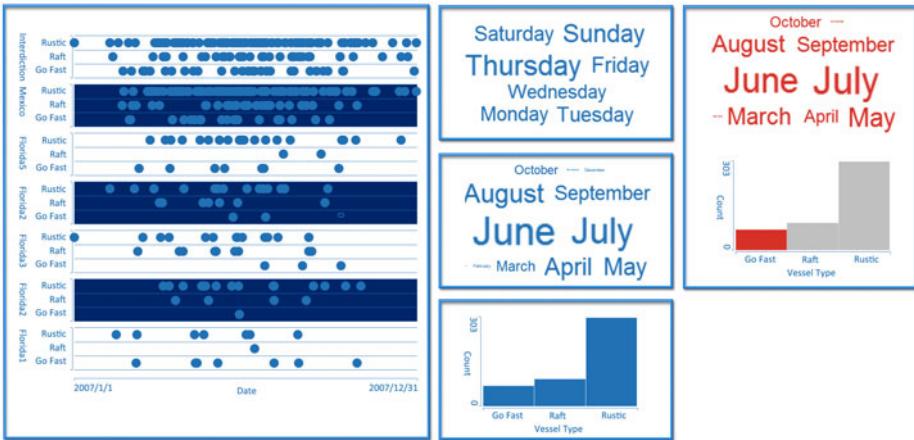
**Fig. 4.** The *event line* view for years 2005, 2006, and 2007. Some interesting things can be noticed. First of all, the increase in the number of events is accompanied by the increase in the number of landing spots over time. We can see that some regions are active during all three years (*Florida3*), and some became active in 2007 (*Florida1* and *Florida2*). Also, looking at week days it can be noticed that *Florida2* is almost never active during weekends and that *Florida5* has a big gap on *Monday*.

## 5 Illustrative Example

In this section we provide examples to illustrate usefulness of the proposed approach. We will start with the complete data set which contains events from years 2005, 2006, and 2007. The data set used is described in Section 2.

We were interested in the events with respect to the regions and days of the week. We wanted to see if there are some days when there are especially many or few events, if there is some rules like going to the same region on certain days, etc. We use the *event line* view with *y*-axis showing regions (there are seven strips: *Florida1* to *Florida5*, *Mexico*, and *Interception*) and days of a week (seven lines per strip, one for each day). Figure 4 shows the view.

From this view many interesting things can be observed. First thing is the visible increase of the number of events during time, and increase of number of landing spots. This could be due to increased activity of the immigrants or due to US Coast Guard becoming more effective (for example it could mean that the fleet became bigger). Other things are, for example, *Florida3* is active during all three years, *Florida4*, *Florida5* and *Mexico* became active in 2006 while *Florida1* and *Florida2* mostly became active in 2007. Looking at the week days we can notice that *Florida2* is almost never active during weekends and that *Florida5* has a big gap on *Monday*.



**Fig. 5.** Events in 2007. It is very obvious that there are no trends and patterns in boat type usage, but also that the number of boats of some type are not the same. While *Go Fast* and *Raft* types are very similar in numbers, *Rustic* type is used in much larger numbers. Very important thing is that there is much much less immigrant activity during fall and winter which is probably due to bad weather. This is also confirmed in the *tag cloud* view with months where *January*, *February*, *November*, and *December* (winter months) are barely visible. The *tag cloud* view with week days shows us that *Thursday* and *Sunday* are the most active days and there is a growing trend towards the end of the week. The histogram confirms that *Rustic* boat type is used most often. At the right we can see that *Go Fast* boat type is rarely used during winter (with the selected brush only option in the *tag cloud* view, the winter months are barely visible or not visible at all).

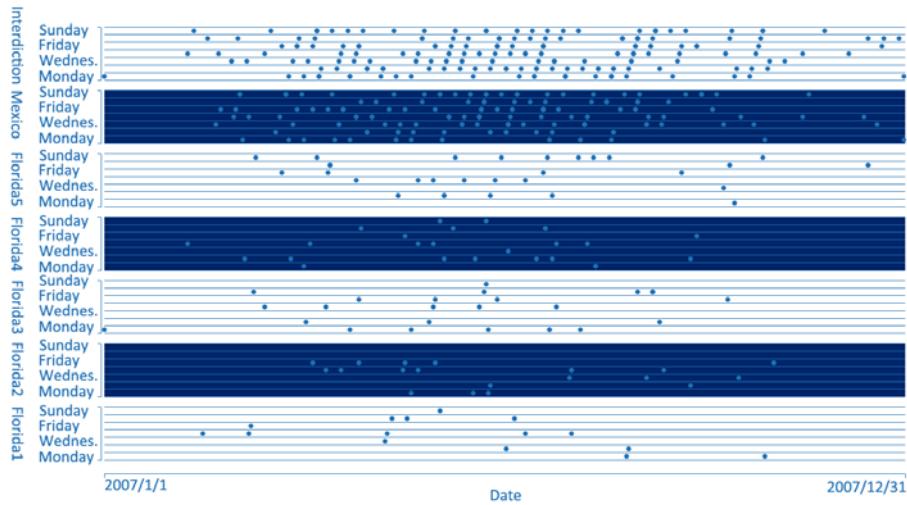
Since there are most of events in 2007, we will focus on this year only from now on. We want to analyze events in respect to a region and boat type now. We are interested if migrants are using specific boats for specific routes, for example, or if there is any other pattern in the data. As we have multiple views we will use the *tag cloud* view to depict weekdays so to see if there are any preferable days in the week when the events occur, and another *tag cloud* view to depict months. We have also included a histogram showing frequency of usage for each boat type (Figure 5).

We were first interested if there any trends or patterns considering boat types being used, which could then be compared to trends in immigrants activities to see is there a specific boat type which is more effective than the others. We can see that *Rustic* vessel is used most often and *Go Fast* vessel the rarest.

It is interesting to see the distribution of events with specific boat and landing spots. *Go Fast* succeeded only once in reaching *Florida2*, e.g., and only twice in reaching *Florida4*. It reached *Mexico* quite often. The usage of *Go Fast* is not frequent at the end of the year. There were very few successful landings and very few successful interdictions in the last months. In order to confirm this finding



**Fig. 6.** Immigrant successfulness. The left view shows unsuccessful (context) as gray and successful (brushed) as red. *March*, *May* and *June* were most successful months for the immigrants. Right view shows same thing but with brush only option selected (so the context is not shown).



**Fig. 7.** Searching for events happening day after day in particular region. This can be identified in events line as dots in line with a slight slope to the right when looking from *Monday* to *Sunday*. We can see that one region (*Florida4*) has none of such cases, and one has many (*Mexico*). *Interdiction* has some examples where there were events in every day of the week. Again, there are no visible patterns and the number of these cases is proportional to the number of events in a region overall.

we have brushed *Go Fast* vessels in the histogram and select option show brush only in the *tag cloud* view with months.

Figure 5 shows the *tag cloud* view and the brushed histogram on the far left. Note that *January*, *February*, *November*, and *December* (winter months) are barely visible in this *tag cloud* view.

The next task could be identifying months when the immigrants are most successful (most landings). We can visualize this using the *tag cloud* view of months with all successful landings selected (Figure 6). There are two views, one with context and one without. The red part represents successful landings. Immigrants are most successful during *March, May and June*.

Last thing we are interested in is how often did the events happen day after day in some region, which could also lead us to patterns. These cases can be identified in the *event line* view as dots in line with a slight slope to the right when looking from *Monday to Sunday* (Figure 7).

*Florida4* has none, *Florida1*, *Florida2*, *Florida3* and *Florida5* have few, *Mexico* has many of those cases and *Interdiction* has few examples when there were events in every day of the week.

## 6 Summary and Conclusions

In this paper we have presented a new way of visualizing irregularly occurring events. We have introduced a new type of view, the *event line* view, and integrated it into a coordinated multiple views (CMV) system. The *event line* view supports events analysis based on two categorical attributes. Since we are mostly dealing with categorical data we also added the *tag cloud* view which is well suited for categorical data visualization. We also provided illustrative examples of usefulness of this new view type.

The *event line* view has been proven very useful in exploring data sets that motivated the development of the view. When combined with the *tag cloud* view and with the other well known views (histogram, scatter plot, parallel coordinates, etc.) in a coordinated multiple views (CMV) system, it provides new analytical capabilities. These capabilities may enable new discoveries which were previously harder (or less intuitive) to do or even impossible.

## Acknowledgments

The authors thank Helwig Hauser for numerous discussions and Hrvoje Šala for help with the first experiments with the *event line* view. Part of this work was done in the scope of the VSOE VCV program at the VRVis Research Center in Vienna (<http://www.VRVis.at>) and at the Center for HCI at Virginia Tech (<http://www.hci.vt.edu>).

## References

1. Aigner, W., Miksch, S., Müller, W., Schumann, H., Tominski, C.: Visual methods for analyzing time-oriented data. *IEEE Transactions on Visualization and Computer Graphics* 14(1), 47–60 (2008)
2. Bausch, P., Bumgardner, J., Fake, C.: Flickr Hacks: Tips & Tools for Sharing Photos Online (Hacks). O'Reilly Media, Inc., Sebastopol (2006)

3. Carlis, J.V., Konstan, J.A.: Interactive visualization of serial periodic data. In: UIST 1998: Proceedings of the 11th Annual ACM Symposium on User Interface Software and Technology, pp. 29–38. ACM Press, New York (1998)
4. VAST 2008 challenge (2008), <http://www.cs.umd.edu/hcil/VASTchallenge08>
5. Francis, B., Pritchard, J.: Visualisation of historical events using Lexis pencils. Advisory Group on Computer Graphics (1997)
6. Havre, S., Hetzler, E., Whitney, P., Nowell, L.: ThemeRiver: Visualizing thematic changes in large documents collections. IEEE Transactions on Visualization and Computer Graphics 8(1), 9–20 (2002)
7. Hochheiser, H., Schneiderman, B.: Dynamic query tools for time series data sets: timebox widgets for interactive exploration. Information Visualization 3(1), 1–18 (2004)
8. Konyha, Z., Matković, K., Gračanin, D., Jelović, M., Hauser, H.: Interactive visual analysis of families of function graphs. IEEE Transactions on Visualization and Computer Graphics 12(6), 1373–1385 (2006)
9. Kosara, R., Bendix, F., Hauser, H.: Parallel sets: Interactive exploration and visual analysis of categorical data. IEEE Transactions on Visualization and Computer Graphics 12(4), 558–568 (2006)
10. Lohmann, S., Ziegler, J., Tetzlaff, L.: Comparison of tag cloud layouts: Task-related performance and visual exploration. In: Gross, T., Gulliksen, J., Kotzé, P., Oestreich, L., Palanque, P.A., Prates, R.O., Winckler, M. (eds.) INTERACT 2009. LNCS, vol. 5726, pp. 392–404. Springer, Heidelberg (2009)
11. Miklin, R., Lipić, T., Konyha, Z., Berić, M., Freiler, W., Matković, K., Gračanin, D.: Migrant boat mini challenge award: Simple and effective integrated display. In: Ebert, D., Ertl, T. (eds.) Proceedings of the IEEE Symposium on Visual Analytics Science and Technology (VAST 2008), October 19–24, pp. 203–204 (2008)
12. Müller, W., Schumann, H.: Visualization for modeling and simulation: Visualization methods for time-dependent data - an overview. In: WSC 2003: Proceedings of the 35th Conference on Winter Simulation, vol. 1, pp. 737–745 (2003)
13. Roberts, J.C.: State of the Art: Coordinated & Multiple Views in Exploratory Visualization. In: Andrienko, G., Roberts, J.C., Weaver, C. (eds.) Proc. of the 5th International Conference on Coordinated & Multiple Views in Exploratory Visualization. IEEE CS Press, Los Alamitos (2007)
14. Suntinger, M., Obweger, H., Schiefer, J., Gröller, M.E.: Event tunnel: Exploring event-driven business processes. IEEE Computer Graphics and Applications 28(5), 46–55 (2008)
15. van Wijk, J.J., van Selow, E.R.: Cluster and calendar based visualization of time series data. In: INFOVIS 1999: Proceedings of the 1999 IEEE Symposium on Information Visualization, p. 4. IEEE Computer Society, Washington (1999)
16. Weber, M., Alexa, M., Müller, W.: Visualizing time-series on spirals. In: Proc. of the IEEE Symp. on Information Visualization, pp. 7–13 (2001)
17. Wegenkittl, R., Löffelmann, H., Gröller, E.: Visualizing the behavior of higher dimensional dynamical systems. In: Proceedings of the IEEE Visualization (VIS 1997), pp. 119–125 (1997)

# Automatic Blending of Multiple Perspective Views for Aesthetic Composition

Kairi Mashio<sup>1</sup>, Kenichi Yoshida<sup>1</sup>, Shigeo Takahashi<sup>1</sup>, and Masato Okada<sup>1,2</sup>

<sup>1</sup> The University of Tokyo, Japan, Chiba, 277-8561 Japan

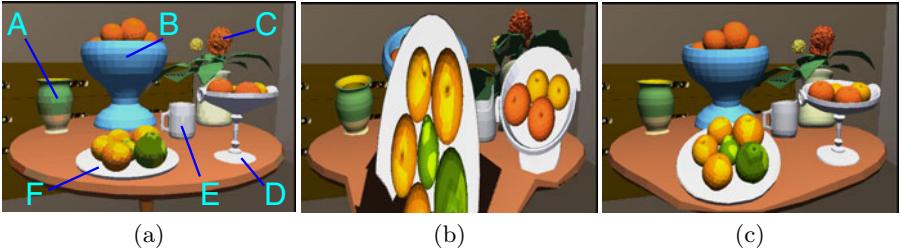
<sup>2</sup> RIKEN, Japan, Saitama, 351-0198 Japan

**Abstract.** Hand-drawn pictures differ from ordinary perspective images in that the entire scene is composed of local feature regions each of which is projected individually as seen from its own vista point. This type of projection, called nonperspective projection, has served as one of the common media for our visual communication while its automatic generation process still needs more research. This paper presents an approach to automatically generating aesthetic nonperspective images by simulating the deformation principles seen in such hand-drawn pictures. The proposed approach first locates the optimal viewpoint for each feature region by maximizing the associated viewpoint entropy value. These optimal viewpoints are then incorporated into the 3D field of camera parameters, which is represented by regular grid samples in the 3D scene space. Finally, the camera parameters are smoothed out in order to eliminate any unexpected discontinuities between neighboring feature regions, by taking advantage of image restoration techniques. Several nonperspective images are generated to demonstrate the applicability of the proposed approach.

## 1 Introduction

In computer graphics, ordinary perspective projection is commonly used to transform 3D scenes onto the 2D screen space to simulate the effects of the pin-hole camera model. On the other hand, hand-drawn pictures often differ from such ordinary perspective images in that the entire scene is composed of local feature areas each of which is projected as seen from its own optimal viewpoint. This type of projection, called *nonperspective projection*, has been synthesized by selecting such local viewpoints so that they retain the visual consistency with the overall scene composition. Furthermore, employing such local viewpoints allows us to emphasize/suppress specific objects and to avoid unexpected occlusions between objects, which is especially useful in drawing 3D map illustrations and artistic paintings.

Several models for nonperspective projection have been proposed by simulating the process of hand-drawn pictures and effects of magnification lenses. Agrawala et al. [1] proposed *artistic multiprojection rendering*, where they projected each individual object from a different viewpoint and merged the projected images pixel by pixel consistently. Kurzion et al. [2] simulated 3D object deformations by bending the sight rays together with hardware-assisted 3D texturing.



**Fig. 1.** Nonperspective image of a 3D scene generated using our method. (a) Ordinary perspective image. (b) Nonperspective image with non-smooth blending of local viewpoints. (c) Nonperspective image with smooth blending of local viewpoints.

Singh et al. [3] presented a generalized approach, called a *fresh perspective*, to generate smoothly deformed nonperspective images by arranging 3D local cameras and interpolating the corresponding camera parameters in the 3D scene space, and extended it to realize an interactive interface that directly manipulates 2D image deformations [4]. Deforming the target 3D objects allows us to simulate the effect of composing nonperspective images as seen from multiple viewpoints. This idea can be traced back to the concept of *view-dependent geometry* proposed by Rademacher [5]. Martin et al. [6] implemented *observer dependent deformations* by relating user-defined nonlinear transformations of 3D objects to the viewing distance and camera orientation. Takahashi et al. optimally deformed 3D terrain models for generating mountain guide-maps [7] and occlusion-free route navigation [8].

Basically, the process of designing nonperspective images consists of three steps: (1) segmenting the entire scene into local feature regions, (2) selecting the optimal viewpoint for each local region, and (3) generating a composite image from local regions each of which is projected as seen from its viewpoint. In practice, the above existing approaches successfully provide us with an interface for designing such nonperspective images, however, its design still needs a time-consuming trial and error process. This is because the design of nonperspective images inherently has excessive degrees of freedom in nature, and existing interfaces do not provide any guidelines for selecting appropriate viewpoints for local feature regions and aesthetically blending the corresponding camera parameters within the 3D scene space. Figure 1(b) shows such an example where excessive image distortions occur due to inappropriate selection and interpolation of local viewpoints assigned to the feature objects as indicated in Figure 1(a).

This paper presents an approach to generating an aesthetic nonperspective image from a set of segmented feature regions in the target 3D scene. Our approach automates the latter two steps in the above nonperspective design process, the selection of optimal viewpoints for feature regions and its visually plausible blending in the final composite image. The viewpoint selection is accomplished by employing adaptive Monte Carlo sampling of the viewpoint entropy function around the user-specified global viewpoint. The obtained local viewpoints will be interpolated over the 3D scene to compose a nonperspective image in a

visually plausible manner, by taking advantage of image restoration techniques. The proposed formulation allows us to synthesize nonperspective images only by adjusting a single parameter that controls the smoothness of the viewpoint transition over the entire 3D scene. Several design examples including Figure 1(c) will be exhibited to demonstrate the feasibility of the proposed approach.

The remainder of this paper is organized as follows: Section 2 explains how we can calculate the optimal viewpoint for each feature region using the Monte Carlo sampling technique. Section 3 introduces a 3D field of camera parameters as the data structure to retain the calculated local viewpoints. Section 4 describes a method of interpolating such local viewpoints over the 3D scene using the image restoration techniques. After presenting several design examples in Section 5, Section 6 concludes this paper and refers to possible future extensions.

## 2 Selecting Optimal Viewpoints

Our design process starts with an ordinary perspective image projected from the user-specified global viewpoint, and then tries to find nonperspective image deformation by incorporating a locally optimal viewpoint calculated for each feature region. In our approach, users are requested to specify a set of representative objects so that the target 3D scene can be segmented into local feature regions. In this section, we explain how to select the optimal viewpoint for each feature region by the Monte Carlo sampling of the corresponding viewpoint entropy function.

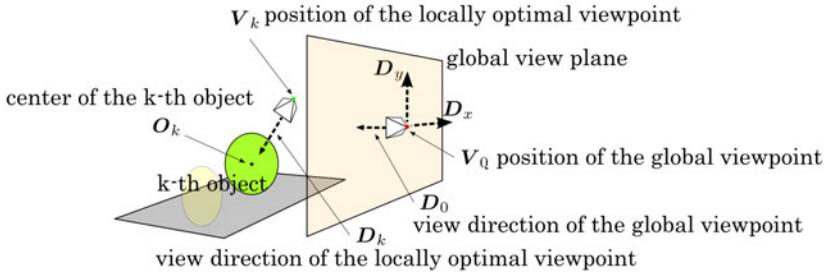
### 2.1 Viewpoint Entropy

For finding the optimal viewpoints for representative objects, we employ the formulation of the *viewpoint entropy* proposed by Vázquez et al. [9]. In this formulation, the optimality of a viewpoint is evaluated by the following entropy function:

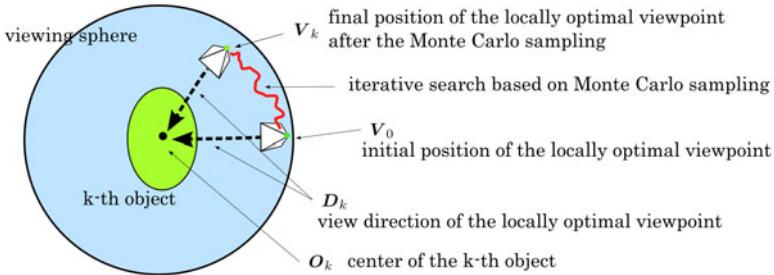
$$I = - \sum_{j=0}^N \frac{A_j}{S} \log_2 \frac{A_j}{S}, \quad (1)$$

where  $N$  represents the number of faces that compose a 3D polygonal model of the target object, and  $A_j$  is the projected area of the  $j$ -th polygonal face of that model on the 2D screen. Here, we assume that  $A_0$  represents the area of the background region and thus the overall area of the screen  $S$  holds the following condition:  $S = \sum_{j=0}^N A_j$ . Eq. (1) implies that the definition of the viewpoint entropy is equivalent to that of the Shannon entropy, when we think of the relative ratio  $A_j/S$  as the probability of the corresponding face visibility. Thus, we can find the optimal viewpoint by exploring the best balance of the face visibilities based on the entropy formulation.

This search for optimal viewpoints can be replaced with a more sophisticated version based on surface curvatures [10] if we put more weight on the salient features of the object shapes, or semantic-based approaches if the target scene has



**Fig. 2.** Global viewpoint and locally optimal viewpoints. Each viewpoint is represented by the 3D geometric position and view direction.



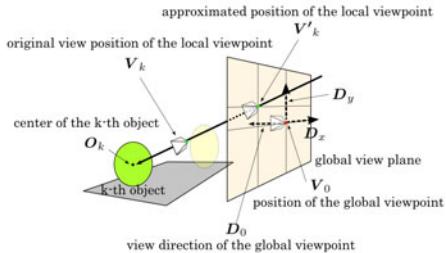
**Fig. 3.** Optimal viewpoint selection on the viewing sphere using Monte Carlo sampling

domain-specific context [11][12][13]. Readers can refer to an excellent overview [14] of this interesting subject.

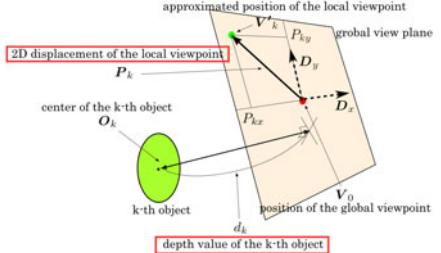
## 2.2 Viewpoint Selection Based on Monte Carlo Sampling

Before going into the details of the actual viewpoint calculation, we introduce our notation. In this paper, we represent each viewpoint by the 3D geometric position and view direction. Figure 2 illustrates this notation where the position and view direction for the  $k$ -th object are denoted by  $\mathbf{V}_k$  and  $\mathbf{D}_k$ , respectively, while  $\mathbf{V}_0$  and  $\mathbf{D}_0$  correspond to those for the user-specified global viewpoint for the initial perspective projection. In addition, we refer to the plane perpendicular to the view direction  $\mathbf{D}_0$  as the *global view plane*, and denote the horizontally and vertically aligned vectors that span the plane by  $\mathbf{D}_x$  and  $\mathbf{D}_y$ .

In practice, we can compute the optimal viewpoint of the  $k$ -th object as follows: First, the viewpoint  $\mathbf{V}_k$  for the  $k$ -th object is initialized to  $\mathbf{V}_0$ , and the view direction  $\mathbf{D}_k$  is set to be a vector emanating from  $\mathbf{V}_0$  to the center of the  $k$ -th object  $\mathbf{O}_k$ . For exploring the best viewpoint, as shown in Figure 3, we iteratively perform the Monte Carlo sampling in the vicinity of the current position  $\mathbf{V}_k$  over the viewing sphere of the  $k$ -th object, and replace it with the new position if it has a higher value of the viewpoint entropy than the current value. We also set  $\mathbf{D}_k$  to the vector from  $\mathbf{V}_k$  to  $\mathbf{O}_k$  at each updating step. This



**Fig. 4.** Projecting the local viewpoint onto the global view plane associated with the global viewpoint



**Fig. 5.** 2D displacement vector and the depth value of the viewpoint to be stored in the 3D field of camera parameters

iterative computation will terminate when we reach a local maximum around the the position of initial global viewpoint, and employ the corresponding  $V_k$  and  $D_k$  as the optimal viewpoint for that object.

### 3 3D Field of Camera Parameters

Before synthesizing nonperspective images by interpolating the obtained local viewpoints, we introduce a 3D field of camera parameters on regular grid samples in the target 3D scene. This is because, from the 3D field of camera parameters, we can retrieve a locally optimal viewpoint at any 3D position in the target scene by trilinearly interpolating the eight nearest neighbor samples. Furthermore, the regular grid samples permit us to naturally incorporate image restoration techniques into this framework when smoothly blending the locally optimal viewpoints in a visually plausible manner. In what follows, we first introduce the representation of a locally optimal viewpoint for each object in our framework, then store the corresponding viewpoint information into the 3D field of camera parameters, and finally synthesize the corresponding nonperspective image with reference to that 3D field.

#### 3.1 Approximating the Locally Optimal Viewpoints

In the 3D field of camera parameters, we first store the 2D displacements of the local viewpoints from their original position for generating the initial perspective image, while having fixed other parameters such as the focal length, view direction, up vector, size of the screen and its orientation. This setting effectively allows us to control the associated nonperspective projection without worrying about the excessive degrees of freedom inherent in the design of such nonperspective images. However, we cannot describe a locally optimal viewpoint assigned to each representative object only with the 2D displacement, and thus have to approximate the position of the local viewpoint by projecting it to the global view plane along the corresponding view direction. Figure 4 shows such

an viewpoint approximation where the original viewpoint for the  $k$ -th object  $\mathbf{V}_k$  is projected onto the new position  $\mathbf{V}'_k$  as:

$$\mathbf{V}'_k = \frac{(\mathbf{O}_k - \mathbf{V}_0) \cdot \mathbf{D}_0}{(\mathbf{O}_k - \mathbf{V}_k) \cdot \mathbf{D}_0} (\mathbf{V}_k - \mathbf{O}_k) + \mathbf{O}_k. \quad (2)$$

This approximation effectively allows us to store the information about the local viewpoints into the 3D field of camera parameters.

### 3.2 Storing Camera Parameters in the 3D Field

As described earlier, the 2D displacement of each locally optimal viewpoint can be obtained by calculating the difference in position between the new viewpoint and original global viewpoint along the global view plane, as  $\mathbf{V}'_k - \mathbf{V}_0$ . Thus, as shown in Figure 5, the 2D displacement vector  $\mathbf{P}_k = (P_{kx}, P_{ky})$  to be stored in the 3D field can be given by

$$P_{kx} = (\mathbf{V}'_k - \mathbf{V}_0) \cdot \mathbf{D}_x \quad \text{and} \quad P_{ky} = (\mathbf{V}'_k - \mathbf{V}_0) \cdot \mathbf{D}_y. \quad (3)$$

In addition to the 2D viewpoint displacement, we also associate the distance between the global view plane to the target representative object, as its depth value. Figure 5 again indicates the depth value of the  $k$ -th object, which can be given by

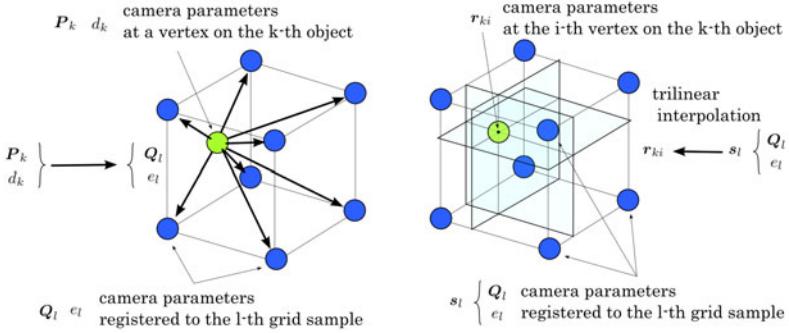
$$d_k = (\mathbf{O}_k - \mathbf{V}_0) \cdot \mathbf{D}_0. \quad (4)$$

These 2D displacement vectors and depth values for local viewpoints are recorded as the camera parameters at the grid samples in the 3D field.

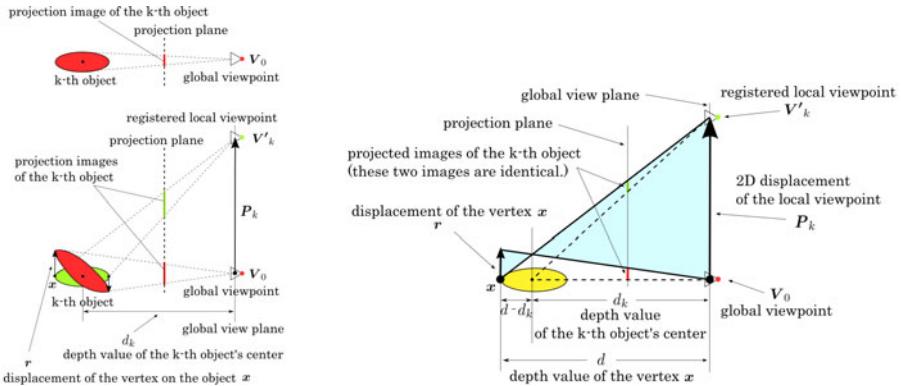
### 3.3 Synthesizing Nonperspectives with the Camera Parameters

Suppose that the target scene has been composed of a set of 3D meshes. For synthesizing the nonperspective image of the target scene, we have to retrieve the camera parameters of each vertex on the meshes from the 3D field of camera parameters. However, since the mesh vertices do not necessarily coincide with the grid samples of the 3D field geometrically, we have to register the aforementioned 2D displacement vectors and depth values of the mesh vertices appropriately to the 3D field. In our implementation, we propagate the camera parameters of each mesh vertex to the eight nearest grid samples as shown on the left of Figure 6. Here, in this figure, the 2D displacement of the viewpoint  $\mathbf{P}_k$  and depth value  $d_k$  for the  $k$ -th object are distributed to the  $l$ -th grid sample as  $\mathbf{Q}_l$  and  $e_l$ , respectively. On the other hand, when synthesizing nonperspective images by referring to the 3D field of camera parameters, we compute the camera parameters at the 3D position of a mesh vertex by trilinearly interpolating those at the eight nearest grid samples as shown on the right of Figure 6.

In practice, we simulate the effects of nonperspective projection by projecting each mesh vertex with reference to the corresponding 2D displacement of viewpoint and depth value. Figure 7 shows such an example where the change in the



**Fig. 6.** Registering camera parameters to the 3D field: (Left) Storing the camera parameters to the nearest eight grid samples. (Right) Retrieving the camera parameters by trilinearly interpolating those at the nearest eight grid samples.



**Fig. 7.** Simulating nonperspective projection by deforming the target object

**Fig. 8.** Displacement of a vertex according to the change in the position of the corresponding viewpoint

viewpoint position results in the deformation of the target object in the final nonperspective image. Note that, when synthesizing the nonperspective image in this scheme, we fix the center of each representative object in the 3D scene space. This implies that, as shown in Figure 8, we have to translate the vertex  $x$  along the global view plane by:

$$\mathbf{r} = \left( \frac{d - d_k}{d_k} \right) \mathbf{P}_k \quad \text{and} \quad d = (\mathbf{x} - \mathbf{V}_0) \cdot \mathbf{D}_0, \quad (5)$$

where  $d$  represents the depth value of the vertex  $\mathbf{x}$  in the global view coordinates. This formulation successfully allows us to synthesize nonperspective images if an appropriate 3D field of camera parameters is provided. Note that the registered locally optimal camera parameters are not used directly; they will be smoothed

out when synthesizing aesthetic nonperspective images as described in the next section.

## 4 Blending Local Viewpoints

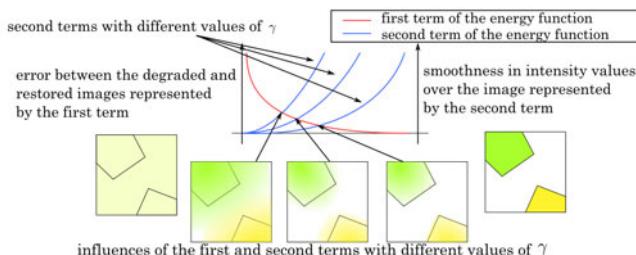
In generating final nonperspective images, we have to fully smooth out the 3D field of camera parameters. Otherwise, we may incur unexpected visual inconsistency such as folds and breaks over the projected images around possible sudden changes in the camera parameters. In this section, we introduce a method of smoothly blending the locally optimal viewpoints that correspond to the representative objects in the 3D scene, by taking advantage of image restoration techniques [15].

### 4.1 Energy Function for Blending Viewpoints

Basically, image restoration techniques restore a degraded noisy image by seeking smooth spatial change in intensity value, while minimizing the difference between the degraded image and newly restored image. For example, we can transform the degraded image  $\mathbf{g}$  into the restored image  $\mathbf{f}$ , by minimizing the following function  $F$  (Figure 9):

$$F = \frac{1}{2} \sum_i^L (g_i - f_i)^2 + \frac{\gamma}{2} \sum_{(j,k)} (f_j - f_k)^2, \quad (6)$$

where  $f_i$  and  $g_i$  represent the intensity values at the  $i$ -th pixel in  $\mathbf{f}$  and  $\mathbf{g}$ , respectively, and  $L$  corresponds to the number of pixels in the images. In addition,  $\sum_{(j,k)}$  indicates that we calculate the summation of the corresponding terms for all the pairs of neighboring pixels in the image. (The  $j$ -th and  $k$ -th pixels are immediate neighbors in this case.) On the right side of Eq. (6), the first term represents the error between the degraded image  $\mathbf{g}$  and restored image  $\mathbf{f}$ , and the second term evaluates the smoothness of the intensity values over the restored image  $\mathbf{f}$ , while  $\gamma$  denotes the relative ratio of the reliability of  $\mathbf{f}$  with



**Fig. 9.** Image restoration process by minimizing an energy function. The parameter  $\gamma$  controls the smoothness of the restored image.

respect to  $\mathbf{g}$ . The best restored image can be obtained by minimizing the energy function  $F$ .

A similar strategy can be used to achieve the smooth interpolation between local viewpoints, by smoothing out the transition of camera parameters in the 3D space while maximally respecting the calculated locally optimal viewpoints. In practice, this problem can be formulated by introducing an energy function that evaluates the quality of the 3D field of camera parameters, with reference to the definition of the energy function in Eq. (6). Let  $s_l$  denote the displacement of the position at the  $l$ -th grid sample in the 3D field of camera parameters, as shown on the left of Figure 6. We define the energy function that evaluates the quality of the 3D field of camera parameters as

$$E = \frac{1}{2} \sum_{i=1}^N (s_i^n - s_i^o)^2 + \frac{\gamma}{2} \sum_{(j,k)} (s_j^n - s_k^n)^2, \quad (7)$$

where  $s_i^o$  and  $s_i^n$  represent the displacement before and after the optimization, and  $\sum_{(j,k)}$  indicates that we calculate the summation of the corresponding terms for all the pairs of neighboring grid samples in the 3D field. In the same way as in Eq. (6), the first term on the right side of Eq. (7) represents the difference between the 3D field of camera parameters obtained by calculated local viewpoints and its updated version, while the second term represents the smoothness of the updated 3D field. Here, the parameter value  $\gamma$  controls the smoothness of the 3D field of camera parameters and can be adjusted by users for tweaking aesthetic continuity of the interpolated local viewpoints in the nonperspective scene.

## 4.2 Iteratively Updating Camera Parameters

Given an appropriate parameter value  $\gamma$ , we explore the 3D field of camera parameters that minimizes the energy function in Eq. (7), which provides us with the optimal interpolation of local viewpoints assigned to the representative objects. In our approach, this is achieved by iteratively updating the camera parameters using the steepest descend method to seek the minimal value of the energy function. For each update, we replace the old displacement of the  $l$ -th grid sample  $s_l^o$  with the new one  $s_l^n$ , by zeroing the derivative of Eq. (7) with respect to  $s_l^n$ , as

$$\frac{\partial E}{\partial s_l^n} = (s_l^n - s_l^o) + \gamma \sum_{m \in \text{nearest}} (s_l^n - s_m^o) = 0. \quad (8)$$

Here,  $\sum_{m \in \text{nearest}}$  represents the summation of the corresponding terms for all the nearest grid samples of the  $l$ -th sample, while the number of the nearest samples is 6 in this 3D case because the  $l$ -th sample has two neighbors along each of the three coordinate axes. This means that we can rewrite Eq. (8) as

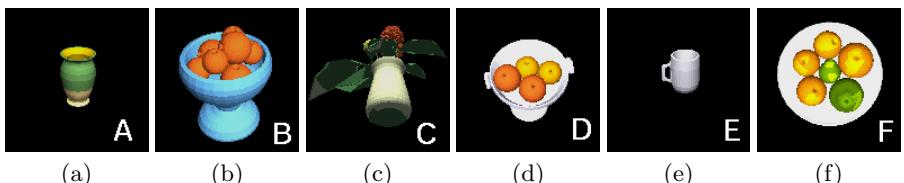
$$s_l^n = \frac{s_l^o + \gamma \sum_{m \in \text{nearest}} s_m^o}{1 + 6\gamma} \quad (9)$$

By applying the above updates to all the grid samples, we can iteratively optimize the 3D field of camera parameters. If the difference in the energy function between the current field and updated field becomes less than some specific threshold, we consider that the 3D field has been converged to the optimal one and terminate this iterative process. Finally, we employ the final 3D field of camera parameters to synthesize aesthetic compositions of nonperspective images.

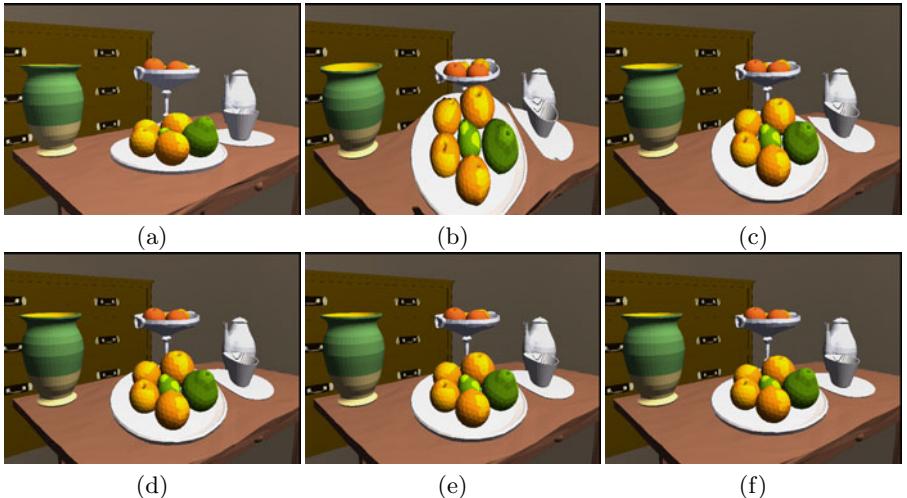
## 5 Results

This section provides several design examples to demonstrate the capability of the proposed method. First, Figure 10 shows the set of representative objects specified by users for synthesizing the nonperspective scene in Figure 1, where each of which is projected as seen from its own optimal viewpoint. Note that Figure 1(a) shows how the representative objects look like in the ordinary perspective projection where they are projected with the given global viewpoint. We cannot inevitably avoid undesirable breaks and folds in the projected image as seen in Figure 1(b) by just storing the locally optimal viewpoints for the representative objects to the 3D field of camera parameters, because we have several stepwise discontinuities in the 3D field without any smoothing operations. After having smoothed out the 3D field using our approach, we can successfully interpolate the locally optimal viewpoints over the scene to synthesize a visually plausible nonperspective image as shown in Figure 1(c). Note that we use  $\gamma = 10.0$  when minimizing the energy function in Eq. (7) in this case.

This design example nonetheless suggests that we cannot necessarily retain the locally optimal view of each representative object, for example, as seen through the comparison between optimal local views in Figure 1(a), (b), and (c) and the final synthesized image in Figure 1(c). Thus, we like to observe how the nonperspective image deformation changes according to the parameter value of  $\gamma$ . Figure 11 presents such an example, where we use another set of representative objects as our target scene, and generate nonperspective images with different values of  $\gamma$ . Figure 11(a) shows the corresponding ordinary perspective image, while Figures 11(b)-(f) present how the different values of  $\gamma$  will influence on the overall interpolation of local viewpoints. The associated results reveal that the small value of  $\gamma$  will not maintain the smoothness of the 3D field of camera parameters, as shown in Figure 11(b). On the other hand, the large value of  $\gamma$  will excessively decrease the flexibility of the nonperspective image deformation,



**Fig. 10.** Representative objects for synthesizing the nonperspective image in Figure 1, where each object is projected as seen from its own optimal viewpoint



**Fig. 11.** Results with different values of  $\gamma$ . (a) Ordinary perspective image. Nonperspective images with smooth fields obtained by optimizing the energy function with (b)  $\gamma = 3.0$ , (c)  $\gamma = 7.0$ , (d)  $\gamma = 10.0$ , (e)  $\gamma = 12.0$ , and (f)  $\gamma = 20.0$ .

and thus make the projected image close to the original perspective image, as shown in Figure 11(f). This implies that we can control the visual plausibility of the nonperspective image deformation by controlling only one parameter  $\gamma$ , which is still manually adjusted by users in our implementation. Adaptively controlling the value of  $\gamma$  in the 3D target scene would help us improve the aesthetic interpolation of local viewpoints.

## 6 Conclusion

This paper has presented an approach for automatically synthesizing aesthetic nonperspective images by interpolating locally optimal viewpoints assigned to the given set of representative objects in the scene. The plausible interpolation of local viewpoints can be accomplished by adjusting only single control parameter so that the optimal view of each representative object can be sufficiently reflected in the final nonperspective image. Currently, our scheme cannot necessarily preserve all the locally optimal viewpoints in the synthesized nonperspective image, for example, when the corresponding viewpoints are quite different in its direction from the global viewpoint, or when different viewpoints are assigned to multiple representative objects within a relatively small region. Adaptively adjusting the control parameter in 3D scene space, together with non-regular samples of camera parameters, will allow us to respect the spatial configuration of representative objects in the final nonperspective image. Applying the present approach to nonperspective animations will demand more systematic control of camera parameters, and thus will be an interesting theme for future research.

## Acknowledgements

This work has been partially supported by Japan Society of the Promotion of Science under Grants-in-Aid for Scientific Research (A) No. 20240020, Scientific Research (B) No. 20300033, and Young Scientists (B) No. 17700092, and the CASIO Science Promotion Foundation.

## References

1. Agrawala, M., Zorin, D., Munzner, T.: Artistic multiprojection rendering. In: Eurographics Rendering Workshop 2000, pp. 125–136 (2000)
2. Kurzion, Y., Yagel, R.: Interactive space deformation with hardware-assisted rendering. *IEEE Computer Graphics and Applications* 17(5), 66–77 (1997)
3. Singh, K.: A fresh perspective. In: Proceedings of Graphics Interface 2002, pp. 17–24 (2002)
4. Coleman, P., Singh, K., Barrett, L., Sudarsanam, N., Grimm, C.: 3D scene-space widgets for non-linear projection. In: Proceedings of GRAPHITE 2005, pp. 221–228 (2005)
5. Rademacher, P.: View-dependent geometry. In: Proceedings of SIGGRAPH 1999, pp. 439–446 (1999)
6. Martín, D., García, S., Torres, J.C.: Observer dependent deformations in illustration. In: Proceedings of the 1st International Symposium on Non-Photorealistic Animation and Rendering (NPAR 2000), pp. 75–82 (2000)
7. Takahashi, S., Ohta, N., Nakamura, H., Takeshima, Y., Fujishiro, I.: Modeling surperspective projection of landscapes for geographical guide-map generation. *Computer Graphics Forum* 21(3), 259–268 (2002)
8. Takahashi, S., Yoshida, K., Shimada, K., Nishita, T.: Occlusion-free animation of driving routes for car navigation systems. *IEEE Transactions on Visualization and Computer Graphics* 12(5), 1141–1148 (2006)
9. Vázquez, P.-P., Feixas, M., Sbert, M., Heidrich, W.: Viewpoint selection using view entropy. In: Proceedings of Vision Modeling and Visualization Conference (VMV 2001), pp. 273–280 (2001)
10. Lee, C.H., Varshney, A., Jacobs, D.W.: Mesh saliency. *ACM Transactions on Graphics* 24(3), 659–666 (2005)
11. Vázquez, P.P., Feixas, M., Sbert, M., Llobet, A.: Realtime automatic selection of good molecular views. *Computers and Graphics* 30(1), 98–110 (2006)
12. Sokolov, D., Plemenos, D., Tamine, K.: Viewpoint quality and global scene exploration strategies. In: Proceedings of International Conference on Computer Graphics Theory and Applications (GRAPP 2006), pp. 184–191 (2006)
13. Mühlner, K., Neugebauer, M., Tietjen, C., Preim, B.: Viewpoint selection for intervention planning. In: Proceedings of Eurographics/IEEE-VGTC Symposium on Visualization, pp. 267–274 (2007)
14. Elmquist, N., Tsigas, P.: A taxonomy of 3D occlusion management for visualization. *IEEE Transactions on Visualization and Computer Graphics* 14(5), 1095–1109 (2008)
15. Geman, D.: Random fields and inverse problems in imaging. In: Y. Vardi, M. (ed.) CAV 1998. LNCS, vol. 1427, pp. 113–193. Springer, Heidelberg (1998)

# Focus and Context in Mixed Reality by Modulating First Order Salient Features

Erick Mendez<sup>1</sup>, Steven Feiner<sup>2</sup>, and Dieter Schmalstieg<sup>1</sup>

<sup>1</sup> Graz University of Technology

{mendez,schmalstieg}@icg.tugraz.at

<sup>2</sup> Columbia University

feiner@cs.columbia.edu

**Abstract.** We present a technique for dynamically directing a viewer’s attention to a focus object by analyzing and modulating bottom-up salient features of a video feed. Rather than applying a static modulation strategy, we inspect the original image’s saliency map, and modify the image automatically to favor the focus object. Image fragments are adaptively darkened, lightened and manipulated in hue according to local contrast information rather than global parameters. The goal is to suggest rather than force the attention of the user towards a specific location. The technique’s goal is to apply only minimal changes to an image, while achieving a desired difference of saliency between focus and context regions of the image. Our technique exhibits temporal and spatial coherence and runs at interactive frame rates using GPU shaders. We present several application examples from the field of Mixed Reality, or more precisely Mediated Reality.

## 1 Introduction

Focus and context (F+C) describes the concept of visually discriminating interesting objects (focus) from nearby related objects (context). In mixed reality, or more precisely mediated reality, F+C can be used to draw the attention of a user to certain objects of a scene, be it to indicate danger, to supplement detailed information, or to guide the user to a destination. There exist several strategies to achieve this, for example, by changing the color, overlaying augmenting artifacts, or distorting the area of attention [13].

Not all of the F+C strategies are universally effective, and the choice of technique depends heavily on the focus and context objects themselves. For example, suppose we were to draw the attention of the user to a particular region by drawing a circle around it. The effectiveness of this technique will depend on parameters such as the color or size of the circle and whether they offered sufficient contrast with the rest of the image.

Consequently, an adaptive discrimination of scene objects is needed, i. e., the F+C strategy has to be constantly adjusted. Specifically, in mixed reality applications based on live video, one cannot easily impose constraints on visible objects or camera movements. The technique presented in this paper therefore



**Fig. 1.** Attention direction with our technique. (Left) Original image. (Right) Result of our modulation technique. The saliency of pixels is automatically decreased in the context area and increased in the focus area. Notice how the reflections of windows in the context area are slightly diminished yet not suppressed entirely. Pixel values of the modulated image differ on average by 1.28% from their counterparts in the original image.

analyzes the video image in real time and computes the saliency for every fragment. In an image, an object is said to be visually salient if it stands out more than its surrounding neighborhood [24]. Our technique modifies the image by changing lightness and color contrast in order to make a desired focus region have the highest visual saliency. This is done in such a way that the applied changes are minimal and spatial and temporal coherence are respected. Consequently, the legibility of the context region is affected as little as possible.

All computations are carried out with GPU shaders in real time. We present several application examples from the field of mixed reality, including directing the attention of the user to an object in a search task, and highlighting a possibly dangerous object during car maintenance.

## 2 Background

There exists an extensive amount of work on trying to model the visual saliency of an image. Techniques that have been tried include adding contextual information [24], non-parametric approaches [10], face detection [3] and using trained samples over large datasets [8].

Saliency is usually defined as a measure of how contrasting a particular location is from its surrounding in dimensions such as color, orientation, motion, and depth. Treisman and Gelade use *dimension* to refer to the range of variations, and *feature* to refer to values in a dimension (e.g., orientation and lightness are dimensions, while horizontal and dark are features) [25]. The *conspicuities* of a location are measures that represent how contrasting this location is to its surroundings in each of the dimensions. The visual *saliency* of a location is the combination of all its conspicuities. A scene's *saliency map* is a map of the saliency values on each location in the image.

In this paper, we consider bottom-up saliency, which only relies on instantaneous sensory input and not on higher level factors such as intentions. Itti et al. [7] provided a computational model for analysis of several bottom-up stimuli. From this work, we adopt those that lend themselves to pixel-wise manipulation, namely lightness, red-green color opponency and blue-yellow color opponency. This can be seen as a form of in-place F+C [13]. Highly conspicuous objects in the lightness dimensions are either dark objects in light surroundings or vice versa. Color opponency is based on the opponent process theory [5], stating that we perceive colors by processing the differences between opponents, red-green and blue-yellow [7]. This means that, for example, if two objects, one blue and one green, were placed on a yellow canvas, the blue will be more conspicuous due to its color opponency to yellow.

There is much evidence that there is a correlation between our visual attention and the saliency map. Ouerhani et al. [17] and similarly Santella et al. [19] used an eye tracker to confirm that there exists a relationship between the saliency map and human visual attention. Lee et al. [14] went one step further by using the saliency map to track objects to which the user is attending.

Practically any change made to the image will modify its saliency map. Blurring, (de-) saturating, harmonizing and distorting are typical operations that implicitly change the saliency of the image. During the last few years there has been an increasing interest in directing the attention of the user through saliency manipulation for volume rendering [12], non-photorealistic stylization [19] and geometry [11]. However, previous work concentrates on creating salient features in focus regions rather than applying subtle modifications to existing images. For example, Kim et al. [12] present a visual-saliency-based operator to enhance selected regions of a volume. This operator is a part of the visualization pipeline and is applied before the image is generated, in contrast, our work receives an existing image as input and pursues the manipulation of its existing salient regions.

Closest to our intentions is the work by Su et al. [23] on de-emphasizing distracting image regions and by Bailey et al. [2] on subtle gaze direction. Su et al. focused on so-called second-order saliency by modulating variations in texture to redirect the user's attention to specific locations. Bailey et al. applied first-order modulations to the focus, only when the user is not looking there, as determined by an eye tracker. In contrast to these techniques, our technique works with dynamic live video and can thus support mediated reality applications with arbitrary scenes and without requiring an eye tracker.

### 3 Conspicuities Analysis

We modulate the image on a frame-by-frame basis, in order to reflect the latest information available in the case of a video feed. Achieving this demands two general steps that depend on the input before composing the final image:

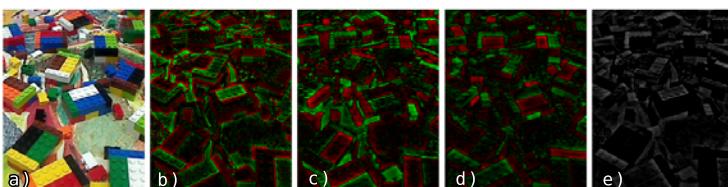
- **Conspicuities analysis.** During this step, we compute the conspicuities of the whole image to have a measure of the naturally salient objects in the scene.
- **Conspicuities modulation.** (Section 4) Once we have quantified the image’s conspicuities, we select and apply the appropriate modulations to the input image. The modulations are done sequentially for each of the conspicuities at multiple levels of coarseness, and ultimately produce an image whose highest saliency is in the focus area.

Since the saliency of a location is given by a combination of its conspicuities, the final goal is then to modulate these conspicuities. We now present how the saliency of the image is analyzed so that modulation can take place. For simplicity, we refer to lightness as  $L$ , red-green color opponency as  $RG$  and blue-yellow color opponency as  $BY$ .

Figure 2 illustrates the calculated conspicuities for  $L$ ,  $RG$  and  $BY$ . For purposes of illustration, we show positive values in green and negative values in red; for example, dark objects near light objects have a negative conspicuity and it is shown in red. The right-most image shows the arithmetical average of the conspicuities representing the total saliency of the image.

In order to compute the conspicuity map of an image one must follow three steps: a) feature extraction, b) conspicuity computation and c) normalization. A feature is the value on a given dimension in a given location, while conspicuity is the difference of the feature value of that location with its surroundings. Finally, the saliency is a combination of the conspicuity values.

*Feature extraction.* We use a slightly modified version of the conspicuity computation provided by Itti et al. [7]. That work computed the saliency of a location in the lightness, red-green color opponent, blue-yellow color opponent and orientation dimensions. We only compute the first three dimensions by converting the image from the RGB to the CIE  $L^*a^*b^*$  space, which already encodes the lightness and opponent colors dimensions similar to the work of Achanta et al. [1] (the initial RGB values are given in simplified sRGB with a Gamma of 2.2; we assume the observer is at  $2^\circ$ , and use the D65 illuminant).



**Fig. 2.** Illustration of conspicuities. These images illustrate the conspicuities of the different dimensions used in this paper. Green is used for positive values, while red is used for negative. a) Original image. b) Lightness conspicuity. c) Red-Green Color Opponency conspicuity. d) Blue-Yellow Color Opponency conspicuity. e) Saliency map.

*Conspicuities computation.* The next step computes the conspicuities for each separate dimension. This is done by calculating the difference between a location and its neighborhood by the center-surround technique. This technique calculates the relation of a location to its surroundings by checking the difference across fine and coarse levels. Accessing finer and coarser levels of the image is done by using the built-in hardware mip-mapping as suggested by Lee et al. [14]. The center-surround technique as described by Itti et al. [7] is:

Let  $k_i$  be the fragment's feature  $k$  on pyramid level  $i$ . The conspicuity  $c_k$  is then defined as:

$$c_k = \frac{\sum_{n=0}^{n=2} \sum_{m=n+3}^{m=n+4} k_n - k_{n+m}}{p}, \quad (1)$$

where  $k_i$  is the conspicuity value  $k \in \{L, RG, BY\}$  at mipmap level  $i$  and  $p = 6$ . The value of  $p$  is the number of levels of coarseness being considered. An important difference between our work and others is that we do not use the absolute value of the conspicuities before adding them up. This allows us to keep the sign of the conspicuity, e.g. if the current location (fragment) has a negative lightness conspicuity then it is a dark location on light surroundings.

*Normalization.* We use a normalization that considers the global conspicuity maxima, as described by Lee et al. [14]. This has the effect of reducing non-contributing high-frequency artifacts on each dimension. The normalized conspicuity is then defined as follows:

Let  $\max(c_k)$  be the maximum conspicuity value of the feature  $k$  of the whole image. The normalized conspicuity at every location  $n_k$  is then

$$n_k = \frac{c_k}{\max(c_k)}, \text{ where } k \in \{L, RG, BY\}. \quad (2)$$

The calculation of the normalization weights is a computationally demanding task. We allow the computation of these weights to be done every few frames. The number of frames is determined by the current framerate of the system in order to maintain at least 15fps.

*Saliency computation.* The saliency of a location is the arithmetical average of its normalized conspicuities. The computation of the saliency  $s$  at a given location is:

$$s = \frac{\sum n_k}{d}, \text{ where } k \in \{L, RG, BY\} \text{ and } d = 3 \quad (3)$$

The value of  $d$  is the number of dimensions being considered.

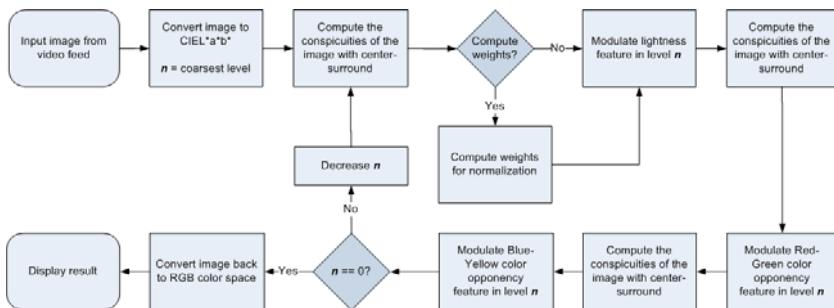
## 4 Conspicuities Modulation

Modulating a location means either reducing its conspicuity (in the case of context) or increasing it (in the case of focus). To modulate the conspicuity of a location we must, for example, lighten or darken it, or reduce or increase its “redness” or “greenness”. A brute force method of attention direction would heavily modify the image to highlight the object of interest, for example, by

turning all the context area to black. Although effective, such a technique also eliminates the information of the context, since all fragments are suppressed whether it was necessary or not. The purpose of our algorithm is to apply the appropriate amount of change to the image to try to minimize the information lost in the context. This means that the modulations are not all applied to every fragment equally. For example, some fragments might need a strong red-green modulation, but no blue-yellow modulation. Modulation is performed in three sequential steps: first, lightness modulation, then red-green opponency and finally blue-yellow opponency. The order of this sequence is given by the sensitivity of the human visual system in each dimension; we are more sensitive to lightness than to chromatic information, and we are more sensitive to red-green stimulus than to blue-yellow [22] [21].

*Classification.* Before we can modulate the image, we need a classification of the objects in the scene. This classification tells us whether we want to direct attention towards a given object (focus) or away from it (context). In this paper, we assume that this classification is given through a-priori knowledge of the scene or user interaction.

*Modulation thresholds.* The set of conspicuities encodes the difference of every location with its surroundings. However, in order to modulate conspicuities adaptively, we need a threshold for every dimension to which a location's conspicuities can be compared. For automatic determination of thresholds, we empirically found the average conspicuity values of the focus object to be a good value. Throughout the rest of this paper, the threshold values will be referred as  $t_k$ , where  $k$  is the given dimension.



**Fig. 3.** Flow chart of our technique. This flow chart illustrates the iterative process of our modulation technique.

*Modulation steps.* The modulation procedure is a series of analyses and adjustments. The analysis step generates information such that the adjustment step can verify further changes in a given location are necessary. The adjustment step modifies the location in each of its dimensions separately in order to reduce its conspicuity (or increase it depending on the classification). These steps are done multiple times from coarse to fine levels of the image pyramid by using the

built-in mip-map capability of the graphics unit. This allows changes affecting a large region to occur early in the process, while later steps progressively refine the result. An implicit benefit of starting with coarse resolutions is that modulation of lower frequencies introduces less noticeable artifacts. Each analysis and adjustment is carried out in a fragment shader that executes in a render pass on the current image. The total number of passes necessary for modulation can be expressed as:  $2 + 6 * n$ , where  $n$  is the number of pyramid passes. Two passes are necessary to convert the image to and from CIE L\*a\*b\* space and six passes are necessary for the adjustments in the three considered dimensions and their respective analyses. The analysis step computes the conspicuity values of the input image as described in section 3. Figure 3 shows a flowchart with a detailed description of all the necessary steps.

*Compute the modulation values to be applied.* To know the modulation value to be applied to the current location (such as making it more blue or less yellow), we first verify that the location's absolute conspicuity value exceeds the given threshold. The conspicuity value is provided by the analysis step. If it does not exceed it, then no modulation is necessary and we leave the feature value unmodified. Otherwise, we compute the difference between the threshold and the current conspicuity value. This difference is then used as a modulation value to be applied to the current feature. At this point, it is important to remember the roles of conspicuity and feature. A conspicuity is the difference between a location and its surroundings. We cannot modify a conspicuity directly, instead, it is modified indirectly by changing the feature values of the location.

Let  $m_k$  be the modulation to be applied to the location,  $c_k$  the conspicuity of the location and  $t_k$  the threshold of the conspicuity, where  $k$  is the given dimension.

$$m_k = \begin{cases} 0 & \text{where } c_k < t_k \\ c_k - t_k & \text{otherwise.} \end{cases} \quad (4)$$

Before applying this modulation, a few checks must be performed in order to avoid unpleasing visual artifacts. For example, in the chromatic channels, we must also take care that the modulation should not flip the hue completely, i.e., blue never becomes yellow, and red never becomes green (or vice versa). This is done by preventing a flip on the sign of the feature value, a positive value (e.g., red) cannot become negative (e.g., green).

*Coherence.* The modulation process seeks to reduce the amount of change in the original image. A naïve implementation only considers the appropriate values for each location without regard to the global coherence of the image. As a result, noise artifacts (typically chromatic) can occur on the final image, as illustrated in Figure 4. Such artifacts happen when two spatially close locations are matched to different modulation values, when the conspicuity of a location is increased (focus) and the original chromatic values are close to zero. To resolve this problem, we compute the average between the modulation computed at the previous pyramid level and the current level. A side effect of this filtering is that the strength of the modulation is smoothed, leading to more visually pleasing results.



**Fig. 4.** Spatial coherence. This figures illustrates the problem arising from the emphasis of contrast in the focus area. A tungsten light on the metallic surface of this car model caused these particular artifacts. Notice the red or green dust in the middle image. a) Original image. b) Image affected by naïve conspicuity enhancement. c) Image after applying our spatial coherence technique.

As stated before, the computation of the normalization weights is amortized over several frames, depending on the current framerate. If the amortization period is too long, the changes in normalization weights may be drastic. This can introduce temporal discontinuity artifacts between two adjacent frames. We therefore compute the weight and thresholds using a sliding average over a history of a few frames.

Once our modulation value has been computed and all checks have been performed that are necessary to ensure that no drastic changes occur (either spatially or temporally), we can apply this value to the location. To decrease the conspicuity of the context we merely subtract the change value we computed from the given location's feature. This has the effect of reducing the distance between the current conspicuity value and the threshold. To increase the conspicuity of a location, instead of subtracting, we simply add the change value we computed to the given location's feature. This has the effect of increasing the distance between the current conspicuity and the threshold.

Let  $f_k$  be the feature value of the location and  $f'_{lk}$  the modulated feature value and  $m_k$  the modulation to be applied, where  $k$  is the given dimension.

$$f'_{lk} = \begin{cases} f_k - m_k & \text{if the location is marked as context} \\ f_k + m_k & \text{if the location is marked as focus} \end{cases} \quad (5)$$

## 5 Results and Applications

We now present several examples that use our technique. First, we contrast naïve modification with our technique. Then, we show applications that involve image modifications to achieve mediated reality effects on either the focus or the context of a scene. The first is a classical search task, where the system tries to direct the attention of the user towards an item in the field of view. The second application tries to direct attention to an object that is not the main actor in the current task. All of the images shown in this paper were computed on a 3.0 GHz Intel Dual Core CPU with an NVIDIA GTX280 graphics card. We used

GLSL for our fragment shaders and framebuffer objects for texture handling. The video feed used for our examples was at a 640x480 resolution. The lowest framerate we experienced was 15fps. Computing the modulation of an image on a single pyramid level was achieved in 1.023ms; computing it with 7 render passes was done in 36.47ms.



**Fig. 5.** Comparison between traditional techniques and our work. a) by Gaussian blur with a kernel size of 4. b) by total suppression of the context. c) by augmentation with a 4-pixel-wide red border. d) by de-saturation. e) our approach.

*Comparison of adaptive saliency modification and naïve image modification.* There exist multiple techniques for attention direction. One may, for example, point at the object of interest with an augmentation. One may also de-saturate the context, blur it, or plainly suppress it entirely. However, the effectiveness of such attention direction techniques depends on the objects in the scene themselves. For example, color, shape and size of augmentations are heavily influenced by the objects in the field of view. An arrow pointing at an object will be only as effective as it is contrasting with its background, and text labels only be effective when displayed on adequate surfaces. In the same sense, traditional pixel-wise attention direction techniques rely on the assumption that the focus object has the necessary properties to stand out from the modified context. A de-saturation technique will be ineffective if the focus object has little saturation itself. Similarly, lightness modulation depends on the objects in the scene, highlighting an item on a game of chess (where multiple instances of each item occur in either black or white) might be give ambiguous results unless combined with other modulations such as saturation. Moreover, properties such as the strength of the modification are typically assigned a priori, for example, the kernel sized used for a Gaussian blur. This becomes critical in a mixed or mediated reality scenario where we do not have control over which objects are visible in the scene, as the user is allowed free camera movements. Figure 5 compares different attention direction techniques side-by-side. The most effective of these techniques is figure 5 b), where the context is entirely suppressed. The least effective is figure 5 d), due to the lack of saturation of the focus itself.

*Reminding the user of dangerous objects in their proximity.* Consider the task of maintaining a car engine. The engine presents surfaces that are dangerous to the touch (e.g., too hot), of which the user should remain constantly aware. These objects, however, are not the main interest of the user. The user is engaged in a task that does not require the direction of the system. Our technique is well suited for this task by constantly reminding the user of the location of the dangerous surfaces (focus), while minimizing the obstruction of the main working area (context). Figure 6 shows an example of our technique before and after modulation. Bailey et al. [2] suggest that modulation does not need to be constantly applied. Instead, modulating the image for one second is already capable of directing the user's gaze towards the focus area. This modulation can then be repeated every few seconds to keep attracting the gaze of the user towards the dangerous surfaces.

*Finding objects task.* Finding a particular object in a collection of similar objects is a common task presented in mixed reality. In this example, we present a shelf where multiple books are visible to the user. The problem is to find a specific book in this shelf. Figure 5 illustrates this application, which is reminiscent of that presented by Schwerdtfeger et al. [20]. The unmodulated image contains multiple items with colorful covers, all competing for the user's attention. Our system then subtly suggests to the user where the target book is. Notice that the target book does not have any particularly salient features, such as a colorful book cover or large letters, yet the system is capable of accentuating its contrast and diminishing that of the context. This modulation can then be repeated every few seconds to keep attracting the gaze of the user towards the dangerous surfaces.



**Fig. 6.** One advantage of our technique is that it is not entirely detrimental to the Context region. This is helpful in situations where we want to highlight an object that is not the main actor on the current task of the user. In this image we highlight an item that is not set for maintenance (below the user's hand), but may be dangerous and should be avoided.

## 6 Discussion and Conclusion

To find out how much our technique changes the image, we computed the average pixel difference between the original image and after our modulation

procedure. This was done by calculating the squared root of the sum of squared differences in the RGB space divided by the number of pixels in the image. The total average pixel difference across all images in this paper between modified and original versions is 1.34%.

We have presented a technique for the modulation of visual saliency for attention direction, specifically designed for interactive mediated reality applications. Image saliency is analyzed in real time, and modulations are made in such a way that a desired distance between focus and context regions is achieved with minimal changes. This technique can be seen as a way of reducing the contrast of the context area and of increasing it in the focus area. This contrast manipulation takes place on the lightness and color opponents dimensions. We have shown a number of application examples that indicate the usefulness of this approach.

However, it should be noted that the use of the technique presented in this article may not always be warranted or even possible. For some applications, the perception of the context may be unimportant. In other cases, there may be moving or blinking objects in the context, which may not be sufficiently suppressed with the pixel-wise techniques we presented. Moreover, the modulations we have shown are incapable of directing the attention to an object that is not present in the image, nor can they show the direction on which this object may be found. In those cases, guiding attention direction through direct augmented overlays, such as described by Schwerdtfeger et al. [20] may be more viable.

We are considering several future directions. Specifically, we are planning on incorporating additional saliency models. This would allow us to possibly modify the strength of our modulation depending on other conspicuity dimensions, such as texture variation as demonstrated by Su et al. [23]. An important future direction is the validation of our research via user studies. We have already performed preliminary user tests with an eye tracker [15]. We found that modulation of bottom up stimuli can effectively direct the attention of users. The initial applications to mixed reality look promising on an informal level, but clearly a quantitative analysis is needed to fully understand the involved phenomena. To this aim, we are currently setting up a user study performed with an eye tracker to investigate the effects of different image modifications. We hope to evaluate not only the effectiveness of attention direction towards the focus, but also the circumstances of preservation of the contextual information.

**Acknowledgments.** This work was partially funded through the HYDROSYS project (EC-FP7 grant 224416).

## References

1. Achanta, R., et al.: Salient Region Detection and Segmentation. In: Gasteratos, A., Vincze, M., Tsotsos, J.K. (eds.) ICVS 2008. LNCS, vol. 5008, pp. 66–75. Springer, Heidelberg (2008)
2. Bailey, R., et al.: Subtle Gaze Direction. In: ACM TOG, vol. 28(4), pp. 1–14 (2009)
3. Cerf, M., et al.: Predicting human gaze using low-level saliency combined with face detection. In: Platt, J.C., Koller, D., Singer, Y., Roweis, S.T. (eds.) NIPS. MIT Press, Cambridge (2007)

4. Engel, S., et al.: Colour tuning in human visual cortex measured with functional magnetic resonance imaging. *Nature* 388(6637), 68–71
5. Hering, E.: *Outlines of a Theory of the Light Sense*. Harvard University Press, Cambridge (1964)
6. Hurvich, L.M., Jameson, D.: An opponent-process theory of color vision. *Psychological Review* 64(6), Part I, 384–404 (1957)
7. Itti, L., et al.: A model of saliency-based visual attention for rapid scene analysis. *IEEE TPAMI* 20(11), 1254–1259 (1998)
8. Judd, T., et al.: Learning to predict where people look. In: *ICCV 2009* (2009)
9. Koch, C., et al.: Shifts in selective visual attention. *Human Neurobiology* 4, 219–227 (1985)
10. Kienzle, F., et al.: A nonparametric approach to bottom-up visual saliency. In: Scholkopf, B., Platt, J.C., Hoffman, T. (eds.) *NIPS*, pp. 689–696. MIT Press, Cambridge (2006)
11. Kim, Y., Varshney, A.: Persuading Visual Attention through Geometry. *IEEE TVCG* 14(4), 772–782 (2008)
12. Kim, Y., Varshney, A.: Saliency-guided Enhancement for Volume Visualization. *IEEE TVCG* 12(5), 925–932 (2006)
13. Kosara, R., et al.: An interaction view on information visualization. In: *EUROGRAPHICS*, pp. 123–137 (2003)
14. Lee, S., et al.: Real-Time Tracking of Visually Attended Objects in Interactive Virtual Environments. In: *ACM VRST*, pp. 29–38 (2007)
15. Mendez, E., et al.: Experiences on Attention Direction through Manipulation of Salient Features. In: *IEEE VR 2010 PIVE Workshop*, pp. 4–9 (2010)
16. Niebur, E.: Saliency Map. Scholarpedia (January 10, 2010),  
<http://www.scholarpedia.org/article/SaliencyMap>
17. Ouerhani, N., et al.: Empirical validation of the saliency-based model of visual attention. *Electronic Letters on Computer Vision and Image Analysis* 3(1), 13–24
18. Rosenholtz, R.: A simple saliency model predicts a number of motion popout phenomena. *Vision Research* 39(19), 3157–3163 (1999)
19. Santella, A., Decarlo, D.: Visual interest and NPR: an evaluation and manifesto. In: *Proceedings of NPAR*, pp. 71–150 (2004)
20. Schwerdtfeger, B., Klinker, G.: Supporting Order Picking with Augmented Reality. In: *IEEE and ACM ISMAR*, pp. 91–94 (2008)
21. Sangwue, S.J., et al.: *The colour image processing handbook*, 1st edn. Chapman and Hall, Boca Raton (1998)
22. Spillman, L.: *Visual Perception: The Neurophysiological Foundations*. Academic Press, London (1990)
23. Su, S., et al.: De-Emphasis of Distracting Image Regions Using Texture Power Maps. In: *Workshop on Texture Analysis and Synthesis at ICCV* (2005)
24. Torralba, et al.: Contextual guidance of eye movements and attention in real-world scenes: The role of global features in object search. *Psychological Review* 113(4), 766–786 (2006)
25. Treisman, A.M., Gelade, G.: A feature-integration theory of attention. *Cognitive Psychology* 12, 97–136 (1980)

# An Interactive Interface for Lighting-by-Example

Hai Nam HA<sup>1</sup>, Christophe Lino<sup>2</sup>, Marc Christie<sup>2</sup>, and Patrick Olivier<sup>3</sup>

<sup>1</sup> Posts & Telecom. Institute of Technology, Hanoi, Vietnam

<sup>2</sup> IRISA/INRIA Rennes

<sup>3</sup> Newcastle University, UK

**Abstract.** Lighting design in computer graphics is essentially not a random process but one driven by both a technical and aesthetic appreciation of lighting. In some applications, the result of the lighting design process is a 2D image derived by rendering a 3D scene. Users with limited understandings of manipulation of lighting parameters may have difficulties in properly modifying the lighting parameters in order to achieve desired lighting effects. We present and demonstrate an approach to lighting design in applications where the expected result of the lighting design process is a 2D image. In this approach, the lighting-by-example method using perception-based objective function is used in combination with an interactive interface in order to optimize lighting parameters for an object or a group of objects individually, and the visual results of these separate processes are combined (utilizing 3D depth information) in the seamless generation of a final 2D image.

**Keywords:** Computer graphics, lighting design, interactive interface, lighting-by-example.

## 1 Introduction

Conventional lighting design is a repeated process of manipulating lighting parameters and testing the changes in lighting effects, and the lighting process stops when desired lighting goals have been achieved. Apparently, conventional lighting design is a knowledge-based process rather than a random process as experienced users would know how to adjust light parameters better than inexperienced users. Manipulating scene parameters, in general, and lighting parameters in lighting design process, in particular, requires heavy interactions between users and graphic tools. Hence, interactions in graphic tools have been continually improved in order to equip users with convenient ways of interactions [1, 6, 7, 8, 9].

In graphic tools such as 3D Studio Max, lighting work starts with identifying the types of light sources and their characteristics that will be used in a scene. By investigating the purpose and intent of lights, a user finds a real-world counterpart for each of every light used in the scene. Lights are then positioned at intended locations. The next step will be editing the properties of the lights. In commercial graphic tools, most of the light properties such as intensities, colour, attenuation and shadow parameters are manipulated through a window using keyboard. For some properties of lights such as light position and direction, there is an alternative for specifying them through mouse-based interactions such as dragging and dropping. Indeed, mouse-based

interactions for light positioning would be more efficient and intuitive than specifying the values through a window using keyboard as users normally know relatively where the light should be and which direction the light should point at rather than the exact values for light position and direction. In reality, commercial graphic tools such as 3D Studio Max, Maya, Light Wave support mouse-based interactions for manipulating light position and direction. Where mouse-based interactions cannot be easily applied such as specifying values for falloff and hotspot of lights, context menus are normally used in graphic tools to enhance the performance of accessibility to functionalities.

The ultimate goal of research on human-machine interactions is to empower users with convenient ways of achieving the desired goals with less effort using limited resources. Particularly, interactions in lighting design aims to provide users with convenient ways of achieving desired lighting effects by taking advantages of the combination of existing lighting design approaches.

## 2 Interactive Design through Independent Object Lighting

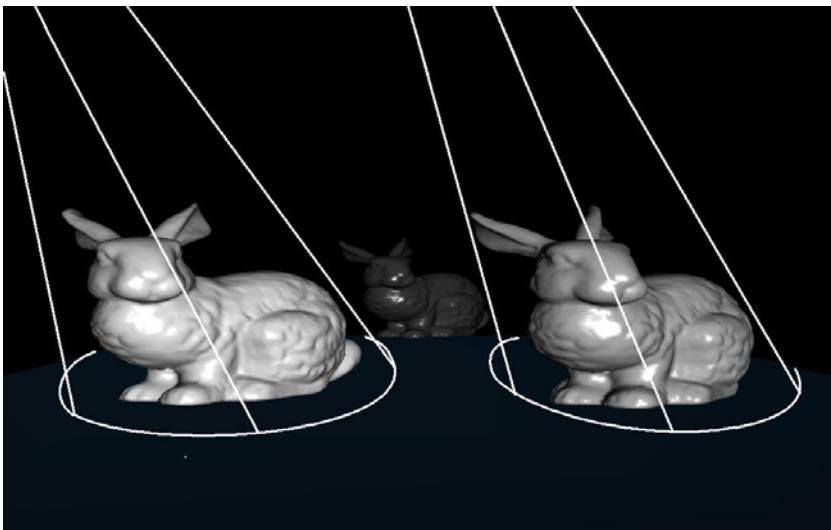
Interactive lighting design is problematic, as although we may have a clear notion of how we want each individual object in a scene to be lit, any change in lighting parameters effects the illumination of all objects simultaneously. One solution, which is already a common practice in graphic design, is to produce 2D images from 3D scenes and use 2D image processing techniques to merge different source images (i.e. source images that have been differently lit or have been manipulated in some way). There are a number of drawbacks to working with 2D images; in particular, that 3D information is not taken into consideration. Indeed, when 2D images are merged together without taking advantages of 3D information, unanticipated effects may occur (for example, if no account is taken of object occlusion).

In fact, creating lighting effects, either using 2D compositing, or by modifying the parameters of scene lights directly, is not an intuitive process, even for experienced graphic artists. To address this problem, we propose the extension of the work presented so far. By integrating the optimization schemes developed in [2,3,4] (for both *perception-based lighting* and *lighting-by-example*) we have developed an interface that allows users to design the scene lighting on an object-by-object basis. Developing an interactive approach requires us to extend of our optimization framework and develop mechanisms by which we can more or less independently specify and modify the lighting for different objects in a scene.

Two methods are proposed which can be contrasted in two respects, the degree to which they treat each object in the scene independently, and the nature of the result of the lighting process. Both approaches take as a starting point (if desired) either the results of a perception-based lighting optimization process, or a lighting-by-example process. In this way we envisage *interactive lighting design* as a matter of fine tuning the illumination of each object in a scene. In the first approach, 3D Interactive Lighting Design (3D-ILD), the outcome of the lighting design process is a 3D scene in which objects are differently lit using spotlights, as is commonly used in interactive graphics applications. In the second approach (2D-ILD) a set of lights that is specific to each object is optimized, and the visual results of these separate processes are combined (utilizing 3D depth information) in the seamless generation of a 2D image.

## 2.1 3D Interactive Lighting Design (3D-ILD)

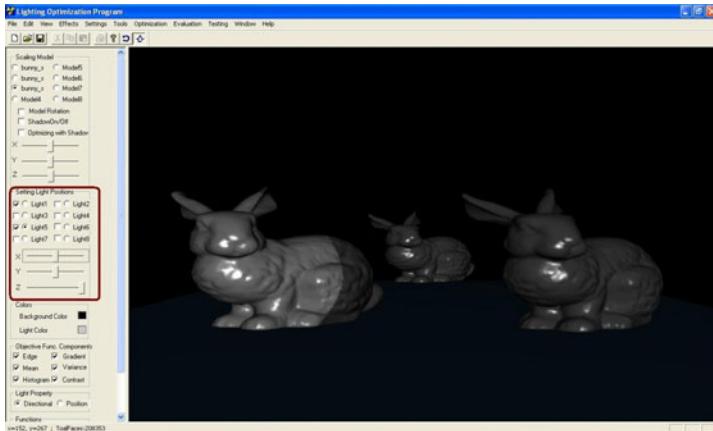
For most graphics applications, we anticipate that the result of a lighting design system is a configuration of lights with respect to the scene elements. Though this design process takes place by reference to a single viewpoint (that is, the optimization process) the viewpoint selected is typically one that is characteristic or exemplifies the contexts in which an object is likely to be viewed. Thus the resulting lighting configuration can be integrated into a scene which includes other objects and light. We have realized this 3D interactive lighting design process through the use of spotlights which are used to light each object individually. A spotlight provides localized and directional illumination. Illumination is restricted to within a specified cone (i.e. the beam of light). This beam of light can be controlled through the specification of both width of cone and the nature of the drop off function.



**Fig. 1.** Object is lit by a spotlight with an appropriate focus cone

Figure 1 illustrates the idea of using spotlights to optimize lighting for objects separately. In this scenario, there are 3 objects in the scene and for simplicity 2 spotlights are used to light the leftmost and rightmost objects, and an ambient light was used to create the background illumination. The cone around each object represents the focus of a spotlight. Apparently, the cone parameters must be adjusted such that whole object is in the cone of the spotlight.

The first step of the interactive design process involves setting up appropriate parameters for the spotlight, the position, orientation, and cut-off angle, such that the cone of the spotlight focuses on the object of interest without lighting adjacent objects (see figure 2(a)). This is sometimes impossible due to the close proximity of other object, and occlusions, in the 3D scene. In such cases the user simply has to adjust the light so as to keep lighting on unintended objects to a minimum. Once the parameters



**Fig. 2.** The leftmost object is not completely in the cone of the spotlight

of spotlights for different objects have been set up, the next step is to select the spotlight to be optimized (see figure 2(b)).

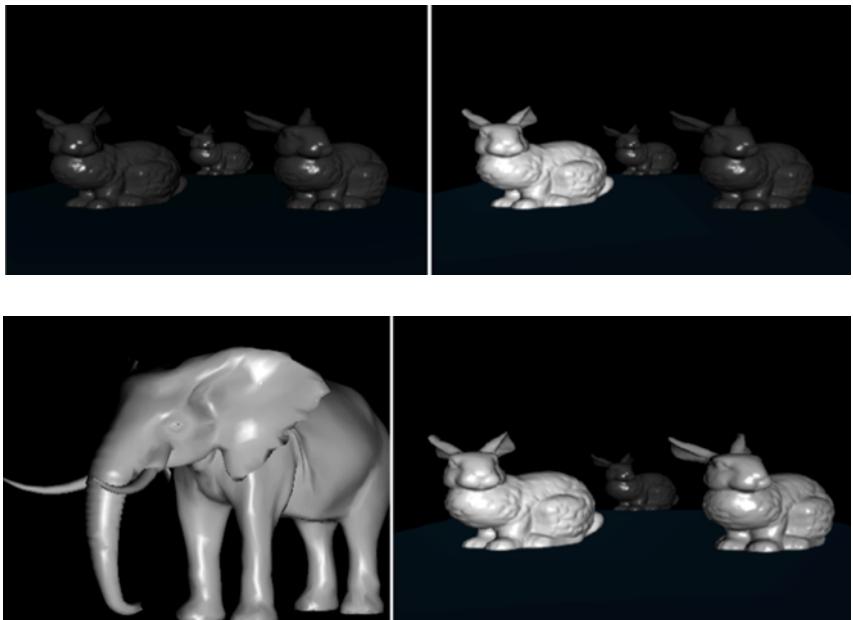
The final step is to choose the optimization technique to be employed. This can be either ideal *perception-based lighting optimization* [3,4] which will optimize the lighting parameters of the spotlight such that the visual properties of the objects in the scene are maximized; or *perception-based lighting-by-example* [5] which aims to capture the lighting effect of a 3D example and recreate it for the current object (see figure 2(c)). Our current experimental implementation supports up to 6 spotlights. Figure 3 shows an example in which lighting for objects are separately optimized using spotlights. Figure 3(a) shows the original scene for which there is a default direction lighting originating from behind and above the camera. Figure 3(b) shows the result of applying the ideal perception-based lighting optimization approach to the left-most bunny rabbit. Figure 3(c) shows the example object (an elephant) that is the target for lighting the rightmost bunny rabbit using perception-based lighting by example. Finally, figure 3(d) shows the final result for both processes. The figures demonstrate how the objects can be independently lit in a 3D scene without significantly affecting the lighting of the other objects in the scene.

The optimization steps of the interaction took 15 steps (93 seconds on a Windows PC, with P4 3.00GHz processor, 1G RAM and GeForce 7600 GT graphics card) for the ideal perception-based lighting optimization and 28 steps (154 seconds on a Windows PC, with P4 3.00GHz processor, 1G RAM and GeForce 7600 GT graphics card) for the perception-based *lighting-by-example* optimization.

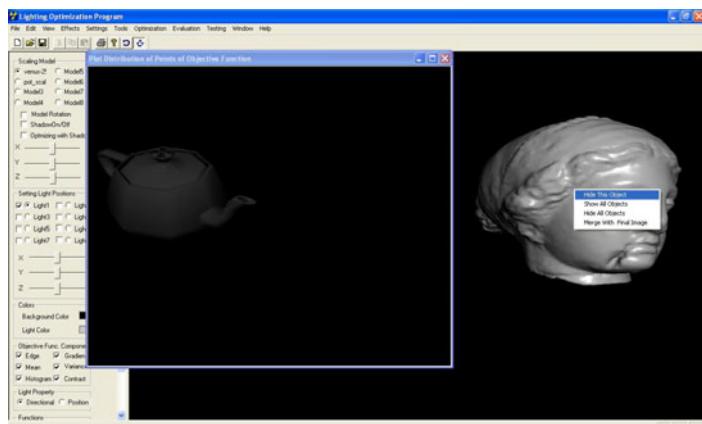
## 2.2 2D Interactive Lighting Design (2D-ILD)

Where the result of the lighting design process is a 2D image (for example, in graphic design for static digital media and print) we can design the lighting for each object in the scene by optimizing each object individually. Figure 4 illustrates this process, by which the other objects in a scene are hidden and the techniques that we developed in [3,4,5] are deployed to optimize the lighting for a single object. Having optimized

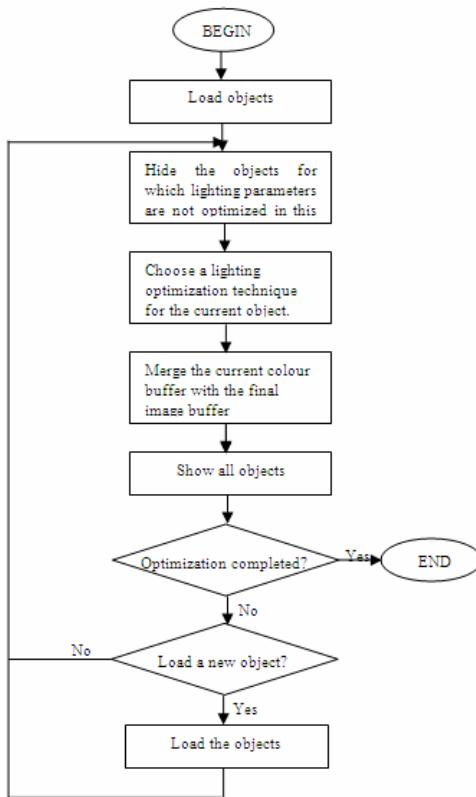
each object separately, the resulting 3D scenes are merged into a final (2D) composition using depth information which is retained from the 3D renderings. The final image buffer is displayed in a separate window.



**Fig. 3.** Lighting parameters separately optimized using spotlights: original set-up (top-left); applied to left-most bunny only (top-right); target for right-most bunny (bottom-left); and final result after both processes (bottom-right).



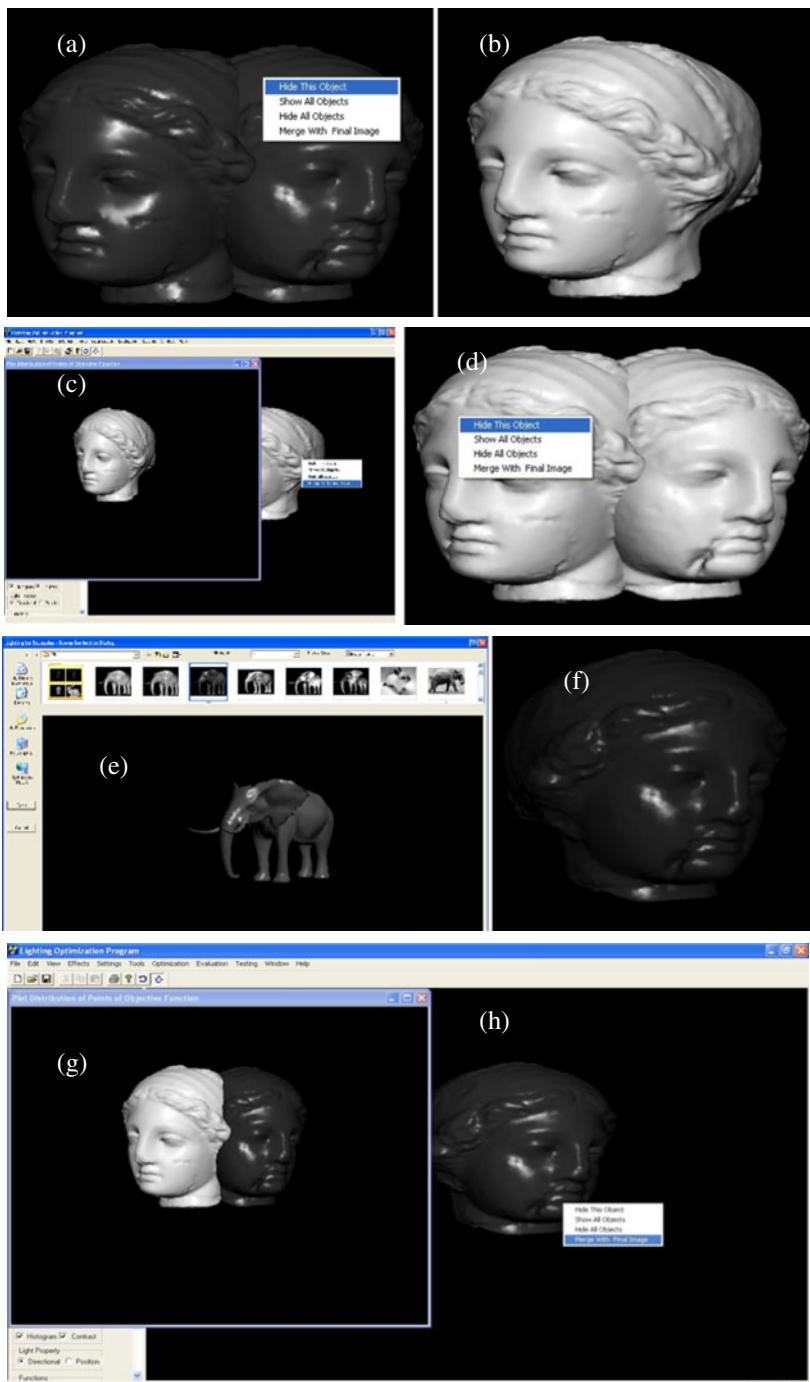
**Fig. 4.** Interactive interface of the lighting design application



**Fig. 5.** The workflow for optimizing lighting parameters for an individual object in 2D Interactive Lighting Design

Figure 5 is a diagram that illustrates the use of interactive interface to individually optimize lighting parameters for objects. Firstly the objects of the 3D scene are loaded and the graphic designer then uses a context menu to hide the objects that are not to be optimized in current loop. The objects for which lighting parameters are to be optimized in this loop remain visible. At this point the user can also manipulate the visible objects with operations such as: rotating, scaling and moving the objects. When the user is satisfied with current set-up of the objects, he can proceed the next step of choosing an appropriate lighting optimization, either ideal perception-based lighting, perception-based *lighting-by-example* or wavelet-based *lighting-by-example*. The optimization for the current setup of objects proceeds and the colour buffer is then merged to the final image buffer. The user reveals all the invisible objects, and if lighting parameters have been optimized for all objects and the optimization process stops. If not, the user will start to optimize lighting parameters for the next object and new objects may also be loaded to the scene.

Figure 6 shows a worked example in which the lighting for each of the faces on a two-headed bust is separately designed and then recombined into a final image. In this process a context menu for an object provides a number of options:



**Fig. 6.** Lighting two heads differently using different lighting design

1. *Hide This Object*: hide the currently selected object, to allow the optimization of the remaining (visible) objects in the scene.
2. *Hide All Objects*: hide all objects of the 3D scene, used when a user is about to load a new object to the scene and lighting parameters are only to be optimized for the newly loaded object.
3. *Show All Objects*: show all objects of the 3D scene, used when the user wants to view the set-up of the whole 3D scene.
4. *Merge With Final Image*: merges the colour buffer with the final image buffer using available depth information to ensure that objects of the two buffers are merged in the right depth order.

Figure 6(a) shows the selection of the right head as the object to hide, in figure 6(b) the ideal perception based framework is deployed, and in figure 6(c) the colour and depth information is merged in the final 2D image (and a separate depth buffer). Figure 6(d) shows the selection of the left head as the object to hide. As the perception-based *lighting-by-example* framework is deployed, figure 6(e) shows the exemplar, and figure 6(f) shows the result for the right head. Figure 6(g) shows the final merged image in which the spatial information for the scene is preserved but the two heads have been lit very differently.

### 3 Conclusion

In this paper we have explored different ways of using lighting design methods. The use of spotlights in 3D context where the outcome is a 3D scene with optimized lighting parameters has been explored in which each object in the scene was lit by a spotlight that can be individually optimized. The challenge of this approach is that the parameters of the spotlights must be carefully adjusted such that each spotlight can light only one object. Interactive methods developed in this paper aim to demonstrate some ways of using lighting design approaches developed in this research. There should be more different ways of using lighting design approaches developed in this research.

In the current interactive interface, we only implemented a number of basic function as for the purpose of exploring their potential, and as a result our prototype interface was rather simple in the range of interactions it supports for the spotlights and the depth information-supported merging technique. We believe the further development of interaction techniques, to combine the lighting approaches, is likely to greatly enhance the appeal of such a system to novice and expert users. For example, an interface that allows users to change the target values for *lighting-by-example* approach would allow significantly more flexibility. Likewise, the interface for manipulation of light parameters is rather limited in current interactive interface. In many cases, the light positions have been fixed and defined by users. A sketch-based interface for manipulating scene parameters would also contribute to making interactions far more natural for users. Also, in some applications, designing lighting parameters for multiple viewpoints is highly desirable, and this requires interaction and display techniques such as allowing multiple viewports for displaying results of the lighting design process from different viewpoints. Finally, in the current development framework, only

standard lights are used in the system. However, for scenes require complex lighting effects, extended light sources (i.e. physically extended light sources that are not point-like) are needed.

## Acknowledgements

This work was supported in part by the EU FP7 network of excellence IRIS (ICT-231824) Integrating Research in Interactive Storytelling.

## References

1. A Lighting Approach for Computer Graphics (SIGGRAPH 96 - Course 30)
2. Shacked, R., Lischinski, D.: Automatic Lighting Design using a perceptual quality metric. In: Proc. Eurographics 2001, vol. 20(3) (2001)
3. Ha, H.N., Olivier, P.: Perception-based lighting design. In: Theory and Practice of Computer Graphics, Middlesbrough, UK (2006)
4. Ha, H.N., Olivier, P.: Explorations in Declarative Lighting Design. In: Butz, A., Fisher, B., Krüger, A., Olivier, P. (eds.) SG 2006. LNCS, vol. 4073, pp. 160–171. Springer, Heidelberg (2006)
5. Ha, H.N., Olivier, P.: Perception-based lighting-by-example. In: Theory and Practice of Computer Graphics, TPCG (2007)
6. Rusinkiewicz, S., Burns, M., DeCarlo, D.: Exaggerated Shading for Depicting Shape and Detail. ACM Transactions on Graphics (Proc. SIGGRAPH) 25(3) (July 2006)
7. John, K.K., Painter, J.S., Cohen, M.F.: Radiotimization: goal based rendering. In: Proceedings of SIGGRAPH 1993, pp. 147–154 (1993)
8. Jung, T., Gross, M.D., Do, E.Y.-L.: Light pen – sketching light in 3d. In: CAAD Futures (2003)
9. Okabe, M., Matsushita, Shen, Y. L., Igarash, T.: Illumination Brush: Interactive Design of All-frequency Lighting. In: Pacific Graphics (2007)

# Stroking a Cymbidium

Young-Mi Kim and Jong-Soo Choi

Graduate School of Advanced Imaging Science, Multimedia, and Film,  
Chung-Ang University,  
221 Huksuk-dong, Dongjak-gu, 156-756, Seoul, Korea(R.O.K)  
[{frontier,jschoi}@imagelab.cau.ac.kr](mailto:{frontier,jschoi}@imagelab.cau.ac.kr)

**Abstract.** This work is an interactive visualization of an oriental cymbidium using modern technology which our oriental ancestors painted for mental training. During the old days in the orient, people used to wipe cymbidium leaves or painted cymbidium for mental training by having a cymbidium always by their side. Through the act of wiping cymbidium leaves with utmost care, a cymbidium instilled with ancient philosophical ideas is visualized. Just as God breathed life into human nostrils and created a living life form, a cymbidium flower with an excellent fragrance is visualized when a breath is breathed into a cymbidium flower.

**Keywords:** Interactive art, ink-and-wash painting, haptic.

## 1 Introduction

When taking a look at Korea history, ancestors have firmly protected our tradition and culture from numerous foreign invasions and handed them down to the present generation. However, entering the brilliant digital era, it is a fact that traditional values and culture are treated as invaluable and thus neglected. Therefore, there is a need to actively utilize and fuse modern media as a means of inheriting and fostering our tradition. The purpose of this dissertation is to make the original meaning of the ink-and-wash painting, the most aesthetic oriental painting, stand out as well as naturally blending it into the lives of modern people by having a different form of expression. It is the intention of this paper to study the potential of expressing ink-and-wash painting through interaction, and present a direction that can coincide with modern paintings by developing ink-and-wash painting from a traditional aspect through analyzing the theories and techniques instilled in my work.

## 2 Feature and Expressivity of Ink-and-Wash Painting

First of all, the spirituality, idea, and unique characteristic of our ink-and-wash painting will be examined. This means that I intend to find the characteristic of ink-and-wash painting from spirituality and express the spirituality through ink-and-wash painting in my work in order to shed new light on the features of ink-and-wash painting from a modern point of view. Prior to this, I would

like to help understand the act of wiping cymbidium as a means of interaction. Asians has been living with cymbidium as if it is a close friend, and an oriental cymbidium is always placed in noble places. The act of wiping each cymbidium leaves with human hands as if sweeping it off is not to maintain cleanliness but rather to uphold the spirit instilled within the cymbidium. In other words, if one's heart is troubled and there are a lot on ones mind, mental training was carried out with a careful act of wiping cymbidium leaves with a calm mind by having a cymbidium close to oneself. Once each cymbidium is wiped off, the things troubling ones mind is forgotten and ones heart is emptied. This act has great meaning as it plays a role of interacting with the cymbidium painting. While the meaning of a cymbidium painting is also similar to the act of wiping a cymbidium, it is instilled with more meaning. The artist always conducts close observation of the subject before painting it. However, the oriental method of sketching is not depicting the subject while looking at it in detail but making a sketch by memory apart from the subject. Oriental drawing technique expands the artists creative space while relying on visual memory and shape memory by making the most use of ones subjectivity and imagination breaking away from the restraint of the actual subject. What is important is drawing with creativity yet keeping several rules. The curves, and bold and thick lines of a cymbidium must be drawn, and its leaves must shake in the wind and have bones. In this work, cymbidium leaves are visualized while being divided up into leaves curved according to the angle of wiping up or down and cymbidium leaves blowing in the wind.

When closely examining the elements necessary for a cymbidium painting and rules of painting, they are as follows.

**First.** It must not be corrected with a brush again and must be finished with just one stroke. Because a cymbidium painting must be finished with just one stroke, an expression, “stroking a cymbidium” is used just as the title of this dissertation says rather than the expression of “painting a cymbidium”. A bold and confident stroke is instilled with the principle of life, and the blank space contains the principle of the universe.

**Second.** There must be blank space and balance. The blank space makes the artistic shape stand out with its space and comfort, and is an intentional expression for expressing more details than the world expressed on the screen. Furthermore, there must be balance in the cymbidium, and this expresses the straight character and spirit of scholars who did not waver or break down from the tide of the world.

**Third.** When Asians paint, they hold the brush with the right hand and support their body on the ground with their left hand. The energy of the ground is received from the left hand and the hand holding the brush receives the energy from the sky. In the writers work, the left hand feels the energy of the ground by taking the closest side near the root of the cymbidium and its leaves are expressed by the act of sweeping down by feeling the energy from the sky with the right hand.

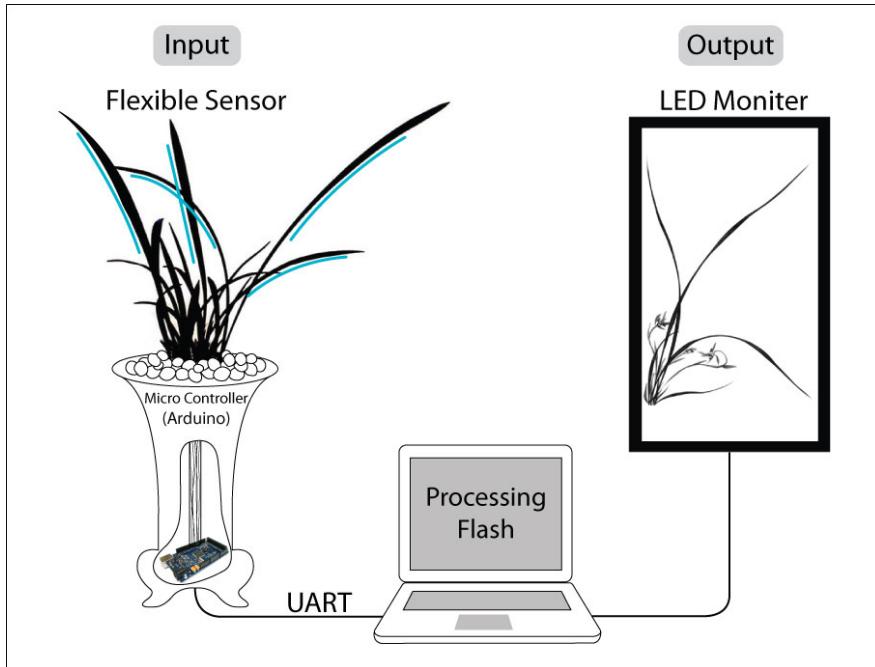
**Fourth.** Ink-and-wash paintings must be read with ones mind. Because an ink-and-wash painting has the artisticity as well as the philosophy and spirituality of the artist, there must be a consensus between the artist and the viewer while looking at the painting. In order to communicate its meaning more clearly, the artists writes the reason for drawing the painting or writes up a poem that goes well with the painting after it is complete. It is a tradition for not only artists but for people observing the painting to write their impressions and stamp their seal on the painting while having balance with blank spaces.

Just like this, even in the writers work, a poem and a writing about its impression going well with the style of painting after the cymbidium and flower is painted is written in harmony and last of all, a piece of an artwork is finished by stamping a seal.

### 3 Methodology

When a cymbidium leaf is selected and wiped using the thumb and index finger as if sweeping it down, a cymbidium leaf is drawn one after another inside the digital frame. A flexible sensor is attached behind cymbidium leaves, and the size of an angle made while a cymbidium leaf is curved appears due to the difference in electrical current through flexible sensors. The micro controller detects this signal and converts the analog signal to a digital signal. Then, the data is sent to the PC using UART communication. The server program can detect UART output signal sent to the desktop from the micro controller and converts this data to a TCP/IP data format. As a result, this format can send this data set from the server to the client. Consequently, the client draws the painting according to this signal after detecting the signal.

When the angle of the curve communicated through the cymbidium leaf is big, the cymbidium leaf in the painting is drawn as curved leaves or leaves blowing in the wind. A total of 6-16 cymbidium leaves are drawn up, and if one breathes air into a flower after drawing a cymbidium leaf, a floral axis and flower is drawn. It takes in the strength of the breath and when it is strong, a full blossomed flower is visualized and when the breath is weak, an unblossomed flower is visualized. When a lot of cymbidium leaves are drawn, only one flower is drawn, and when relatively less leaves are drawn, 2-3 flowers are drawn up. Due to the regularity of the cymbidium painting, the flowers and leaves dont compete with each other and raises the level of perfect as it strikes a balance as a painting of a humble yet strong cymbidium. When the interactive work is finished, a caption or a poem that goes well with the style of the painting is written in the remaining blank space with a stamped red seal. All of these methods communicate that the painting was produced in the same way cymbidium paintings were painted in the old days. The regularity and creativity of the cymbidium painting striking a balance, writing a poem and writing which goes well with the blank space, the seal of the artist, and above all, the greatest significance is that interaction was achieved through the act of wiping the cymbidium leaves with the attitude and spirituality of the person drawing the painting.



**Fig. 1.** Procedures

## 4 Conclusions

Ink-and-wash painting representing oriental art is a collection of a diverse range of art that can be enjoyed along with the beauty of poems, calligraphies, and seals within the painting. This is an artwork for examining the spirituality and figurative perception of oriental cymbidium, finding and succeeding a modern way of expression which was completed by achieving mutual harmony between analog contents and digital technology. This is an interactive art that can complete the artwork only through the active behavior of the viewer and not a work which can be viewed onesidedly by achieving a consensus between the ancestors and modern people.

**Acknowledgments.** This work was supported by Korean Research Foundation under BK21 project, Seoul R&BD Program(TR080601), Seoul Future Contents Convergence (SFCC) Cluster established by Seoul R&BD Program(10570).

# WAVO: An Interactive Ball to Express Light Waves with Wave Equation

Kazushi Mukaiyama

Media Architecture Department, Faculty of System Information Science,  
Future University Hakodate,  
116-2 Kamedanakancho Hakodate, Hokkaido 041-8655, Japan  
[kazushi@fun.ac.jp](mailto:kazushi@fun.ac.jp)  
<http://www.kazushi.info/>

**Abstract.** This paper describes an art work named WAVO. WAVO consists of a PoSC micro-computer, an accelerometer and a 16x16 matrix LED and express moving waves with LED. These waves are calculated by Wave Equation in real-time. Therefore, people can feel beautiful moving of lights which this equation makes. Yet, people can enjoy interacting waves as WAVO has an accelerometer. Interacting with people, this art work notices them fascinating aspects of mathematics.

**Keywords:** art, interaction, physics, wave equation.

## 1 Background

In recent years, we became making electronic crafts using micro-computers at ease, sophisticating electronic technology. And electronic illuminations also spread out in usual after using LED (Light Emitting Diode) mainly. This situation has given us the way to control enriched lighting, for example blinking by an audience's action with sensors. Additionally, it became usual to show illuminations with a not big difficult but small smart control system. In the point of view of Art, this means that new materials and tools are increased for new expressions. It is important for contemporary art expression to use new materials and tools. The result gives us new possibility expands our perception. Therefore, many artists are actively releasing their art works used with electronic technology and computer in these days.

## 2 Purpose

There are people who feel beauty looking at art pieces. And there are people who feel beauty from mathematical formulas. For the people who can't know mathematics well, however, it is difficult to find beauty out from mathematical formulas because they recognize just a list of marks and symbols. WAVO has been made to fix this gap of mathematics. It visualizes a Wave equation (Section 4) and expresses natural water surface to show the beautiful aspects of mathematics. And audiences not only just see wide and dissolve of wave but also make waves swinging WAVO to enjoy much more.

### 3 Related Work

#### 3.1 Illumination Art

Illumination Art means a kind of art works made of neon tubes, light bulbs, LED and so on to express beautiful lightings. We can usually see many illuminations on not only trees but also building walls in Christmas season. We can also decorate illuminations ourselves in our home with ease. There are works which deal with these illuminations as Fine Art. For instance, one of them is Moriwaki's "Rayo=Graphy". [1] This work is a big board hanging on wall, attached many LEDs and light sensors in a grid pattern. Each LED turn off if there is a light and turn on if not. Therefore, an audience's shadow shines when he/she stands in front of it (Figure [1]).

#### 3.2 Generative Art

Generative Art means a kind of art works generated, calculated or composed with mathematical and computing algorithms. It can be explained one of computer graphics to visualize mathematical formulas which man can't image as pictures like Mandelbrot set. But, most of purposes are not to analyze mathematic equations but express something with mathematical formulas as tools. Kimoto's "Imaginary-Numbers" [2] is computer graphics generated dots made with one of non-liner dynamics formulas. We can see wonderful images generated by numbers (Figure [2]).



**Fig. 1.** rayo=graphy [1]



**Fig. 2.** Imaginary-Numbers [2]

#### 3.3 Device Art

Device Art means a kind of art works using electronic parts such as LED, sensors and so on. Most of works are compact and mobility. Also, they looks like some kind of devices in usual. They are not useful but inspiring an audience's creativity as an audience can enjoy their interactions touching them. Hiruta's "Mr. Rolling" [3] is a 80 mm diameter cylinder displaying a human abstract character on a red LED. This abstract character, "Mr. Rolling" is animated comically in the cylinder. Once an audience pushes it, it rolls like a car wheel and "Mr. Rolling" begins to run inside like a hamster (Figure [3]).

**Fig. 3.** Mr. Rolling**Fig. 4.** WAVO

## 4 System

WAVO has a sphere shape (Figure 4) and consists of a PSoC micro-computer (CY8C29466), a 16x16 matrix LED (C-2AA0SRD), an accelerometer (KMX52-1050), a 4-16 line decoder (74HC154), 330 $\Omega$  resistors, a 5 voltage 3-Terminal regulator and a switch. All parts are on two stack circuit boards. A matrix LED is put on the upper board. The other parts are put on the lower. And, there is a 9 voltage cell battery at the bottom. This battery acts a role of the weight, so that WAVO swings like a pendulum when an audience pushes it on a table. In a 16x16 matrix LED the anode common side (row) is controlled in PWM connected to a PSoC micro-computer, and the cathode common side (column) is connected through a 4-16 line decoder for dynamic drive blinking. The LED can display 16x16=256pixels, 32 steps of brightness updating 30 frames per second. The color is red only because there was no other matrix LED in this proper size except it. An accelerometer which gets an audience's action is connected to a PSoC micro-computer. A clear sphere case is made from acrylic. This has been selected the proper case for aesthetics.

Wave equation is a partial differential equation which describes the diffusion of waves. Wave equation in two dimension is expressed as follows; [4]

$$\frac{\partial^2 u}{\partial t^2} = v^2 \left( \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \right) \quad (1)$$

In WAVO, there is a virtual dot which moves on the matrix LED surface sensing a tilt with an accelerometer. The micro-computer calculates wave topological values made by this dot with Wave equation. Then, WAVO expresses the light water surface mapping wave values to LED brightness.

## 5 Conclusion

WAVO has been shown the exhibition called "Make: Tokyo Meeting 03" at Digital Hollywood University Hachioji campus from May 23rd to 24th, 2009. We prepare 3 WAVOs and there is no information for audiences except the title

and abstract and “touching ok”. As the result, we got the following comments and actions of audiences. At first, most of audiences said that light waves of LED were beautiful. Then, they were impressed the mathematical beauty after explaining about Wave equation. On the other hand, there were audiences who screwed WAVO more than audiences swung it. Then, the wave became volute shape and audiences enjoyed looking at volute moves. Also, we got technical comments. For instance, “it might be interesting if there are many of them not just 3 ones”, “what if building a small motor in for dancing automatically?” From the above comments and actions, we could show audiences the aspect of mathematics in which a simple formula makes fascinating effects.

## 6 Future of Media Art

From around 1995, art works which made from electronic technology began to call Media Art and had become popular. [5] At that time, we must use expensive hardware and software to make high quality works. Today, we can personally get hardware and software to make same level quality works at that time. In this situation, we anticipate Media Art will diversify much more. Small and compact works is one of the diversifications. As you can see the contents of mobile phone, a Media Art work is becoming from a big one for only gallery exhibition to a smaller one which can be taken on palm of hand. [6][7] At least, we already have new materials and tools which mean electronic sensors, switches and so on instead of paint brushes. WAVO is one of the evidences in this situation.

## References

1. Moriwaki, H.: rayo=graphy, 1580x2700(mm) (1990)
2. Kimoto, K.: Imaginary-Numbers, Kousakusha (2003)
3. Hiruta, S.: Mr.Rolling, 80x80x50(mm) (2007)
4. Sudo, S.: Physical Mathematics One Point, vol. 9. Kyoritsu Shuppan Co., Ltd. (2002)
5. Sirai, M., Mori, K., Towata, M., Tomari, M.: Media Art Exercise. Filmart-Sha Co., Ltd. (2008)
6. Kobayashi, S., et al.: IAMAS Ubiquitous Interaction Research Group,  
<http://www.iamas.ac.jp/project/ui/>
7. Levin, G., et al.: Mobile Art && Code, <http://artandcode.ning.com/>

# OverWatch: Real-Time Narrative Visuals from Live Performance

Guy Schofield, Rachel Casey, and Patrick Olivier

School of Computing Science, Newcastle University Culture Lab, Kings Walk,  
Newcastle upon Tyne, NE165UL  
[g.p.schofield@ncl.ac.uk](mailto:g.p.schofield@ncl.ac.uk)  
<http://www.theturingtest.co.uk/guyschofield>

**Abstract.** OverWatch is an audio/visual performance for two musicians, involving the production of narrative visuals on-the-fly. Presented in a theatrical context, with the musicians playing beneath a large projection screen, the piece references early silent movies, specifically the expressionist films of the 1920s. However, rather than responding to the formal and narrative content of the images onscreen, the musicians in OverWatch have a direct impact upon the content and structure of the movie. The improvised musical score, translated into MIDI and audio data is used to provide camera, lighting and animation cues for a 3D engine, allowing the musicians to effectively direct a Computer Graphic film in real-time.

**Keywords:** Live visuals, improvisation, mediated performance, electronic music, live score.

## 1 Introduction

OverWatch is an audio/visual performance combining real-time 3D projections with a live musical score. Inspired by silent Expressionist cinema of the 1920s [1] but operating as the diametric opposite of a silent movie, OverWatch presents a scenario where the musical performance directs the projected visuals, determining content, action and editing. Led by the authors: electronic musicians 'The Turing Test', OverWatch presents an empty and eerie world viewed through a vast and labyrinthine network of surveillance cameras. Populated only by bizarre monuments and abandoned machinery, OverWatch takes the viewer into a world where human agency has withered and vanished, leaving only the remnants of industrial processes.

The performance involves a cinematic tour through a real-time 3D environment: a deserted wasteland surrounding the Clinic, a medical facility which carries the traces of weird surgical processes. The viewer is invited to piece together the events that led to this status quo, as the view flicks between cameras that open on a world littered with clues in the form of posters, objects and recorded messages.

## 2 Concept

OverWatch is a narrative piece, in which the action is largely figurative and staged according to cinematographic conventions. However, the narrative is driven not by the actions of actors within the frame of the image but by the musical interaction of the performers outside it. This approach explores the boundary between digetic and non-diagic approaches to music in film and introduces a third possibility: music which does not exist within the context of the story or as an accompaniment but actually defines it, scripting the narrative in real-time. Techniques borrowed from film music such as style topics and leitmotifs<sup>[2]</sup> are used in OverWatch to cue animations and camera movements.



**Fig. 1.** The Turing Test performing with live visuals at Sierra Metro Gallery

OverWatch arose from our interest in an increased integration of visuals and music while performing as live performers under the name 'The Turing Test'. Having used increasingly figurative and narrative sets of pre-recorded visuals (mainly using pre-rendered 3D CGI videos), we began to experiment with systems which responded to the music in real time. This has led to an exploration of the relationship of visuals to musical narrative content in live performance. In a primarily aural medium such as live music, performance visuals often form the equivalent of a musical score in visual media such as film, heightening aesthetic and psychological effects and elucidating narrative structures within the music. However, while in conventional cinema, the screen is favoured with the audience's undivided attention, in most live music settings, the visual attention of the audience is divided between the musicians and screen(s). OverWatch attempts to reconcile this by presenting a narrative space in which the subject is largely absent: the 'action' takes place largely through the camera's

interrogation of objects and scenes allowing the role of subject is to be fulfilled by the musician's onstage.

The performance of OverWatch presents a world composed of familiar pop-culture memes. The dystopian themes and graphical style of video games are combined with exaggerated references to the apocalyptic wastelands of Hollywood cinema, stitched together in a deliberately uneasy mix of visual styles. Taking as its subject the exponential growth of surveillance, OverWatch explores a notional future where surveillance is complete: every centimetre of the world is within the view of a security camera. The cameras now watch an empty landscape, turning only to respond to the fall of rubble from crumbling buildings or the passing of automatic barriers. Factory production lines still disgorge distorted cars, half-wrecked trams wait at stops for pedestrians long vanished, automatic doors open for drifts of falling leaves. Over the course of the performance, the action drifts through a series of weird environments, each scene corresponding to a movement in the score. The narrative is structured by the introduction of objects and scenes by specific combinations of sounds. As the visuals are composed and edited in real-time, no two performances of OverWatch are the same.



**Fig. 2.** Still from OverWatch

### 3 Technical Realisation

OverWatch is a live performance piece designed primarily for a theatre setting. Two musicians (audio/visual act, The Turing Test) occupy the stage area, beneath a large projection screen. As we play, live audio and MIDI streams from synthesisers, an electric guitar, bass and microphones are transmitted via a USB audio interface into a laptop. This information is first processed in Ableton Live and then passed to MAX/MSP which analyses and translates it into a stream of

OSC messages. This stream is then transmitted via a UDP network connection to a second laptop running a 3D graphics engine (Blender GE) which uses these messages to control not only much of the action but also camera movements, framing and cutting.

Scenes and active elements within the visuals are introduced using simple leitmotifs: musical structures which are read as combinations of MIDI messages and are used to trigger loading or visibility of objects. The pace of the editing is set by counting MIDI clock messages and adjusting cuts according to the resulting speed. Camera switching is affected by various instrumental effects depending on context: sometimes changes of chord, mode or key. Camera zoom and small camera movements are affected by the velocity of MIDI notes and amplitude of audio signals, connecting intensity in the score to psychological intensity in the narrative. Discord/harmony is calculated by comparing the notes played to the perceived key signature and is connected to the implementation of unsettling techniques: unbalanced compositions, slight camera roll and tilt etc.

## References

1. Adorno, T., Eisler, H., McCann, G.: Composing for the Films. Continuum Intl. Pub. Group (2005)
2. Kalinak, K.: Settling the score: music and the classical Hollywood film. Univ. of Wisconsin Pr. (1992)

# Sticking Point

Tom Schofield

School of Fine Art, Newcastle University Culture Lab, Kings Walk, Newcastle upon Tyne, NE17RU  
[tomschofieldart@gmail.com](mailto:tomschofieldart@gmail.com)  
<http://www.tomschofieldart.com>

**Abstract.** Sticking point is a data visualisation project which compares sections of national constitutions pertaining to fundamental rights, freedoms, duties and obligations of citizens. Individual words from each constitution are analysed for their distinctness in comparison to other constitution texts. This analysis is presented as a visualisation which also shows relationships between different constitution texts.

**Keywords:** Data Visualization, constitution, human rights.

## 1 Introduction

Sticking point is a data visualisation project which aims to examine core issues of human or social values. Realised in openframeworks and php, Sticking Point is a visualisation of the texts of different world constitutions. Using an algorithm (term frequency vs inverse document frequency), Sticking Point displays the relative uniqueness of words in a particular constitution text compared to a corpus of other constitutions. Thus if a word appears frequently in the text of one constitution but infrequently across the corpus, that word will be deemed significant. By examining these differences an audience can investigate and consider the significance of the differences between these texts which serve as statements of the fundamental nature of who we are as nations.

## 2 Concept

As societies we declare our moral values in public documents such as constitutions. These documents may be realised at a national level, such as the U.S constitution or at an international level such as the European Convention on Human Rights and Fundamental Freedoms or the United Nations, Universal Declaration of Human Rights. Within these titles it is the words ‘fundamental’ and ‘universal’ which are the starting place for this project. The assumption of an intrinsic commonality, a shared core of human qualities is a message which is also at the heart of international art fairs and biennales; a free-flowing world of human exchange where tolerance and respect for cultural difference is lauded as a new religion. Julian Stallabrass in ‘Art Incorporated’<sup>[1]</sup> argues that in actual fact, this apparently commendable ethos is flawed, in both its conception

and application: Rather than engaging with real difference in outlook the real subject of the biennale can be the very hegemony expressed in its stated aims of tolerance and freedom. As Slavoj Zizek points out:

‘Liberalist multiculturalism preaches tolerance between cultures, while making it clear that true tolerance is fully possible only in the individualist Western culture, and thus legitimizes even military interventions as an extreme mode of fighting the other’s intolerance...’<sup>[2]</sup>

Zizek describes contemporary Western attitudes to dealing with cultural otherness in ‘Against Human Rights’:

‘Liberal attitudes towards the other are characterized both by respect for otherness, openness to it, and an obsessive fear of harassment. In short, the other is welcomed insofar as its presence is not intrusive, insofar as it is not really the other. Tolerance thus coincides with its opposite. My duty to be tolerant towards the other effectively means that I should not get too close to him or her, not intrude into his space...’<sup>[3]</sup>

Against this background, Sticking Point attempts to locate the areas within national, stated conceptions of human rights, freedoms and duties where we differ. By examining the sections of world constitutions which pertain specifically to rights, freedoms and duties of citizens, the project aims to deepen our understanding of what makes us different.

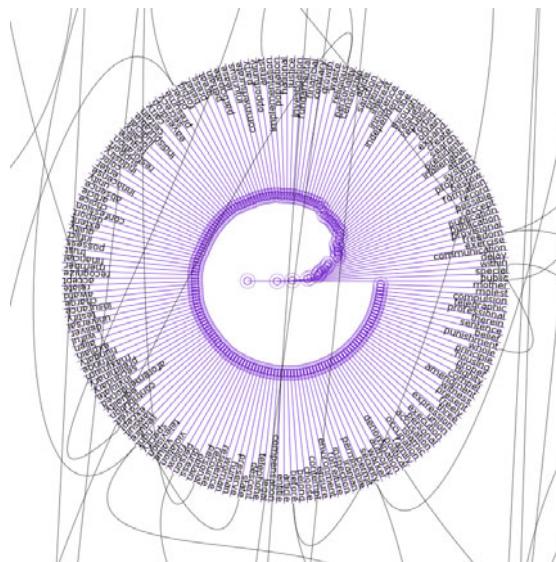
### 3 Technical Realisation

Sticking Point is realised in c++ using the open source openframeworks.cc libraries. The text parsing uses php to create a tf idf (term frequency versus inverse document frequency) equation. Constitution texts are loaded in php, strings are stripped of punctuation and white space and the resulting word lists are saved in text files. Those files are then opened using the wordnet/RITA libraries for processing.org (a java-based I.D.E. for artists and designers). The words are stemmed (for example ‘running’ and ‘runs’ will be reduced to ‘run’) and POS-tagged (Part of Speech). Content words (nouns, verbs, adjectives and adverbs) are retained while grammar words (articles, prepositions etc) are discarded. The resulting lists are once again opened in php and the tf-idf algorithm is applied. A further script analyses those lists to find common words and rank them in order of frequency.

### 4 Images

The final visualisation of this project displays results from 73 countries. Shown in figures 1 and 2 are details from the constitutions of Iran and the Chechen Republic. In Figure 1 we can observe words which have an immediate interest in the context of Western conceptions of Islamic society; ‘mother’ and ‘woman’. One passage (article 21) describing these rights reads;

‘The government must ensure the rights of women in all respects, in conformity with Islamic criteria, and accomplish the following goals: 1) create a favorable



**Fig. 1.** Sticking Point (Detail Iranian Constitution)

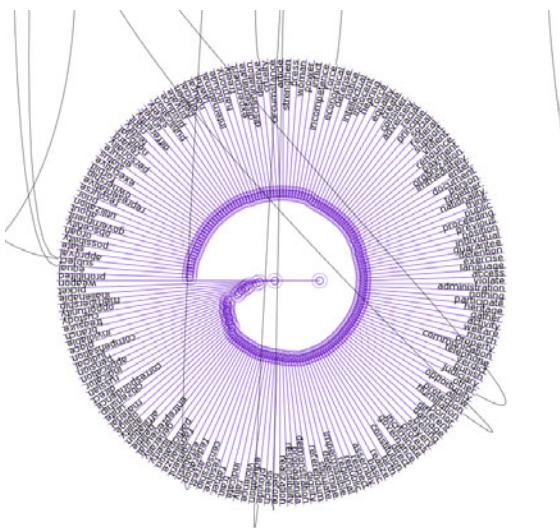
environment for the growth of woman's personality and the restoration of her rights, both the material and intellectual; 2) the protection of mothers, particularly during pregnancy and child-rearing, and the protection of children without guardians.'

Further development of this project will attempt to find ways of introducing this contextual material. Indeed a possible criticism of this visualisation is that words in isolation are difficult to interpret. It is difficult to infer meaning from this level of information. That said, this project was designed to stimulate interest and imagination in viewers rather than provide definitive interpretive or philosophical statements.

In Figure 2 we can observe the word 'Russia' and indeed the constitution of the Chechen Republic contains many references to the Russian legal system due to its status as a federal subject of the Russian Federation. In fact the below passage (article 52.5) partly describes the relationship between those two constitutions:

'Rights and liberties of individuals and citizens may be restricted through federal law only to the extent required for the protection of the fundaments of the constitutional system of the Russian Federation, for morality, health, rights and lawful interests of other persons, and for ensuing the defense and security of the Russian Federation.'

These small insights are of course available to readers of the main constitution texts, however Sticking Point augments this experience by firstly acting as an accessible, engaging interface with the texts and secondly by showing, through the use of connecting lines, the common areas between these constitutions.



**Fig. 2.** Sticking Point (Detail Chechen Constitution)

## References

1. Stallabras, J.: Art Incorporated: the story of contemporary art. Oxford University Press, Oxford (2004)
2. Zizek, S.: Tolerance as an Ideological Category. University of Chicago Press, Chicago (2008)
3. Zizek, S.: Against Human Rights. New Left Review (2005)

# *Phantasmagoria: Composing Interactive Content for the humanaquarium*

Robyn Taylor<sup>1</sup>, Guy Schofield<sup>2</sup>, John Shearer<sup>2</sup>, Pierre Boulanger<sup>1</sup>,  
Jayne Wallace<sup>2</sup>, and Patrick Olivier<sup>2</sup>

<sup>1</sup> Advanced Man-Machine Interface Laboratory, University of Alberta  
Edmonton, Alberta, Canada

{robyn,pierreb}@cs.ualberta.ca

<sup>2</sup> Department of Computing Science, Newcastle University,  
Newcastle upon Tyne, UK

{g.p.schofield,john.shearer,jayne.wallace,p.l.olivier}@ncl.ac.uk

**Abstract.** *humanaquarium* is a mobile performance space which draws upon the traditions of busking and street performance to engage audiences in collaborative, creative play. We describe how the conceptual and physical nature of the performance space affected the way we composed the audio/visual performance content in *Phantasmagoria*, an interactive art piece built for the *humanaquarium* environment.

## 1 Humanaquarium

*humanaquarium* is an interactive performance environment [3] which consists of a 1.5m cube, large enough to contain two musicians who play acoustic instruments, sing, and generate audio/visual content using laptop-based electronic music applications - Ableton Live and Max/MSP/Jitter (see Figure II) The rear wall of the cube is a plain white surface upon which visualizations are projected. The front face of the cube is a transparent touch-sensitive window, through which the performers and projected imagery are visible to the audience. *humanaquarium* is a participatory installation, allowing audience members to interact with the ongoing performance by touching the transparent window. The window uses frustrated total internal reflection (FTIR) technology [2] to detect when participants touch the screen. Infra-red light is shone into the sides of the clear acrylic, and when audience members place their hands upon the window the change in surface tension caused by their touches frustrates the contained light. Each touch generates an infra-red bright spot (invisible to human eyes) which is reflected back towards a camera, mounted at the rear of the *humanaquarium*. By tracking these bright spots, the camera can detect the movements of the hands, and send control data to the *humanaquarium* performance system. This is then used to affect the audio and visual content of the performance, and to manipulate the output of the two musicians' playing and singing. In this way, participants are able to interact with the performers, affect the development of the performance, and essentially co-create the performance content.



**Fig. 1.** A participant interacts with the two musicians inside the *humanaquarium*

## 2 Creating Compositions

Our core creative team consists of two electronic artists and an interaction designer, however, in practice, the process of composing performance content and interactive strategies for *humanaquarium's* participatory platform blurs these boundaries and each member of the team contributes jointly to the creative process. In our practice, we do not separate issues of interactivity from issues of composition, as the interaction between the audience and the musicians is critical to how the composition is realised during performance. Each composition is defined as an aesthetic and temporal structure upon which interactions between the audience and musicians will occur. In order to compose appropriate creative content which properly supports the performance, the interaction mechanism and its impact on the performance experience must be considered concurrently to any musical and visual choices that are made. Our team handles this approach by composing audio/visual content alongside the development of interaction strategies, simultaneously composing new media content (video manipulations, audio tracks, etc.) while testing the mappings between participant touch and control system response.

We have written several different compositions for the *humanaquarium* environment. Our first set of performances, *Darkshines* and *Mariana*, used abstract imagery and a fluid soundscape to create an ethereal space within the *humanaquarium*. We used video footage featuring the movements of jellyfish (filmed by UK-based documentarian and videographer David Green) as well as a drifting, pulsating soundtrack which intensified in timbre and colour in response to participant touches. During the making of the works, interaction paradigms were established iteratively and concurrently with the content composition. The skeleton of the audio/visual content was outlined, providing a

starting point for the development and refinement of interactions and musical ideas. To complete the composition, musical and interactive ideas were proposed, quickly implemented and then evaluated against the existing composition. Ideas which seemed to encourage interesting interactions were explored more deeply, refined, and integrated into the composition, gradually adding increasing complexity and content to the piece. During this process, a huge overlap occurred in the roles of team members, with interactions setting the musical scene, and musical ideas suggesting new interactions – audio/visual content was developed so closely with interaction design that all three creators contributed simultaneously to the musical, visual and interactive aspects of the experience.

The eventual interaction strategy for *Darkshines* and *Mariana* was a relatively simple mapping of vertical touch placement to the proportion of heavy and light instrumentation, and left-right touch placement to various audio/visual effects on the two performers. Audience response to these aquatic pieces indicated that we had created a mysterious and imaginative space within the purposefully spartan confines of the sharp-edged *humanaquarium* structure. Encouraged by this response, we decided to explore this concept of ‘spaces within and without’ further, and intensify the surrealism of the fantastical environment inside the aquarium in our subsequent performance, *Phantasmagoria*.

### 3 Composition Strategies in *Phantasmagoria*

The concept of ‘phantasmagoria’ – a feverish dreamlike series of shifting scenes, images and figures – not only describes well our creative intentions behind the piece but also refers to a 19th century device which incorporates projected imagery into theatrical performance [4]. As *humanaquarium* was designed from the outset to borrow from traditional theatrical practice, we decided to further emphasise the theatrical aspects of the performance in several ways. The cuboid shape and single-sided aspect referenced conventional proscenia with a projected backdrop similar to stage flats. In the design of *Phantasmagoria*, we intended to reinforce this by using visuals featuring stage-like setups. Previous compositions had featured semi-abstract moving canvases of jellyfish in aquaria. The variable depth of



**Fig. 2.** Scenes from *Phantasmagoria*

the image in these pieces was contrasted with the clearly defined layering of images in *Phantasmagoria*, where each scene comprised two distinct planes of action depicting clearly delineated architectural spaces (see Figure 2) In contrast, again, to *Darkshines* and *Mariana*, *Phantasmagoria* featured human subject matter in the form of dancers and moving figures. These figures were used to clearly distinguish ‘live’ interactive visual material from the non-interactive backgrounds. Besides the symbolic and aesthetic effect of this decision (i.e. allowing the audience to literally ‘bring the piece to life’) it also yielded a practical benefit, as it made the audience’s effect on the image clearly legible.

Two distinct sources were used for the imagery in the piece. As the background in each scene we used photos taken by Robyn Taylor on a number of visits to the neo-Baroque Palais Garnier opera house in Paris. These images were brought into After Effects by Guy Schofield (who had never visited the original building) and collaged together into surreal imaginary spaces, creating an interpretative representation exploring the character of the spaces in response to Taylor’s descriptions and photographic footage. The looping, co-creational nature of this process is, once again, a defining feature of the composition process in *humanaquarium*, with one member of the team responding to material proposed by another in a continuous dialogue. The second source was the 1925 film production of *The Phantom of The Opera* [4], a classic horror tale set in the Palais Garnier, which formed yet another conception of the space and also provided the moving figurative elements. The ‘penny dreadful’ subject matter and the distinctive formal qualities of this footage – a grainy, low resolution video of an old silver nitrate print – reinforced the casting of the *humanaquarium* as a theatrical domain in the tradition of nineteenth-century magic lantern shows [1].

*Phantasmagoria* was designed to evoke the type of late Victorian entertainment popular immediately before the birth of conventional cinema, when mediated forms of entertainment used innovative technology to explore new methods of connecting performers and audiences. Coupled with elaborate formal costumes, the whole effect was intended to establish an aesthetic space where the audience could clearly read *humanaquarium* as a theatrical stage but also be enticed into interacting with the performance in unfamiliar ways.

## References

1. Barber, X.T.: Phantasmagorical Wonders: the Magic Lantern Ghost Show in Nineteenth-Century America. *Film History* 3(2), 73–86 (1989)
2. Han, J.Y.: Low-cost multi-touch sensing through frustrated total internal reflection. In: Proc. of UIST 2005, pp. 115–118. ACM, New York (2005)
3. Taylor, R., Schofield, G., Shearer, J., Boulanger, P., Wallace, J., Olivier, P.: *Humanaquarium*: Designing for Participatory Performance. In: Proc. of NIME 2010 (in press, 2010)
4. The Phantom of the Opera. Universal Pictures, USA (1925)

# Self Portraits with Mandelbrot Genetics

Jeffrey Ventrella

Jeffrey@Ventrella.com

**Abstract.** This paper is an artist statement describing a manipulation of the Mandelbrot Set equation as the basis for creating semi-figurative images, using a genetic algorithm. Modernist painting is referenced in terms of the play between representation and abstraction. The Platonic implications of the Mandelbrot Set are considered as a point of departure for manipulation of the complex function.

**Keywords:** Mandelbrot Set, evolutionary art, genetic algorithm.

## 1 Introduction

As a young art student, I had an appetite for abstract expressionism and surrealism. I would gaze at the paintings of Bacon, Motherwell, and Gorky, and marvel at their ability to mix beauty and ugliness, and force me to see the world more intensely. These artists explored forms that lay between representation and abstraction. I did not know that I was exercising my eye-brain for what lay ahead: a new art medium that would disrupt the existing paradigms of representation and abstraction.



**Fig. 1.** Four examples of self-portraits based on the Mandelbrot equation

The Mandelbrot Set is a curious discovery. It is as if a distant planet has been found. The intoxication caused by the Mandelbrot Set's fantastic, infinite form is offset by the simplicity of its genetic expression – these are two very different representations. The elegance of the Set's underlying code can lead Math-artists to lose the sense of irreverence that is sometimes necessary to pull a subject out of context and expand its expressivity.

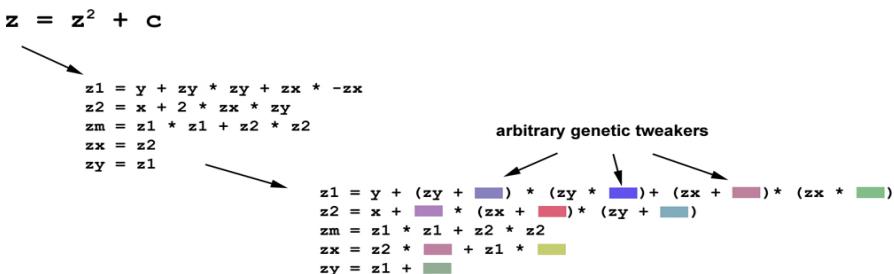
When considered as a tool for visual expression, I believe the Mandelbrot equation has not yet been given a thorough workout, tweaked, deconstructed – using the tools of visual language. I decided that this would make an interesting challenge. The Set,

so thoroughly complex and beautiful as to create near religious admiration – what a great candidate for the subversive act: The *Big Tweak*.

## 2 Process

The Mandelbrot-based images presented here are a recent variation from a series that has been evolving since the late 1980s. My personal vision of organic form, generated by way of genetic algorithms, has had its own evolution. The many thousands of images explored, discarded, and refined, constitute a Darwinian process of its own.

Introduced by Benoit Mandelbrot [1], the Mandelbrot equation is  $z = z^2 + c$ , where  $z$  and  $c$  are complex numbers, and  $c$  is a location in the complex plane being tested. The function is applied many times, with the output value of  $z$  from each iteration being used as input for the next iteration. During iteration, if the value of  $z$  exceeds a magnitude of 2, iteration halts, and the location  $c$  is declared outside of the Mandelbrot Set. Coloration of these two domains inside and outside the Set reveals an infinity of self-symmetry and filigree. Coloration, and adjusting the location and scaling in the complex plane constitutes much of what is called *fractal art*. The process I use involves altering the equation, inserting arbitrary real numbers (the *genes* of the image) into its code expression, as shown in Figure 2. I then employ a genetic algorithm to evolve families of images. I do not use color, and instead render the forms in sepia tones.

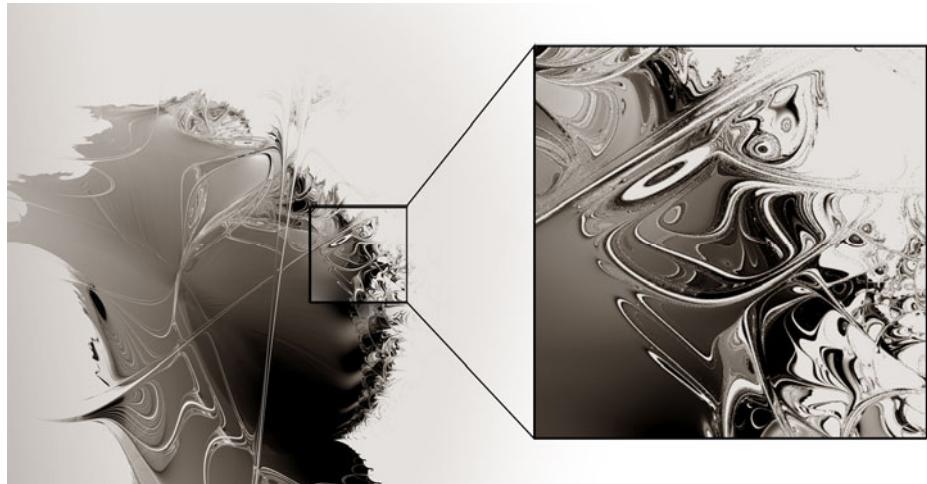


**Fig. 2.** Adding arbitrary genetic tweakers to the code expansion of the Mandelbrot equation

Yanking the Mandelbrot equation out of the realm of complex analysis can be offensive to some mathematicians. A gentler description is *sliding it off* the complex plane: when these genetic tweakers are set to default values, there is no change to the Set, but tweaking any one gene by the tiniest epsilon causes the Set to change its attitude ever so slightly. I ask the mathematician to think like a biologist for a moment. Studying messy nature shifts scrutiny to families of forms that are interrelated, imperfect, and always in a state of becoming.

The philosophical debate as to whether the Mandelbrot Set is completely human-made, or whether it is an eternal Platonic form, provides a backstory. Imagine if the Mandelbrot Set had come from a lineage of imperfect beings, whose genetic codes were wildly varied and continually evolving. Imagine now that the human mind, with its Platonic reach, had extracted perfection out of this menagerie, and revealed the Set as we now know it. My manipulation if the Set has allowed me to cast many shadows

of the Set on the walls of Plato's cave. Some attributes seem to persist throughout this diverse family of forms. For instance, circular, spiraling, and curvilinear forms that are in the Set persist, though distorted, upon subversive tweakage, as Figure 3 shows.



**Fig. 3.** Curvilinear forms exist in most tweaked variations of the Set



Abstract expressionism; automatism; action painting – require giving up part of yourself to the raw energy that generates a painting. There is likewise a raw energy beneath the complex plane, having its own laws of nature, offering strange beasts. I have conjured many from the underworld. Like Pollack's drips, these curvilinear forms are determined by a dynamic process. But instead of the viscosity of paint and the gesture of a brush, it's the spin in the complex plane and the acceleration of that spin by recursion.

**Fig. 4.** A heavily-tweaked self portrait

### 3 Portraiture

These are self-portraits. The genetic algorithm uses a digital image of my face as a fitness function. This process is described in detail in a paper I wrote for the book, *Design by Evolution* [2]. Instead of trying to design variables in the equation that can evolve so as to generate recognizable, realistic imitations, I allow the underlying dynamic to rear its ugly head, while the genetic algorithm attempts optimization towards a resemblance. In most cases, the only features common to the target image are the overall shape of my head and differences in shading. For me this is *just right*.



**Fig. 5.** Two examples of self-portraits based on the Mandelbrot equation

A painter working on a portrait might stop as soon as the essence of the subject has been captured, even as there are bare patches of canvas left, rough brush strokes, dribbles, and splotches. The physics of paint and canvas are allowed to have a voice in the final statement. These Mandelbrot-based images likewise permit the beasts lying beneath the complex plane to become a part of the expression. Since I have been exploring this family of beasts for more than twenty years, I have established kinship with them, and that is the main reason I consider them to be self-portraits.

## References

1. Mandelbrot, B.: *The Fractal Geometry of Nature*. Freeman, New York (1982)
2. Ventrella, J.: Evolving the Mandelbrot Set to Imitate Figurative Art. In: Hingston, P., Barone, P., Michalewicz, Z. (eds.) *Design by Evolution*. Springer, Heidelberg (2008)

# Art 101: Learning to Draw through Sketch Recognition

Tracy Hammond, Manoj Prasad, and Daniel Dixon

Sketch Recognition Lab  
Computer Science & Engineering Department  
Texas A&M University  
srl@tamu.edu

**Abstract.** *iCanDraw* is a drawing tool that can assist novice users to draw. The goal behind the system is to enable the users to perceive objects beyond what they know and improve their spatial cognitive skills. One of the early tasks in a beginner art class is to accurately reproduce an image, in an attempt to teach users to draw what they see, rather than what they know, improving spatial cognition skills. The *iCanDraw* system assists users to reproduce a human face, providing real-time drawing feedback enabled by face and sketch recognition technologies. We are presenting an art installation piece, where the conference participants using the *iCanDraw* ‘smart graphics’ system create the art in real-time at the conference.

**Keywords:** spatial cognition, sketch recognition, face recognition.

## 1 Introduction

*iCanDraw* is a system to assist users in drawing the human face. The system teaches a novice how to expand the realm of his knowledge in sketching. It enables the artist to draw what they see and focus their attention on the parts of an object, contours, proportions and spatial relationships between them. The art of drawing is commonly perceived as hard to learn, learning then mostly involves how one sees and processes the object being drawn. We present this system for the participants at the conference using our program. This allows them to create the art from the installation piece themselves. We use before and after photos to show examples of how an artist can grow.

## 2 Concept

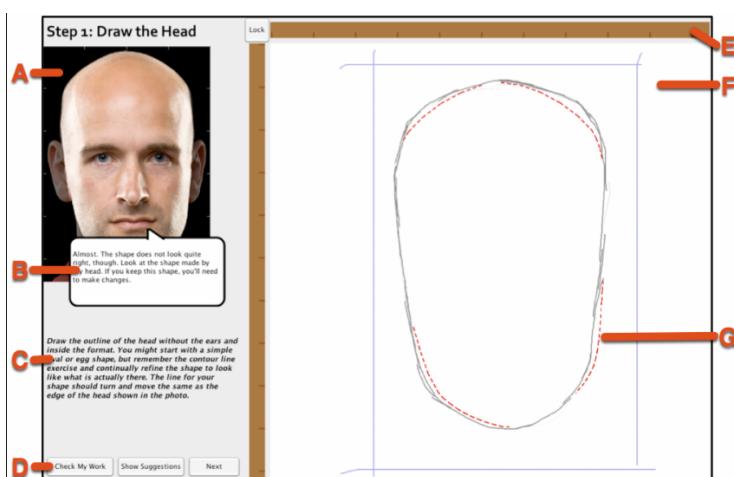
The art of drawing involves a balance in creativity and the drawing skills. Empirically, people are quick to confess their inability to draw. This is due to the perceived difficulty in learning how to draw. *iCanDraw* is an intelligent tutoring system on drawing, built to stimulate creative thinking by helping overcome this perceived difficulty. The difference between an experienced artist and a novice is the ability to see beyond what one knows, the ability to perceive an object through the contours, spaces, and relationships of an object. One of the first tasks in teaching

novices how to draw is to reproduce an image. This task teaches how to visualize an object into its parts, their proportions and spatial relationships between them. Joshua Bienko, a professional artist and art professor states, “Beginning drawers draw what they KNOW. I go to great depths in my courses to erase what artists know so that we can develop what we see.” Our application assists users to reproduce a human face. In the step-wise tutorial to draw, each step provides detailed instructions on how to draw specific, singular pieces, forcing the user to see the parts of face and their relation to the whole picture.

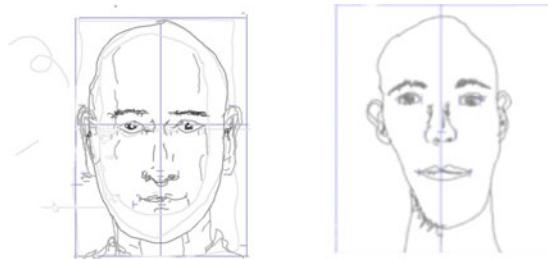
Our presentation includes a tour of the system, where we show the aspects of the drawing being taught, well defined steps involved in teaching a drawing, interpretations and feedback mechanism involved in the system illustrating the ability of the system to act as art tutor.

### 3 Technical Realization

We chose a photo of the human face because it is a stable idea for new artists to accept, unlike a cartoon figure. When a new artist sees a photo they can immediately see the inaccuracies in their own drawings; unlike a cartoon where there are more arbitrary choices based on perception. The drawing procedure is divided into well-defined steps. Figure 1 provides the screenshot of the program at one such step. Every step helps the artist to draw what they see, rather than what they perceive using written guidelines and corresponding adjustments to the reference image. Instructions provide a detailed explanation (including size and location) about a part of the face to be drawn, the parts of the face being the outline of face, eyes, ears, nose and mouth. This forces the user to concentrate on one part of the face, its size, symmetry and its relation to the face as a whole.



**Fig. 1.** Screenshot demonstrating many of the interaction techniques. (A) The reference image manipulated to show the head. (B) Text feedback. (C) Step instructions (D) Options available. (E) Straightedge tool. (F) Drawing area. (G) Visual feedback.



**Fig. 2.** Faces drawn by participants using the application at ArtReach

**Table 1.** Sample drawings from a user study

APPENDIX	Exercise 1 Freehand	Exercise 2 Corrective Feedback	Exercise 3 Freehand (Repeated)	Exercise 4 Corrective Feedback	Exercise 5 Freehand
	(A)	(B)	(A)	(C)	(D)
Male #1					
Male #2					
Male #3					
Male #4					
Male #5	 <b>A: 466 SD: 119</b>	 <b>A: 167 SD: 41</b>	 <b>A: 336 SD: 70</b>	 <b>A: 122 SD: 16</b>	 <b>A: 237 SD: 87</b>

The photo image is first processed using face recognition software to obtain information about the location and size of the facial features. Throughout the drawing process, sketch recognition interprets the soon-to-be artist's freeform strokes, and

provides real-time constructive feedback to guides the user. Example feedback: “The eyes are a bit high on the face. Hint: Because few perceptually important features exist on the forehead, users tend to draw the eyes too high on the head. However, the eyes are usually approximately half way down on the face.” Or: “The eyes are a bit too close together. Hint: The space between the eyes is approximately equal the size of an eye itself.” Once the user is completely finished with the drawing, face recognition is performed on the final image, and the two facial templates are compared to provide a overall evaluation, which can be used to measure the progress that a user has made in learning how to draw. Figure 1 shows a screenshot of the system in progress. Figure 2 shows examples of face drawn by the users using the application. Table 1 illustrates the improvement in drawing after using the application across 5 users in terms of our evaluation metric.

This project is an interactive art piece where members of the conference use the *iCanDraw* system during the art exhibit and throughout the entire conference. The art produced becomes the art exhibit itself. Digital and paper copies of the images will be displayed, and participants will be able to take home a copy of their drawings.

## Acknowledgements

This work is supported in part by NSF 0757557. The authors also thank Kerre Cole for her help in the completion of this document.

## Reference

1. Dixon, D., Prasad, M., Hammond, T.: iCanDraw? Using Sketch Recognition and Corrective Feedback to Assist a User in Drawing Human Faces. In: Proceedings of Computer Human Interaction (CHI 2010), April 10-17. ACM Press, Atlanta (2010)

# ColourVision—Controlling Light Patterns through Postures

Alexander Wiethoff and Andreas Butz

University of Munich, Mediainformatics, Amalienstr. 17,  
80333 Munich, Germany

**Abstract.** *ColourVision* is an interactive installation that empowers people to step into an intensive dialogue with colors. Physical seating postures such as active, relaxed or reflective positions are captured, translated and trigger a rapid change of the room's color. The installation was planned and executed for the Museum of Perception in Upper Austria, where we wanted to create seamless communication between the visitor and the color that would lead to an aesthetic experience in the space and let people experience the psychological effects of different colors.

**Keywords:** light installation, responsive environment, body interface, interactive art.

## 1 Introduction

In an article that analyzed the psychological effects of environmental colors on the human body, Stone et al. [5] claimed that “red and yellow are naturally experienced as stimulating and disagreeable,” and that “these colors focus people on the outward environment,” and that “they produce forceful, expansive behavior, whereas green and blue are experienced as quieting and agreeable, focusing people inward, and produce reserved, stable behavior.” Olafur Eliasson’s *Room 360°* for all colors [2] is a good artistic example on how a person can be encapsulated within a light-space and perceive an intensive experience of colors (see Figure 1a). In contradiction to monochromatic light-spaces with one light source, his installation captures the visitor constantly through continuous changes. In addition to Eliasson, we wanted to create a dialogue between the body language of the visitors and color changes. Responsive environments, such as those described by Moeller [4] in his participatory light installation *Electro Clips*, were inspiring in the ideation phase . Bullivant [1] documented how spaces can rearrange themselves and include a wide audience, such as the light installation *ICE* from Iwai, where participants can *paint* colors on a digital surface with their hands (see Figure 1b) or *Sky Ear* [3], a playful public installation based on color interactions (see Figure 1c). Winkler described the challenges for artists to create such an experience in his participatory sound and light installation *Light around the Edges* [6].

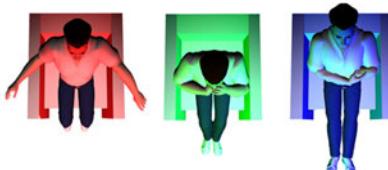
Combining a lighting installation with an interface that interprets the body positions of a person and visualizes them in a space through color seemed a way to



**Fig. 1.** Different artworks that served as inspirations (from left) (a) Olafur Eliasson: 360° Room for all colors (Photograph by <http://www.flickr.com/photos/fabianmohr/259818925/> reproduced under Creative Commons License, <http://creativecommons.org/licenses/by/2.0/>) (b) Toshio Iwai : ICE (Photograph by <http://www.flickr.com/photos/mikichui/3596926827/> reproduced under Creative Commons License, <http://creativecommons.org/licenses/by/2.0/>) (c) Usman Haque : Sky Ear (Photograph by <http://www.flickr.com/photos/vrocamp/538832031/> reproduced under Creative Commons License, <http://creativecommons.org/licenses/by/2.0/>).

enable participants to quickly communicate with a light space. The natural, seamless interaction with light was important for creating this installation as visitors should step into an intensive color experience and reflect on the effects of colors on themselves without being distracted. *ColourVision* combines color changes corresponding to the participants' body language and lets them glide into an intensive dialogue with the environment.

## 2 ColourVision



**Fig. 2.** Initial animation from the ideation phase showing different seating positions and the resulting room colors: (from left) (a) activity, (b) reflectiveness and (c) relaxation

### 2.1 Technical Setup

A camera on the ceiling in the center of the space is attached to a PC with video analysis software written in C++. Participants were tracked in a chair from above and the resulting data was sent to a DMX converter that digitally controlled the different colors in the space. The installation consists of 45 x 58 watt fluorescent RGB lamps arranged in a half circle, with 15cm of space between them. A rear projection foil serves as a diffuser with sites at a distance of 65cm to the lamps in order to present a planar lighting experience. We shaped the projection foil in a curve so that no spatial edges or corners are perceived in the view area (Figure 3).



**Fig. 3.** Implementation: Different colors generated through diverse seating positions

## 2.2 Posture and Color

The implemented body interface controls the room through posture. Red, for example, is activated through an open, active seating position. Green is the color for introverted reflectiveness as generated if a person takes a thoughtful, closed position. A person, sitting on the chair in a stretched, relaxed position plunges the room into a cool blue as the color for calmness (Figures 2 and 3). Playful interactions with the body interface were perceived positively. Some of the participants were exploring the interface in different ways and climbed on the chair or sat on the floor to experience the responses of the installation. Different emotional color impressions were reported from the participants: Asked about their emotional state when exposed longer to one color, for example red, the participants reported an increased level of “nervousness.” An intensive exposure in a relaxed posture to the color blue was described as “peaceful.”

## 2.3 Personal View

From an artistic standpoint, the interaction with light and colors is a magical experience for me. People can let themselves glide into a fluid color experience, while all edges disappear and a space made out of light encapsulates them fully. The longer perceived colors triggered different emotions in me that I had not previously associated with colors. With my background as an electrician and my later studies in design and art, I feel that many different aspects of my education are applied to this work. My knowledge of the technical feasibility of the project was interesting as well as the spatial design of the installation. Even more intriguing were the aspects of how a person controls such an installation. The fluid level of interaction that incorporates a rapid change gives the visitor the freedom to participate in the installation and become a part of the artwork.

### 3 Discussion

By applying an interface that lets people influence a color space, we hope to contribute an inspiration about how people can interact seamlessly with embedded technology, and—if applied correctly—provide a smart, usable solution for a very simple interaction. By bridging the gap between experiencing colors and controlling them, we hope to encourage more people to explore the area of interactive lighting design. Regarding the received positive feedback, we were encouraged to expand the topic on a larger scale. Smart media façades, equipped with color mixed LEDs that can be embedded in the architecture of buildings, provide an interesting area of interactive lighting design that has just briefly been explored thus far. If these structures know how to interpret people's behavior, an interesting dialogue between humans and buildings can be established. In this sense, such installations can make spaces appear smart and reactive.

**Acknowledgments.** We would like to thank Zumtobel Staff and Feno for their support with materials and information. This project was funded by the University of Art & Industrial Design, Linz (A), and the Museum of Perception, Rohrbach (A).

### References

1. Bullivant, L.: *4dspace: Interactive Architecture*. John Wiley & Sons, Chichester (2005)
2. Eliasson, O., Tuyl, G.v., Broeker, K.: *Olafur Eliasson: Your lighthouse: works with light 1991-2004*. Hatje Cantz (2004)
3. Haque, U.: *Sky Ear* (2004), <http://www.haque.co.uk/skyear/> (access:14/02/10)
4. Moeller, C.: *A Time and Place: Christian Moeller: Media Architecture*. Lars Müller (2004)
5. Stone, N.J., English, A.J.: Task type, posters, and workspace color on mood, satisfaction, and performance. *Journal of Environmental Psychology* 18 (1998)
6. Winkler, T.: Audience participation and response in movement-sensing installations. In: *Proc. ISEA* (2000)

# Interaction with a Virtual Character through Performance Based Animation

Qiong Wu, Maryia Kazakevich, Robyn Taylor, and Pierre Boulanger

Advanced Man-Machine Interface Laboratory  
Department of Computing Science, University of Alberta  
T6G 2E8 Edmonton, Alberta, Canada  
`{qiong,maryia,robyn,pierreb}@cs.ualberta.ca`

**Abstract.** While performance based animation has been widely used in film and game production, we apply a similar technique for recreational/artistic performance purposes, allowing users to experience real-time natural interaction with virtual character. We present a real-time system that allows a user to interact with a synthetic virtual character animated based on the performance of a dancer who is hidden behind the scenes. The virtual character responds to the user as if she can “see” and “listen” to the user. By presenting only the virtual character animated by our system to the observing audiences within an immersive virtual environment, we create a natural interaction between the virtual world and the real world.

**Keywords:** synthetic characters, virtual reality, motion capture, advanced man-machine interfaces, behavioral systems, interaction techniques, immersive entertainment, artistic installations.

## 1 Introduction

As computer generated graphics gain sophistication and become part of our culture through our exposure to films, video games, and interactive applications, one of the great challenges of our time is to achieve seamless interaction between the virtual world and real world. For example, natural and intuitive responses from virtual characters can dramatically improve the end-user experience in applications such as video games, communication tools, and live performances. Such online applications require the virtual character to efficiently and accurately interpret the user’s intention and to respond accordingly and naturally.

We have created a system which enables one-to-one real-time interaction between a user and a virtual character, a character which is animated through the live performance of a real actor in a remote location. The virtual character can “see” and “listen” to the user through camera and microphone installed in our system, and “respond” in real time driven by the performance based animation. Such an environment allows the user to naturally interact with a virtual character as if interacting with a real person, by providing audio, visual and even vocal ability to the character that is controlled by the hiding performer.

Such a system may serve various purposes, ranging from the artistic (choreographed animation maybe driven by a dancer), the recreational (users maybe entertained by figuring the intelligence reason behind the system), or as a way to enhance live performances (the performer may perform with a virtual character together).

## 2 Related Work

There exists extensive research addressing interfaces to allow users to interact with virtual characters. For applications of games, novel interfaces that have been proposed seek the use of limited input resources, such as a simple mouse [1], a keyboard [2], video cameras [3], Nintendo Wiimotes [4], or a pressure sensor [5] etc. In these systems, once the input is interpreted, a pre-defined animation is played and cannot be interrupted until the next input is read. The similar idea is also applied as the interaction means for live performances such as music driven animations, which strive to reduce the barrier between the musician and the virtualized environment. Such mediated performance practice usually parameterizes the vocal [6], movements [7], and/or keyboard [8] input, extracts meaningful features and maps to pre-defined character behavior. Such “new media” performance mixes traditional performances with digital contents, which bring new multimedia elements to the performance itself. Using live input to trigger pre-defined animations usually permits only limited variation, however, and therefore lacks a real feeling of interaction for the user.

## 3 Real-Time Interactive Dancing System

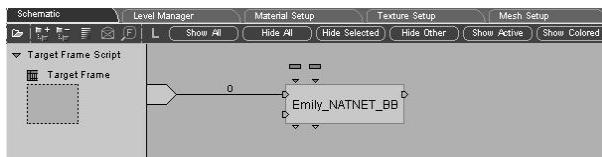
The idea of our system is to achieve the real interaction between a user and a virtual character through controlling the character by a hiding performer/dancer. The performer is invisible to the user but is able to see the user by a surveillance camera, and performs according to the user’s intention. The character is then animated exactly the same way as the performance through motion capture system and animation engine. Without knowing the existence of the performer, the audiences experience real interaction with a virtual character, bringing a new form of multimedia to the live performance itself.

The system is composed of following three parts:

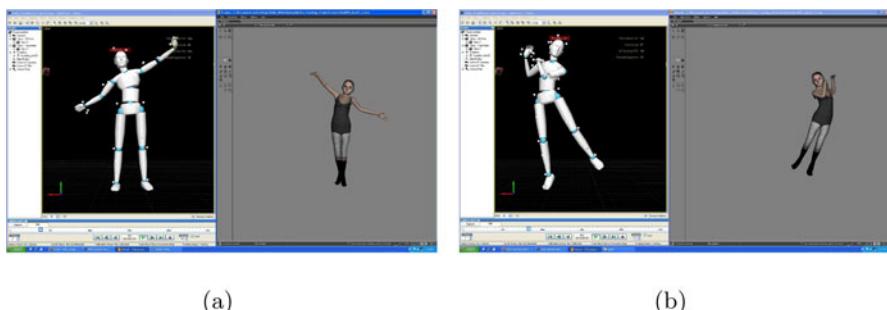
- **Sensing Interface:** To achieve natural interaction, the interface interacting with the virtual character is hands free. We use a high-end visualization system which is capable of displaying stereoscopic images to display the life-sized and three-dimensional character. A five-walled life sized immersive display environment (called a CAVE) allows the user to experience the virtual world inhabited by the believable character. The CAVE is equipped with a microphone as well, so that the user may not only dance with the character but also speak to the character.

- **Motion Capture:** We use the motion capture system, OptiTrack by NaturalPoint, to capture the performance motion. Six infrared cameras are synchronized and connected to a computer. The dancer wears a special black suit with optical markers attached on the suit, so that when the dancer moves within the capturing volume the system can compute the position and orientation of markers as well as skeleton joints of the character. In total, 24 skeleton joints are used in our system to control the virtual character.
- **Animation Engine:** Animation is rendered using a programmable game engine Virtools. We programmed a plug-in, “Emily\_NATNET\_BB” building block in the Virtools (as shown in Fig. 1). This plug-in mainly completes three tasks. First, acting as a client side, read real-time data (each joint’s position and orientation) from the motion capture server (Arena) using the Natnet transport (a customized streaming SDK from the NaturalPoint). Second, the plugin retargets the motion joints to the skeleton joints of the character which is modeled in Maya and exported into Virtools with the skeleton. Third, update the character’s motion. As one can see from Fig. 2, the character’s animation is synchronized in real-time with the motion capture data.

Using Virtools, we may further program computer generated animations based on the motion of the character. For example, the clapping motion of the character may elicit sparkles from the clapping position based on the motion detection. Such composing of real performance based animation with the computer generated contents has been wildly used in the post-production



**Fig. 1.** Building block in Virtools sets up the animation engine



**Fig. 2.** Performance based animation. The left side of the image is the OptiTrack software Arena which captures the motion data. The right side of the image is the animation rendered using Virtools.

of film and game, yet most of people never have chance to interact in real-time with such a believable virtual world as what is currently seen in films or games.

## 4 Future Work

It is our goal that user can experience believable and natural interaction with the virtual character. This requires that the character is equipped with as many senses like a human being, through which the user can interact with. The current system only allows the user to see the life-sized three-dimensional character performing in an interactive way within an immersive environment. We may add a haptic device for the user to touch the character, a speaker to hear the character. A face capture system may be added to capture the performer's facial expression, which adds more life to the character, and the virtual world will be enriched with a more complex cognition layer to respond to the motion as well as facial expression. We aim to create such a system that the user can experience real-time interacting with virtual characters like Navi in an immersive pandora environment in the movie Avatar rather than just watching static screens.

## References

1. Zhao, P., Van De Panne, M.: User interfaces for interactive control of physics-based 3d characters. In: Symposium on Interactive 3D Graphics and Games, pp. 87–94 (2005)
2. Thorne, M., Burke, D., Van De Panne, M.: Motion doodles: an interface for sketching character motion. ACM Trans. on Graphics, 424–431 (2004)
3. Chai, J., Hodgins, J.K.: Performance animation from low-dimensional control signals. ACM Trans. on Graphics 24(3), 686–696 (2005)
4. Shiratori, T., Hodgins, J.K.: Accelerometer-based user interfaces for the control of a physically simulated character. ACM Trans. on Graphics 27(5), 123:1–123:9 (2008)
5. Yin, K., Pai, D.: FootSee: an interactive animation system. In: ACM SIGGRAPH/Eurographics symposium on Computer Animation, pp. 329–338 (2003)
6. Levin, G., Lieberman, Z.: Insitu. speech visualization in real-time interactive installation and performance. In: Proceedings of the 3rd International Symposium on Non-Photorealistic Animation and Rendering, pp. 7–14. ACM Press, New York (2004)
7. Camurri, A., Coletta, P., Ricchetti, M., Volpe, G.: Expressiveness and Physicality in Interaction. Journal of New Music Research 29(3), 187–198 (2000)
8. Taylor, R., Torres, D., Boulanger, P.: Using music to interact with a virtual character. In: The International Conference on New Interfaces for Musical Expression (2005)

# Author Index

- Ammer, Andreas 208  
Andujar, Carlos 115  
Argelaguet, Ferran 115  
Ashley, Peter 152
- Baker, Andrew 1  
Boulanger, Pierre 269, 285  
Butz, Andreas 281
- Casares, Laura 196  
Casey, Rachel 261  
Cheema, Salman 13  
Chittaro, Luca 103  
Choi, Jong-Soo 253  
Christie, Marc 91, 244  
Chung, Cheng-Hsuan 80  
Company, Pedro 152
- Dickmann, Lutz 139  
Dixon, Daniel 277
- Ebert, David S. 172  
Echelman, Janet 68  
Endo, Yuki 160
- Feiner, Steven 232  
Fukui, Yukio 160
- Gajananan, Kugamoorthy 44  
Gračanin, Denis 56, 208
- HA, Hai Nam 244  
Hammond, Tracy 277  
Heppel, Peter 68  
Hildebrandt, Andreas 44
- Ieronutti, Lucio 103  
Igarashi, Takeo 127
- Johnson, Andrew 184
- Kanamori, Yoshihiro 160  
Kazakevich, Maryia 285
- Kim, Ji-Sun 56  
Kim, SungYe 172  
Kim, Young-Mi 253
- Lai, Chan-Yet 156  
LaViola Jr., Joseph J. 13  
Lee, Sangyoon 184  
Leigh, Jason 184  
Lensing, Tobias 139  
Lež, Alan 208  
Lino, Christophe 244  
Lischka, Christoph 139  
Liu, Damon Shing-Min 80
- Maciejewski, Ross 172  
Malaka, Rainer 139  
Mashio, Kairi 220  
Matković, Krešimir 56, 208  
Mendez, Erick 232  
Mitani, Jun 160  
Mukaiyama, Kazushi 257
- Nakasone, Arturo 44
- Okada, Masato 220  
Olivier, Patrick 244, 261, 269
- Pintilie, Grigore D. 68  
Porzel, Robert 139  
Prasad, Manoj 277  
Prendinger, Helmut 44  
Purgathofer, Werner 208
- Quek, Francis 56
- Ranon, Roberto 91, 103  
Runions, Adam 1
- Samavati, Faramarz 1  
Schmalstieg, Dieter 232  
Schofield, Guy 261, 269  
Schofield, Tom 265  
Shearer, John 269  
Shen, Rui 33  
Stiver, Marc 1

- Sun, Yiwen 184  
Suveeranont, Rapee 127  
Takahashi, Shigeo 220  
Taylor, Robyn 269, 285  
Therón, Roberto 196  
Urli, Tommaso 91  
Varley, Clifford 152  
Ventrella, Jeffrey 273  
Wallace, Jayne 269  
Wiethoff, Alexander 281  
Woo, Insoo 172  
Wu, Qiong 285  
Yoshida, Kenichi 220  
Yung, Chun-Hao 80  
Zakaria, Nordin 25, 156