Geometry and Topology Extraction and Visualization from Scalar and Vector Fields

John Congote

EAFIT UNIVERSITY
COLLEGE OF ENGINEERING
DOCTORAL PROGRAM IN ENGINEERING
MEDELLIN, COLOMBIA
2013

DOCTORAL PUBLICATION

COPENDIUM ON

Geometry and Topology Extraction and Visualization from Scalar and Vector Fields

DOCTORAL STUDENT

John Congote

ADVISOR

Prof. Dr. Eng. Oscar Ruiz

CO-ADVISOR

Dr. Eng. Jorge Posada

DISSERTATION

Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Engineering in the College of Engineering of the EAFIT University

EAFIT UNIVERSITY
COLLEGE OF ENGINEERING
DOCTORAL PROGRAM IN ENGINEERING
MEDELLIN, COLOMBIA
JUNE 2013

FINAL EXAM OF DISSERTATION
DOCTORAL PROGRAM IN ENGINEERING
EAFIT UNIVERSITY

UNIVERSIDAD EAFIT
Abierta al mundo

Doctoral Student:	**John Edgar Congote Calle**
Thesis Supervisors:	**Prof. Dr. Ing. Oscar Ruiz (U. EAFIT, Supervisor)** **Dr. Ing. Jorge Posada (Vicomtech, Co-supervisor)**
Topic of Examination:	**GEOMETRY AND TOPOLOGY EXTRACTION AND VISUALIZATION FROM SCALAR AND VECTOR FIELDS**
Room:	**27 – 103. Audience Room. Medellin Campus**
Date:	**June 07, 2013**
Time:	**10:00h**
Jury:	**Dr. Ing. Aitor Moreno** **Vicomtech - Visual Interaction and Communication Technologies Centre, San Sebastian, Spain.** **Prof. Dr. Ing. Juan Guillermo Lalinde** **Universidad EAFIT, Colombia** **Prof. Dr. Diego Acosta** **Universidad EAFIT, Colombia**

The Jury considers that the Doctoral Student has **APPROVED** (APPROVED / FAILED) this Final Examination, faced by the Doctoral Student in the mentioned place and date.

Juror Dr. Ing. Aitor Moreno

Juror Prof. Dr. Ing. Juan G. Lalinde

Juror Prof. Dr. Ing. Diego Acosta

witnesses:

Thesis Supervisor
Prof. Dr. Ing. Oscar Ruiz

Thesis Co-supervisor
Dr. Ing. Jorge León Posada

President of the Doctoral Board
Dean Alberto Rodriguez

The College of Engineering of

EAFIT University

announces the

final examination of

John Edgar Congote Calle

for the degree of

Doctor of Philosophy in Engineering

Friday, June 07, 2013
at 10:00 am
Room 27 – 103

Medellin Campus

COLOMBIA

Major Field of Study
Computer Graphics

Dissertation
GEOMETRY AND TOPOLOGY EXTRACTION AND VISUALIZATION FROM SCALAR AND VECTOR FIELDS

Dissertation Committee Chairperson
Prof. Dr. Ing. Oscar Ruiz. Universidad EAFIT, Colombia

Directors of Dissertation Research
Prof. Dr. Ing. Oscar Ruiz. Universidad EAFIT, Colombia
Dr. Ing. Jorge León Posada. VICOMTECH Institute, Spain

Examining Committee
Prof. Dr. Ing. Diego A. Acosta
Universidad EAFIT, Colombia

Dr. Ing. Aitor Moreno
VICOMTECH Institute, Spain

Prof. Dr. Ing. Juan G. Lalinde
Universidad EAFIT, Colombia

President of the Board of the Doctoral Program in Engineering
Dean Alberto Rodriguez Garcia

This examination is open to the public

ABSTRACT

This doctoral work contributes in several major aspects: (1) Extraction of geometrical and topological information of 1- and 2-manifolds present in static and video imaging. This endeavor includes several variants of Marching Cubes algorithm, including enhanced precision iso-surfaces for non-negative scalar fields. (2) Hybrid visualization of scalar and vector fields, along with simplex geometry and topology, underlying in medical, geo-sciences and entertainment data. (3) Optimization of parallel visualization algorithms and usage of texture - optimized hardware as a means of efficient scientific computation and visualization. (4) Definition of WEB ISO-standards for interactive homogeneous visualization of simplex, scalar and vector field data in mobile devices (smart cell phones, pads, tablets, etc.). (5) A novel technique for volume rendering in low-level mobile devices, along with practical, feasible solutions for overcoming current architecture limitations. This doctoral work has been thoroughly screened by the international academic community in the scenario of a vigorous publication activity of the doctoral team, scrutinized by world top experts in the relevant fields.

VITAE

Doctoral Student M.Sc. Eng. John Edgar Congote Calle
John Edgar Congote Calle (Medellín, Colombia, 1982) obtained the Diploma in Computer Science (2007) and the Magister in Science of Engineering (2009) at EAFIT University. He has been Research Assistant of the CAD/CAM/CAE Laboratory EAFIT since 2007, under supervision of Prof. Oscar Ruiz. Since. John Congote has been research intern in the Institute Vicomtech, Spain, under co-supervision of Dr. Ing. Jorge Posada. His research interests are application of Computational Geometry in Medical, Entertainment and Environmental applications, and the specification of cutting edge Web standards for interactive graphics in mobile devices.

Doctoral Supervisor Prof. Dr. Eng. Oscar E. Ruiz
Professor Oscar Ruiz (Tunja, Colombia 1961) obtained Diploma degrees in Mechanical Engineering (1983) and Computer Science (1987) at Los Andes University, Bogotá, Colombia, a M.Sc. degree with emphasis in CAM (1991) and a Ph.D. with emphasis in CAD (1995) from University of Illinois at Urbana- Champaign, USA. Prof. Ruiz has been Visiting Researcher at Ford Motor Co. (USA. 1993, 1995), Fraunhofer Institute for Computer Graphics (Germany 1999, 2001), University of Vigo (Spain 1999, 2002), Max Planck Institute for Informatik (Germany 2004) and Purdue University (USA 2009). In 1996 Prof. Ruiz was appointed as Faculty of the Mechanical Eng. and Computer Science Depts. and as Coordinator of the CAD CAM CAE Laboratory at EAFIT University, Medellin, Colombia. He has supervised several Ph.D. pupils and has coauthored a significant number of scientific publications. His academic interests are Computer Aided Geometric Design, Geometric Reasoning and Applied Computational Geometry.

Doctoral Co-supervisor Dr. Eng. Jorge Leon Posada
Dr. Jorge Posada is Associate Director at Vicomtech (Visual Interaction and Communication Technologies Centre) since 2001. He earned a Ph.D. in Computer Science & Engineering from the Technische Universitaet Darmstadt (Germany), an Executive MBA from IE Business School (Spain), and a degree in Mechanical Engineering (honors) from EAFIT University (Colombia). After working for 4 years as full-time researcher at the Fraunhofer Institute for Computer Graphics (Germany), he joined in 2001 the R&D centre Vicomtech (Spain) as Associate Director. His research interests are in computer graphics, engineering and multimedia technologies applied to the industry. He has co-authored more than 50 publications and has been co-director of several Ph.D. Thesis. He is member of the Editorial Board of the Journal Multimedia Tools and Applications (Springer). He is also active in scientific conferences such as ACM Web3D 2013 (General Chair), ACM Web3D 2012 (General Co-Chair) and Knowledge Engineering Systems KES 2012 (General Co-Chair).

dédiée à Isabelle Robin

Abstract

This doctoral work contributes in several major aspects: (1) Extraction of geometrical and topological information of 1- and 2-manifolds present in static and video imaging. This endeavor includes several variants of Marching Cubes algorithms, including enhanced precision iso-surfaces for non-negative scalar fields. (2) Hybrid visualization of scalar and vector fields, along with simplex geometry and topology, underlying in medical, geo-sciences and entertainment data. (3) Optimization of parallel visualization algorithms and usage of texture - optimized hardware as a means of efficient scientific computation and visualization. (4) Definition of WEB ISO-standards for interactive homogeneous visualization of simplex, scalar and vector field data in mobile devices (smart cell phones, pads, tablets, etc.). (5) A novel technique for volume rendering in low-level mobile devices, along with practical, feasible solutions for overcoming current architecture limitations. This doctoral work has been thoroughly screened by the international academic community in the scenario of a vigorous publication activity of the doctoral team, scrutinized by world top experts in the relevant fields.

Acknowledgements

I would like to express my gratitude to everybody involved to any extend in this thesis and in my life. As, my memory serves me not good enough to remember all the people who participated in this process, I apologize in advance to the ones I forget.

I would like to thank my advisor Prof. Dr. Oscar Ruiz and my co-advisor Dr. Jorge Posada for their help in this endeavor; without them this thesis would be certainly impossible.

I would like to thank also their teams at EAFIT University and Vicomtech-IK4 Research Centre, both of them offered great help during my research and in part this document is the result of their trust and companionship during the toughest moments.

I want dedicate this work to my family; their support and love is certainly unique. To my friends who give me the courage to carry on and last but not less important, I dedicate this work to GOD who gives me the life necessary allowing also these things to happen.

Contents

Introduction

Visualization vs. Computation Geometry is a usual dichotomy in Computer Graphics. While Visualization has as goal to satisfy perception expectations of a human, Computational Geometry aims to model real world objects in a mathematically faithful manner. Visualization has as a prime requisite the speed of producing an acceptably realistic sequence of images. On the other hand, Computational Geometry must produce a sound set of geometrical and topological objects, which are subsequently used for Finite Element Analysis, Computer Aided Design and Manufacturing, Robotics, etc. Traditionally, Visualization is taken as a lesser discipline of the more formal Computational Geometry one. However, such a view is inexact: there are many Computational Geometry techniques that have been inherited from entertainment and games. Examples are the stereo lithography and layer based manufacturing processes, which are 3D extensions of ray-casting techniques. In addition, the significant financial influx of the entertainment industry has permitted research and development of side products for manufacturing, medical imaging, meteorology, and many other serious applications. The present doctoral work aims to reduce the gap between Visualization and Computational geometry, as follows.

Part I of this compendium presents a study of surface reconstruction algorithms which generate geometry from real data emerging from still images and video streams. This field is known as Computer Vision and it provides a bridge between light models and scene geometry. The main problem of Computer Vision is the complexity of both, the light and scene models, which makes the problem computationally intractable. In this domain, this compendium treats the real-time geometry recovery from still and time-dependent images in Face Reconstruction with Structured Light, Real-Time depth map generation architecture for 3D videoconferencing, Evaluation of interest point detectors for image information extraction and Tuning of Adaptive Weight Depth Map Generation Algorithms.

Part II of this compendium addresses the obtainment of geometrical (0- , 1- and 2-simplexes) information from (grid or random) samples of scalar fields, with the ulterior goal of rendering this estimated geometry. This double step is called Indirect Volume Rendering in the present work. Extensions of the Marching Cubes algorithm are implemented to avoid overly detailed meshes and to take into consideration non negative scalar fields. These two problems are ubiquitous in Computer Vision. Publications of this domain are the Extending of Marching Cubes with Adaptive Methods to obtain more accurate iso - surfaces and Marching Cubes in unsigned distance fields for Surface Reconstruction from Unorganized Point Sets.

Part III of this compendium deals with Direct Volume Rendering from scalar fields. In this area, the algorithms do not pass through geometry generation to achieve visualization. Our efforts are directed to accelerate and popularize a usually otherwise expensive rendering in heterogeneous

(including low-end) devices. The publications in this area relate to Interactive visualization of volumetric data with WebGL in real-time, Volume Visual Attention Maps (VVAM) in Ray-Casting Rendering and the contribution of ISO standards MEDX3DOM, MEDX3D and X3DOM.

Part IV of the compendium works in connecting Visualization and Computational Geometry. It presents the implementation of Computational Geometry tasks on current optimized graphic Hardware and Software. In particular, this effort efficiently renders vector and scalar fields (from geo sciences) by using the texture optimized, parallel Graphics processors. Specifically, we present work on Real-time volume rendering and Tractography visualization on the WEB, Visualization of Flow Fields in the Web Platform, and Web Based Hybrid Volumetric Visualization of Urban GIS data Integration of 4D Temperature and Wind Fields with second Level of Detail city (CityGML) models.

Prof. Dr. Ing Oscar Ruiz
Dr. Ing Jorge Posada
Doctoral Student John Congote

Publications

(1) Realtime Dense Stereo Matching with Dynamic Programming in CUDA. John Congote, Javier Barandiaran, Iñigo Barandiaran, Oscar Ruiz. XIX Spanish Congress of Graphical Informatics (CEIG 2009). ISBN 978-3-905673-72-2, pp 231-234. September 2009. Donostia - San Sebastian, Spain. CONFERENCE

(2) Adaptative Cubical Grid For Isosurface Extraction John Congote, Aitor Moreno, Inigo Barandiaran, Javier Barandiaran, Oscar E. Ruiz. 4th International Conference on Computer Graphics Theory and Applications GRAPP-2009. ISBN 978-989-8111-67-8, Thomson Reuters, IET=Inspec, DBPL=uni-trier.de, pp 21-26. Feb 5-8, 2009. Lisbon, Portugal. CONFERENCE

(3) Marching Cubes in an Unsigned Distance Field for Surface Reconstruction from Unorganized Point Sets. John Congote, Aitor Moreno, Iñigo Barandiaran, Javier Barandiaran, Jorge Posada, Oscar Ruiz. GRAPP-2010. International Conference on Computer Graphics Theory and Applications. Angers, France, May 17-21, 2010. ISBN: 978-989-674-026-9, , Thomson Reuters, IET=Inspec, DBPL=uni-trier.de, pp: 143-147. CONFERENCE

(4) Real-time depth map generation architecture for 3D videoconferencing. John Congote, Iñigo Barandiaran, Javier Barandiaran, Tomas Montserrat, Julien Quelen, Christian Feran, Pere Mindan, Olga Mur, Francesc Tarres, Oscar Ruiz. 3DTV-CON2010 Conference on Capture, Transmission and Display of 3D Video. June 7-9 2010, pp. 1-4. Tampere, Finland. ISBN: 978-1-4244-6377-0 (Books), ISBN: 978-1-4244-6378-7 (CD). CONFERENCE

(5) Sistema integrado de generacion de mapas de profundidad para videoconference 3D en tiempo real. John Congote, Iñigo Barandiaran, Javier Barandiaran,Tomas Montserrat, Julien Quelen, Christian Ferran, Pere Mindan, Olga Mur,Genis Aguilar, Francesc Tarres, Oscar Ruiz. Proceedings of XXV Simposium Nacional de la Union Cientifica Internacional de Radio URSI 2010, September ISBN: 978-3-642-11840-1. CONFERENCE

(6) Extending Marching Cubes with Adaptative Methods to Obtain More Accurate Iso-surfaces . John Congote, Aitor Moreno, Iñigo Barandiaran, Javier Barandiaran, Oscar Ruiz. In: book "Computer Vision, Imaging and Computer Graphics. Theory and Applications" , Ranchordas, Alpeshkumar and Pereira, Joao Manuel R.S. (Ed.). Publisher: Springer Berlin Heidelberg. Vol. 68, 2010. pp 35-44. ISBN: 978-3-642-11840-1. BOOK CHAPTER

(7) Face Reconstruction with Structured Light. CONGOTE, John; BARANDIARAN, Iñigo; BARANDIARAN, Javier; NIETO, Marcos; RUIZ, Oscar. VISIGRAPP 2011 6th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications, Thomson Reuters, Publisher=SciTePress, IET=Inspec, DBPL=uni-trier.de, ISBN 978-989-8425-47-8, Vilamoura, Portugal 5-7, March 2011. CONFERENCE

(8) Interactive visualization of volumetric data with WebGL in real-time. CONGOTE, John; KABONGO, Luis; MORENO, Aitor; SEGURA, Alvaro; POSADA, Jorge; RUIZ, Oscar. Sixteenth Annual International Conference on 3D Web Technology (2011) , Paris, France. June 20-22, 2011, pages 137-146, ISBN 9781450307741. CONFERENCE

(9) Statistical Tuning of Adaptive-Weight Depth Map Algorithm. HOYOS, Alejandro; CONGOTE, John; BARANDIARAN, Iñigo; ACOSTA, Diego; RUIZ, Oscar. CAIP 2011. 14th International Conference on Computer Analysis of Images and Patterns. Seville (Spain) 29-31 August 2011. Eds: Real, Pedro; Diaz-Pernil, Daniel; Molina-Abril, Helena; Berciano, Ainhoa; Kropatsch, Walter. pp 563-572. Publisher: Springer Verlag. Series Lecture Notes in Computer Science, Vol 6855. ISBN 978-3-642-23677-8. 2011. CONFERENCE

(10) Hardware-Accelerated Web Visualization of Vector Fields. Case Study in Oceanic Currents.(poster) ARISTIZABAL, Mauricio; CONGOTE, John; SEGURA, Alvaro; MORENO, Aitor; ARRIEGUI, Harbil; RUIZ, Oscar. VISIGRAPP 2012 7th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications, Publisher=SciTePress, Thomson Reuters, ISBN 978-989-8565-02-0, Rome, Italy. Feb 24-26 2012. CONFERENCE

(11) Volume Ray Casting in WebGL. Chapter in Book Computer Graphics. Editor: Prof. Nobuhiko Mukai. Authors: John Congote, Luis Kabongo, Aitor Moreno, Alvaro Segura, Andoni Beristain, Jorge Posada and Oscar Ruiz, 2012. Publisher: Intech. ISBN: 978-953-51-0455-1. InTech. BOOKCHAPTER

(12) Tuning of Adaptive Weight Depth Map Generation Algorithms Exploratory Data Analysis and Design of Computer Experiments (DOCE). Diego Acosta, Iñigo Barandiaran, John Congote, Oscar Ruiz, Alejandro Hoyos and Manuel Graña. Journal of Mathematical Imaging and Vision, Springer Verlag Netherlands. Online First, 6 July 2012. ISSN 0924-9907. DOI: 10.1007/s10851-012-0366-7. JOURNAL

(13) Realtime Visualization of the Connectome in the Browser using WebGL. Ginsburg Daniel, Gerhard Stephan, Calle John Congote, Pienaar Rudolph. Frontiers in Neuroinformatics. doi:10.3389/conf.fninf.2011.08.00095. ISSN=1662-5196. JOURNAL

(14) Realtime Volume Rendering and Tractography Visualization on the Web. J. Congote and E. Novo and L. Kabongo and D. Ginsburg and S. Gerhard and R. Pienaar and O. Ruiz. Journal of WSCG. v.20, n.2, 2012. ISSN 1213-6972. pages 81-88. WoS ISI and Tompson Reuters. url http://wscg.zcu.cz/wscg2012/ JOURNAL

(15) Visualization of Flow Fields in the Web Platform. Mauricio Aristizabal and John Edgar Congote and Alvaro Segura and Aitor Moreno and Harbil Arriegui and Oscar E. Ruiz. Journal of WSCG. v.20, n.3, 2012. ISSN 1213-6972. pages 189-196. WoS ISI and Tompson Reuters. url= http://wscg.zcu.cz/wscg2012/ JOURNAL

(16) Volume Visual Attention Maps (VVAM) in ray-casting rendering. Beristain, Andoni, John Congote, and Oscar Ruiz.. Studies in Health Technology and Informatics 173 (2012): 53. PMID: 22356956 [PubMed - indexed for MEDLINE], JOURNAL

(17) Web based hybrid volumetric visualisation of urban GIS data. Congote, J., Moreno, A., Kabongo, L., Perez, J. L., San-Jose, R., Ruiz, O. published on-line: 24 October 2012. DOI http://dx.doi.org/10.1051/3u3d/201203001 , Event 3u3D2012 Usage, Usability, and Utility of 3D City Models European COST Action TU0801 2012. EDP Sciences. CONFERENCE

(18) Medx3dom: Medx3d for x3dom. Congote, J. (2012, August). 17th International Conference on 3D Web Technology (pp. 179-179). ACM. ISBN 978-1-4503-1432-9. doi: 10.1145/2338714.2338746. CONFERENCE
(19) Advanced HCI and 3D Web over Low Performance Devices.David Oyarzun and Arantza del Pozo and John Edgar Congote and Igor Garcia and Iñaki Sainz and Igor Leturia and Xabier Arregi and O. Ruiz. Proceedings of the 1st International Workshop on Declarative 3D for the Web Architecture (Dec3D2012 at WWW2012) Lyon, France, 2012 CONFERENCE
(20) Evaluation of interest point detectors for image information extraction. Iñigo Barandiaran, John Congote, Jon Goenetxea, Oscar Ruiz. Pages 2170 - 2179. DOI 10.3233/978-1-61499-105-2-2170. Frontiers in Artificial Intelligence and Applications, Volume 243: Advances in Knowledge-Based and Intelligent Information and Engineering Systems CONFERENCE
(21) ReWeb3D - Enabling Desktop 3D Applications to Run in the Web. Tassilo Glander, Aitor Moreno, Mauricio Aristizabal, John Congote, Jorge Posada, Alejandro Garcia-Alonso, Oscar Ruiz. 18th International Conference on 3D Web Technology. Donostia - San Sebastian, 2013. CONFERENCE

Part 1

Surface Reconstruction

CHAPTER 1

Tuning of Adaptive Weight Depth Map Generation Algorithms

- Diego Acosta[3]
- John Congote[1,2]
- Iñigo Barandiaran[2]
- Oscar Ruiz[1]
- Alejandro Hoyos[1]
- Manuel Graña[4]

[1] CAD CAM CAE laboratory, Universidad EAFIT
Carrera 49 No 7 Sur - 50, Medellín, Colombia
[2] Vicomtech Research Center
Donostia - San Sebastián, Spain
[3] DDP Research Group, Universidad EAFIT
Carrera 49 No 7 Sur - 50, Medellín, Colombia
[4] Dpto. CCIA, UPV-EHU

Context

Tuning of Adaptive Weight Depth Map Generation Algorithms Exploratory Data Analysis and Design of Computer Experiments (DOCE). Diego Acosta, Iñigo Barandiaran, John Congote, Oscar Ruiz, Alejandro Hoyos and Manuel Graña. Journal of Mathematical Imaging and Vision, Springer Verlag Netherlands. Online First, 6 July 2012. ISSN 0924-9907. DOI: 10.1007/s10851-012-0366-7. JOURNAL

The Spanish project VISION develops new models for immersed and virtual communication. 3D videoconference is one of the studied models, and a new kind of real-time depth-map algorithms were researched. These are a methodology to reconstruct 3D surfaces from two or more cameras, several methodologies had been proposed in the literature. However, researchers select the running parameters of the algorithms empirically. This work presents a formal methodology to select the parameters for these algorithms optimizing the results.

The research was made by a collaboration between Vicomtech research center and the laboratories of DPP and CAD/CAM/CAE of EAFIT University. This work has been partially supported by the Spanish Administration Agency CDTI under project CENIT-VISION 2007-1007, the Colombian Administrative Department of Science, Technology, and Innovation; and the Colombian National Learning Service (COLCIENCIAS-SENA) grant No. 1216-479-22001.

As co-authors of such publication, we give our permission for this material to appear in this document. We are ready to provide any additional information on the subject, as needed.

Prof. Dr. Eng. Oscar E. Ruiz
oruiz@eafit.edu.co
Coordinator CAD CAM CAE Laboratory
EAFIT University, Medellin, COLOMBIA

Eng. Alejandro Hoyos
ahoyossi@eafit.edu.co
CAD CAM CAE Laboratory
EAFIT University, Medellin, COLOMBIA

M.Sc. Iñigo Barandiaran
ibarandiaran@vicomtech.org
eHealth & Biomedical Applications
Vicomtech, San Sebastian, SPAIN

Prof. Dr. Eng. Diego Acosta
dacostam@eafit.edu.co
Research Group DDP
EAFIT University, Medellin, COLOMBIA

Pr. Manuel Graña
manuel.grana@ehu.es
Dpto. CCIA
Universidad del Pais Vasco-EHU

Abstract

In depth map generation algorithms, parameters settings to yield an accurate disparity map estimation are usually chosen empirically or based on unplanned experiments. Algorithms' performance is measured based on the distance of the algorithm results vs. the Ground Truth by Middlebury's standards. This work shows a systematic statistical approach including exploratory data analyses on over 14000 images and designs of experiments using 31 depth maps to measure the relative influence of the parameters and to fine-tune them based on the number of bad pixels. The implemented methodology improves the performance of adaptive weight based dense depth map algorithms. As a result, the algorithm improves from 16.78% to 14.48% bad pixels using a classical exploratory data analysis of over 14000 existing images, while using designs of computer experiments with 31 runs yielded an even better performance by lowering bad pixels from 16.78% to 13%.

Keywords: Stereo Image Processing, Parameter Estimation, Depth Map, Statistical Design of Computer Experiments

1. Introduction

Depth map calculation deals with estimation of multiple object depths on a scene. It is useful for applications like vehicle navigation, automatic surveillance, aerial cartography, passive 3D scanning, industrial inspection, or 3D videoconferencing ([**1**]). These maps are constructed by generating, at each pixel, an estimation of the distance from the camera to the object surface.

Disparity is commonly used to describe inverse depth in computer vision, and to measure the perceived spatial shift of a feature observed from close camera viewpoints. Stereo correspondence techniques often calculate a disparity function $d(x, y)$ relating target and reference images, so that the (x, y) coordinates of the disparity space match the pixel coordinates of the reference image. Stereo methods commonly use a pair of images taken with a known camera geometry to generate a dense disparity map with estimates at each pixel. This dense output is useful for applications requiring depth values even in difficult regions like occlusions and textureless areas. The ambiguity of matching pixels in these zones requires complex and expensive global image processing or statistical correlations using color and proximity measures in local support windows. The steps generally taken to compute the depth maps may include: (i) matching cost computation, (ii) cost or support aggregation, (iii) disparity computation or optimization, and (iv) disparity refinement.

In this article, section 2 reviews the state-of-the-art. Section 3 describes our algorithm (filters, statistical analyses and experimental set-up). Section 4 discusses the results. Section 5 concludes the article.

2. Literature Review

Depth-map generation algorithms and filters use several user-specified parameters to generate a depth map from an image pair. The settings of these algorithms are heavily influenced by the evaluated data sets ([**2**]). Published works usually report the settings used for their specific case studies without describing the procedure followed to fine-tune them ([**1, 3, 4**]), and some explicitly state the empirical nature of these values ([**5**]). The variation of the output as a function of several settings on selected parameters is briefly discussed while not taking into account the effect of modifying them all simultaneously ([**2, 3, 6**]). Reference [**7**] compares multiple stereo methods whose parameters are based on experiments. Only some parameters are tuned, without explaining the choices made. In the present article, we improve upon this work. In [**8, 9**], Depth Maps are generated from single images instead of image pairs.

2.1. Literature Review Conclusions. Used approaches in determining the settings of depth map algorithm parameters show all or some of the following shortcomings: (i) undocumented procedures for parameter setting, (ii) lack of planning when testing for the best settings, and (iii) failure to consider interactions of changing all parameters simultaneously.

As a response to these disadvantages, this article presents a methodology to fine-tune user-specified parameters on a depth map algorithm using a set of images from the adaptive weight implementation in [**1**]. Multiple settings are used and evaluated on all parameters to measure the contribution of each parameter to the output variance. A quantitative evaluation uses main effects plots and variance on multi-variate linear regression models to select the best combination of settings. Performance improves by setting new estimated values of user-specified parameters, allowing the algorithm to give much more accurate results on a rectified image pair.

Since it is not always feasible to have a large set of images available, a fractional factorial design of computer experiment (DOCE) with only eight runs is used to find out which parameters have a major influence on the images tested. To optimize the parameters and to have the lowest percentage of bad pixels a central composite DOCE with 23 runs is used with the most influential parameters found in the fractional factorial design. To the best of our knowledge the systematic and efficient application of DOCE in the field of depth maps generation has not been done yet.

3. Methodology

3.1. Image Processing. In adaptive weight algorithms ([**1, 3**]), a window is moved over each pixel on every image row, calculating a measurement based on the geometric proximity and color similarity of each pixel in the moving window to the pixel on its center. Pixels are matched on each row based on their support measurement with larger weights coming from similar pixel colors and closer pixels. The horizontal shift, or disparity, is recorded as the depth value, with higher values reflecting greater shifts and closer proximity to the camera.

The strength of grouping by color $(f_s(c_p, c_q))$ for pixels p and q is defined as the Euclidean distance between colors (Δc_{pq}) by Equation (1). Similarly, grouping strength by distance $(f_p(g_p, g_q))$ is defined as the Euclidean distance between pixel image coordinates (Δg_{pq}) as per Equation (2). γ_c and γ_p are adjustable settings used to scale the measured color delta, represented as aw_col in the study, and window size represented as aw_win respectively.

$$f_s(c_p, c_q) = exp\left(-\frac{\Delta c_{pq}}{\gamma_c}\right) \tag{1}$$

$$f_p(g_p, g_q) = exp\left(-\frac{\Delta g_{pq}}{\gamma_p}\right) \tag{2}$$

The matching cost between pixels shown in Equation (3) is measured by aggregating raw matching costs, using the support weights defined by Equations (1) and (2), in support windows based on both the reference and target images.

$$E(p, \bar{p}_d) = \frac{\sum\limits_{q \in N_p, \bar{q}_d \in N_{\bar{p}_d}} w(p,q)\, w(\bar{p}_d, \bar{q}_d) \sum\limits_{c \in \{r,g,b\}} |I_c(q) - I_c(\bar{q}_d)|}{\sum\limits_{q \in N_p, \bar{q}_d \in N_{\bar{p}_d}} w(p,q)\, w(\bar{p}_d, \bar{q}_d)} \tag{3}$$

where $w(p,q) = f_s(c_p, c_q) \cdot f_p(g_p, g_q)$, $\bar{p}_d$ and $\bar{q}_d$ are the target image pixels at disparity d corresponding to pixels p and q in the reference image, I_c is the intensity on channels red (r), green (g), and blue (b), and N_p is the window centered at p and containing all q pixels. The size of this movable window N is a derived

TABLE 1.1. Input and Output Variables of Depth Maps Generation Algorithms

INPUT VARIABLES

Adaptive Weight ([3]): Disparity estimation and pixel matching with γ_{aws}: similarity factor, and γ_{awg}: proximity factor related to the W_{AW} pixel size of the support window as user-adjustable parameters

Parameter	*Description*	*Values*
aw_win	Adaptive Weights Window Size	[1 3 5 7]
aw_col	Adaptive Weights Color Factor	[4 7 10 13 16 19]

Median: Smoothing and incorrect match removal with W_M: pixel size of the median window as user-adjustable parameter

Parameter	*Description*	*Values*
m_win	Median Window Size	[N/A 3 5]

Cross-check ([8]): Validation of measurement per pixel with Δ_d: allowed disparity difference as adjustable parameter

Parameter	*Description*	*Values*
cc_disp	Cross-Check Disparity Delta	[N/A 0 1 2]

Bilateral ([9]): Intensity and proximity weighted smoothing with edge preservation with γ_{bs}: similarity factor, and γ_{bg}: proximity factor related to the W_B pixel size of the bilateral window as user-adjustable parameters

Parameter	*Description*	*Values*
cb_win	Cross-Bilateral Window Size	[N/A 1 3 5 7]
cb_col	Cross-Bilateral Color Factor	[N/A 4 7 10 13 16 19]

OUTPUT VARIABLES

`rms_error_all`	Root Mean Square (RMS) disparity error (all pixels)
`rms_error_nonocc`	RMS disparity error (non-occluded pixels only)
`rms_error_occ`	RMS disparity error (occluded pixels only)
`rms_error_textured`	RMS disparity error (textured pixels only)
`rms_error_textureless`	RMS disparity error (textureless pixels only)
`rms_error_discont`	RMS disparity error (near depth discontinuities)
`bad_pixels_all`	Fraction of bad points (all pixels)
`bad_pixels_nonocc`	Fraction of bad points (non-occluded pixels only)
`bad_pixels_occ`	Fraction of bad points (occluded pixels only)
`bad_pixels_textured`	Fraction of bad points (textured pixels only)
`bad_pixels_textureless`	Fraction of bad points (textureless pixels only)
`bad_pixels_discont`	Fraction of bad points (near depth discontinuities)

parameter of (aw_win). Increasing the window size reduces the chance of bad matches at the expense of missing relevant scene features.

3.2. Post-Processing Filters. Algorithms based on correlations depend heavily on finding similar textures at corresponding points in both reference and target images. Bad matches happen more frequently in textureless regions, occluded zones, and areas with high variation in disparity, such as discontinuities. The winner-takes-it-all approach enforces uniqueness of matches only for the reference image so that points on the target image are matched more than once, creating the need to check the disparity estimates and to fill any gaps with information from neighboring pixels using post-processing filters like the ones discussed next (Table 1.1).

Median Filter (m) is widely used in digital image processing to smooth signals and to remove incorrect matches and holes by assigning neighboring disparities at the expense of edge preservation. The median filter provides a mechanism for reducing image noise, while preserving edges more effectively than a linear smoothing filter. It sorts the intensities of all q pixels on a window of size M and selects the median value as the new intensity of the p central pixel. The size M of the window is another of the user-specified parameters.
Cross-check Filter (cc) performs twice the correlation by reversing the roles of the two images (reference and target) and considering valid only those matches having similar depth measures at corresponding points in both steps. The validity test is prone to fail in occluded areas where disparity estimates will be rejected. The allowed difference in disparities between reference and target images is one more adjustable parameter.
Bilateral Filter (cb) is a non-iterative method of smoothing images while retaining edge detail. The intensity value at each pixel in an image is replaced by a weighted average of intensity values from nearby pixels. The weighting for each pixel q is determined by the spatial distance from the center pixel p, as well as its relative difference in intensity, defined by Equation (4).

$$O_p = \frac{\sum_{q \in W} f_s\,(q-p)\, g_i\,(I_q - I_p) I_q}{\sum_{q \in W} f_s\,(q-p)\, g_i\,(I_q - I_p)} \tag{4}$$

O_p is the output image, I the input image, W the weighting window, f_s the spatial weighting function, and g_i the intensity weighting function. The size of the window W is yet another parameter specified by the user.

3.3. Experimental Set-up. Our depth maps are calculated with an implementation developed for real time videoconferencing ([**1**]). We use well-known rectified image sets: Cones from [**7**], Teddy and Venus from [**10**], and Tsukuba head and lamp from the University of Tsukuba. Our dataset consists of 14688 depth maps, 3672 for each data set, like the ones shown in Figure 1.1.

(a) (b) (c) (d)

FIGURE 1.1. Depth Map Comparison. Top: best initial, bottom: new settings. (a) Cones, (b) Teddy, (c) Tsukuba, and (d) Venus data set.

Many recent stereo correspondence performance studies use the Middlebury Stereomatcher for their quantitative comparisons ([**2, 6, 11**]). The evaluator code, sample scripts, and image data sets are available from the Middlebury stereo vision site, providing a flexible and standard platform for easy evaluation.

The online Middlebury Stereo Evaluation Table gives a visual indication of how well the methods perform with the proportion of bad pixels metric (`bad_pixels`) defined as the average of the proportion of bad pixels in the whole image (`bad_pixels_all`), the proportion of bad pixels in non-occluded regions (`bad_pixels_nonocc`), and the proportion of bad pixels in areas near depth discontinuities (`bad_pixels_discont`) in all data sets. A bad pixel represents a pixel where the estimated disparity is wrong with respect to a ground thruth disparity value.

3.4. Statistical Analyses. The user-specified input parameters and output accuracy data are statistically analyzed to correlate them (see Table 1.1). Box plots give insights on the influence of settings on a given response variable. Equation (5) relates $\hat{y}$ (predicted response) with x_i (input factors). β_0 and β_i are the coefficients fit by multi-variable linear regression. Constant Variance and Null Mean of Residuals help to validate the assumptions of the regression model. When those assumptions are not fulfilled, the model is modified ([**12**]). The parameters are normalized to fit the range $(-1, 1)$ at their values shown in Table 1.1.

$$\hat{y} = \beta_0 + \sum_{i=1}^{n} \beta_i x_i + \epsilon \tag{5}$$

Having a large data set (in this case 14688 images) to perform statistical analyses is not always feasible. DOCE is applied here to obtain an equivalently good model for the depth map, by having a much smaller number of runs. A 2^{6-3} fractional factorial DOCE with just eight runs allows to establish which ones of the parameters aw_win, aw_colo, m_win, cc_disp, cb_win, and cb_col are the most influential on the `bad_pixels` output by using a Daniel plot ([**13**]). The parameters whose distribution cannot be considered as normal standard are statistically relevant in the fractional DOCE. Therefore, they are used to optimize the depth map generation algorithm.

A surface response central composite DOCE with 23 runs was performed afterward with aw_win, aw_colo, m_win, and cb_win as studied factors while keeping constant the remaining parameters (i.e., $cc\_disp = 2$ y $cb\_col = 13$) to yield a mathematical model of the form:

$$\hat{y} = \beta_0 + \sum_{i}^{k} \beta_i x_i + \sum_{ii}^{k} \beta_{ii} x_i^2 + \sum_{i<j} \beta_{ij} x_i x_j \tag{6}$$

where, as in equation 5, $\hat{y}$ is the predicted variable, x_i are the parameters, and β_0, β_i, β_{ii} and β_{ij} are constants adjusted by minimum least squares regression. Data from DOCE was analysed with the software for statistical computing R with Bayes Screening and Model Discrimination -BsMD- and Response Surface Method -rsm- add-on packages ([**14**]).

4. Results and Discussion

4.1. Selection of Input Variables for Mathematical Model. Response variables for depth map generation algorithms are shown with their meaning in Table 1.1. Pearson multiple correlation coefficients for the response variables shown in Table 1.2 evidences that `bad_pixels_all` is strongly correlated to the remaining response variables. This means that all response variables follow a similar trend as `bad_pixels_all` and that modeling `bad_pixels_all` is sufficient to reach statistically sound results for depth map generation algorithms optimization.

On the other hand, low Pearson coefficients for the input variables indicate that those variables are independent, that there is no co-linearity among them and that each independent variable must be included in the exploratory analysis.

TABLE 1.2. Pearson correlation coefficient for the evaluator outputs over all data sets

	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)	(11)	(12)	(13)
(1)bad_pixels	1.00	0.81	0.82	0.59	0.83	0.77	0.84	1.00	1.00	0.86	1.00	0.95	0.99
(2)rms_error_all	0.81	1.00	1.00	0.69	1.00	0.98	0.99	0.82	0.82	0.64	0.85	0.70	0.79
(3)rms_error_nonocc	0.82	1.00	1.00	0.71	1.00	0.98	0.99	0.83	0.82	0.67	0.85	0.71	0.80
(4)rms_error_occ	0.59	0.69	0.71	1.00	0.70	0.77	0.74	0.62	0.61	0.68	0.61	0.63	0.53
(5)rms_error_textured	0.83	1.00	1.00	0.70	1.00	0.98	0.99	0.83	0.83	0.67	0.86	0.72	0.81
(6)rms_error_textureless	0.77	0.98	0.98	0.77	0.98	1.00	0.98	0.78	0.78	0.64	0.80	0.68	0.73
(7)rms_error_discont	0.84	0.99	0.99	0.74	0.99	0.98	1.00	0.85	0.84	0.67	0.87	0.73	0.82
(8)bad_pixels_all	1.00	0.82	0.83	0.62	0.83	0.78	0.85	1.00	1.00	0.85	1.00	0.96	0.98
(9)bad_pixels_nonocc	1.00	0.82	0.82	0.61	0.83	0.78	0.84	1.00	1.00	0.85	1.00	0.96	0.98
(10)bad_pixels_occ	0.86	0.64	0.67	0.68	0.67	0.64	0.67	0.85	0.85	1.00	0.83	0.87	0.86
(11)bad_pixels_textured	1.00	0.85	0.85	0.61	0.86	0.80	0.87	1.00	1.00	0.83	1.00	0.93	0.99
(12)bad_pixels_textureless	0.95	0.70	0.71	0.63	0.72	0.68	0.73	0.96	0.96	0.87	0.93	1.00	0.93
(13)bad_pixels_discont	0.99	0.79	0.80	0.53	0.81	0.73	0.82	0.98	0.98	0.86	0.99	0.93	1.00

4.2. Exploratory Data Analysis. Box plots analyses of bad_pixels presented in Figure 1.2 shows lower output values from using filters, relaxed cross-check disparity delta values, large adaptive weight window sizes, and large adaptive weight color factor values. The median window size, bilateral window size, and bilateral window color values do not show a significant influence on the output at the studied levels.

The influence of the parameters is also shown by the value of the slopes of the main effects plots in Figure 1.3 and confirms the behavior found with the analysis of variance (ANOVA) of the multi-variate linear regression model. The optimal settings from this analysis (i.e., $aw\_win = 9$, $aw\_col = 22$, $m\_win = 5$, $cc\_disp = 1$, $cb\_win = 3$ and $cb\_col = 4$) to minimize bad_pixels yields a result of 14.48%.

4.3. Multi-variate Linear Regression Model. The analysis of variance on a multi-variate linear regression (MVLR) over all data sets using the most parsimonious model quantifies the parameters with the most influence as shown in Table 1.3. The most significant input variable is cc_disp, since it accounts for a [33%-50%] of the variance in every case.

TABLE 1.3. Linear model ANOVA with the contribution to the sum of squared errors (SSE) of bad_pixels.

Data set	cc_disp	aw_win	aw_col	cb_win
Cones	34.35%	14.46%	17.47%	–
Teddy	41.25%	13.75%	8.10%	–
Tsukuba	50.25%	–	–	7.16%
Venus	47.35%	9.42%	–	5.62%
All	47.01%	8.11%	–	–

Interactions and higher order terms are included on the multi-variate linear regression models to improve the goodness of fit. Reducing the number of input images per dataset from 3456 to 1526 by excluding the worst performing cases ($cc\_disp = 0$, $aw\_col = 4$ and $aw\_col = 7$), using a cubic model with interactions yields a very good multiple correlation coefficient of $R^2 = 99.05\%$. However, for the model selected the residuals distribution is not normal even after transforming the response variable and removing large residuals values. Another constraint for the statistical analyses is that any outliers from the data set can not be excluded. Nonetheless, improved algorithm performance settings are found using the model to obtain lower bad_pixels values comparable to the ones obtained through the exploratory data analysis (14.66% vs. 14.48%).

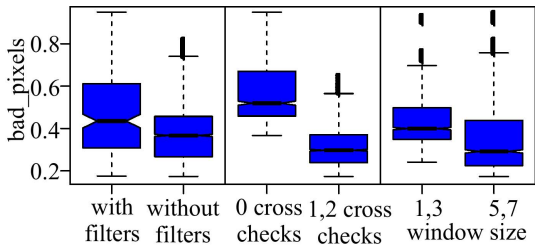


FIGURE 1.2. Box Plots for Input Variable Analysis

In summary, the most noticeable influence on the output variable comes from having a relaxed cross-check filter, accounting for nearly half the response variance in all the study data sets. Window size is the next most influential factor, followed by color factor, and finally window size on the bilateral filter. Increasing the window size on the main algorithm yields better overall results at the expense of longer running times and some foreground loss of sharpness, while the support weights on each pixel have the chance of becoming more distinct and potentially reduce disparity mismatches. Increasing the color factor on the main algorithm allows better results by reducing the color differences, and slightly compensating minor variations in intensity from different viewpoints.

A small median smoothing filter window size is faster than a larger one, while still having a similar accuracy. Low settings on both the window size and the color factor on the bilateral filter seem to work best for a good trade-off between performance and accuracy.

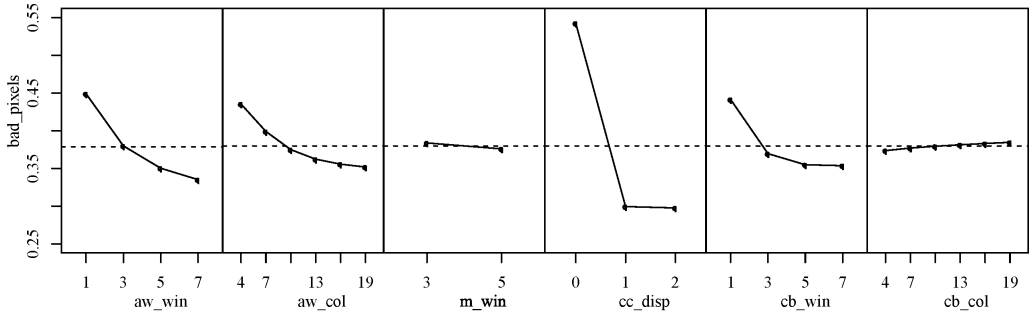


FIGURE 1.3. Main Effects Plots of each factor level for all data sets. Steeper slopes relate to bigger influence on the variance of the `bad_pixels` output measurement.

The optimal settings in the original data set are presented in Table 1.4 along with the proposed settings. **Low settings** comprise the depth maps with all their parameter settings at each of their minimum tested values yielding 67.62% `bad_pixels`. **High settings** relates to depth maps with all their parameter settings at each of their maximum tested values yielding 19.84% `bad_pixels`. **Best initial** are the most accurate depth maps from the study data set yielding 16.78% `bad_pixels`. **Exploratory analysis** corresponds to the settings determined using the exploratory data analysis based on box plots and main effects plots yielding 14.48% `bad_pixels`. **MVLR optimization** is the optimization of the classical data analysis based on multi-variate linear regression model, nested models, and ANOVA yielding 14.66% `bad_pixels`.

The exploratory analysis estimation and the MVLR optimization tend to converge at similar lower `bad_pixels` values using the same image data set. The best initial and improved depth map outputs are shown in Figure 1.1. The best runs for fractional factorial and central composite DOCEs lower the value of the `bad_pixels` variable to 14.72% and 13.05%, respectively. Notice that to achieve these results only 31 depth maps are needed (DOCE) as opposed to analyzing over 14000 depth maps (Exploratory Analysis).

TABLE 1.4. Model comparison. Average `bad_pixels` values over all data sets and their parameter settings.

Run Type	**`bad_pixels`**	aw_win	aw_col	m_win	cc_disp	cb_win	cb_col
Low Settings	67.62%	1	4	3	0	1	4
High Settings	19.84%	7	19	5	2	7	19
Best Initial	16.78%	7	19	5	1	3	4
Exploratory analysis	14.48%	9	22	5	1	3	4
MVLR optimization	14.66%	11	22	5	3	3	18
Best Treatment for							
Fractional Factorial DOCE	14.72%	10	25	3	3	1	3
Best Treatment for CCD DOCE	13.05%	7	14	3	4	1	13

4.4. Depth-map optimization by design of computer experiments (DOCE). 2^{6-3} *Fractional Factorial Design of Experiment*.

The goal of this type of design of experiment is to screen the statistically most significant parameters. Details on how to set up the runs are discussed in [**12**]. The design matrix describing all experimental runs can be set so that the high and low levels for each parameter are chosen by assigning them the maximum and minimum values allowed by the algorithm respectively. This was done for all of the parameters but for m_win (i.e., it was set at the levels 3 and 5), to avoid bias from the results and conclusions obtained from the exploratory and multivariate regression analysis. The results for this DOCE range from 14.72% and 72.17% bad pixels for all images which is quite promising because already with only eight runs a set of parameters values that is very close to the optimum obtained by exploratory analysis of 14.48% bad pixels and the multivariate linear regression analysis of 14.66% on the 14688 data points is delivered. The alias for the parameters and Daniel plot showing the most relevant ones are shown in Figure 1.4.

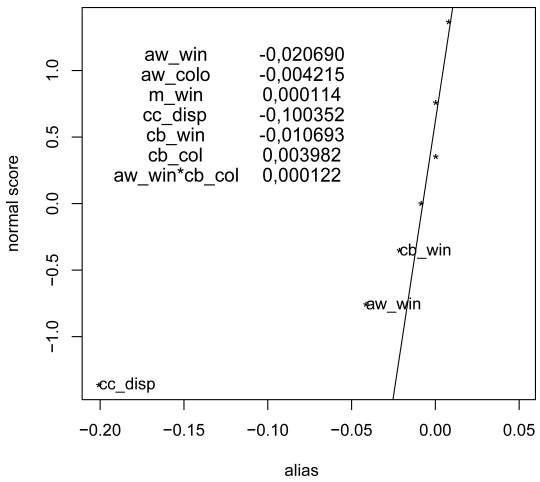


FIGURE 1.4. Daniel Plot for determining the significance of input variables.

Daniel's plot indicate that the most influential parameters are cc_disp, aw_win and cb_win which deviate the most from the normal distribution curve. These parameters and m_win at levels 0, 3 and 5 are used for the surface response methodology central composite design of experiment that follows.

Central Composite Design of Experiment.

To further optimize the depth maps generation algorithm a central composite design of experiment is used. As with the fractional factorial design of experiment, the best run with 13.05% bad pixels is obtained

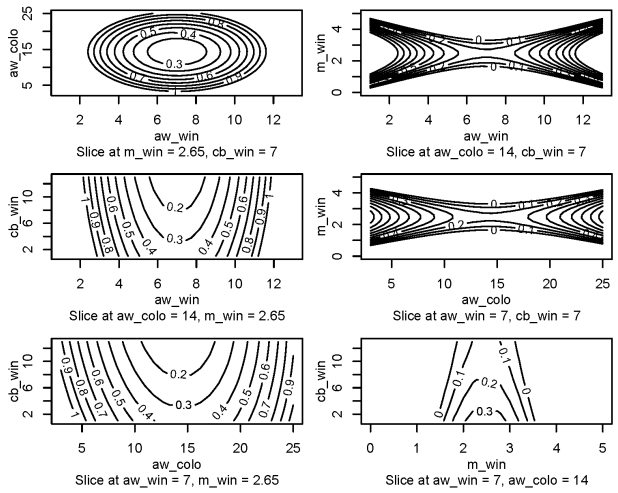


FIGURE 1.5. Contour Plots for Central Composite DOCE.

amongst the 23 treatments which surpasses the results obtained thus far. The outputs from R using the rsm package at the levels tested for each parameter are shown in Table 1.5.

As it can be seen the second order model depicted before in equation 6 fits very well the data as indicated by the multiple correlation coefficient 0.9695. The most significant variables include aw_win, aw_col, m_win, cb_win, aw_win^2, aw_col^2, and m_win^2. Nonetheless, the complete model with all coefficients is used to draw the contour plots shown later. The rsm package also allows to detect stationary points. In this case the stationary point detected is a saddle point because one of the eigen-values is negative while the remaining ones are positive

Graphically the iso-lines for `bad_pixels_all` are seen in slices by looking at two parameters simultaneously for the analysis while keeping the remaining ones constant as shown in Figure 1.5. The graphs allow to see that the stationary point does indicate a local minimum when analyzing for aw_win and aw_col. With m_win though, the graph indicates that a saddle is detected and that it is better to use values not in the $1,5 < m\_win < 3,5$ interval (which is physically imposible). For cb_win the stationary point apparently corresponds to a minimum. The settings for the stationary point closer to what rsm's package detects are $aw\_size = 7$, $aw\_col = 14$, $m\_size = 3$, $cc\_disp = 2$, $cb\_size = 21$ and $cb\_col = 13$ and this yields 26% `bad_pixels_all` leading to conclude that the best treatment for the rsm yielding 13.05% of `bad_pixels_all` is the local minimum optimum at the settings shown on Table 1.4.

5. Conclusions and Future Work

Previously published material in [**15**] showed how Exploratory Analysis, applied on over 14000 images, allowed the sub-optimal tuning of the parameters for Disparity Estimation algorithms, lowering the percentage of bad pixels from from 16.78% (manual tuning) to 14.48 %. The present work shows how to use DOCE to optimize the tuning, by running a dramatically smaller sample (31 experiments). The result of applying DOCE allowed to reach 13.05% of bad pixels, without the need of Exploratory Analysis. Using DOCE reduces the number of depth maps needed to carry out the study when a large image database is not available. The DOCE methodology itself is independent of the particular algorithms used to generate the disparity maps and it can be used whenever a systematic tunning of process parameters is required.

An improvement from 16.78% (manual tuning) to 13.05% in the `bad_pixels_all` variable might seem negligible at first glance. However, such figures imply a jump of the optimized algorithm of almost 10

TABLE 1.5. Summary of RSM Central Composite DOCE

Parameters	Levels				
aw_win	1	4	7	10	13
aw_colo	3	8.5	14	19.5	25
m_win	0	3	5		
cb_win	1	4	7	10	13
cb_colo	13				
cc_disp	2				

Call: rsm(formula=`bad_pixels_all` $\sim$ SO(aw_win,aw_colo,m_win,cb_win))

Coefficients	Estimate	Std. Error	t-value	$p > \|t\|$	Signif.
(Intercept)	1.634	2.49×10^{-1}	6.561	0.00018	***
aw_win	-5.25×10^{-1}	8.03×10^{-2}	-6.538	0.00018	***
aw_colo	-1.9×10^{-1}	4.78×10^{-2}	-3.971	0.00411	**
m_win	1.606	4.22×10^{-1}	3.802	0.00522	**
cb_win	-3.96×10^{-2}	8.03×10^{-2}	-0.493	0.63495	
aw_win:aw_colo	4.15×10^{-5}	1.59×10^{-4}	0.26	0.80128	
aw_win:m_win	-6.13×10^{-5}	7.01×10^{-4}	-0.087	0.93243	
aw_win:cb_win	1.73×10^{-4}	2.92×10^{-4}	0.592	0.56990	
aw_colo:m_win	3.01×10^{-4}	3.82×10^{-4}	0.788	0.45339	
aw_colo:cb_win	5.33×10^{-4}	1.59×10^{-4}	3.347	0.01013	*
m_win:cb_win	5.56×10^{-4}	7.01×10^{-4}	0.793	0.45083	
aw_win^2	3.73×10^{-2}	5.73×10^{-3}	6.508	0.00019	***
aw_colo^2	6.44×10^{-3}	1.70×10^{-3}	3.78	0.00539	**
m_win^2	-3.25×10^{-1}	8.45×10^{-2}	-3.846	0.00490	**
cb_win^2	7.19×10^{-4}	5.73×10^{-3}	0.126	0.90314	

Significance codes: 0 *** 0.001 ** 0.01 * 0.05 . 0.1 1
Residual standard error: 0.04208 on 8 degrees of freedom
Multiple R-squared: 0.9695, Adjusted R-squared: 0.916
F-statistic: 18.13 on 14 and 8 DF, p-value: 0.0001577

Stationary point at response surface		Eigen-values	
aw_win	6.987	λ_1	0.0373
aw_colo	13.788	λ_2	0.0064
m_win	2.495	λ_3	0.0007
cb_win	20.615	λ_4	-0.3249

positions in the Middlebury Stereo Evaluation ranking. It must be noticed that many algorithms competing in such a rank could benefit from the systematic tunning presented here.

A Surface Reconstruction application with DOCE uses the optimal tuning of disparity maps between two stereoscopic images scanning a scene. The disparity between the images, in turn, allows the triangulation of the 3D points on the surface of objects in the scene. This point cloud is an input to surface reconstruction algorithms. This process is discussed in detail in [**1**]. DOCE applications in other domains are indeed possible.

CHAPTER 2

Face Reconstruction with Structured Light

- John Congote[1,2]
- Iñigo Barandiaran[2]
- Javier Barandiaran[2]
- Marcos Nieto[2]
- Oscar Ruiz[1]

[1] CAD CAM CAE laboratory, Universidad EAFIT
Carrera 49 No 7 Sur - 50, Medellín, Colombia
[2] Vicomtech Research Center
Donostia - San Sebastin, Spain

Context

Face Reconstruction with Structured Light. CONGOTE, John; BARANDIARAN, Iñigo; BARANDIARAN, Javier; NIETO, Marcos; RUIZ, Oscar. VISIGRAPP 2011 6th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications, Thomson Reuters, Publisher=SciTePress, IET=Inspec, DBPL=uni-trier.de, ISBN 978-989-8425-47-8, Vilamoura, Portugal 5-7, March 2011. CONFERENCE

The Spanish project VISION develops new models for immersed and virtual communication. Augmented environments for the integration of real and virtual elements required the reconstruction virtual models of real persons. One of the main elements of attention and recognition in the communication process is the face. This work present a hybrid method for reconstruction of model of persons and special emphasis was made for the correct reconstruction of the face.

The research was made by a collaboration between Vicomtech research center and the laboratory of CAD/CAM/CAE of EAFIT University. This work has been partially supported by the Spanish Administration Agency CDTI under project CENIT-VISION 2007-1007, the Colombian Administrative Department of Science, Technology, and Innovation (COLCIENCIAS)

As co-authors of such publication, we give our permission for this material to appear in this document. We are ready to provide any additional information on the subject, as needed.

Prof. Dr. Eng. Oscar E. Ruiz
oruiz@eafit.edu.co
Coordinator CAD CAM CAE Laboratory
EAFIT University, Medellin, COLOMBIA

M.Sc. Iñigo Barandiaran
ibarandiaran@vicomtech.org
eHealth & Biomedical Applications
Vicomtech, San Sebastian, SPAIN

Eng. Javier Barandiaran
jbarandiaran@vicomtech.org

Vicomtech, San Sebastian, SPAIN

Dr. Marcos Nieto
mnieto@vicomtech.org

Vicomtech, San Sebastian, SPAIN

Abstract

This article presents a methodology for reconstruction of 3D faces which is based on stereoscopic images of the scene using active and passive surface reconstruction. A sequence of Gray patterns is generated, which are projected onto the scene and their projection recorded by a pair of stereo cameras. The images are rectified to make coincident their epipolar planes and so to generate a stereo map of the scene. An algorithm for stereo matching is applied, whose result is a bijective mapping between subsets of the pixels of the images. A particular connected subset of the images (e.g. the face) is selected by a segmentation algorithm. The stereo mapping is applied to such a subset and enables the triangulation of the two image readings therefore rendering the (x, y, z) points of the face, which in turn allow the reconstruction of the triangular mesh of the face. Since the surface might have holes, bilateral filters are applied to have the holes filled. The algorithms are tested in real conditions and we evaluate their performance with virtual datasets. Our results show a good reconstruction of the faces and an improvement of the results of passive systems.
Keywords: 3D reconstruction, structured light, Gray codes, depth-map.

1. Introduction

1.1. Mathematical Context. In general, surface reconstruction from optical samples requires a function G relating pixels in an image of the scene $A \times B$ ($A, B \subset \mathbb{N}$) with points $p \in \mathbb{R}^3$. This function, $G : A \times B \to \mathbb{R}^3$, is an injection since the image only records the visible part of the scene. G is not an onto function, as there are many points $p \in R^3$ for which there is no pixel $(i, j) \in A \times B$ in the image that records them.

Once this geometry function G is known, it is relatively simple to build a triangular mesh of the portion of the object visible in the image. Under a threshold of geometrical proximity, $G(i, j), G(i + i, j), G(i + 1, j+1)$ may be considered the vertices of a triangular facet of the sought surface M. Moreover, the triangles being natural neighbours to triangle $t = [G(i, j), G(i + i, j), G(i + 1, j + 1)]$ are the ones involving pixels $(i, j + 1), (i + 2, j + 1), (i, j - 1)$, again, under thresholds of geometrical proximity. Stitching the different M triangular meshes originated in different views of the scene is known as zippering, and is not in the scope of our article. Literature on the topic might be found in [**16**], [**17**] and [**18**].

The discussion in this article involves two images, which may be labelled, without losing generality, as *right* and *left*, I_R and I_L. Simplifying the discussion, a color image is a mapping $I : A \times B \to [0, 255]^3$. For example, $I(i, j) = (143, 23, 112)$ means that the color registered in the pixel (i, j) of I corresponds to Red=143, Green=23 and Blue=112. A grey scale image has the form $I(i, j) = (k, k, k)$ due to the fact that in it the Red, Green and Blue graduations are identical ($k \in [0, 255]$).

Let S_L and S_R be the coordinate systems associated to images Left and Right, respectively. In the general configuration of the set-up, S_L and S_R such that (1) the Z axis of the coordinate system is normal to the capture plane of the image, and (2) the two cameras point to a common point $p \in \mathbb{R}^3$. In this article we assume that the images are *rectified*. This means, both of them have been rotated inside their own $X - Y$ plane (i.e. rotation around the Z axis of the image) in such a manner that the epipolar plane of the set-up is seen as the plane $y = E_e$ in both images. That means, the epipolar plane is seen as the same horizontal line in both images. We call the rectified images I_R and I_L and their rectified and their rectified coordinate systems S_L and S_R, respectively.

Let us consider a point $p \in \mathbb{R}^3$ recorded in both images I_R and I_L. Because the previous assumptions we have that $G_L(i, j) = p$ and $G_R(i, k) = p$. This means, the point p appears *in the same row* i of pixels in both images. The value $|k - j|$ is an offset that only occurs in the same pixel row of both images. Since we know that pixels (i, j) in image I_L and $(i, k) = p$ in image I_R record the same point $p \in \mathbb{R}^3$, the point p can be recovered by a usual triangulation procedure.

1.2. Informal Context. Human face reconstruction is a common problem in computer vision and computer graphics [**19**], where one possible objective is the generation and animation of a virtual model of it.

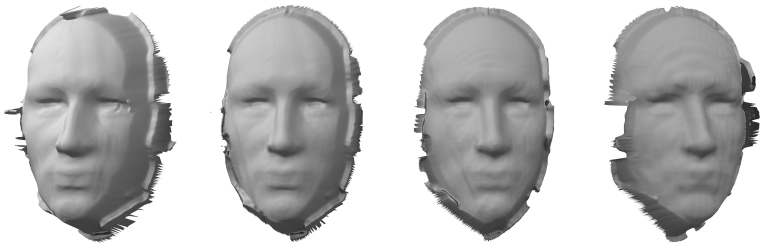


FIGURE 2.1. Results of the algorithm with the virtual dataset. Smooth surfaces are obtained with wider baselines.

The face is one of the most important identification regions of the human body, presenting commensurate technical challenges [**20**]. A correct reconstruction of human faces is a precondition to both augmented reality and face recognition.

3D surface reconstruction may be achieved by both passive and active methods. passive ones do not change the environment in the process of reconstruction. Even thought passive methods obtain very good results, their setups are very expensive because they required a very high resolution required for obtaining reasonable results[**21**].

Active systems modify or adapt the environment during the capture process. Our active system uses the projection of a light pattern (i.e *structured light*), which is widely used for face surface reconstruction. In structured light systems any change on the setup requires new algorithms for face (surface) reconstruction.

The 3D surface reconstruction system implemented and informed in this article is part of a system used for full body reconstruction with visual hull algorithm [**22**]. Our setup applied to a face-body model produces a triangular mesh with high detail in the face region and low detail in the rest of the body. The reason for this differential resolution is that, while for the face region one requires high frequency details (e.g. texture of the skin), for the rest of the body such details are not required in our applications.

This article presents a system for face reconstruction which articulates non-proprietary hardware and our own software to obtain geometrical information from two images (possibly originated in 3D video - conference set ups). Our system also recovers the 3D geometry form the body region, although intentionally using lower resolution for neighbourhoods other than the face.

This paper, Section 2 reviews previous works in face reconstruction. Section 3 presents the methodology implemented, including generation of the light patterns, capture, segmentation and reconstruction. Section 4 discusses the hardware set-up for the experiment and its configuration. Section 5 presents the results of the 3D surface reconstruction set-up and algorithms, and evaluates the reconstructed models against with real data. Section 6 concludes the work and proposes the future actions in this domain.

2. Related work

Face reconstruction is a widely studied topic. [**23**] presents a tutorial on face reconstruction, describing different problems and approaches from an artistic point of view, looking for a correct representation of the face and its expressions in multi-media. [**19**] presents a survey of 3D face reconstruction methods, classifying them in three different categories: *single image*, *stereo images* and *videos*.

passive systems are commonly used for face reconstruction. One of the advantages of these systems is their non interaction with the environment, allowing to capture the geometry without interfering with other systems. [**24**] uses a system with four calibrated cameras applying a multi - view algorithm.

A stochastic model is generated for the identification of the geometry, by minimizing a cost function. [**25**] compares different stereo algorithms for face reconstruction, and proposes an correct geometrical configuration of cameras to obtain accurate results. [**26**] presents a complex setup to a high resolution face reconstruction system. The methodology is based on an iterative reconstruction of the face by incrementing

the size of the image and the number of stereo pairs used in each step. [**21**] extends the approach proposed in [**26**] by adding a post-processing step that modifies the geometry of the face by using a texture, assuming that small dark regions of the face represent small hollows. This approach obtaining a more rich geometry.

Structured light for 3D reconstruction have been study for several years. The information obtained with this kind of systems is already being used as ground truth data for the evaluation of passive reconstruction systems, such as stereo algorithms [**27**]. An extensive survey of structured light 3D reconstruction approaches can be found in [**28**], where a classification of different coding techniques are presented and evaluated. They identify the best coding approach for each one of the possible scenario configurations such as static o moving scenarios or if the light conditions are controlled.

Real time capture of facial expressions is also an important feature in some systems. Several problems have to be addressed to accomplish this objective. One of these problems is the difficulty of projecting several patterns for a reconstruction in non static scenes where small moves generate artefacts in the reconstructions, so other patterns have been employed which uses color models as [**29**] which allows a denser codification of the pattern, also single frame reconstruction with 2D coding is possible[**30**]. Another problem is hardware calibration to obtain several frames per second with a correct synchronization process between the projector and the cameras. An accepted synchronization approach can be found in [**31**]. Finally, for a correct pattern codification of time variant patterns, a motion compensation should be implemented. This issue is especially critical for face reconstruction systems, where the person being reconstructed could move in a involuntary way,during acquisition [**32**]

3. Methodology

Our algorithm of face reconstruction uses a set of stereo images captured at the same moment when a pattern is projected into the face. The images are captured in a setup previously calibrated. We assume that the object does not move between the different captures and the face is assumed to be a smooth surface without hair or beard and without highlight reflections. The result is a mesh of triangles correctly located in the space which represent the face region.

3.1. Stereo Calibration. Stereo calibration refers to the task of finding the relative pose between the cameras of a stereo pair. The objective is to feed subsequent stereo rectification processes that align the images such that the epipolar lines are horizontal and thus matching algorithms for 3D reconstruction can be implemented as one-dimensional searches.

Typically, stereo calibration is carried out by means of finding a number of point-correspondences between the images of the pair and retrieving the fundamental matrix. Let $\mathbf{x}$ be the image of a 3D point in the left image, and $\mathbf{x}'$ the image of the same point in the right image. The fundamental matrix restricts the position of $\mathbf{x}'$ to the epipolar line associated to $\mathbf{x}$, such that $\mathbf{x}'^{\top} F \mathbf{x} = 0$. It has been shown [**33**], that the knowledge of the fundamental matrix can be used to retrieve the projection matrices of the two cameras of the pair, up to a projective ambiguity that can be solved with known restrictions of the camera.

Besides, images captured by real cameras show some tangential and radial distortion, which can be corrected applying the following functions:

$$u = p_x + (u - px)(1 + k_1 r + k_2 r^2 + k_3 r^3 + \ldots)$$
$$v = p_y + (v - py)(1 + k_1 r + k_2 r^2 + k_3 r^3 + \ldots)$$

where $r^2 = (u - p_x)^2 + (v - p_y)^2$ and $k_1, k_2, k_3, \ldots$ are the coefficients of the Taylor expansion of an arbitrary radial displacement function $L(r)$.

Parameter identification of the camera stereo pair is extracted from the calibration information of the full body reconstruction setup; which is further explained in [**34**]. For our purposes we select the camera pair which are focused to the face region of the body, and we follow a 3D stereo reconstruction process with them.

Input: bin
Output: $gray$
return $bin^{(bin/2)}$

Algorithm 1: Gray function to convert from binary to gray code

Input: $gray$
Output: $bin, nPat$
$ish, ans, idiv \in \mathbb{N}$
$ish \leftarrow 1$
$ans \leftarrow gray$
while 1 **do**
  $idiv \leftarrow \frac{ans}{ish}$
  $ans \leftarrow ans \oplus idiv$
  if $idiv \leq 1 \vee ish = 32$ **then**
    return ans
  end
  $ish \leftarrow ish \times 2$
end

Algorithm 2: Gray function to convert from Gray code to binary

3.2. Pattern Generation. Pattern generation refers to the task of creating of a set of synthetic binary images to be projected as structured light in the scene. The objective is to identify the coordinates of the projected pattern in the image scene and thus allowing a point matching algorithm to become independent of the color in the captured scene.

The used patterns are represented as a matrix of boolean values. Let P be a matrix of M columns and N rows thus $P = \{P_{m,n} \in \{0,1\}\}$ with $0 < m < M$ and $0 < n < N$. Let C be a matrix of the same dimensions of P thus $C = \{C_{m,n} \in (0, M) \subseteq \mathbb{N}\}$. The restriction of the number of values in the matrix C is the same that the number of columns allows the correct identification of the column in the images. Let g be a function such as $g : \mathbb{N} \to \mathbb{N}$ which is bijective and transforms the numbers from binary representation to Gray representation as described in algorithm 1, the inverse Gray function g^{-1} is described in algorithm 2. The number of images to be projected depends of the number of columns of the matrix C, so $nPat = \lceil \log_2 M \rceil$

The $nPat$ patterns represented by the matrix P are generated as follows:

$$P^i_{j,k} = g(j) \bullet 2^i$$

where $0 < i < nPat$ represent the number of the pattern, j, k the coordinates in the matrix P. The pattern structure can be depicted as a sequence of columns as can be visualized in the figure 2.2. The nature of this kind of patterns is 1D because the calibration setup already give us an epipolar constrain of the images. Therefore it is not necessary to use of 2D patterns in this case.

3.3. Pattern Recognition. Pattern recognition refers to the task of creating a pair of images which maps the position of the projected patterns P in the set of stereo pair of images. The objective is the identification of the projected pattern in the set of images, and calculate the value of the matrix C for each

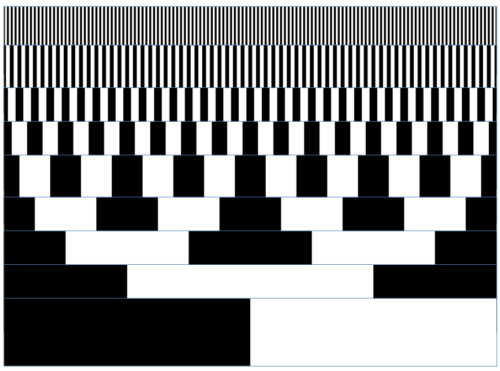


FIGURE 2.2. Gray code

pixel. This matrix allow the point matching algorithm to become unambiguous since each point in the maps is labelled uniquely in each epipolar line.

Let $s = L, R$ be matrices of W columns and H rows, thus $L = \{L_{w,h} \in (0, 255)\}$ with $0 < w < W$ and $0 < h < H$. The matrices L and R represent the information of the stereo pair cameras in grayscale. Let $O = \{O_{w,h} \in (0, M)\}$ be the decode maps OL, OR. Let $t : (0, 255) \to \{0, 1\}$ be a threshold function which binarizes the images L and R. The threshold value could be calculated with the Otsu algorithm as explained in [**35**] or by means of calculating the albedo of the images with the process described in [**27**] where each pattern is projected two times, each one with the original version and their negative.

$$O^s_{m,n} = g^{-1}\left(\bigvee_i t\left(s^i_{m,n}\right) \cdot 2^i\right)$$

As shown in figure 2.3 the set of L and R images are binarized. Stereo reconstruction images are combined, and a unique map OL and OR is processed. The maps should be rectified as shown in figure 2.4: this rectification process is possible because the camera information is already known as explained in section 3.1.

3.4. Stereo Reconstruction. Stereo reconstruction task calculates the point correspondence between two images. The objective is to calculate the 3D point coordinates of each pixel in the stereo images. The previous steps of the algorithm such as the stereo calibration assures the epipolar constrain. Pattern generation process gives the color independence of the images. Pattern recognition maps the set of images into a map without ambiguities. For dense depth-map calculation we used a simple box filter algorithm, based on sum of square differences *SSD* and a Winner Takes All *WTA* for pixel selection [**36**].

Let D be a matrix of W columns and H rows, thus $D = \{D_{w,h} \in \mathbb{N}\}$. where D describes the pixel difference between the same point in the image L and R. The identification of the correct disparity the following function is applied.

$$D_{m,n} = \textit{minarg}_l\left(\textit{SSD}(OL, OR, m, n, l)\right)$$

$$SSD(L, R, m, n, l) = \sum_{e=m-b}^{m+b} \sum_{f=n-b}^{n+b} \left(L_{e,f} - R_{e,f+l}\right)^2$$

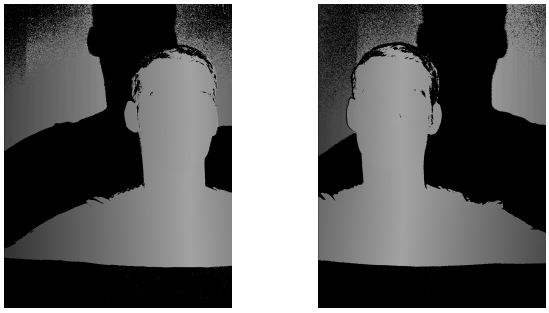
(a) Left camera capture

(b) Right camera capture

(c) Left binarization

(d) Right binarization

FIGURE 2.3. Stereo images captured from the cameras and their result of the threshold function

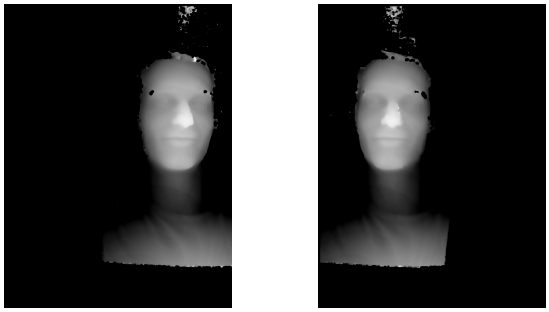


(a) Left Gray decoded

(b) Right Gray decode

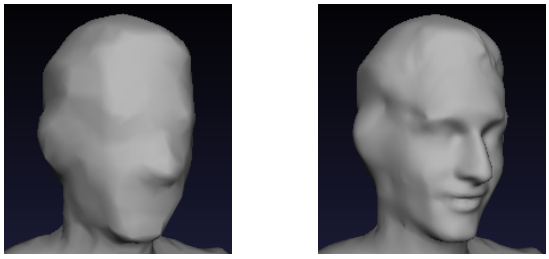
FIGURE 2.4. O maps of the images, the Gray value in each pixel represents the value of the C matrix in the stereo images

The figure 2.5 shows the result of the disparity maps DL and DR. The identification of wrong matched points is carried out by applying process such as *cross checking* and *joint bilateral filter*. It is assumed that the biggest connected region represents the face, then a max blob algorithm is applied to filter regions outside the face. The mesh is generated by joining adjacent pixels in the image with their 3D coordinates. The topology of the mesh is correct since the difference between the coordinates of adjacent pixels are small.



(a) Left disparity (b) Right disparity

FIGURE 2.5. Stereo reconstruction of the face, the disparity images for the L and R



(a) Visual Hull (b) Structured Light

FIGURE 2.6. 3D face reconstruction

4. Setup configuration

As our algorithm is part of a bigger chain of process where a full body reconstruction process is done. We tried to maintain the same setup for our algorithm, even we tried to use a passive system for face reconstruction, the resolution obtained were insufficient to fulfil our needs. Then, we put a stereo camera setup with a projector in the middle of the cameras. We identified that a small distance between the cameras does not give enough information for recovering 3D positions accurately. In opposite, a wide baseline between cameras generates occlusion regions, although projection information is used for hole filling in post processing step.

The cameras used where tested with different resolutions, 1024x768, 1240x960 and 1600x1200. Finally, the resolution was set to 1240x960. Also, the projector were defined at a resolution of 800x600 because an increase of resolution generates very high frequency patterns, that are very difficult to identify accurately at that resolution. We found that a minimum width of 4 pixels for each column of the pattern is necessary for a correct identification.

Different kind of binary patterns were used. Gray pattern generates the best results. Using binary or golay patterns shows in some aspects impossible to generate a workable results. In this way, we didn't consider these methods for the final version, and used only Gray codes. The binarization of the images present one of the biggest problems of the structured light setup. We used the Otsu method but it exhibits some problems, such as high sensitivity to areas of specular reflection. We finally choose the projection of the negative images with good results and a threshold t with a value of the half of the range of the gray image.

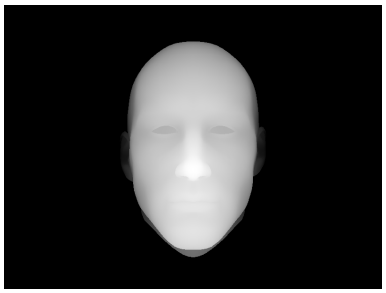


FIGURE 2.7. Ground truth depth-map image of a face

FIGURE 2.8. Result from the different cameras in the virtual setup

5. Results

The evaluation of the algorithm presents several problems since the ground-truth information it is not available. However, we implemented a virtual environment which resembles a real setup. This approach allowed us to test our method and validate our results. We use Blender software for the generation of the setup and the identification of the ground-truth data, as shown in figure 2.8. The ground-truth was defined as a normalized depth-map with values between 0 and 255 using all the possible values in the image format as shown in the figure 2.7.

Different camera positions were tested in our virtual setup, for the identification of the best baseline distance. The results obtained with the algorithm are shown in the figure 2.8. and the position of the cameras are shown in the figure 2.9.

The ground-truth information and the results of the algorithm show a difference in scale, but not in position. We measured the difference of the results and the ground-truth with a image correlation algorithm. The correlation gives us a value between the range of 0 and 1 where 0 is a bad results and 1 is the ground-truth. The table 2.1 presents the correlation values obtained for different camera position, and the figure 2.1 shows the 3D mesh generated.

6. Conclusions and future work

We present a methodology for face reconstruction in a mixed environment of active-passive setup. Structured light shows a quality improvement against the results obtained with passive setups. Time multiplexing codification has the problem of motion between the captured images generating a waving effect in the reconstructions. Even robust algorithms of point matching for dense depth-maps were tested there were no

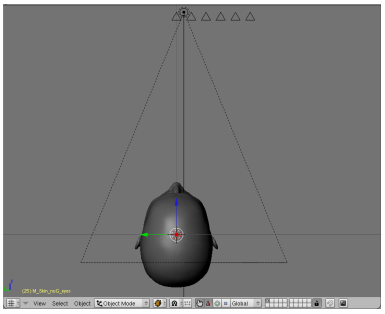


FIGURE 2.9. Camera positions for virtual framework

Baseline distance	Value
1	0.904372
2	0.938449
3	0.958089
4	0.974051

TABLE 2.1. Correlation Results

real improvement in the results. We will try with color or 2D patterns which only requires one exposition that present a better approach for the reconstruction of faces since the motion problem is not present.

CHAPTER 3

Real-Time depth map generation architecture for 3D videoconferencing

- John Congote[1,2]
- Iñigo Barandiaran[2]
- Javier Barandiaran[2]
- Tomas Montserrat[3]
- Julien Quelen[3]
- Christian Ferran[3]
- Pere J. Mindan[4]
- Olga Mur[4]
- Francesc Tarres[4]
- Oscar Ruiz[1]

[1] CAD CAM CAE laboratory, Universidad EAFIT
Carrera 49 No 7 Sur - 50, Medellín, Colombia
[2] Vicomtech Research Center
Donostia - San Sebastián, Spain
[3] Telefonica Research
Barcelona, Spain
[4] UPC GTAV Group
Barcelona, Spain

Context

Real-time depth map generation architecture for 3D videoconferencing. John Congote, Iñigo Barandiaran, Javier Barandiaran, Tomas Montserrat, Julien Quelen, Christian Feran, Pere Mindan, Olga Mur, Francesc Tarres, Oscar Ruiz. 3DTV-CON2010 Conference on Capture, Transmission and Display of 3D Video. June 7-9 2010, pp. 1-4. Tampere, Finland. ISBN: 978-1-4244-6377-0 (Books), ISBN: 978-1-4244-6378-7 (CD). CONFERENCE

The Spanish project VISION develops new models for immersed and virtual communication. A final result of the project was the generation of an architecture for the implementation of a 3D video-conference system.

This work was made by a collaboration between the Telefonica I+D, Vicomtech Research Center, GTAV group of UPC University and the laboratory of CAD/CAM/CAE of EAFIT Univerty. It has been partially supported by the Spanish Administration agency CDTI, under project CENIT-VISION 2007-1007. and COLCIENCIAS with GENERACIN DEL BICENTENARIO program.

As co-authors of such publication, we give our permission for this material to appear in this document. We are ready to provide any additional information on the subject, as needed.

Prof. Dr. Eng. Oscar E. Ruiz
oruiz@eafit.edu.co
Coordinator CAD CAM CAE Laboratory
EAFIT University, Medellin, COLOMBIA

M.Sc. Iñigo Barandiaran
ibarandiaran@vicomtech.org
eHealth & Biomedical Applications
Vicomtech, San Sebastian, SPAIN

Javier Barandiaran
jbarandiaran@vicomtech.org
Vicomtech Research Center
Vicomtech, San Sebastian, SPAIN

Tomas Montserrat

Telefonica Research
Barcelona, SPAIN

Julien Quelen

Telefonica Research
Barcelona, SPAIN

Christian Ferran

Telefonica Research
Barcelona, SPAIN

Pere J. Mindan

UPC GTAV Group
Barcelona, SPAIN

Olga Mur

UPC GTAV Group
Barcelona, SPAIN

Francesc Tarres

UPC GTAV Group
Barcelona, SPAIN

Abstract

In this paper we present a reliable depth estimation system which works in real-time with commodity hardware. The system is specially intended for 3D visualization using autostereoscopic displays. The core of this work is an implementation of a modified version of the adaptive support-weight algorithm that includes highly optimized algorithms for GPU, allowing accurate and stable depth map generation. Our approach overcomes typical problems of live depth estimation systems such as depth noise and flickering. Proposed approach is integrated within the versatile GStreamer multimedia software platform. Accurate depth map estimation together with real-time performance make proposed approach suitable for 3D videoconferencing.
Keywords: depth map, stereo vision, 3D videoconferencing, autostereoscopic displays, stereo matching

1. Introduction

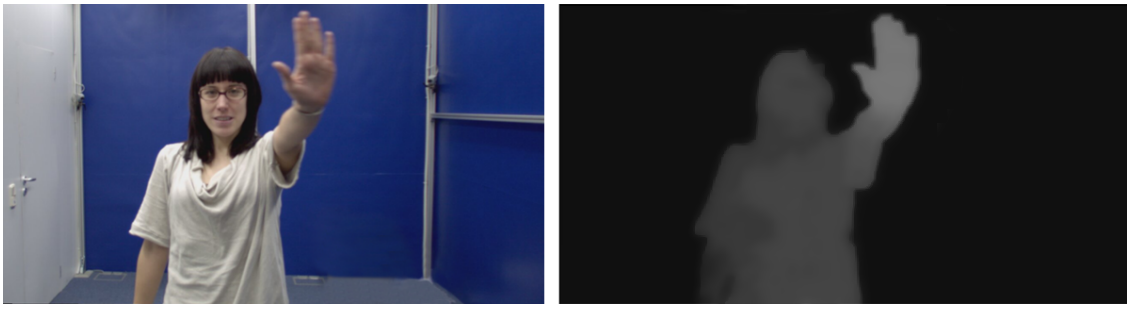


FIGURE 3.1. Image-plus-depth input format

During the last years, videoconferencing has become a widely extended application. Companies and individuals use this technology to reduce transportation costs and to improve communication between distant parties. However, traditional 2D videoconferencing still fails to produce natural impressions to the remote conferees. 3D videoconferencing is a logical evolution: a better feeling of presence is provided to conferees by leading them to believe that they are closer to each others. Currently, the emerging 3D displays and the increase of mainstream hardware computation capabilities make telepresence feasible.

Our contribution is a 3D videoconferencing approach which implements a visually accurate stereo algorithm running over GPU. The outline of the paper is as follows: In section 2, a short review of different approaches and techniques related with 3D videoconferencing is presented. Methods and algorithms implemented in our approach are shown in 3. Then, an architecture where proposed approach is validated is described in section 4. Finally, we detail some conclusions and suggestions for future work in section 5.

2. Related Work

Many important topics related to videoconferencing must be addressed to enhance user experience. One of such issue is eye-contact or gaze. Research [**37, 38**] has shown that gaze is one of the most important non-verbal cues, responsible for giving feedback or expressing feelings and attitudes. Another problem related with videoconference is the lack of perception of depth wrt the analyzed scenario. In a 2D videoconference, traditional displays are used. The absence of depth information in this type of displays, in addition to the problem of gaze results in an unnatural experience.

An accurate and efficient depth map estimation mechanism is needed for both problems to be addressed. For example, depth information is used by autostereoscopic displays to render multiple novel views. These generated views can be used to tackle the problem of eye-contact, by generating views that agreed with the

user eye-sight, as in [**39**]. Moreover, autoestereoscopic displays typically use depth maps information to generate 3D perception, [**40**] (See Figure 3.2).

In [**36**] Scharstein presents a set of stereo matching methodologies used to calculate dense disparity maps from stereo images, which can be classified in local, global and semi-global algorithms. The depth calculation process is done by identifying correspondences between pixels in two images. Given the location of corresponding pixels in the images, their 3D coordinates can be retrieved by means of triangulation. Applications like 3D Videoconferencing imposes strong time restrictions, thus only real-time methods are suitable.

FIGURE 3.2. 2D+Depth image result of our system.

3. Methodology

In this section we describe in detail the methods implemented in our architecture, which allow us to achieve accurate depth information in real-time.

We identify in the process of 3D videoconference 3 different process which are important to obtain an effective 3d conference experience. Image rectification step is used to correct and setup the image inputs for further processing, the correct and fast implementation of this step allows an smooth test base for the testing of algorithms of depth map calculation. Next depth map calculation as previously mentioned is a widely studied topic, but a carefully selection of a good algorithm which could be implemented in real time which good quality results is necessary. We use the adaptive weight algorithm as starting point for our implementation. Finally the post processing steps are very important to obtain real time and good quality images, because this process could be implemented in a fast way and could save time of processing of the previous step.

3.1. Image Rectification. Depth map calculations rely on the estimation of relative pixel positions between two images. The position of the cameras and lens distortions make necessary some previous corrections. These corrections are carried out before depth map calculation takes place in order to improve the process efficiency. Moreover, it is also necessary to align the horizontal lines of both images in order to reduce the search process of the pixels to a 1D problem. The tools needed to carry out this transformation are the camera calibration parameters, which should have been calculated previously. Results of this process can be observed in Figure 3.3.

Although camera calibration can be carried out just once and off-line, rectification should be done online and for each pair of acquired images. As we work with large images, we need a fast and effective algorithm. The solution adopted in our approach is the use of GPU algorithms implemented in CUDA. Our

FIGURE 3.3. Top: Left and right captured images. Bottom: Undistorted and rectified images, yellow frame denotes cropping area.

implementation processes the pixels in parallel, taking advantage of the multi-core architecture of the GPU. Execution of our algorithms on the GPU outperforms General Purpose Processors (GPP) based solutions and obtains real-time performance at a reasonable cost. Rectified images have a resolution of 960×540 pixels as required by our 3D display. After undistortion and rectification processes, some points of the image may lay outside the working frame, producing some empty parts (see Figure 3.3). In order to eliminate these unwanted regions, input images are captured at higher resolution (1200×600) and cropped after rectification process takes place as seen in Figure 3.3.

3.2. Adaptive support-weight. Adaptive support-weight *AW* is a local algorithm to calculate dense stereo disparity maps, presented by Yoon[**41**]. The algorithm is a window based method for correspondence search using varying support weights. The support weights of the pixels in a given support window are based on a combination of color similarity and geometric proximity to reduce ambiguity. *AW* gives a cost to a pair of pixels depending on their support windows, representing how related the pixels are. Afterwards, each pixel is correlated to the pixel that has the associated minimal cost in the matching set.

A dense disparity map can be represented as an image D of the same size of the input rectified images L, R. Where $D_{i,j}$ represents the displacement of the coordinates of a given pixel from L to R, consequently $L_{[i+D_{[i,j]},j]} \approx R_{[i,j]}$. Image pixels p, q are commonly represented as an *RGB* 3-tuple then p_r is the *Red* component of the pixel p as p_g and p_b the *Green* and *Blue* components, hence the color distance between two pixel is defined as (eq: 7, 8) following the L_1 and L_2 distance norms.

$$f_c(p,q) = |p_r - q_r| + |p_g - q_g| + |p_b - q_b| \tag{7}$$

$$g_c(p,q) = \sqrt{(p_r - q_r)^2 + (p_g - q_g)^2 + (p_b - q_b)^2} \tag{8}$$

Equation 9 shows the m image (left or right) weight contribution of the pixel in the position (k,l) in a window centered at coordinates (i,j). The γ_c and γ_s values are empirical factors that modify the color and space similarity respectively. In our implementation, we set these factors to $\gamma_c = 20$ and $\gamma_s = 1$. The γ_c

factor depends on how close the values in the *RGB* color space are, while γ_s modifies the space similarity, which is closely related to the support window size.

$$f_w(m,i,j,k,l) = e^{\left(-\left(\frac{g_c\left(m_{i,j},m_{k,l}\right)}{\gamma_c}+\frac{\sqrt{(i-k)2+(j-l)2}}{\gamma_s}\right)\right)} \tag{9}$$

Calculation of the disparity map D for each pixel $D_{i,j}$ in the reference image L is made by the *Winner Takes All* method as show in (eq: 12), where d represents the range of values where the correlation of pixels will be tested. This value depends on the camera parameters and the depth search range defined, furthermore the parameter w presented in (eq: 11) indicates the size of the square support window for pixel correlation. Finally SW (eq: 10) is a component used in (eq: 11) representing Support Weights used to influence the difference of the color and position in the window (eq:7). Notice that the cost support weight for a given pixel in L depends on the matching candidate, so normalization is required in (eq: 11).

$$SW := f_w(L,i,j,s,t) f_w(R,i+d,j,s+d,t) \tag{10}$$

$$f_{aw}(L,R,i,j,d) = \frac{\sum_{s=i-w}^{i+w}\sum_{t=j-w}^{j+w} SW\, f_c(L_{s,t},R_{s+d,t})}{\sum_{s=i-w}^{i+w}\sum_{t=j-w}^{j+w} SW} \tag{11}$$

$$f_d(L,R,i,j) = \arg\min_{d\in\mathbb{Z}} f_{aw}(L,R,i,j,d) \tag{12}$$

Although *AW* is a local stereo matching algorithm, real-time performance is difficult to achieve with CPU implementations. However, the local nature of the algorithm allows an easy implementation in parallel architectures such as GPUs, achieving real time performance. We use a support window of 1×15 in size and only for the reference image, a maximum disparity of 30 pixels and an image size of 480×270 pixels. With this specifications we can calculate the depth map for both images in 8ms. Note that the depthmap is processed at a quarter of the resolution of rectified images. Post-processed depth map is upscaled afterwards to match the reference image resolution. Mentioned modifications to the original AW algorithm caused a loss of quality in the resulting depth map. However, our proposed post-processing technique is able to correct most of the wrong matches introduced by the algorithm simplification while still keeping real-time performance. Cross-checking is a common process for identifying unreliable pixels in the calculated depth maps. Matched pixels in left and right disparity maps are tested for consistency, where inconsistent matches are discarded. These pixels are later approximated by an anisotropic diffusion post-processing step. Even though regions with no texture cannot be correctly calculated, it is assumed that these regions are part of the background which is removed from our depth map. This background segmentation process is obtained using the statistical method described in [**42**], also implemented on GPU. The integration of a segmentation process allows us to obtain a more accurate depth map even on untextured regions, avoiding uncomfortable flickering effects during the conference.

3.3. Post-Processing. As mentioned before, cross-checking is used in our system to detect wrong correspondences by building a mask with them. This mask also includes information regarding foreground segmentation. The main objective of post-processing is to assign correct disparity values to all unreliable pixels detected in the previous step. Moreover, the disparity map is improved by removing the quantization effect introduced by the lack of sub-pixel accuracy during disparity estimation.

Mismatches basically occur in occluded regions as well as in homogeneous texture areas. A common hole-filling criterion is that occlusions must not be interpolated from the occluder, but only from the occluded

to avoid incorrect smoothing of discontinuities. Thus, an extrapolation of the background into occluded regions is necessary. In contrast, holes generated due to mismatches can be smoothly interpolated from all neighbouring pixels. If we assume that discontinuities in both disparity and color images are co-aligned, the previous interpolation criterion can be achieved by relying only on reference image color cues.

We use a variation of the classic Perona-Malik anisotropic filter [**43**] in order to propagate correct disparity values through the image. Anisotropic diffusion is a non-linear smoothing filter. It produces a Gaussian smoothed image, which is the solution to the heat equation, with a variable conductance term to limit smoothing at edges. For this particular implementation, the conductance term is a function of the reference image gradient magnitude at each pixel and for each color channel. As we can partially rely on foreground segmentation information, unreliable pixels are initialized to their known background disparity value. Another difference between standard Perona Malik discrete filter implementation and our approach is that in initial iterations we only modify values of unreliable pixels, avoiding them to influence on correctly assigned disparities. These initial iterations are intended to fill the disparity map holes. Subsequently, after convergence of discarded pixel values, we apply a reduced number of additional diffusion iterations modifying all pixels (see figure 3.4). This second diffusion step is intended to remove the quantization effect of the disparity map. This post-processing algorithm also allows an efficient and easy implementation in GPU architectures.

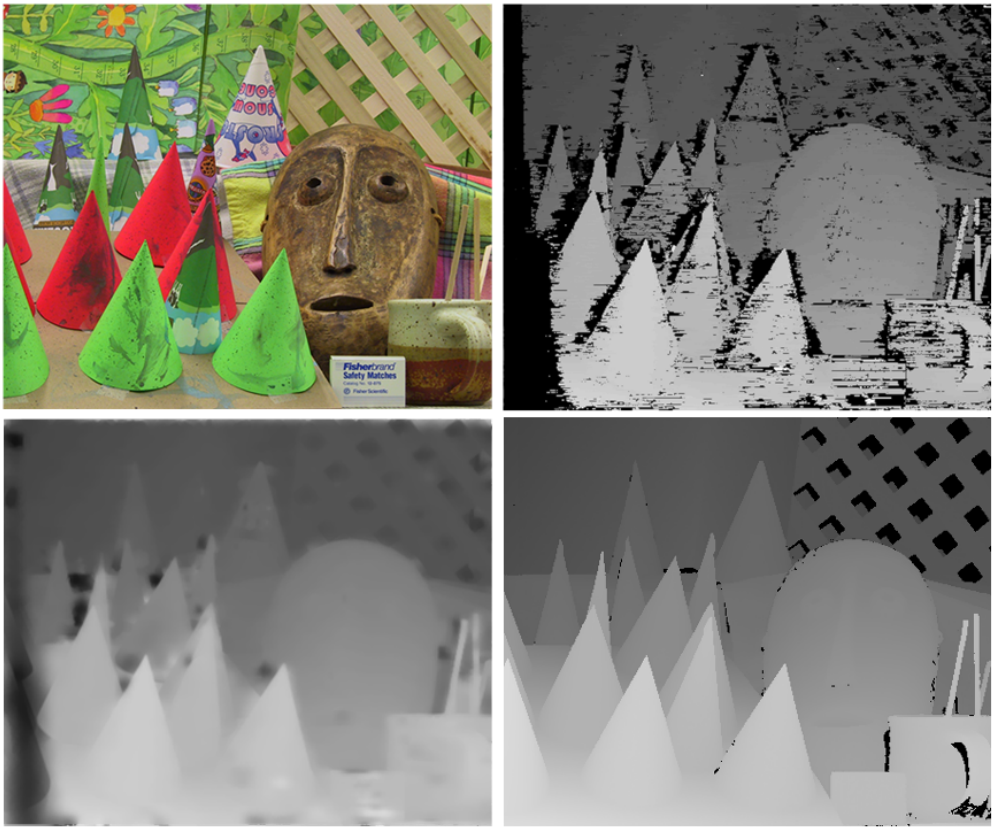


FIGURE 3.4. Post-Processing (from left to right and up to down): Reference Image. Initial Disparity Map (Black pixels are wrong pixels). Final Disparity Map. Ground truth

4. Results

Current 3D displays are capable of providing multiple stereoscopic views in order to support head motion parallax. Hence, the user is able to perceive depth when uses these types of displays. This 3D perception can reinforce the feeling of volume and therefore the feeling of "tele-presence". In our prototype, we are using a 42" Wow display designed by Philips. This display can generate several views of the same scene, by using a 3D data interface in the form of image-plus-depth with a resolution of 960×540. This input format was standardized by MPEG and is currently implemented in many commercial 3D displays. This format proposes the use of images that are double the size of the native resolution, where the reference frame is disposed in one side of this image, and the corresponding disparity or depth map on the other, as depicted in figure 3.2. In order to be able to estimate a disparity map, two or more images of the same scene are needed. In our prototype we use two Pixelink digital cameras for stereo image acquisition. These cameras capture images at a resolution of 1200×600 pixels and with 25 fps framerate. We use RAW Bayer native image format, and compute Bayer demosaicing and white balance, inside the GStreamer pipeline. By using RAW images the required bandwidth is reduced, optimizing memory usage and allowing better off-camera Bayer demosaicing.

4.1. Integration Pipeline. Our streaming architecture is based on GStreamer software[**44**]. This library is an open source framework for creating streaming media applications. The design of the framework is based on modularity, where any process in the streaming pipeline can be encapsulated in an independent module or plug-in. This modularity provides flexibility and allows the user to easily test different configurations and approaches. Moreover, threading and buffering architectures are controlled. Our architecture is divided in two separated parts. One of these parts works as the content generator or server, where the videoconference content, i.e, 3D images for visualization, is generated. The second part of the architecture represents the client side, where these images are received and displayed to the conferees. In the following section, we describe in more detail both parts of the proposed architecture.

4.1.1. *Server Side.* This part of the architecture runs as a content generator. It manages the digital cameras that capture the scene, the depth estimation and encoding modules. Cameras are externally triggered by using specific hardware, in order to have a perfectly synchronized image acquisition. This synchronization significantly increases the accuracy of disparity estimation. Figure 3.5 shows the complete processing pipeline integrated into the server side of the architecture.

The core of the pipeline is the plugin called *Depthestplugin*. This plugin receives two video streams from two synchronized threads, and executes image rectification and depth map estimation procedures. The output of the plugin is composed of a reference image and the corresponding disparity map. These streams are queued in two buffers before encoding them by using the h.263 video compression format. Finally, encoded video streams are rtp-packetized and sent through udp socket.

4.1.2. *Client Side.* This part of the architecture runs as a content receiver. It represents the end point of the entire pipeline, where the output device, i.e. the autostereoscopic display, is active. As shown in Figure 3.6, the first part of the processing pipeline consists of the network and video decoding of both streams.

Before the video content can be visualized on the display, it must be processed by the *SideBySide* image plug-in. This plug-in adapts the content by merging the reference image and disparity map with a resolution of 960×540 into one single 1080p video stream compliant with the format supported by the Philips Wow display, (See Figure 3.2).

The system is specially designed to show stereo images in two different displays. On the one hand, a stereoscopic monitor which only needs a pair of undistorted and rectified images. On the other hand, an

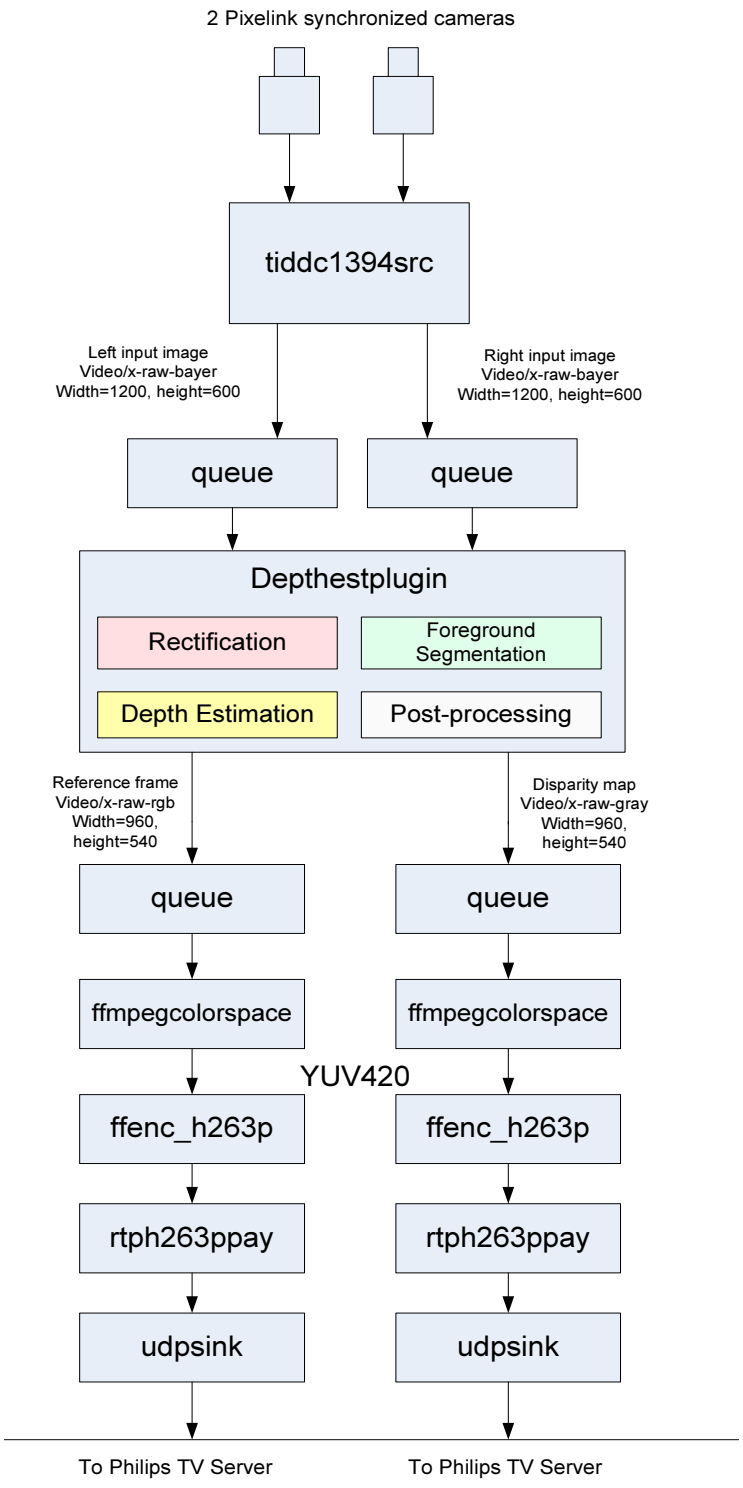


FIGURE 3.5. GStreamer 3D videoconference Server Side

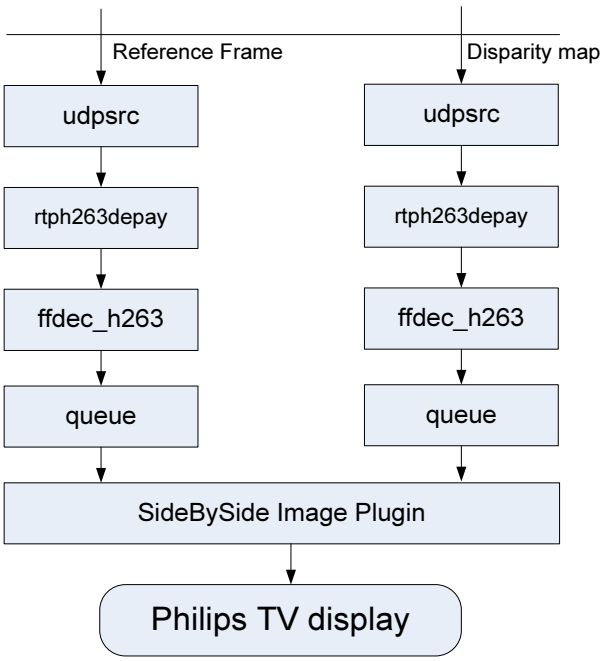


FIGURE 3.6. GStreamer 3D videoconference Client Side

autostereoscopic monitor which needs a depth map and a texture image corresponding to this depth map. The approach specified in this paper provide advantages for both solutions.

For the first case, the stereoscopic display, the system takes benefit from the undistort and rectifying CUDA implementation, increasing the capability of the system and also providing higher frame-rates. By increasing the frame-rate, the 'tele-presence' experience of the viewer is enhanced as a constant video flow is displayed in the monitors.

In the second case, the autostereoscopic display, apart from the advantages of the high performance of the rectification and undistort GPU based algorithms, it also increases the viewer experience as the depth maps obtained have higher quality. This gain of quality is possible thanks to the powerful algorithms ran over GPU as the use of this kind of algorithms with a standard CPU is not an option for real-time depth estimation. As in the previous case, the increase of presence sense and image quality really increases with the use of the described method.

5. Conclusions and Future Work

We have proposed an approach that achieves real-time performance in a 3D videoconferencing application using autosterescopic displays. We have implemented and optimized several algorithms on GPU trying to find the best compromise between quality of visualization and computational cost. Our approach runs at 25 fps at SD image resolution on an usual PC, with commercial hardware (dual core Intel processor with a NVIDIA GTX 295 GPU), using a multimedia architecture based on GStreamer. We achieve this performance by implementing an optimization of the Adaptive support-weight algorithm. This optimization is able to calculate disparity maps accurately in a short time, with low computational consumption. This efficiency allows us to add some other optimizations such as a background subtraction, cross-check consistency testing or error correction mechanisms such as gap-filling with Anisotropic diffusion. All of these

developments have been successfully implemented in GPU using CUDA in order to obtain the highest levels of parallelization and performance possible.

We are currently integrating a Time of Flight (ToF) camera into our architecture. In the future we plan to use this type of camera in combination with digital camera pairs. This combination will result in an hybrid approach that could overcome the problems of each technology when working independently. With this approach we hope to obtain better disparity maps at higher resolutions.

CHAPTER 4

Evaluation of interest point detectors for image information extraction

- Iñigo Barandiaran[2]
- Jon Goenetxea[1]
- John Congote[1,2]
- Manuel Graña[3]
- Oscar Ruiz[2]

[1] CAD CAM CAE laboratory, EAFIT University
Carrera 49 No 7 Sur - 50, Medellín, Colombia
[2] Vicomtech Research Center
Donostia - San Sebastián, Spain
[3]Computer Science and Artificial Intelligence Department, University of the Basque Country Donostia - San Sebastián, Spain

Context

Evaluation of interest point detectors for image information extraction. Iñigo Barandiaran, John Congote, Jon Goenetxea, Oscar Ruiz. Pages 2170 - 2179. DOI 10.3233/978-1-61499-105-2-2170. Frontiers in Artificial Intelligence and Applications, Volume 243: Advances in Knowledge-Based and Intelligent Information and Engineering Systems CONFERENCE

The generation of robust point detector algorithms is the base for several methodologies in computer vision like camera calibration, object tracking, image comparison. Vicomtech Research Center, the department of Computer Science and Artificial Intelligence of the Basque Country University and the laboratory of CAD/CAM/CAE of EAFIT University participate in the generation of a survey study of Point Detector Algorithms for implementation of the best result in projects related to computer graphics.

As co-authors of such publication, we give our permission for this material to appear in this document. We are ready to provide any additional information on the subject, as needed.

Prof. Dr. Eng. Oscar E. Ruiz
oruiz@eafit.edu.co
Coordinator CAD CAM CAE Laboratory
EAFIT University, Medellin, COLOMBIA

Jon Goenetxea
jgoenetxea@vicomtech.org

Vicomtech, San Sebastian, SPAIN

M.Sc. Iñigo Barandiaran
ibarandiaran@vicomtech.org
eHealth & Biomedical Applications
Vicomtech, San Sebastian, SPAIN

Prof. Manuel Graña

Computer Science and Artificial Intelligence Department
University of the Basque Country, San Sebastian, SPAIN

Abstract

Interest points extraction and matching is a common task in many computer vision based application, which are used in different domains, such as 3D reconstruction, object recognition, or tracking. We present an evaluation of current state of the art about interest point extraction algorithms to measure several parameters, such as detection quality, invariance to rotation and scale transformation, and computational efficiency.
Keywords : Computer vision, Interest points, point matching

1. Introduction

Image analysis or computer vision based applications deal with information extraction from the images acquired by a camera sensor. This information is usually discretized in form of relevant pixels or set of pixels (regions) having distinctiveness or discriminative characteristics, i.e, retain information about the structures in the scene being captured, compared with surrounding or neighbouring pixels or regions. Different terms in the literature referring to the same structures can be found, such as feature points, interest points, key points, or corner points. There has been progress in the last decade within the area of feature point extraction algorithms with image and video descriptors that led to substantial improvements in many computer vision areas including registration, 3D reconstruction, motion estimation, image registering, matching,and retrieval, or object and actions recognition. These types of techniques are usually found in the first stages of many computer vision algorithms, and can therefore be considered as low level image information extractors or descriptors. This information is then delivered to other processes in a bottom-up manner, until a semantic knowledge or meaning is recovered. In this article we show the results of an evaluation to measure the behaviour of several points detection algorithms, representative of the current state of the art.

The paper is structured as follows: In section 2 an overview of point detectors followed by a brief description of every detection algorithms evaluated in this study are given. In section 2.2, a description of the methods, measures and data sets used during the evaluation are shown. In section 3 details about the results obtained during the evaluation are given. Finally, in section 4 some general conclusions are made and future work perspectives are proposed.

2. Methods

2.1. Evaluation Parameters. A review about feature or interest point detectors can be found in [**45**]. The authors suggest that there are several parameters of a point detector that can be measured, and point out which are the most relevants that a point extractor should address:

Repeatability: : Ideally, the same points should be extracted by a detector even if a transformation to the image is applied.

Distinctiveness: : Detected features should be different enough in order to be identified or matched, and the same feature should not vary in images of the same scene.

Quantity: : This factor measures the number of interest points a detector is able to extract from an image.

Accuracy: : Measures the difference in spatial (image) localization of the same point. This value should be as minimal as possible, and ideally having zero or close to zero variance.

Efficiency: : Measures how fast an interest point detector is able to process an image.

It's worth noticing that all these factors are somehow application dependent, i.e, every computer vision application or image analysis has different needs. While for applications such as real-time simultaneous location and mapping (SLAM) mainly the quantity, and efficiency factors are critical, for an application such as off-line object recognition, repeatability and distinctiveness are more relevant. In any case, aforementioned factors can be reduced to two: quality and efficiency. Quality represents the feasibility of a detector to delegate accurate, precise, dense and robust set of points to the next process in the image analysis pipeline, while efficiency represents how fast and with which computation resources is able to carry out that task. Depending on these factors, the following processes in the pipeline should or should not apply different

mechanisms, filters, estimators, or heuristics in order for the application to succeed, running efficiently and obtaining accurate results. Interest points have to be ideally invariant or robust to any image transformations in order to be identified or matched from one image to another. Image transformations can be categorized in two different classes: geometric and radiometric transformations. Geometric transformations are those that modify the shape or the location of a feature in the image space, while radiometric transformations influence the feature appearance, i.e, the intensity or color value of the pixels. Changes in lighting conditions or camera acquisition parameters, i.e, sensor sensibility, exposure time or lens aperture, directly affect radiometric appearance of image by changing their luminance and/or chrominance information. Several geometric transformations are involved in the process of image formation, starting from the most general ones such as projectivities or homographies, to the most specific ones such as euclidean transformations, rotations, translations and scaling. In our study we focus only in geometric transformations excepting those related to optics such as lens distortions are out of the scope of this study.

2.2. Point Detectors. We have included in our evaluation many of the point detectors that currently represent the state of the art and also some older approaches that today are still broadly used by the computer vision community. In the following section, a brief description of every point detector included in the evaluation is given.

2.2.1. *HARRIS.* We have included Harris corner detector in the evaluation because it's one of the most used feature extractors by computer vision community since its publication [**46**]. Harris' approach can be seen as an evolution of one of the earliest corner detectors published, such as the Moravec's detector [**47**]. Harris' approach improves Moravec's detector by taking into consideration different orientations around the candidate pixel, instead of shifting patches by computing the second moment matrix, also called the auto-correlation matrix. Harris cornerness measure is still used by many point extractor approaches as a mechanism of non-maxima suppression. In this evaluation we have used a pyramidal version of Harris proposed in [**48**].

2.2.2. *GFTT.* (GoodFeatureToTrack) detector [**49**] is, like Harris detector, one of the most common feature detectors used by computer vision community. This approach was proposed as interest point detector for to be used in camera tracking algorithms, based on previous work of [**50**]. This approach proposes to different measures named texturedness and dissimilarity for solving the feature matching in order to obtain accurate tracking results, by estimating both a pure translations and affine motion respectively. Texturedness measure is used to extract good candidate points to be tracked, by examining the minimum eigenvalues of auto-correlation matrix.

2.2.3. *SIFT.* (Scale Invariant Feature Transformation) descriptor [**51**] is one of the most successful approaches for feature or interest point extractor and description. Detection is based on the convolution of images with difference of Gaussians (DoG) operator $\eta = (g_\sigma - g_{\sigma'})$. Images are arranged in a pyramidal representation, where every level(octave) of the pyramid represents a down sampled and smoothed version of the image in previous level. Smoothed images are obtained by convolving with a Gaussian operator with different values of scale σ. This arrange of images allows SIFT to work in a scale-space representation [**52**] or framework. SIFT detector and respective descriptor is intended to be invariant to either rotation and scale transformation, but not to perspective transformation, such as affine transformation.

2.2.4. *SURF.* (Speed Up Robust Feature) extractor follows an similar approach to SIFT, but addressing the problem of reducing computation cost. SURF searches for local maxima of the Hessian determinant in the scale-space. SURF calculates Hessian determinant efficiently by using a discrete approximation of the Gaussian second order partial derivatives, in conjunction with integral images representation [**53**]. Differently from SIFT approach, the scale is not obtained by decreasing the image size after smoothing, but by increasing the discrete kernels size.

2.2.5. *FAST.* detector [**54**] uses a different approach than SIFT or SURF detectors. This approach uses supervised classification to label pixels as their values or pertinence to two classes, interest point or background, by examining the values of surrounding pixels around a candidate one in a circular manner.

A feature is detected at pixel p if the intensities of at least n contiguous pixel of a surrounding circle of j pixels are all below or above the intensity of p by some threshold t. FAST approach does not use scale-space representation, and therefore is not invariant to scale nor rotation transformation.

2.2.6. *MSER.* (Maximally stable extremal regions)[**55**] is an approach based on the detection of blob like structures. MSER detects blobs by using local extrema in intensity or luminance space, obtained by iteratively applying watershed based segmentation. A region R_i is considered a feature if for all its n joined connected components $R_1, ..., R_n$ obtained after n watershed segmentations, it attains to a local minimum in the function $q_i = |R_{i+\alpha} - R_{i-\alpha}|/|R_i|$, where α is a used defined parameter and $|.|$ represents the cardinality of the blob measured in pixels.

2.2.7. *STAR.* This point extractor is also known as Censure (Center Sorround Extrema) [**56**]. This approach approximates the Laplacian not using DoG operator as SIFT does, but using bi-level center-surround filters of different shapes such as boxes, octagons, or hexagons. The computation of these filters in combination with integral images allows the detection of interest points in scale-space much faster than SIFT. In our evaluations we used bi-level star shaped filter as proposed and implemented in [**57**].

2.2.8. *ORB.* is the acronym of Oriented BRIEF. This algorithm proposes a modified version of FAST detector for computing orientation during detection step, and an efficient computation of BRIEF based approach for generating descriptors. This approach tries to merge the rotation and scale invariance of SIFT and the computational efficiency of FAST detector.

3. Evaluation

In our evaluation we have used the popular data set, publicly available in [**58**]. Original data set is composed of 6 different set of images, covering many problems addressed by computer vision or image analysis such as geometric transformations between images, image blurring, changing light conditions or image artifacts due to compression. We used only those set of images covering aspects related with geometric transformations, named Graffiti, Bark and Brick, as shown in figure 4.1. All tests were carried out using the point detector implementations integrated in Open Source computer vision Library OpenCV[**57**]. We have focused our evaluation by measuring two key factors in any interest point extractor: repeatability, also known as stability, and accuracy. Stability is referred as the ability of a detector of detect the same feature point across even in a change in radiometric or geometric conditions occurs between two different image acquisitions. Accuracy is related with the consistency and precision, in image space coordinates, in the location of every point extracted by a detector after a change occurs between two images. We have evaluated how accurate every point extractor is by measuring the matching ratio, i.e, the number of matched points or correspondences between two images divided by the total number of points detected, knowing a priori the geometric transformation applied between both images. A candidate pair of points in $images_i$ and $image_j$ respectively is considered a correspondence, if the true match is within a search window of radius r of its estimated correspondence given known rotation, scale or homography transformation. All tests were carried out by using a computer running Windows 7 OS, with 4Gb of RAM and an Intel QuadCore 2.7Ghz CPU. In 2.3.1 version of OpenCV not every CPU point detector algorithm has its own GPU version, so we decided not to distort the results by mixing CPU and GPU implementations.

3.1. Number of points detected. In this test we evaluated how many interest point every extractor is able to detect. Depending of the specificities of every algorithm, the number of extracted features may vary significantly, even if they are apply on the same image. Furthermore, depending on the spatial frequencies present in the images the number of detect points must be different. For example, images taken from Brick data set exhibit high frequency details represented by the bricks on the wall and therefore is the data set that clearly generates the higher number of detections in every extractor.

FIGURE 4.1. (Left) Graffiti Data set. (Center) Bark Data set. (Right) Brick Data set

TABLE 4.1. Number of Points detected in Image 1 of every Data Set

N of features	Graffiti	Brick	Bark
FAST	**7244**	**41266**	**11735**
STAR	870	1461	168
SIFT	2749	8371	4161
SURF	3203	7142	2253
ORB	702	702	702
MSER	516	2287	837
GFTT	1000	1000	100
HARRIS	855	1000	1000

As shown in table 4.1 FAST is clearly the one that gets more dense clouds of points, follow by SIFT and SURF, while STAR detector exhibits irregular results. ORB detector seems to have a maximum number of detections allowed, because the same number of 702 features is detected in every image, independently on the content. Similar results are obtained with GFTT and HARRIS having a maximum threshold of 1000 detections. It is important to notice that not only the number of points detected means a successful detector, but also how accurate and repeatable they are against geometric transformations.

3.2. Rotation Transformation. In this test we have evaluated how different approaches are robust against image rotation. We have used image 1 of every data set, and generate artificial images by apply different angle of in-plane rotation starting from 0 (same image), to 57.3 degrees in steps of 11.4 degrees, not varying scale nor perspective transformation. We included the same image without rotation, i.e 0 degrees, as a matter of measure the consistent of the detectors. In this test all detectors but MSER perform similarly, being SIFT, GFTT and ORB the ones that obtain better results.

3.3. Scale Transformation. In the following test we evaluated how the extractors are robust against scaling transformation. We used the image labeled as 1 of every data set, and generated artificial images by applying different isotropic image scaling factors starting from 1.0 (no scaling) to a factor of 3, not applying rotation nor perspective transformations. As expected, as scale factor increases results of all detectors decrease. STAR detector shows the worst performance while GFTT and ORB exhibits the most stable results.

3.4. Homography Transformation. In the following test we have evaluated how well the extractors are robust against homography transformation. An homography is a plane projective transformation. The

TABLE 4.2. Results of Rotation transformation

Graffiti	0,0	11,4	22,9	34,3	45,8	57,3	*Mean*
2 FAST	100	40,0	41,0	39,3	38,6	40,2	*39,8*
STAR	100	57,3	49,8	50,5	52,9	49,0	*51,8*
SIFT	100	62,4	56,8	56,1	54,3	56,6	*57,2*
SURF	100	44,9	29,2	24,5	24,2	26,2	*29,0*
ORB	100	**82,6**	**82,4**	**78,9**	**76,9**	74,5	***79,0***
MSER	100	33,3	26,7	24,2	20,7	24,2	*25,5*
GFTT	100	79,7	75,5	74,1	71,2	74,2	*74,9*
HARRIS	100	73,4	76,3	75,7	71,7	**75,4**	*74,5*

Bark	0,0	11,4	22,9	34,3	45,8	57,3	*Mean*
FAST	100	52,7	51,4	47,6	46,8	44,6	*56,6*
STAR	100	57,7	44,6	45,8	48,8	39,8	*60,8*
SIFT	100	68,5	61,5	59,1	**57,3**	**55,7**	*65,4*
SURF	100	62,0	45,4	40,6	37,4	35,7	*31,7*
ORB	100	**72,6**	59,4	53,1	49,0	45,2	*66,6*
MSER	100	30,7	22,4	14,0	11,5	11,4	*35,7*
GFTT	100	70,1	**66,3**	**59,1**	55,8	53	*63,8*
HARRIS	100	36,1	38	17	11,1	50,6	***69,0***

Brick	0,0	11,4	22,9	34,3	45,8	57,3	*Mean*
FAST	100	59,6	58,2	56,6	55,0	53,9	*48,5*
STAR	100	70,4	63,3	60,0	56,5	55,0	*47,0*
SIFT	100	69,5	67,3	65,1	63,3	62,0	*60,3*
SURF	100	46,4	31,7	28,6	27,8	27,2	*43,3*
ORB	100	70,2	68,3	66,3	**66,2**	62,3	*55,1*
MSER	100	45,4	37,3	34,0	32,3	31,0	*16,6*
GFTT	100	68,1	63,8	63,7	61,6	61,9	***60,5***
HARRIS	100	**72,5**	**69,6**	**69,6**	66,2	**67,5**	*26,5*

All Images	100	50,1	49,7	47,3	46,3	45,9	47,8
STAR	100	61, 5	52,0	51,8	52,6	47,5	52,9
SIFT	100	66,7	61,7	60,0	58,2	58,0	60,8
SURF	100	50,5	34,8	30,5	30,1	29,4	34,3
ORB	100	74,9	69,4	65,2	62,9	59,4	**66,2**
MSER	100	35,9	28,2	22,6	19,7	20,5	24,7
GFTT	100	72,5	68,4	65,3	62,6	62,4	66,1
HARRIS	100	57,7	58,7	44,7	37,5	63,6	51,5

TABLE 4.3. Results of Scale transformation

Graffiti	1,0	1,4	1,8	2,2	2,6	3,0	*Mean*
FAST	100	37,7	29,8	28,8	29,5	24,3	*29,7*
STAR	100	43,2	31,6	22,0	21,7	13,9	*24,6*
SIFT	100	45,3	27,3	18,6	16,5	12,7	*21,7*
SURF	100	41,5	31,3	28,0	20,1	18,7	*26,7*
ORB	100	**69,2**	51,7	37,7	31,7	**31,0**	*42,1*
MSER	100	31,5	23,8	15,8	8,3	5,4	*13,9*
GFTT	100	67,9	**56,4**	**48,1**	**45,2**	30,5	***47,9***
HARRIS	100	64,6	50,7	44,0	41,0	26,6	*43,5*

Bark	1,0	1,4	1,8	2,2	2,6	3,0	*Mean*
FAST	100	43,2	33,6	30,9	29,8	26,2	*32,3*
STAR	100	50,5	40,4	32,1	24,4	13,6	*29,3*
SIFT	100	46,9	27,5	19,3	13,3	9,6	*19,9*
SURF	100	49,3	34,6	25,2	18,3	12,8	*25,1*
ORB	100	**70,2**	**53,7**	**46,1**	**41,5**	**42,1**	***49,7***
MSER	100	25,0	17,5	10,9	6,3	4,4	*10,5*
GFTT	100	55,5	43,4	36,2	35,3	25,9	*38,0*
HARRIS	100	35,7	23,9	25,3	26,7	15,2	*24,4*

Brick	1,0	1,4	1,8	2,2	2,6	3,0	*Mean*
FAST	100	41,1	33,0	32,6	33,4	27,0	***33,1***
STAR	100	24,9	8,3	2,9	1,5	0,6	*3,5*
SIFT	100	34,0	15,6	8,9	4,8	3,2	*9,3*
SURF	100	37,6	24,7	17,5	11,2	8,6	*17,3*
ORB	100	**53,1**	**33,9**	**26,6**	**24,0**	**20,3**	*29,7*
MSER	100	42,1	29,2	14,2	3,1	0,4	*7,3*
GFTT	100	40,2	27,6	22,7	19,7	18,3	*24,6*
HARRIS	100	43,8	28,6	21,2	21,5	18,8	*26,1*

All Images	1,0	1,4	1,8	2,2	2,6	3,0	*Mean*
FAST	100	40,6	32,1	30,7	31,5	25,9	31,7
STAR	100	37,95	22,0	12,7	9,3	4,8	13,6
SIFT	100	41,6	22,7	14,7	10,2	7,0	16,0
SURF	100	42,5	29,9	23,1	16,0	12,7	22,6
ORB	100	**63,7**	**45,5**	**35,9**	**31,6**	**29,8**	**39,6**
MSER	100	32,1	23,0	13,5	5,5	2,1	10,3
GFTT	100	53,3	40,7	34,1	31,6	24,4	35,5
HARRIS	100	46,6	32,6	28,7	29,9	19,7	30,3

TABLE 4.4. Results of Homography transformation

Graffiti	Image1	Image2	Image3	Image4	Image5	Image6
FAST	100	25.9	23.4	20.4	16.8	**13.9**
STAR	100	22.2	14.5	6.4	4.8	1.6
SIFT	100	30.8	14.3	9.4	5.8	3.7
SURF	100	20.0	11.8	7.3	5.3	3.9
ORB	100	**54.1**	**36.7**	**27.6**	**17.3**	3.2
MSER	100	16.2	12.7	6.9	5.2	1.7
GFTT	100	33.4	21.5	13.3	10.7	1.9
HARRIS	100	39.4	25.1	14.5	13.6	1.7

transformation between any two images of the same planar structure in space can be described by an homography transformation. Homography transformation estimation are widely used by computer vision community in many applications such as image rectification, image registration, camera calibration or camera pose estimation. We have used the six images of Graffiti data set with homography ground truth transformation,

TABLE 4.5. Mean pixel error for Homography transformation

Graffiti	image1	image2	image3	image4	image5	image6
FAST	0,00	0,73	0,73	0,74	0,75	0,76
STAR	0,00	0,70	0,72	0,72	0,73	0,77
SIFT	0,01	**0,66**	**0,68**	0,75	0,75	0,81
SURF	0,05	0,73	0,74	0,74	0,77	0,79
ORB	0,25	0,70	0,73	0,73	0,75	0,75
MSER	0,10	0,72	0,70	0,74	**0,70**	**0,74**
GFTT	0,00	0,69	0,70	0,77	0,77	0,84
HARRIS	0,00	0,69	0,71	**0,71**	0,73	0,81

mapping points from image 1 to the rest of images of the data set. In this test we obtained overall worst results compared to rotation or scale tests. The worst results are obtained comparing image1 with image6 (column 6) because of the severe distortion generated by the homography to patches around each feature point. These results shows that no one of the tested extractors are truly invariant to perspective transformation. Most of the current approaches propose to generate feature point detectors and descriptors to be invariant to affine geometric transformation. Affinities preserves parallelism between lines, while projectivities only preserves straightness of lines not giving enough discriminative or distinctiveness power allowing to have a detector fully invariant to such type of transformations. However, perspective transformations can be effectively approximated by piecewise local affinities so affine invariants detectors should perform best in this test. ORB detector clearly outperforms the rest of detectors, being the one that gets best matching ratios in all images of all data sets.

3.5. Accuracy. We wanted also to measure how precise and consistent every point extractor is related with the locations, in image space coordinates, of every point detected, after a viewpoint change occurs between two images. We measure the accuracy by calculating the difference in pixels between location of detected point, being a correct match, and the true location. We depict here the averaged results obtained with Graffiti, brick and bark data sets. Results depicted in table 4.5 shows that there is no clear winner with respect to feature point location accuracy, given a mean pixel error value of 0,62 between all detectors.

3.6. Efficiency. In addition to accuracy of feature point detectors and their invariance to geometric transformation, how fast and efficient they perform such task is also critical in some computer vision applications. we have measured the time they expend to perform detection task operation in every of the six images of Graffiti data set. As mentioned previously, depending on the content of the images, i.e low or high frequencies, the number of detected points vary and therefore computation time. The lower computation time the better. Results in image 3.6 shows clearly that FAST is the best performer related with efficiency. Even if being the one that extracts the highest number of points, it only takes 22ms per image on average. It worth notice also how ORB is very close to FAST taking only 42.7 ms per frame, and taking into account that this approach estimates feature orientation while FAST does not. As expected, SIFT extractor is the slower due to the computation of several DoG operators, taking 1000ms on average.

4. Conclusions

We present an evaluation of different feature or interest point extractors. We have measured their invariance and robustness to several geometric image transformations such as rotations, translations, scaling or perspective transformations such as homographies. We have also evaluated their capability of generating information by measuring the number of points they generate. Finally, we also measured their efficiency related with computational resources. As mentioned before, the choice of the feature detector very much

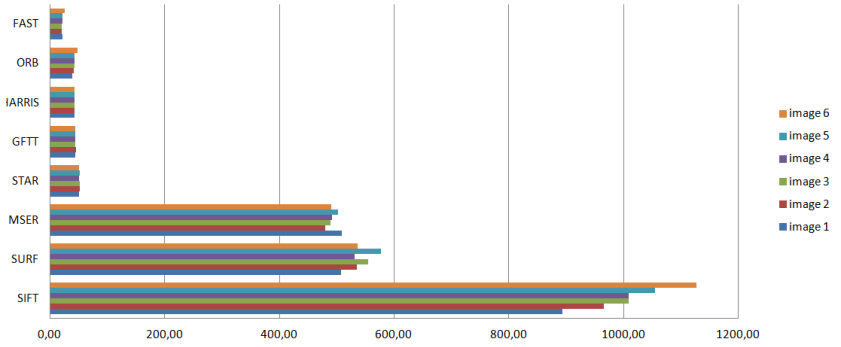


FIGURE 4.2. Computation Time measured in miliseconds (lower time is better)

depends on the application specifications. Overall, we can conclude that the recent ORB detector obtains the best ratio between accuracy and efficiency. ORB shows the best performance on rotation and homography tests and also being the second related with computational cost only exceeded by FAST, however this one does not have on orientation component and does not produce multi-scale features and therefore is not as accurate as ORB in rotation and scaling transformations. One very important aspect nowadays is efficiency, as more and more applications are being migrated to mobile devices, such as iPad or iPhone. In this way, those approaches similar to FAST or ORB detector requiring low computation and memory resources are very useful and promising. The next step is to evaluate all these algorithms running on a mobile device, taking into account that some implementation may be optimized for running on a specific processor architecture using specific instructions, not supported by mobile processors. We are now interested in evaluating how current feature descriptors, in addition with random sampling robust estimation strategies, can overcome problems of robustness and invariance to scale, rotation, and perspective transformations. Recent feature descriptor approaches such as BRIEF, GLOH, DAISY, or ORB opens new possibilities for computer vision applications, such as simultaneous location and mapping, or image registering and reconstruction to run robustly in real-time.

Part 2

Indirect Volume Rendering

CHAPTER 5

Extending Marching Cubes with Adaptative Methods to obtain more accurate iso-surfaces

- John Congote[1,2]
- Aitor Moreno[2]
- Iñigo Barandiaran[2]
- Javier Barandiaran[2]
- Oscar Ruiz[1]

[1] CAD CAM CAE laboratory, Universidad EAFIT
Carrera 49 No 7 Sur - 50, Medellín, Colombia
[2] Vicomtech Research Center
Donostia - San Sebastin, Spain

Context

Extending Marching Cubes with Adaptative Methods to Obtain More Accurate Iso-surfaces . John Congote, Aitor Moreno, Iñigo Barandiaran, Javier Barandiaran, Oscar Ruiz. In: book "Computer Vision, Imaging and Computer Graphics. Theory and Applications" , Ranchordas, Alpeshkumar and Pereira, Joao Manuel R.S. (Ed.). Publisher: Springer Berlin Heidelberg. Vol. 68, 2010. pp 35-44. ISBN: 978-3-642-11840-1. BOOK CHAPTER

The VISION project is a Spanish project for the development of new models of communication. The generation of high quality 3D models is required for a satisfactory communication process. This work explore a new methodology for the generation of high quality 3D models from scalar data. This work has been partially supported by the Spanish Administration agency CDTI, under project CENIT-VISION 2007-1007. CAD/CAM/CAE Laboratory - EAFIT University and the Colombian Council for Science and Technology -Colciencias-.

As co-authors of such publication, we give our permission for this material to appear in this document. We are ready to provide any additional information on the subject, as needed.

Prof. Dr. Eng. Oscar E. Ruiz
oruiz@eafit.edu.co
Coordinator CAD CAM CAE Laboratory
EAFIT University, Medellin, COLOMBIA

Dr. Aitor Moreno
amoreno@vicomtech.org

Vicomtech, San Sebastian, SPAIN

M.Sc. Iñigo Barandiaran
ibarandiaran@vicomtech.org
eHealth & Biomedical Applications
Vicomtech, San Sebastian, SPAIN

Eng. Javier Barandiaran
jbarandiaran@vicomtech.org

Vicomtech, San Sebastian, SPAIN

Abstract

This work proposes an extension of the Marching Cubes algorithm, where the goal is to represent implicit functions with higher accuracy using the same grid size. The proposed algorithm displaces the vertices of the cubes iteratively until the stop condition is achieved. After each iteration, the difference between the implicit and the explicit representations is reduced, and when the algorithm finishes, the implicit surface representation using the modified cubical grid is more accurate, as the results shall confirm. The proposed algorithm corrects some topological problems that may appear in the discretization process using the original grid.
Keywords: adaptive isosurface extraction, adaptive tessellation, isosurfaces, volume warping, marching cubes.

1. Introduction

Surface representation from scalar functions is an active research topic in different fields of Computer Graphics such as medical visualization of Magnetic Resonance Imaging (MRI) and Computer Tomography (CT) [**59**]. This representation is also widely used as an intermediate step for several graphical processes [**60**], such as mesh reconstruction from point clouds or track planning. The representation of a scalar function in 3D is known as implicit representation and is generated using continuous algebraic iso-surfaces, radial basis functions [**61**] [**62**], signed distance transform [**63**], discrete voxelisations or constructive solid geometry.

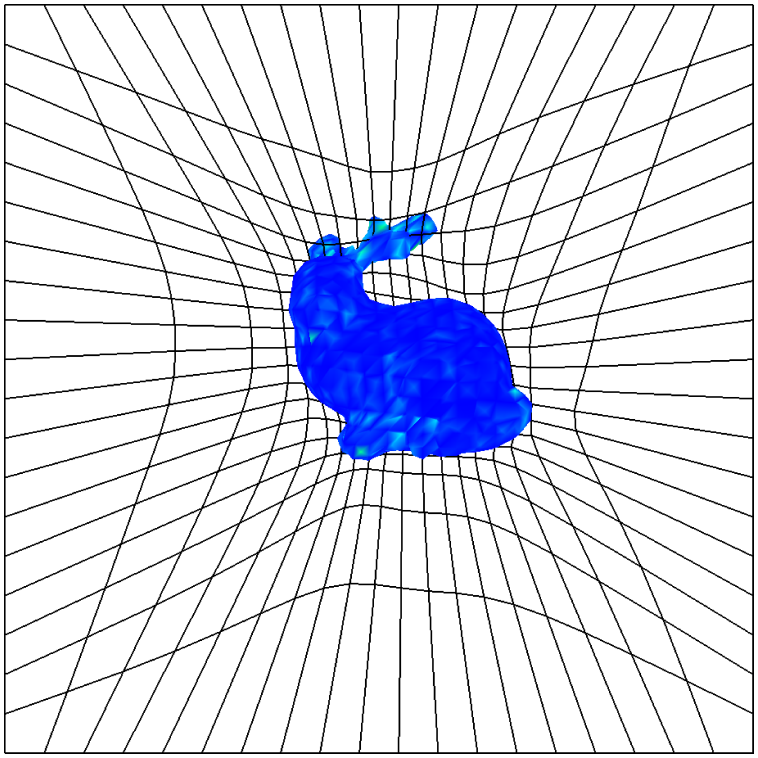


FIGURE 5.1. Optimised Grid with 20^3 cubes representing the bunny.

The implicit functions are frequently represented as a discrete cubical grid where each vertex has the value of the function. The Marching Cubes algorithm (MC) [**64**] takes the cubical grid to create an explicit

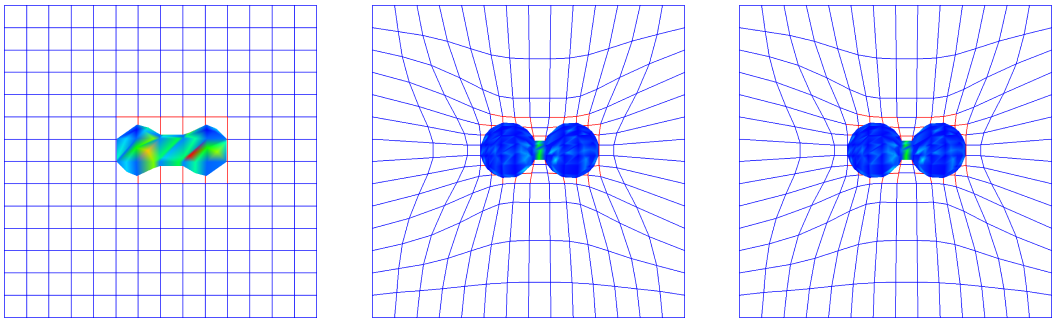
representation of the implicit surface. The MC algorithm has been widely studied as has been demonstrated by Newman [**65**]. The output of the MC algorithm is an explicit surface represented as a set of connected triangles known as polygonal representation. The original results of the MC algorithm presented several topological problems as demonstrated by Chernyaev [**66**] and have already been solved by Lewiner [**67**].

The MC algorithm divides the space in a regular cubical grid. For each cube, a triangular representation is calculated, which are then joined to obtain the explicit representation of the surface. This procedure is highly parallel because each cube can be processed separately without significant interdependencies. The resolution of the generated polygonal surface depends directly on the input grid size. In order to increase the resolution of the polygonal surface it is necessary to increase the number of cubes in the grid, increasing the amount of memory required to store the values of the grid.

Alternative methods to the MC algorithm introduce the concept of generating multi-resolution grids, creating nested sub-grids inside the original grid. The spatial subdivision using octrees or recursive tetrahedral subdivision techniques are also used in the optimization of iso-surface representations. The common characteristic of these types of methods is that they are based on adding more cells efficiently to ensure a higher resolution in the final representation.

This work is structured as follows: In Section 2, a review of some of the best known MC algorithm variations is given. Section 3 describes the methodological aspects behind the proposed algorithm. In Section 4 details the results of testing the algorithm with a set of implicit functions. Finally, conclusions and future work are discussed in Section 5.

2. Related Work



(a) Original Grid. The two spheres are displayed as a singular object due to the poor resolution in the region.

(b) Intermediate Grid. Both spheres are displayed well, but are still joined.

(c) Final Grid. The new resolution displays two well shaped and separated spheres with the same number of cubes in the grid.

FIGURE 5.2. 2D slides representing three different states in the evolution of the algorithm of two nearby spheres.

Marching Cubes (MC) [**64**] has been the *de facto* standard algorithm for the process generating of explicit representations of iso-surfaces from scalar functions or its implicit definition The MC algorithm takes as an input a regular scalar volumetric data set, having a scalar value residing at each lattice point of a rectilinear lattice in 3D space. The enclosed volume in the region of interest is subdivided into a regular grid of cubes. Each vertex of all cubes in the grid is set by the value of the implicit function evaluated at the vertex coordinates. Depending on the sign of each vertex, a cube has 256 (2^8) possible combinations, but using geometrical properties, such as rotations and reflections, the final number of combinations is reduced

to 15 possibilities. These 15 surface triangulations are stored in Look-Up Tables (LUT) for performance reasons. The final vertices of the triangular mesh are calculated using linear interpolation between the values assigned to the vertices of the cube. This polygonal mesh representation is the best suitable one for the current generation of graphic hardware because it has been optimized to this type of input.

MC variations were developed to enhance the resolution of the generated explicit surfaces, allowing the representation of geometrical lost details during MC discretization process. Weber [**68**] proposes a multi-grid method. Inside an initial grid, a nested grid is created to add more resolution in that region. This methodology is suitable to be used recursively, adding more detail to conflictive regions. In the final stage, the explicit surface is created by joining all the reconstructed polygonal surfaces. It is necessary to generate a special polygonization in the joints between the grid and the sub-grids to avoid the apparition of cracks or artefacts. This method has a higher memory demand to store the new values of the nested-grid.

An alternative method to refine selected regions of interest(*ROI*) is the octree subdivision [**69**]. This method generates an octree in the region of existence of the function, creating a polygonization of each octree cell. One of the flaws of this method is the generation of cracks in the regions with different resolutions. This problem is solve with the Dual Marching Cubes method [**70**] and implemented for algebraic functions by Pavia [**71**].

The octree subdivision method produces edges with more than two vertices, which can be overcome by changing the methodology of the subdivision. Instead of using cubes, tetrahedrons were used to subdivide the grid, without creating nodes in the middle of the edges [**72**]. This method recursively subdivides the space into tetrahedrons.

The previous methodologies increment the number of cells of the grid in order to achieve more resolution in the regions of interest. Balmelli [**73**] presented an algorithm based on the warping of the grid to a defined region of interest. The warping of the vertices is performed in a hierarchical procedure, the volume is considered as a single cell of the grid, and then, the central point of the grid is warped in the direction of the *ROI*. Then, this cell is divide then in eight cells and the process is repeated until the number of selected cells is achieved. The result is a new grid with the same number of cells, but with higher resolution near to the *ROI*. The algorithm was tested with discrete datasets, and the *ROI* is created by the user or defined by a crossing edges criteria.

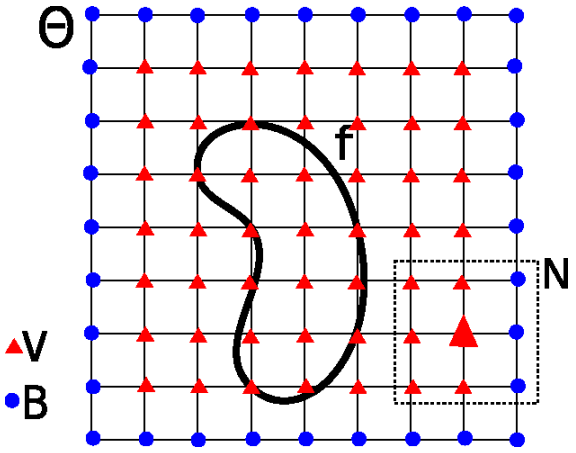
The presented method generates a similar warping grid as Balmelli does, but we avoid the use of hierarchical procedures and our region of interest is automatically generated based in the input implicit function, obtaining dense distribution of vertices near the iso-surface. (see Figure 5.2)

3. Methodology

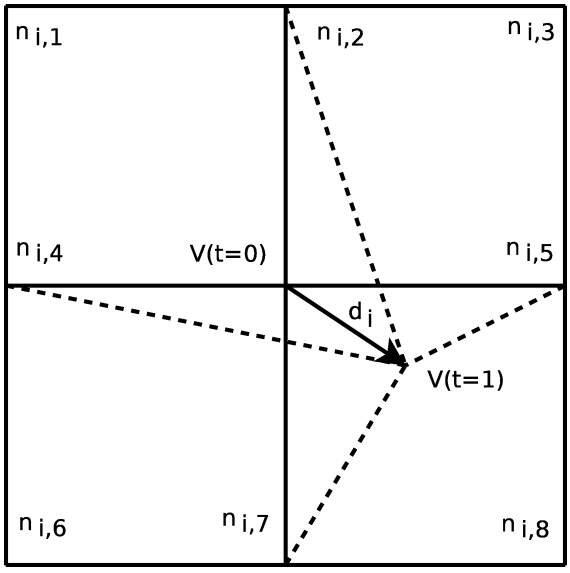
The presented algorithm in this work is an extension of the MC algorithm. The main goal is to generate a more accurate representations of the given implicit surfaces with the same grid resolution.

Applying a calculated displacement to the vertices of the grid, the algorithm reconfigure the position of the vertices of the grid to obtain more accurate representations of the iso-surface. In order to avoid self-intersections and to preserve the topological structure of the grid, the vertices are translated inside the cells of the neighbour of the vertex. The displacement to be applied to all the vertices are calculated iteratively until a stop condition is satisfied.

Let be Θ a rectangular prism tessellated as a cubical honeycomb, W the vertices of Θ [Eq. 13], B the boundary vertices of Θ [Eq. 14], and V the inner vertices of Θ [Eq. 15]. For each vertex $v_i \in V$, a N_i set is defined as the *26 adjacent* vertices to v_i, denoting each adjacent vertex as $n_{i,j}$ [Eq. 16]. (see Figure 3(a)). $f(w)$ is the value of the function in the position w and A is the scale value for the attraction force for the displacement of the vertices.



(a) Grid nomenclature, Θ cubical grid, $f(x, y, z) = 0$ implicit function, N vertex neighbourhood, V vertices inside the grid, B vertices at the boundary of the grid.



(b) two consecutive iterations are show where the vertex v is moved between the iterations $t = 0$ and $t = 1$. The new configuration of the grid is shown as dotted lines.

$$W = \{w_i / w_i \in \Theta\} \tag{13}$$

$$B = \{b_i / b_i \in \delta\Theta\} \tag{14}$$

$$V = W - B \tag{15}$$

$$N_i = \{n_{i,j} / n_{i,j} \text{ is } j\text{th neighborhood of } v_i\} \tag{16}$$

The proposed algorithm is an iterative process. In each iteration, each vertex v_i of the grid Θ is translated by a d_i displacement vector [Eq. 18], obtaining a new configuration of Θ, where *i)* the topological connections of the grid are preserved, *ii)* cells containing patches of f are a more accurate representation of the surface, and *iii)* the total displacement [Eq. 19] of the grid is lower and is used as the stop condition of the algorithm when it reach a value Δ(see Figure 3(b)).

The distance vector d_i is calculated as shown in [Eq. 18] and it can be seen as the resultant force of each neighbouring vertex scaled by the value of f at the position of each vertex and the attraction value A. In order to limit the maximum displacement of the vertices and to guarantee the topological order of Θ, the distance vector d_i is clamped in the interval expressed in [Eq. 17]

The attraction value A is empirical value which scale the value of the function in all the grid. This value control the attraction factor of the vertices of the grid to the iso-surface, values between 0 and 1 produces a grid avoids the iso-surface, values lesser than 0 generate incorrect behaviour of the function. The recommended and useful values are equal or greater than 1, very high values of A could generate problems for the grid, produces big stepping factors for the displacement vectors d_i and then some characteristics of the iso-surface could be lost. The A value is highly related to the value of the distance of the bounding box of the grid, and the size of the objects inside the grid.

$$0 \leq |d_i| \leq \text{MIN}\left(\frac{|n_{i,j} - v_i|}{2}\right) \tag{17}$$

repeat
  s := 0;
  foreach *Vertex* v_i **do**
    $d_i := \frac{1}{26} \sum_{n_{i,j}} \frac{n_{i,j}-v_i}{1+A\lvert f(n_{i,j})+f(v_i)\rvert}$; mindist := MIN $\left(\frac{\lvert n_{i,j}-v_i\rvert}{2}\right)$;
    $d_i := \bar{d_i}$CLAMP $(\lvert d_i\rvert, 0.0, \text{mindist})$; $v_i := v_i + d_i$;
    s := s + $\lvert d_i\rvert$;
  end
until s $\geq \Delta$;

Algorithm 3: Vertex Displacement Pseudo-algorithm. $|x|$ represents the magnitude of x, $\bar{v}$ represents the normalized vector of v

$$d_i = \frac{1}{26} \sum_{n_{i,j}} \frac{n_{i,j} - v_i}{1 + A|f(n_{i,j}) + f(v_i)|} \tag{18}$$

$$\sum_{v_i} |d_i| \geq \Delta \tag{19}$$

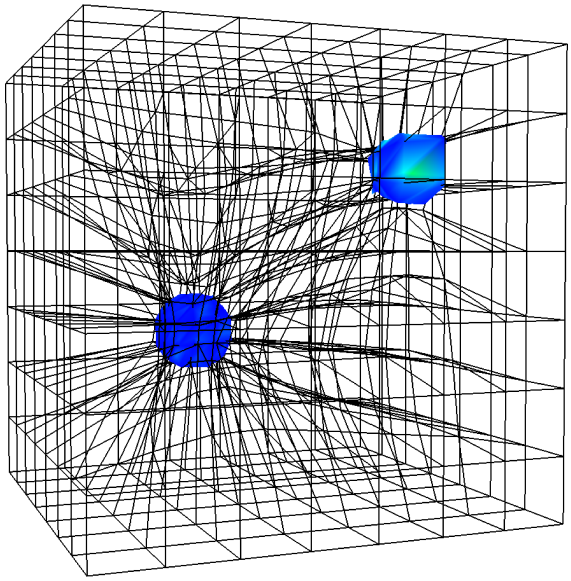
The algorithm stops when the sum of the distances added to all the vertices in the previous iteration is less that a given threshold Δ [Eq. 19] (see Algorithm 3).

4. Results

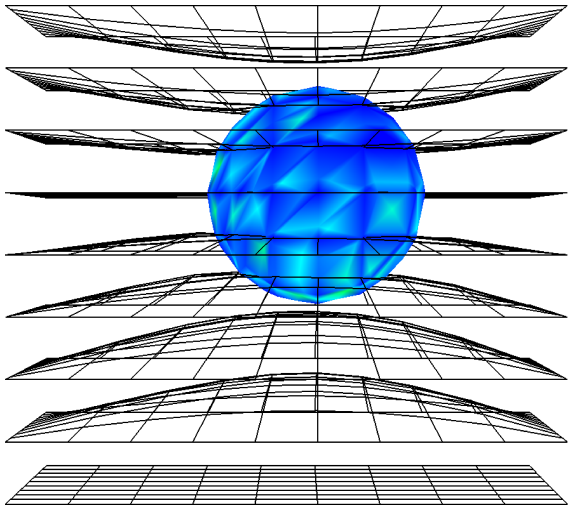
The proposed algorithm was tested with a set of implicit functions as distance transforms (see Figure 5.3) of a set of spheres, the spheres are define as a point in the space with their radius. The result of the sphere data set (3(d),4(a),5(a)) and the two-sphere data set (3(c),4(b),5(b)) are presented, but the algorithm also has been tested with datasets composed of more than 1000 spheres (see Figure 5.1). The algorithm has been tested also with other non-distance transform implicit functions, but the generation of false *ROI* in the grid degenerates the structure of the grid resulting in bad representations of the iso-surface. For demonstration purposes, the number of cells has been chosen to enhance visual perception of the improvements produced by the algorithm. For the visualization process we use Marching Tetrahedra [**74**] because it produces correct topological representation of the iso-surface, and allows the identification of the topological correctness of the algorithm.

The obtained results of the algorithm are visually noticeable, as shown in Figure 5.2. Without using the algorithm, the two spheres model is perceived as a single object (see Figure 5.2). In an intermediate state the spheres are still joined, but their shapes are more rounded. In the final state, when the algorithm converges, both spheres are separated correctly, each one being rendered as a near-perfect sphere. Thus, using the same grid resolution and the proposed algorithm, the resolution of the results has been increased and also topological errors of the original explicit representation were considerably reduce with the algorithm.

Accuracy of the explicit representations of the algorithm were measured using the methodology of De Bruin [**75**] which is based on the Hausdorff distance explain by Dubuisson [**76**]. The figures 5.4 and 5.5 shows the behaviour of the algorithm where, in almost all the iterations, the Hausdorff distance between the implicit iso-surface and the explicit surface are decreasing. The iterations where there is an increment of the



(c) Two spheres in different positions with a scalar function as the distance transform, representing the behavior of the algorithm with different objects in the space.



(d) Sphere in the center of the space with a scalar function as the distance transform. The slides shows the different warping of the grid in the different positions

FIGURE 5.3. Implicit function of spheres as distance transforms

distance, could represent a point where the configuration of the grid is not suitable for optimization and then the algorithm needs to modify the grid looking for suitable configurations.

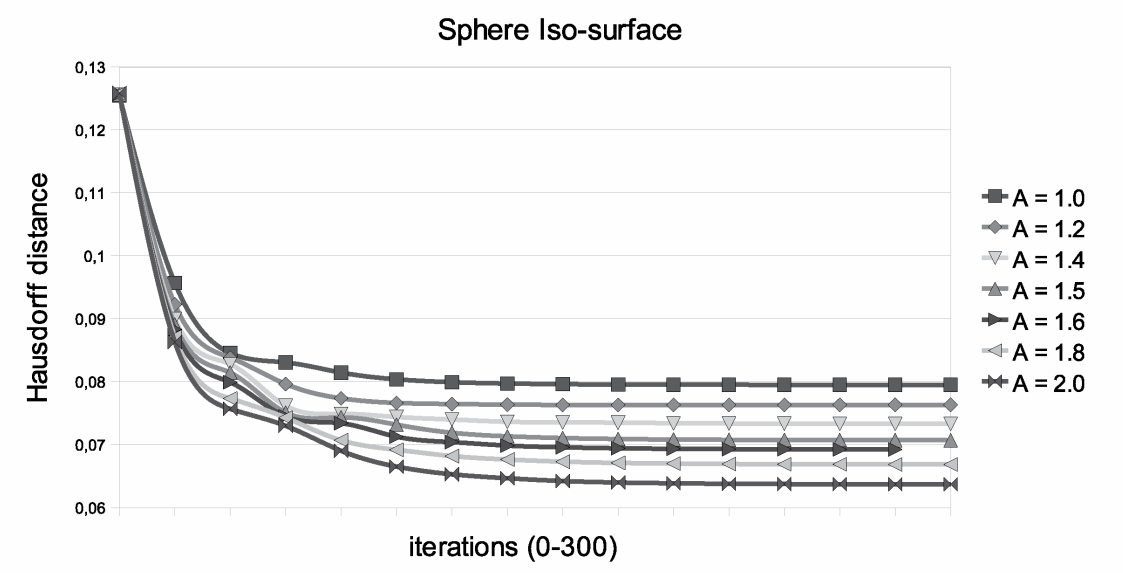
Figure 5.4 presents the results of the algorithm with the same implicit surfaces but with different attraction factor values (A). As the results show the accuracy of the iso-surface is highly related with the (A) value, because the allowed warping of the grid is bigger obtaining a dense grid near to the iso-surface, but this over-fit of the grid can be dangerous if the grid is going to be used for other implicit functions, like time varying functions, because the grid need more iterations to adapt to the new iso-surface.

Figure 5.5 shows the behaviour of the algorithm with different grid sizes. The accuracy of the final explicit representation of the iso-surfaces shows an improvement of the accuracy of the representation. Then it is possible with the algorithm to represent iso-surfaces with good accuracy and quality without increasing grid sizes. This characteristic allow us to use smaller grids for the representation without loss of accuracy or quality in the representation and saving computational resources.

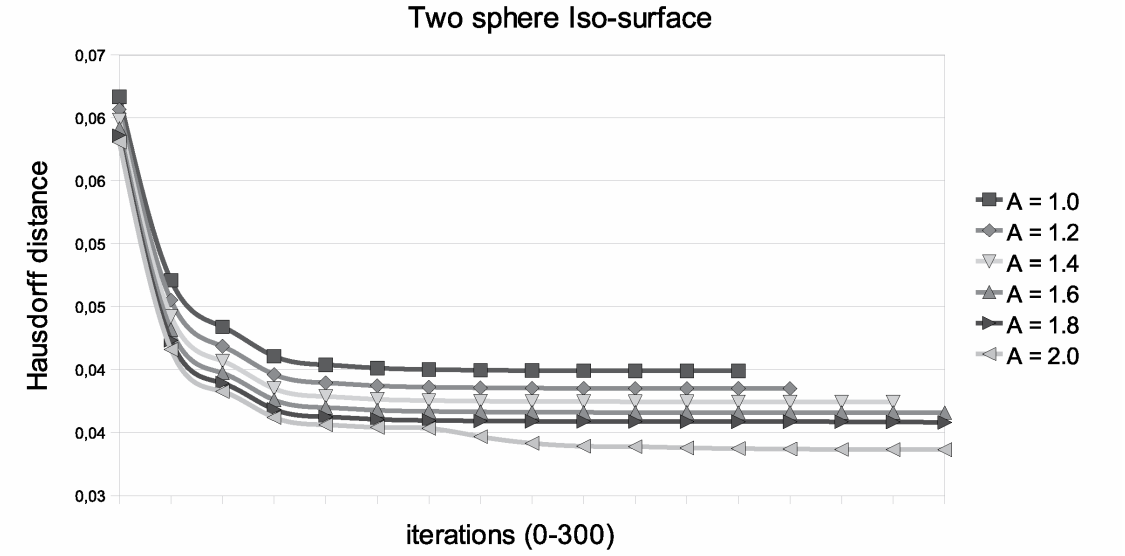
5. Conclusions and Future Work

Our proposed iterative algorithm has shown significant advantages in the representation of distance transform functions. With the same grid size, it allows a better resolution by displacing the vertices of the cube grids towards the surface, increasing the number of cells containing the surface. The algorithm was tested with algebraic functions, representing distance transforms of the models. The generated scalar field has been selected to avoid the creation of regions of false interest [**77**], which are for static images in which these regions are not used.

The number of iterations is directly related to the chosen value Δ as it is the stop condition. The algorithm will continuously displace the cube vertices until the accumulated displacement in a single iteration is less than Δ. The accumulated distance converges quickly to the desired value. This behaviour is very convenient to represent time varying scalar functions like 3D videos, where the function itself is continuously



(a) Sphere implicit function.

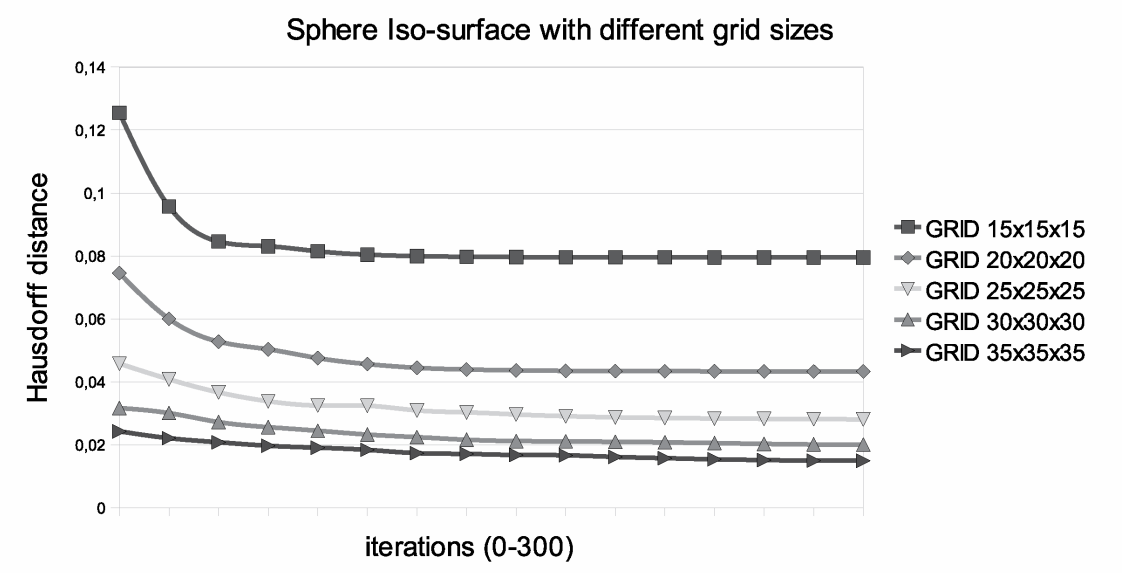


(b) Two sphere implicit function.

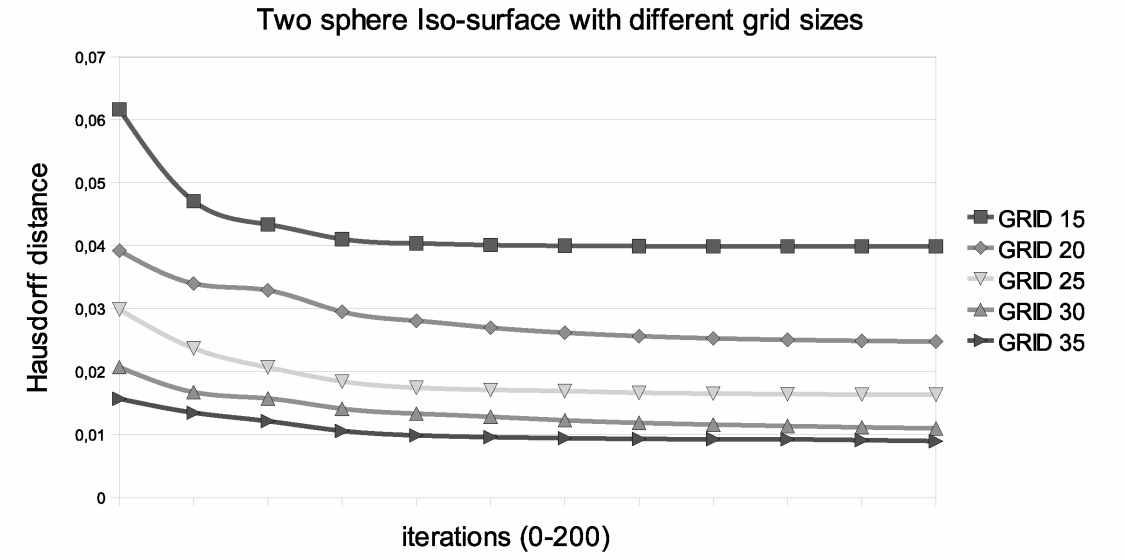
FIGURE 5.4. Hausdorff distance of the explicit representations of the figures in each iteration of the algorithm with different attraction values

changing. In this context, the algorithm will iterate until a good representation of the surface is obtained. If the surface varies smoothly, the cube grid will be continuously and quickly re-adapted by running a few iterations of the presented algorithm. Whenever the surface changes can be considered no be small, the number of iterations until a new final condition is reached will be low. Then the obtained results will be a better real-time surface representation using a coarser cube grid.

The value Δ is sensitive to the grid size, so a better stop condition should be evaluated which represent the state of the quality of the representation and reduce the number of iteration which are unnecessary.



(a) Sphere implicit function.



(b) Two sphere implicit function.

FIGURE 5.5. Hausdorff distance of the explicit representations of the figures in each iteration of the algorithm with different grid sizes

The model used in this algorithm is close to a physics spring model, a close comparison of the proposed algorithm with the spring model could be done.

CHAPTER 6

Marching Cubes in a unsigned distance field for Surface Reconstruction from Unorganized Point Sets

- John Congote[1,2]
- Aitor Moreno[2]
- Iñigo Barandiaran[2]
- Javier Barandiaran[2]
- Jorge Posada[2]
- Oscar Ruiz[1]

[1] CAD CAM CAE laboratory, Universidad EAFIT
Carrera 49 No 7 Sur - 50, Medellín, Colombia
[2] Vicomtech Research Center
Donostia - San Sebastián, Spain

Context

Marching Cubes in an Unsigned Distance Field for Surface Reconstruction from Unorganized Point Sets. John Congote, Aitor Moreno, Iñigo Barandiaran, Javier Barandiaran, Jorge Posada, Oscar Ruiz. GRAPP-2010. International Conference on Computer Graphics Theory and Applications. Angers, France, May 17-21, 2010. ISBN: 978-989-674-026-9, , Thomson Reuters, IET=Inspec, DBPL=uni-trier.de, pp: 143-147. CONFERENCE

The VISION project is a Spanish project for the development of new models of communication. The generation of fast 3D models is required for a satisfactory communication process. This work explore a new methodology for the generation of fast 3D models from point clouds. This work has been partially supported by the Spanish Administration agency CDTI, under project CENIT-VISION 2007-1007. CAD/CAM/CAE Laboratory - EAFIT University and the Colombian Council for Science and Technology -Colciencias-.

As co-authors of such publication, we give our permission for this material to appear in this document. We are ready to provide any additional information on the subject, as needed.

Prof. Dr. Eng. Oscar E. Ruiz
oruiz@eafit.edu.co
Coordinator CAD CAM CAE Laboratory
EAFIT University, Medellin, COLOMBIA

Dr. Aitor Moreno
amoreno@vicomtech.org

Vicomtech, San Sebastian, SPAIN

M.Sc. Iñigo Barandiaran
ibarandiaran@vicomtech.org
eHealth & Biomedical Applications
Vicomtech, San Sebastian, SPAIN

Javier Barandiaran
jbarandiaran@vicomtech.org

Vicomtech, San Sebastian, SPAIN

Dr. Jorge Posada
jposada@vicomtech.org

Vicomtech, San Sebastian, SPAIN

Abstract

Surface reconstruction from unorganized point set is a common problem in computer graphics. Generation of the signed distance field from the point set is a common methodology for the surface reconstruction. The reconstruction of implicit surfaces is made with the algorithm of marching cubes, but the distance field of a point set can not be processed with marching cubes because the unsigned nature of the distance. We propose an extension to the marching cubes algorithm allowing the reconstruction of 0-level iso-surfaces in an unsigned distance field. We calculate more information inside each cell of the marching cubes lattice and then we extract the intersection points of the surface within the cell then we identify the marching cubes case for the triangulation. Our algorithm generates good surfaces but the presence of ambiguities in the case selection generates some topological mistakes.
Keywords: marching cubes, surface reconstruction, point set, unsigned distance field

1. Introduction

Surface reconstruction from an unorganized point set has been a widely studied problem in the computer graphics field. The problem can be defined as given a set of unorganized points which are near to a surface. A new surface is reconstructed from that points. Which is expected to be near to the original one.

Acquisition systems like laser scanners, depth estimation from images, range scan data, LIDAR and other methodologies results in an unorganized point set. Geometry analysis [**78**] and visualization process [**79**] could be done with this input, however the presence of noise, out-layers or non scanned regions could generate problems in the reconstruction and analysis process.

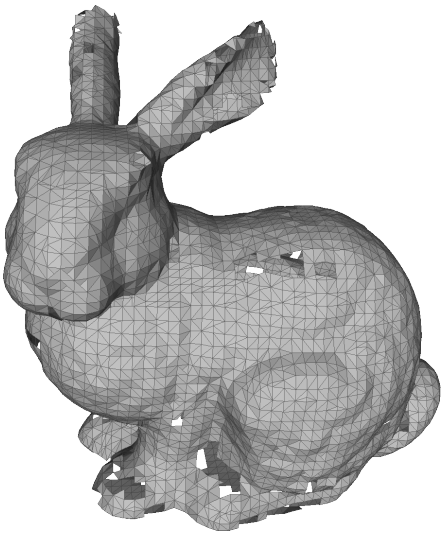


FIGURE 6.1. Reconstructed bunny model using marching cubes in an unsigned distance field with a grid of 50^3 cells with more than 140.000 points representing the surface.

Reconstruction of the surface could be made in several forms, like NURBS, polygonal meshes, implicit or explicit functions. Triangular meshes are the most common form of representation of surfaces. This representation allows an easy extraction of the properties of the surface like curvatures, boundaries, and other characteristics. Also the possibility of extension from this output to visualization process are straightforward with the current graphic hardware.

The contribution of this paper is the extension of Marching Cubes algorithm by analyzing more information inside each cell, for each edge inside the cell the value of the middle point in the edge is calculated as a other vertice. With the new information the intersection points of the surface with the edges of the cell are calculated, and the triangulation case for each cell is selected generating the triangular surface of the 0-level iso-surface.

The remainder of the paper is structure as follows: In section 2 we review previous work in surface reconstruction from point sets and their different approximations. In section 3 we present the methodology of surface reconstruction of our algorithm. And in section 4 we present the results of the algorithm testing different quality measurements in the outputs of the algorithm, conclusions and future work is presented in section 5.

2. Related Work

Surface reconstruction from unorganized point set has been solved with two different and predominant methodologies. The first one is by Voronoi-Delone and by implicit functions, the difference between the methods are the base division of the process. Voronoi-Delone the methods are point centric data calculation process and the triangulation is created following the point set. Implicit functions methodology divide the space and the local characteristics of the surface inside are identified for further rendering with methods like Marching Cubes.

Reconstruction of surfaces with Voronoi-Delone methodologies are explored in [**80**] where a 3D voronoi diagram is created of the point set and a crust is generated. [**81**] a medial axis transform is calculated form the point set and the surface is generated following the distance to the medial axis.

Approximation made by [**82**] which for each point the normal direction is calculated using the neighbourhood of the point and then calculating the tangent plane in that point with a least squares algorithm. A consistent plane orientation for all the points should be made assuring that all the point faces to the same direction which is an NP problem. Finally a signed distance function is generated and a triangular mesh is generated with the marching cubes algorithm. This methodology is improved in the work of [**83**].

Marching cubes [**64**] is an algorithm to generate triangular meshes from volumetric data. This algorithm defines a type of triangulation for each cell in the volume or space given the values of the vertices, given a desired iso-value the vertices are compare against this value and then a triangulation case is selected (Fig. 6.2) and then a triangulation is generated for each cell. The algorithm has been widely studied [**65**] and several extensions had been made to solve topological problems with techniques [**84**], [**85**] also methodologies to obtain more accurate representations were studied [**86**]. One of the biggest advantages of the algorithm is their complexity $O(n)$ where n is the number of cells in the volume, so the time taken by the algorithm to the process could be easily controlled in real time process because it is constant and can be modified changing the number of cells.

The distance field generated from the set of points represents a challenge for the marching cubes algorithm because the distance is always a positive value, and the surface which represents the points are defined in the isovalue 0 which can not be represented without the presence of negative values in the field. To solve this problem a signed distance field must be generated where, for each point in the set, the normal of the surface is calculated, and must be congruent in all the surface. This process is very complex and we avoid this step in our work.

Previous approximations of modified marching cubes algorithm from point sets has been made. [**87**] which uses the vertices as the places where the surface pass, to obtain better geometry approximation the vertices are translated to the nearest point, and a new marching cubes cases are formulated where the surface is in the vertices. [**88**] creates an offset surface around the points and then a cut between the offsets represent the surface of the points.

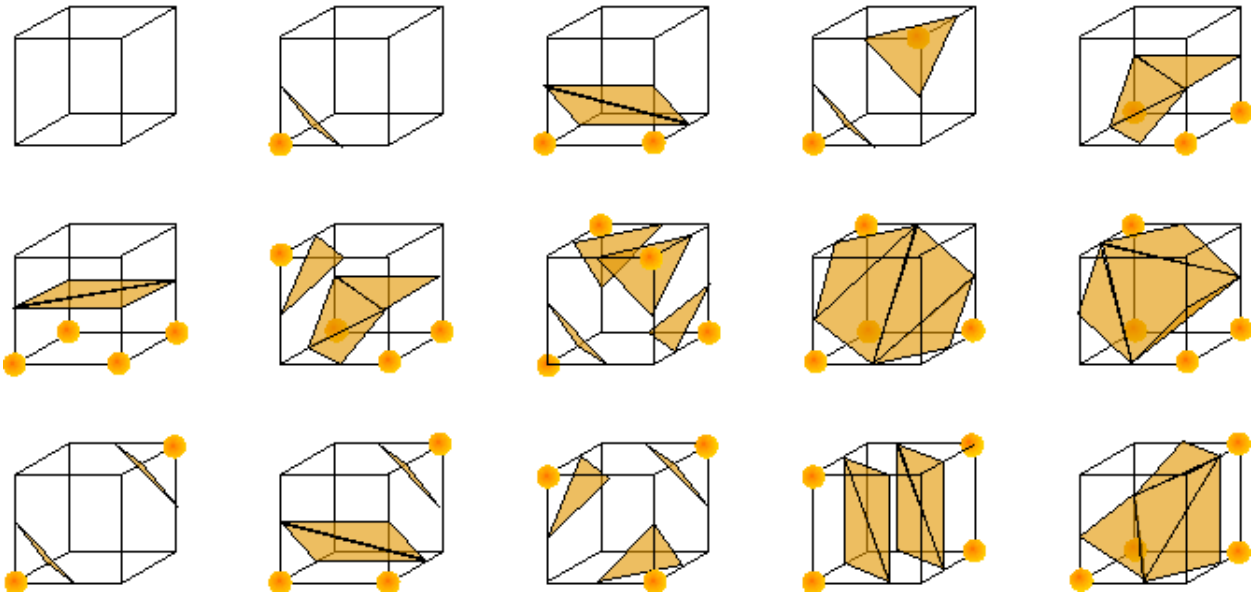


FIGURE 6.2. Marching Cubes cases.

3. Metodology

The problem as defined by [**82**] takes as input an unorganized point set $\{x_1, \ldots, x_n\} \subset \mathbf{R}^3$ on or near an unknown manifold $\mathbf{M}$, and reconstruct a manifold $\mathbf{M}'$ which approximates $\mathbf{M}$. No information about the manifold $\mathbf{M}$ as the presence of boundaries or geometry of $\mathbf{M}$ are assumed to be knew in advance.

Given a set of unorganized points $P = \{p_1, p_2, \ldots, p_n\}$ which are on or near to a manifold δM, our goal is to construct a manifold $\delta M'$ which are close to δM. Each point of P is represented by a 3-*tuple* of real values. The manifold $\delta M'$ is represented as a set of planar triangles of $\mathbf{R}^3$

The algorithm is subdivided in the following steps:

(1) Bounding box B calculation of P
(2) Generation of a cubical grid G inside B
(3) Calculate the minimum distance for each vertex $v_{i,j,k} \in G$ to P
(4) Calculate the intersection points I in the edges of the cells of G
(5) Identify for each cell of G the marching cube case which match the intersection points I
(6) Generate the simplexes of each cell of G given the calculated case.

3.1. Bounding box calculation. The calculation of the bounding box B could be made following the word axis identifying the minimum and maximum values of (x, y, z) in P. There is no a exact rule for the creation of the bounding box, which can be parallel to the global axis, rotated or static. The bounding box define the region in the space where the triangular mesh is going to be generated.

3.2. Cubical Grid generation. Inside the region defined by B a cubical grid G is defined, the number of cells that are inside G define the resolution of the surface, but a high number of cells without a proper density of points defining the case inside each cell could generate ambiguities in the reconstruction process. also the number of cells defines the level of detail of the object. For each cell the vertices of the grid are define as v, also a number of auxiliary vertices in the middle of each edge of the cell must be created v_{aux}.

3.3. Distance calculation for each vertex. For all the vertices v and v_{aux}defined in G the minimum distance from the vertex to the point cloud should be calculated (Fig 6.3). Given the regular form of the grid

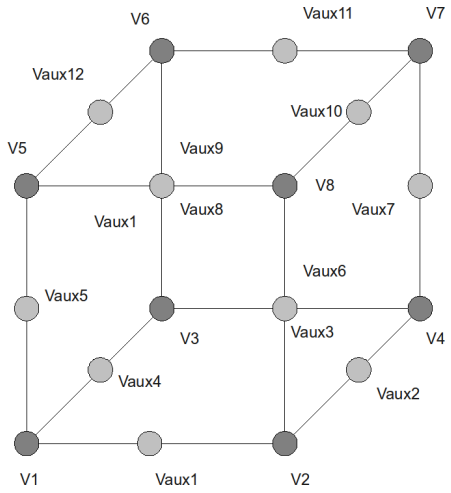


FIGURE 6.3. Vertices v and auxiliary vertices v_{aux} for each cell.

G for each point in P the nearest vertex could be calculated in $O(1)$ time, then the near vertex is assigned with the value calculated, if various points are near to the same vertex v then only then minimum distance is stored.

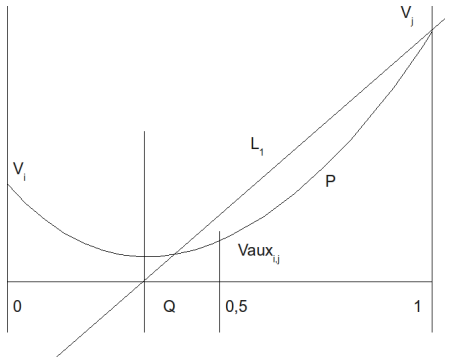


FIGURE 6.4. Parabola showing the intersection point q for the unsigned distance field as a comparison between the original distance function L_1 and a parabola p which values v_i, v_j, v_{aux} represent the distance to the surface.

3.4. Intersection point calculation. For each edge e of the cells C, we have 3 vertices, v_1, v_2, v_3 where v_1 and v_3 are the vertices of the lattices of the grid and v_2 is the middle auxiliary vertex calculated v_{aux}. Our methodology suppose a plane D where the vertices of the edge are aligned in the x axis of the D plane with the values of $x_1 = 0$, $x_2 = 0.5$ and $x_3 = 1$, and the values of $y_1 = v_1, y_2 = v_2, y_3 = v_3$, which are a distance from vertices to the point set, are values in y axis of D plane. Then we can assume that a normal signed surface which cut the edge forms a line in the plane D, as show in the figure 6.4 but as the distance is a positive value the intersection is presented as a line in the positive region of D. We approximate the intersection assuming the as the intersection point can be expressed as a parabola, which is aligned with the y axis in D. Given the 3 points measured in the plane a parabola p could be calculated solving the equations of 20 given the parameters of p, a, b, c in the equation 21. v_x, v_y are the coordinates of the vertice of the

parabola p as defined in the equation 22, the value v_x gives the position in the edge where the surface cuts the edge and v_y could be assigned as a tolerance value, which define how close is the surface to the edge.

$$
\begin{aligned}
y_1 &= ax_1^2 + bx_1 + c \\
y_2 &= ax_2^2 + bx_2 + c \\
y_3 &= ax_3^2 + bx_3 + c
\end{aligned} \tag{20}
$$

$$
\begin{aligned}
a &= -\frac{-x_2y_1+x_3y_1+x_1y_2-x_3y_2-x_1y_3+x_2y_3}{(x_2-x_3)(x_1^2-x_2x_1-x_3x_1+x_2x_3)} \\
b &= -\frac{y_2x_1^2-y_3x_1^2-x_2^2y_1+x_3^2y_1-x_3^2y_2+x_2^2y_3}{(x_1-x_2)(x_1-x_3)(x_3-x2_)} \\
c &= -\frac{x_3y_2x_1^2-x_2y_3x_1^2-x_3^2y_2x_1+x_2^2y_3x_1+x_2x_3^2y_1-x_2^2x_3y_1}{(x_2-x_3)(x_1^2-x_2x_1-x_3x_1+x_2x_3)}
\end{aligned} \tag{21}
$$

$$
\begin{aligned}
v_x &= \frac{-b}{2a} \\
v_y &= c - \frac{b^2}{4a}
\end{aligned} \tag{22}
$$

3.5. Cell case identification. Given the intersection points for each cell, the marching cubes lookup table is used, to define which case of marching cubes had the same edges intersected, if the case is found then the cell is labelled with that case, if there is not case which is defined by the marching cubes look up table, then this cells had an ambiguity case. These cases are produced because the points around the cell had a lot of noise or there is no enough density of points to identify the triangulation in that region. At the time we don't make any processing to solve this ambiguity and that cells are simply are assigned as empty, but global process could be done to eliminate this ambiguities.

3.6. Triangulation generation. Each cell of the Grid G had been labelled with the marching cubes case which represents. Then following the rules of generation of triangles or simplexes of marching cubes. The number of cases which marching cubes gives is 14 in their original definition, because the number of possible combination of edges is $2^8 = 256$ but eliminating redundant cases with rotations or mirrors is only 14 cases. With out approach the number of cases is equal to the number of possible intersections of the edges $2^{12} = 4096$, where we map to the 14 cases of marching cubes, as explained before there is some configurations which gives ambiguity forms this cases are not generated in our algorithm.

4. Results

Our algorithm generates triangular mesh which is a good representation of the surface defined by the set of points. The mesh is not created to be correct in a topological level because the original marching cubes algorithm used to generate the triangulation have topological errors, and the process of calculation is very sensitive to noise or poorly density regions of points. The mesh is generated in a fast way. Even though the results represent a qualitative good representation of the surfaces. This algorithm focus in the problem of generation of the surfaces without the calculation of the normal direction of the surface extending the uses of the Marching Cubes algorithm for unsigned implicit functions in the representation of the isolevel 0.

FIGURE 6.5. Reconstructed elephant model with a grid of 200^3 cells.

5. Conclusions and Future Work

We present an extension of the marching cubes algorithm for the extraction of the 0-level iso-surface in an unsigned distance field. The generated meshes present a lot of topological mistakes because the algorithm can not identify correctly the marching cubes cases where the edges in the cell are intersected with a different pattern as the marching cubes cases present. New set of marching cubes cases have to be generated to correct implement the triangulation step.

The algorithm could represent a big part of the surface without calculating the normal, eliminating an costly step commonly used in other algorithms opening a new method for reconstruction for non-orientable surfaces, and because the local nature of the algorithm it can process without problems surfaces with holes or boundaries.

The quality of the meshes without considering the errors produce by the ambiguities are the same as the marching cubes original algorithm. For further improvements in the quality of the meshes could be applied using the different techniques developed for marching cubes algorithm because the modification of our extension to the original algorithm is minimal.

Part 3

Direct Volume Rendering

CHAPTER 7

Interactive visualization of volumetric data with WebGL in real-time

- John Congote[1,2]
- Luis Kabongo[2]
- Aitor Moreno[2]
- Alvaro Segura[2]
- Jorge Posada[2]
- Oscar Ruiz[1]

[1] CAD CAM CAE laboratory, Universidad EAFIT
Carrera 49 No 7 Sur - 50, Medellín, Colombia
[2] Vicomtech Research Center
Donostia - San Sebastián, Spain

Context

Interactive visualization of volumetric data with WebGL in real-time. CONGOTE, John; KABONGO, Luis; MORENO, Aitor; SEGURA, Alvaro; POSADA, Jorge; RUIZ, Oscar. Sixteenth Annual International Conference on 3D Web Technology (2011) , Paris, France. June 20-22, 2011, pages 137-146, ISBN 9781450307741. CONFERENCE

Volume rendering is a methodology for the rendering of scalar fields, very common in the fields of medical image and geo-sciences. Vicomtech Research Center and CAD/CAM/CAE Laboratoy supports the study of this methodology. Interesting results were obtained from this research and the application of the algorithms in low power computational devices were demonstrated.

This work was partially supported by the Basque Government's ETORTEK Project (ISD4) research programme and CAD/CAM/CAE Laboratory at EAFIT University and the Colombian Council for Science and Technology –COLCIENCIAS–.

As co-authors of such publication, we give our permission for this material to appear in this document. We are ready to provide any additional information on the subject, as needed.

Prof. Dr. Eng. Oscar E. Ruiz
oruiz@eafit.edu.co
Coordinator CAD CAM CAE Laboratory
EAFIT University, Medellin, COLOMBIA

Dr. Jorge Posada
jposada@vicomtech.org
Vicomtech Research Center
Donostia - San Sebastian, SPAIN

Dr. Luis Kabongo
lkabongo@vicomtech.org
Vicomtech Research Center
Donostia - San Sebastian, SPAIN

Dr. Aitor Moreno
amoreno@vicomtech.org
Vicomtech Research Center
Donostia - San Sebastian, SPAIN

Eng. Alvaro Segura
asegura@vicomtech.org
Vicomtech Research Center
Donostia - San Sebastian, SPAIN

Abstract

This article presents and discusses the implementation of a direct volume rendering system for the Web, which articulates a large portion of the rendering task in the client machine. By placing the rendering emphasis in the local client, our system takes advantage of its power, while at the same time eliminates processing from unreliable bottlenecks (e.g. network). The system developed articulates in efficient manner the capabilities of the recently released WebGL standard, which makes available the accelerated graphic pipeline (formerly unusable). The dependency on specially customized hardware is eliminated, and yet efficient rendering rates are achieved. The Web increasingly competes against desktop applications in many scenarios, but the graphical demands of some of the applications (e.g. interactive scientific visualization by volume rendering), have impeded their successful settlement in Web scenarios. Performance, scalability, accuracy, security are some of the many challenges that must be solved before visual Web applications popularize. In this publication we discuss both performance and scalability of the volume rendering by WebGL ray-casting in two different but challenging application domains: medical imaging and radar meteorology.
Keywords: Direct Volume Rendering, Ray Casting, Real-Time visualization, WebGL, Weather Radar Volume, Medical Imaging

1. Introduction

Real-time 3D computer graphics systems usually handle surface description models (i.e. B-Rep representations) and use surface rendering techniques for visualization. Common 3D model formats such as VRML, X3D, COLLADA, U3D (some intended for the Web) are based entirely on polygonal meshes or higher order surfaces. Real-time rendering of polygon models is straightforward and raster render algorithms are implemented in most graphics accelerating hardware. For many years, several rendering engines, often via installable browser plug-ins, have been available to support 3D mesh visualization in Web applications.

However, some scientific fields (e.g. medicine, geo-sciences, meteorology, engineering) work with 3D volumetric datasets. Volumetric datasets are irregular or irregular samples of either scalar ($f : \mathbb{R}^3 \rightarrow \mathbb{R}$) or vector ($f : \mathbb{R}^3 \rightarrow \mathbb{R}^3$) fields. For the purpose of this article, we will use the term *volumetric data sets* to refer to scalar fields and will ignore for the time being vector fields. Surface-based raster rendering techniques are obviously not suitable for visualizing such datasets and specific Direct Volume Rendering algorithms are needed, which are not available for the Web. Therefore, our work uses *Volume ray-casting*, which is a common technique in Computer Graphics for volume visualization originally presented in [**89**] and further studied in [**90**]. The rendering is not photo-realistic but shows the important characteristics of the set.

In medical imaging, diagnostic techniques such as computer tomography (CT), magnetic resonance imaging (MRI) and positron emission tomography (PET) produce sets of parallel slices that form a volumetric dataset. Volume rendering is a common technique for visualizing volumetric datasets along with multi-planar reconstructions (MPR). Storage and distribution of these 3D images usually requires a Picture Archiving and Communication Systems (PACS), which normally uses specialized workstation software ([**91**], [**92**]) for interactive visualization ([**93**]). Few solutions exist for regular desktop computers ([**94**]) among others. Weather radar data is also a volumetric dataset. A weather radar scans a volume around it by collecting values in families of circular, elliptical or conical scan surfaces.

Doppler radars sample several physical variables (reflectivity, differential reflectivity, radial velocity and spectral width) for each location of the sky. Radar data is usually visualized for a single slice of the volume, either conical, as in *plan position indicator* (PPI), or planar, as in *constant altitude plan position indicators* (CAPPI). Initial approaches for Web-based Volume Rendering of radar data rely on pre-computed in-server rendering ([**95**]), which normally hinders interaction in the visualization.

WebGL is a new standard for accelerated 3D graphics rendering in the Web that complements other technologies in the future HTML5 standard ([**96**]). Some of the major Web browsers, including Google

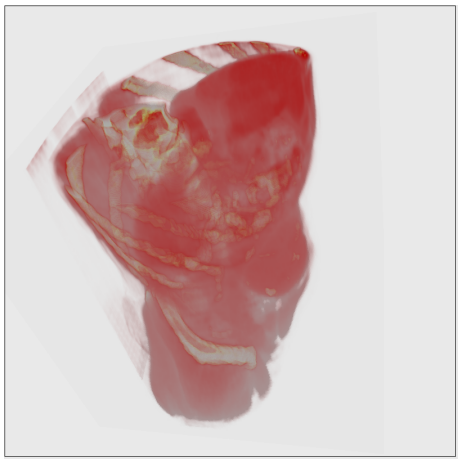


FIGURE 7.1. Medical data rendered with volume ray-casting

Chrome, Mozilla Firefox, WebKit, Safari and Opera have already implemented it in their latest releases or release candidates. WebGL is basically a Javascript binding of the OpenGL ES API and enables low level imperative graphics rendering based on programmable shaders.

1.1. Contribution of this Article. Our contribution is a practical implementation of a volume rendering system for the Web, based on the Volume Ray-casting algorithm and implemented on WebGL. Our system is capable of obtaining interactive visualization with diverse volume datasets (Figure 7.1). The original Volume Ray-casting algorithm was slightly modified to work with the input structures needed for the Web environment. Special care was taken to avoid the use of dynamic server content. This avoidance allows for the algorithm to be used without increasing the demands on the server and shifts, as much as possible, the processing to the client.

Our work is tested in two scenarios with volumetric datasets: medical imaging and radar meteorology. Their main practical difference is the pattern in which the volume is sampled: for medical data a uniform cartesian grid is used. For weather data a non-uniform spherical coordinate grid is used. A side-effect of our work is the proof-of-concept that interactive platform-independent visualization of 3D data on the Web is feasible by means of the WebGL standard.

The paper is organized as follows. Section 2 presents a brief status of the different technologies present in this publication: Volume Rendering, Web rendering, medical and radar visualization. Section 3 presents our methodology for volume rendering with special attention to the modifications of the algorithm for the Web environment. Section 4 presents the output obtained by the implemented algorithm and the performance values in different conditions. Section 6 presents the conclusions of our work and future directions.

2. Related Work

2.1. Direct volume rendering techniques. In 3D scalar field interactive visualization, two solutions prevail: Surface Rendering and Direct Volume Rendering. Surface Rendering has the advantage of being easy to compute due to its low geometric complexity. Its main disadvantages are: (1) A surface must be synthesized first, which is not a trivial task as it depends on the quality of the sample. (2) Since it must be precomputed, the result is static and cannot be easily adjusted in real time.

Recent advances in Direct Volume Rendering and graphic card capabilities allow the representation of volumes with good quality by projecting volumetric data into a 2D image, depending on the position of a virtual camera. The main advantage of this technique is the visualization of all inner characteristics at once.

Preprocessing does not intervene since most of the computations are performed when the camera is displaced. In order to project the volumetric data, several methods exist ([**97**]). Reference [**98**] discusses Volume Splatting and represents each scalar value by a simple geometrical shape that will face the camera, allowing fast rendering. Its main disadvantage is the loss of quality. A technique called *Shear Warping* ([**99**]), consists of applying shear warp transformations to the volume slices to imitate the real orientation of the camera. Since the technique is based on simple transformations this method is quite fast but its main drawback is a low sampling power. With the constant improvement in graphic card capabilities, the *Texture Mapping* method has been popularized in video-games. It consists in re-slicing the volume depending on the orientation ot the camera viewpoint and representing all the slices at once taking advantage of eventual occlusion optimizations ([**100**]).

Volume Ray-casting was initially presented in [**89**] and has become one of the most common methods for volume rendering. The set of rays from the camera reach the 3D scene and hit the objects, generating parametric (scalar) landmark values. By defining a blending function it is possible to give priorities to the different values encountered along the ray, allowing the visualization of different internal structures. Additional modifications to the algorithm, such as *transfer functions*, and *Phong illumination* ([**101**]) were developed in order to improve the perception and make the volume look realistic. Compared to the other techniques, this one is older and more accurate in sampling. However, the computational power required made initially difficult its usage in real-time interactive representations, allowing other approximations to settle. Nowadays, the increasing computational power of graphic cards allows fast calculations ([**102**]) which give new interest to Volume Ray-casting. Reference [**90**] presents a tutorial with all the basic explanation on volume ray-casting. We used this tutorial as starting point for the theoretical foundations in our implementation and for technical details. Open Source implementations such as [**91**] and [**92**] were also used.

2.2. Web 3D rendering. The fact that the Web and 3D graphics are currently ubiquitous in desktop and palm devices makes their integration urgent and important. Several standards and proprietary solutions for embedding 3D in the Web have been devised, such as VRML, X3D or vendor-specific Web browser plug-ins, implementations on general purpose plug-ins, etc. A review of these techniques could be found in [**103**].

In the case of Medical Imaging and other computing-intensive visualization scenarios, a partial solution has been the use of in-server rendering ([**104**]). In this approach, the rendering process is remotely performed in the server and its resulting image is sent to the client. This solution increases the load on the server when many clients are present. In addition, the high latency times make the system non-responsive and unsuitable for smooth interactive visualization.

Outstanding issues among solutions for Web 3D graphics are: dedicated languages, plug-in requirements for interpretation, portability across browsers, devices and operating systems and advanced rendering support. While writing this article, the Khronos Group released the WebGL 1.0 specification, which has been under development and testing. In practice, the WebGL 1.0 is a Javascript binding of the OpenGL ES 2.0 API. Calls to the API are relatively simple and serve to set up vertex and index buffers, to change rendering engine state such as active texture units or transform matrices, and to invoke drawing primitives. Most of the computation is performed in vertex and fragment shaders written in GLSL language, which are run natively on the GPU hardware. Unlike previous Web 3D standards which define declarative scene description languages, WebGL is a low-level imperative graphic programming API. Its imperative character enables a great flexibility and exploits the advanced features of modern graphics hardware.

The WebGL 1.0 standard takes advantage of already existing OpenGL-based graphics applications, such as accurate iso-surface computation ([**105**]) or optimized shader programming ([**106**]). The usage of an interpreted language to manage the behavior of scene elements and animations might be considered as a drawback. However, the performance of Javascript interpreters is constantly improving. Current optimized

just-in-time compilation in the latest engines provides performance not far from that of natively compiled languages.

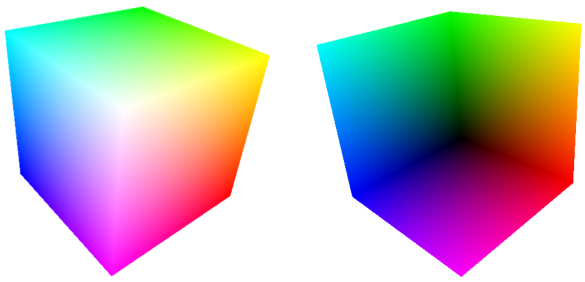
2.3. Medical visualization. From the different scientific fields, Medical Visualization is one of the most challenging since the user interpretation directly translates into clinical intervention. Quality is one of the most important factors, but fast interactive response is also central in this domain. Medical Visualization has already produced some implementations of volumetric visualization in Web, mainly for educational purposes ([**107**][**108**]). These approximations require third party systems for the correct visualization, or the presence of a rendering server ([**109**], [**110**]), which limits the scalability of the application. Using standards such as VRML and Texture Mapping ([**111**]) visualization of volumes in the Web has been achieved.

2.4. Radar visualization. Radar data visualization also poses new challenges as the data are acquired in a spherical coordinate system ([**112**]). This particularity makes difficult the optimization of ray-casting, which usually traverses cubic-shaped volumes. Nevertheless, this problem has already been addressed in [**113**].

3. Methodology

Direct Volume Rendering is a set of Computer Graphics algorithms to generate representations of a 3D volumetric dataset. The produced image is a 2-dimensional matrix $I : [1, h] \times [1, w] \to \mathbb{R}^4$ (w: width and h: height in pixels). A pixel has a color representation expressed by four-tuple (R, G, B, A) of red, green, blue and alpha real-valued components, $(R, G, B, A \in [0, 1])$.

The volume is a 3-dimensional array of real values $V : [1, H] \times [1, W] \times [1, D] \to [0, 1]$ (H: Height, W: Width, D: Depth of the represented volume, in positive integer coordinates). Therefore, $V(x, y, z) \in [0, 1]$. The volume-rendering algorithm is a projection of a 3D model into a 2D image. The projection model used in this work is known as a pin-hole camera ([**114**]). The pin-hole camera model uses an intrinsic $K \in M_{3\times4}$ and an extrinsic $R \in M_{4\times4}$ real-valued matrices. These matrices project a 3D point $p \in \mathbb{P}^3$ onto a 2D point $p' \in \mathbb{P}^2$.



(a) Front Faces (b) Back Faces

FIGURE 7.2. Color cube map coordinates

A volume is normally represented as a set of images. Each image represents a slice of the volume. Usually slices are parallel and evenly-spaced but this is not always the case. For example, volumes can also be sampled in spherical coordinates with the angular interval being variable. Both cases (cartesian and spherical samples) are handled by our algorithm.

Volume ray-casting is an algorithm which defines the color for each pixel (i, j) in the image or projection screen I, calculated in function of the values of a scale field $V(x, y, z)$ associated with the points (x, y, z) visited by a ray originated in such a pixel. The ray is casted into the cuboid that contains the data to display (i.e the scalar field V). The ray is equi - parametrically sampled. For each sampled point p_s on the ray

one computes an approximation of the scalar field $V(p_s)$, by usually calculating a tri-linear interpolation. In addition, a shade might be associated to p_s, according to the illumination conditions prevailing in the cuboid. The color associated to p_s might be determined by axis distances as shown in figure 7.2. As last step, the pixel in the image which originated the ray is given the color determined by the sampled point p_s nearest to the screen, in such a ray.

Alternatively, the samples on the ray may also cast a vote regarding the color that their originating pixel will assume by using a composition function (Eq:23-26), where the accumulated color A_{rgb} is the color of the pixel (i, j), and A_a is the alpha component of the pixel which is set to 1 at the end of the render process. Given an (x, y, z) coordinate in the volume and a step k of the ray, V_a is the scalar value of the volume V, V_{rgb} is the color defined by the transfer function given V_a, S are the sampled values of the ray and O_f, L_f are the general Opacity and Light factors.

$$S_a = V_a * O_f * \left(\frac{1}{s}\right) \tag{23}$$

$$S_{rgb} = V_{rgb} * S_a * L_f \tag{24}$$

$$A^k_{rgb} = A^{k-1}_{rgb} + \left(1 - A^{k-1}_a\right) * S_{rgb} \tag{25}$$

$$A^k_a = A^{k-1}_a + S_a \tag{26}$$

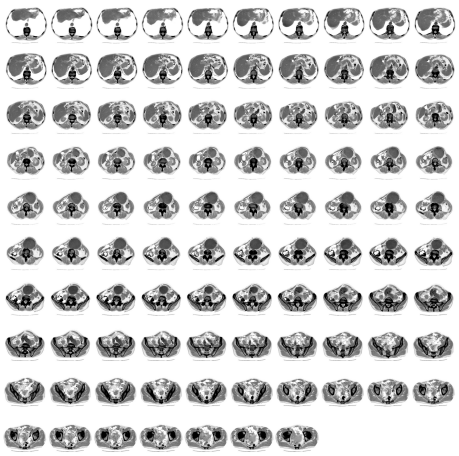


FIGURE 7.3. Aorta dataset in mosaic form to be read by the shader

3.1. Data Processing and volume interpolation.

3.2. Contribution. The images for one volume are composed into a single image containing all slices that will be stored in a texture as shown in Figure 7.3. This texture is generated by tilling each slice one besides other in a matrix configuration, this step was implemented as a preprocessing step in our algorithm. The size of the texture in GPU memory could change from 4096×4096 in PC to 1024×1024 for handheld devices. The reduction in quality in the image is explained in Figure 7.7. The number of images per row and the number of rows as well as the total number of slices must be given to the shader.

In medical images the sample bit depth is commonly bigger than 8 bits. This is hard to handle in Web applications where commonly supported formats are limited to 8 bits per sample. In this work, medical data sets were reduced to 8 bits.

Higher depths could be supported using more than one color component to store the lower and higher bits of each pixel but this representation is not currently implemented in our shader.

$$s_1 = \text{floor}(z * S) \tag{27}$$

$$s_2 = s_1 + 1 \tag{28}$$

$$dx_1 = \text{fract}(\frac{s_1}{M_x}) \tag{29}$$

$$dy_1 = \frac{\text{fract}\left(\frac{s_1}{M_y}\right)}{M_y} \tag{30}$$

$$dx_2 = \text{floor}(\frac{s_2}{M_x}) \tag{31}$$

$$dy_2 = \frac{\text{fract}\left(\frac{s_2}{M_y}\right)}{M_y} \tag{32}$$

$$tx_1 = dx_1 + \frac{x}{M_x} \tag{33}$$

$$ty_1 = dy_1 + \frac{y}{M_y} \tag{34}$$

$$tx_2 = dx_2 + \frac{x}{M_x} \tag{35}$$

$$ty_2 = dy_2 + \frac{y}{M_y} \tag{36}$$

$$v_1 = \text{tex2D}\,(tx_1, ty_1) \tag{37}$$

$$v_2 = \text{tex2D}\,(tx_2, ty_2) \tag{38}$$

$$V_a(x, y, z) = \text{mix}\,(v_1, v_2, (x \times S) - s_1) \tag{39}$$

For the correct extraction of the value of the volume two equations were implemented. The equations 27-39 show how to get the value of a pixel in coordinates x, y, z from images presented in an cartesian grid. s is the total number of images in the mosaic and M_x, M_y are the number of images in the mosaic in each row and column as the medical dataset show in Figure 7.3.

The functions presented in the equations are defined by the GLSL specification. This allow us to manipulate the images as continuous values because the functions of data extraction from the texture utilize interpolation.

3.2.1. *Identification of Ray coordinates.* The geometry of a cube is generated with coordinates from $(0, 0, 0)$ to $(1, 1, 1)$. This cube represents the boundary of the volumetric dataset and is painted with colors representing the coordinates at each point x, y, z coordinates (Figure 7.2)are stored in the r, g, b color component of each pixel. The cube is then rendered in the scene from the desired view point. The rendering process has several steps. The first two steps are the rendering of the color cube with the depth function change. Then one of the passes presents the closest region of the cube to the camera (Figure 2(a)), and the second pass presents the far region (Figure 2(b)).

With these two renders a ray is calculated from each point in the cube for the render with the faces closest to the eye and the end of the ray with the point of the back region. The colors in the cube represent the exact coordinates of the ray for each pixel in the image. We store the color information of the cube as 24 bit RGB values. This range of values seems to be small and not precise enough for big images, but color interpolation gives enough precision for the ray coordinates.

Cartesian coordinates Most voxel-based volume datasets are arranged in a cartesian uniform grid. A medical CT or MRI scanner, computes parallel slices of the specimen at different positions with a normally

constant spacing. Each image contains a matrix of samples of relative to the specific signal measured by the equipment. By stacking all slices aligned together, a discretely sampled volume is defined. Each sample can be addressed by cartesian x, y, z coordinates, one being a slice selector and the other two coordinates of a point in that slice image.

Spherical coordinates A weather radar scans the surrounding sky in successive sweeps. Beginning at a low angular elevation, the radar performs a 360° azimuth scan (Figure 7.4, fig:radar). At each one-degree space direction a ray is emitted and a number of samples along the ray are measured back from its echoes (400 samples or buckets in our case). The radar then proceeds step by step increasing elevation at each successive swept scan. Elevation angles are not normally uniformly incremented because most interesting data is at the lower levels. Our datasets use 14 such elevations.

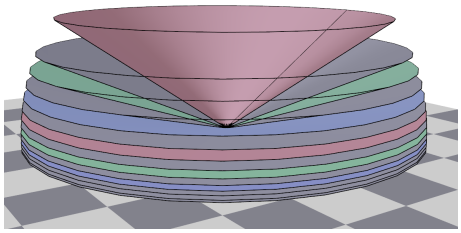


FIGURE 7.4. Simplified geometry of a radar scan. Each scan can be approximated as a cone. Therefore, a radar volume dataset is approximated as a set of co-axial cones with the radar in the common apex.

Such a process results in a discrete sampling of the sky volume in which each sample has *elevation*, *azimuth* and *range* coordinates. Thus, samples can then be addressed by spherical coordinates. In the conversion of raw radar data into input images suitable for the WebGL implementation the sample values become pixel values. Each swept scan for a fixed elevation angle forms one image in which pixel columns correspond to each azimuth direction (there are 360 columns), and rows correspond to distances along each ray (400 rows). Each image maps to a conical surface in space as shown in figure 7.4. The images from consecutive elevations are joined to form a single image for the whole volume, presented in figure 7.5 (the figure is rotated 90°).

$$r = \sqrt{x^2 + y^2 + z^2} \tag{40}$$

$$\varphi = \arctan(y, x) + \pi \tag{41}$$

$$\theta = \arccos(z/\varphi) \tag{42}$$

For the spherical coordinates volume dataset from a radar the following Equations 40-42. Where used. The interpolation process used for the identification of the volume value in an arbitrary point is presented in Equations 27-39. We use a simple interpolation method because the data is expected to be normalized from the capture source. The problems presented in this topic were explained by [**115**].

3.2.2. *Ray generation.* The ray is generated for each pixel in the image I, geometrically the start and end positions of the ray are extracted from the previous render passes with the information of the color cube. The ray is divided by S steps, which indicates the number of samples of the volume. For each sample the x, y, z inside the volume is calculated and the value of that position is interpolated from the texture.

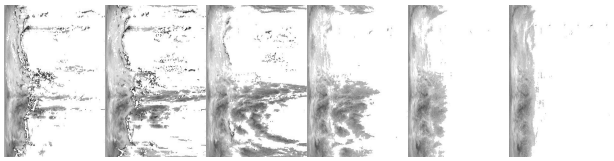


FIGURE 7.5. Original Doppler radar image. Each vertical band represents data along the cones described in Figure 7.4

3.2.3. *Transfer function.* This value of the texture $t_{x,y,z}$, is then used to identify the color to be used in the composition function (Eq:23). When the composition function reaches the end of the ray in the cube or the accumulated alpha A_a reaches its maximum, the ray is interrupted and the resulting color A_{rgb} for the ray in the corresponding pixel is the cumulated value.

4. Results

The proposed GPU implementation of the Volume Rendering technique presented in the previous sections has been tested with different settings and with different datasets. As the interactive and real-time results depend on both hardware and software, it is very important to begin with the platform specification used for the testing. In the following sections, medical volumetric datasets and weather radar volume samples are used to validate that WebGL is a valid and promising technology for real time and interactive applications.

4.1. Hardware and Software configuration. The tests for this article have been conducted using an Intel Quad Core Q8300 processor, 4GB of RAM and a GeForce GTX 460, Windows 7 PRO 64 bits (Service Pack 1) with the latest stable graphics drivers. Among all the Web browsers with full implementation of WebGL standard, we have selected the following ones: FireFox Minefield 4.0b12Pre (2011/02/16)[1], Chrome 9.0.597.98[2] and Opera 11.50 labs (build 24661[3].

It is worth to point out that both Chrome and Firefox in its default configuration use Google's Angle library [4] to translate WebGL's native GLSL shaders to Microsoft's HLSL language and compile and run them through the DirectX subsystem. This procedure improves compatibility with lower-end hardware or older graphics drivers. Firefox Minefield has been configured with two different settings by modifying some keys in the configuration page *about:config*: (1) the default value of *webgl.prefer-native-gl* was set to `TRUE`. (2) The default value of *webgl.shader_validator* was set to `FALSE`. These changes basically disable Angle as the rendering back-end end validator of shaders, thus directly using the underlying native OpenGL support. See Table 7.1 for terminology precisions.

A LightTPD Web server[5] was installed and configured in the same computer, to serve the dataset images, the sample webpages (HTML and Javascript files) and the vertex and fragment shaders.

4.2. Medical Dataset. Figure 7.6 shows some graphical output for the medical dataset introduced in the previous section. The 6 different axial views have been generated using 80 steps in the shaders implementation (800×800 canvas rendered in the *FirefoxDX* configuration). Table 7.2 displays the recorded statistics: Column 1: Browser configuration. Column 2: Number of steps selected in the shaders. Column

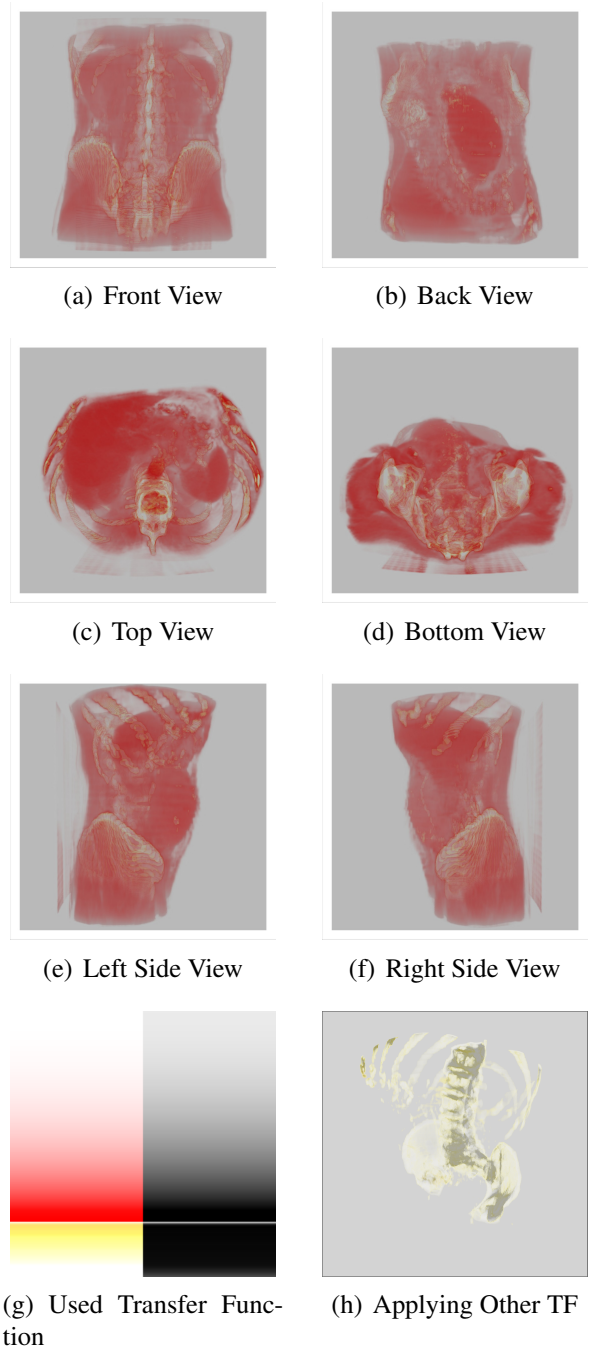
[1]http://ftp.mozilla.org/pub/mozilla.org/firefox/nightly/2011-02-16-03-mozilla-central

[2]http://www.google.com/chrome

[3]http://snapshot.opera.com/labs/webgl/Opera_1150_24661_WebGL_en.exe

[4]http://code.google.com/p/angleproject

[5]http://www.lighttpd.net



(a) Front View (b) Back View

(c) Top View (d) Bottom View

(e) Left Side View (f) Right Side View

(g) Used Transfer Function (h) Applying Other TF

FIGURE 7.6. Subfigures (a), (b), (c), (d), (e) and (f) illustrate renderings of the axial views of the sample volume dataset. The output was generated in 800×800 with 80 steps using FirefoxDX. Subfigure (g) depicts the applied transfer function, where the left side represents the color and the right side the transparency (black=opaque, white=transparent). With different transfer functions other outputs are obtained, as subfigure (h) shows.

Terminology	Meaning
Chrome	Chrome 9.0.597.98
Opera	Opera 11.50 Labs b24661
FirefoxDX	Firefox Minefield 4.0b13pre with Angle and shader validation
FirefoxGL	Firefox Minefield version without Angle nor shader validation

TABLE 7.1. Terminology specification.

Browser	N. Steps	Load Time (msec)	frame rate (frame/sec)	Memory (MB)
FirefoxDX	80	16677	69	204
FirefoxGL	80	308	94	102
Chrome	80	18062	77	153
Opera	80	82	108	74
FF-nA	140	302	60	95
Opera	140	118	66	73
FirefoxGL	170	281	51	95
Opera	170	102	54	77
FirefoxGL	200	312	44	96
Opera	200	111	48	81

TABLE 7.2. Results table for the medical dataset.

3: Loading times (milisec). Column 4: Qualitative frame rate (fps). Column 5: Memory usage (MB). The dataset parameters and volume rendering shaders proved to be very influential in the experiment with WebGL rendered volume datasets. In order to realize the tests under comparable conditions, the web-browsers were stopped and their caches emptied after each test execution. For numbers of steps larger than 80 *Chrome* and *FirefoxDX* failed to compile the shader. Therefore, only *Opera* and *FirefoxGL* statistics are shown. For numbers of steps smaller than 80, the measured FPS was truncated to 128 in all configurations due to the selected measurement method.

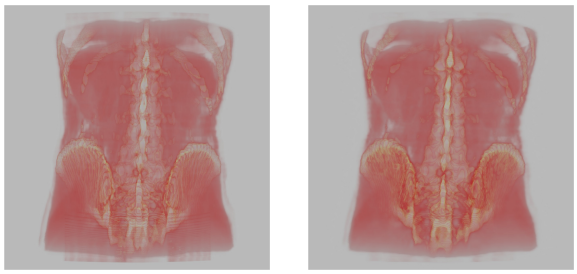
4.2.1. *Memory Usage.* Regarding memory usage, *Opera* showed to be the least demanding browser, being *FirefoxDX* the most consuming one. *FirefoxGL* dropped the memory usage resembling the one of *Opera*. This fact leads us to infer that the current Angle implementation is the key factor in the memory management since *Chrome* has consumption similar to *FirefoxDX*. For *Opera*, *FirefoxDX* and *FirefoxGL*, a simple subtraction between the final and initial memory allocation suffices to estimate the memory consumption. On the other hand, since *Chrome* implements the *Out of Process* technology, we have included *all* running processes of *Chrome* (2 processes at the starting time and 3 processes after WebGL was loaded).

4.2.2. *Loading Times.* The loading time includes the (1) downloading of all the required files (assuming a locally installed Web server), (2) compilation of the shaders and (3) first rendering of the webpage, including the volume rendering. The results follow a similar schema in which *Opera* and *FirefoxGL* are significantly faster than *Chrome* and *FirefoxDX*, due to the longer compiling time and shader validation of Angle.

4.2.3. *Frame Rate.* Since the frame rate cannot be precisely determined, we have devised an empirical measuring method. We forced the GL canvas to be redrawn continuously and then we have counted how

many times the scene was rendered in a 5-seconds interval. We have repeated this procedure 5 times, choosing the median value (i.e. after removing the two highest and the two smallest values) as the effective frame rate. The results show that the frame rate is truncated to 128 fps at maximum (not shown in Table 7.2). This is considered to be a side effect of the chosen measurement method, based on the Javascript `setTimeout()` function. Even with a parameter of 0 ms the browsers take a minimum time to call the corresponding function, being it 10 ms in average for desktop Web browsers. Therefore, it is preferable to increase the number of steps in order to get smaller frame rates and reduce this side effect. With higher values of steps (only usable with *Opera* or *FirefoxGL*) *Opera* is slightly faster, consuming less memory and requiring smaller loading times.

4.2.4. *Dataset Resolution.* This qualitative test was intented to show how the input dataset resolution affects the final rendering quality. Using the same dataset, a modified version was created by reducing the input resolution per slice from 5120×5120 to 1280×1280 (a reduction to 25% per dimension or to 6.25% globally). The number of steps in the shaders were also varied, using 20, 50 and 80 steps with *FirefoxDX* setup and a selection of the results can be shown in Figure 7.7. If the number of steps is small the banding artifacts of the algorithm are noticiable, some aproximations could be implemented to solve this problem as show by [**106**].



(a) 100% - 80 steps (b) 25% - 80 steps

(c) 100% - 20 steps (d) 25% - 20 steps

FIGURE 7.7. Resolution qualitative test. Cases (a) and (b) use 80 steps in the rendering. Cases (c) and (d) use 20 steps. Cases (a) and (c) correspond to the full resolution dataset. Cases (b) and (d) correspond to the reduced dataset. Even with the dramatic reduction of the resolution, the volume render allows to identify the main structures.

4.3. Medical Dataset in Portable Devices. The Mozilla Firefox Development Group has released a mobile version of the browser for ARM devices called Fennec[6]. We have tested it on 2 Android-based

[6]http://www.mozilla.com/en-US/mobile

devices: Samsung Galaxy Tab[7] and Samsung Galaxy S smartphone[8]. Taking into account the hardware limitations of such devices, we have scaled down the *Aorta* dataset to half resolution, reduced the HTML canvas size and chosen a suitable number of steps to get quality results with the highest possible interactivity. The test under this browser was quite straight-forward. No further modification in the implementation of the shaders, Glue Javascript code or HTML Web page were required. Although we achieved a low frame rate (about 2 or 3 frames per second), it was proved as possible the rendering of volume datasets in such mobile devices. Further optimizations in the data or the implementation of the shaders, specifically oriented to such devices, might result in better overall performance. We left such issues for future efforts.

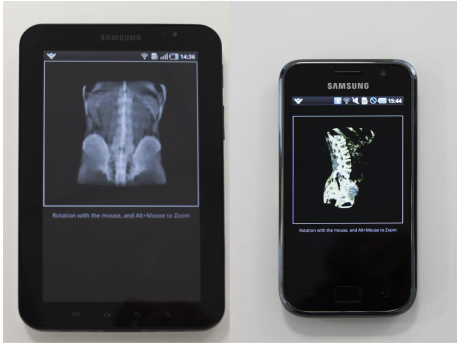


FIGURE 7.8. A Samsung Galaxy Tab (left) and a Galaxy S Smartphone (right) volume - rendering medical datasets.

4.4. Weather Radar Volumetric Dataset. Weather radars are devices used to scan the surrounding atmosphere and determine its structure and composition, typically using the Doppler effect ([**115**]). Radar scans are represented as 2D images in the form of either PPI (plan position indicator) or CAPPI (constant altitude PPI) formats. The volumetric radar-scanned data may be approximated by a set of concentric cones (figure 7.4), with each cone containing a sample set of the volume (figure 7.5). The volume-rendering algorithms, therefore, must work with spherical coordinates for this purpose. This implementation of the volume rendering can only be tested under *Opera* and *FirefoxGL*. Otherwise the shader compilation fails.

[7]http://galaxytab.samsungmobile.com/2010/index.html
[8]http://galaxys.samsungmobile.com

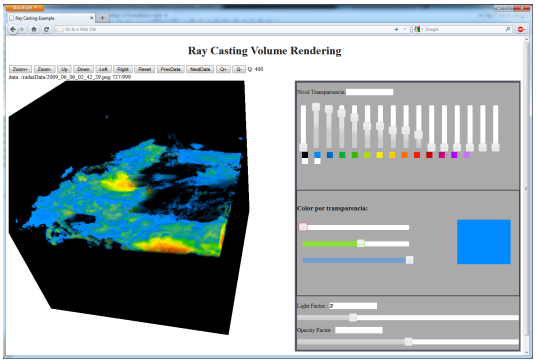


FIGURE 7.9. Application to define the transfer function for radar reflectivity visualization.

In the case studies we have created a simple HTML user interface using jQuery (Figure 7.9) to interact with the shader. It allows the tunning of parameters such as the window (zoom and offset), quality (number of steps) and the transfer function (adapted specifically for this weather radar information). The zoom and pan allow the users to conveniently interact with the radar data, which is not as regular as the medical images. For instance, the useful information is found in the bottom of the volume (i.e. near the terrain). In addition, the resolution in outer zones is lower than near the radar source. Due to their geometrical configuration, the large area over radars is rarely scanned. Therefore, additional navigation controls for zoom and pan have been implemented in the sample Web page, allowing in- and out- zooming in the volume and panning the view. The radar captures periodic scans every 10 minutes. Therefore, some navigational functionality has been added to help users to access the next or previous data in the sequence. As the loading time is rather short, the possibility to create fully interactive 4D animations is totally open.

A very simple HTML-based editor for transfer functions was implemented, which allows the proper analytic inspection of the radar data by changing the values and colors with the provided sliders. Figure 7.10 shows different visualization of the same radar sample, obtained by changing the transfer function (affecting colors and opacity). The figure also shows the effect of variations in camera parameters (zoom, pan and view orientation). The chosen number of steps was high enough to display a 800×800 canvas with high quality images and yet to keep the visualization frame rate above 26 frames/second.

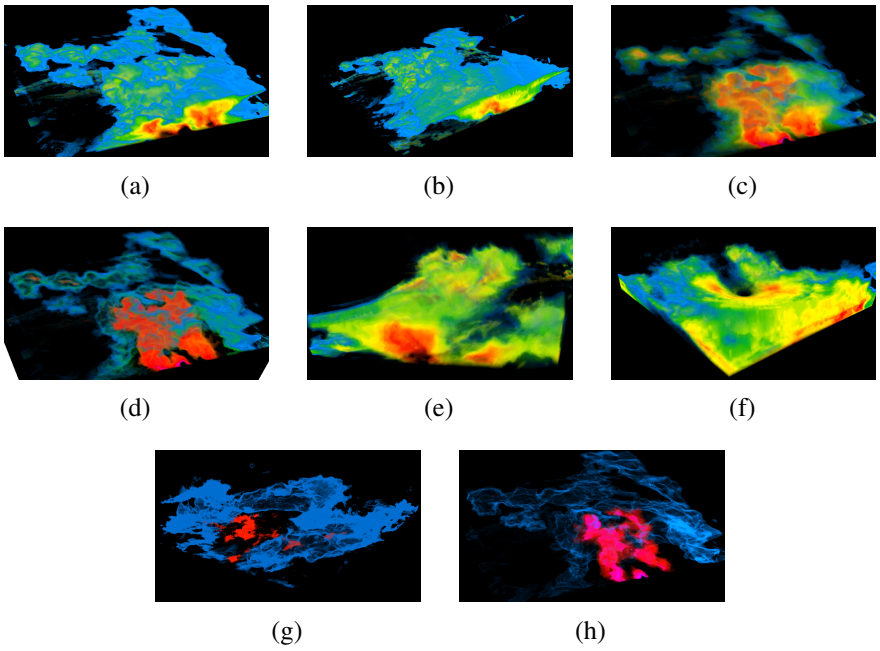


FIGURE 7.10. Different weather radar volume renderings. Images (a) and (b) use the traditional color mapping for reflectivity scans (measured in decibels, dBZ). Images (c), (d), (e), (f), (g) and (h) have been generated by varying the transfer function (color and transparency) and the window zoom and pan.

5. Contribution and Complexity Analysis

Our contribution is an implementation of a volume rendering system for the Web. The system is based on the Volume Ray-Casting algorithm with a complexity of $O(M * S)$, where M is the number of pixels

to be drawn and S is the number of steps of the ray that traverses the volume. Since the algorithm is implemented in WebGL, its visualization speed is similar to native applications because it uses the same accelerated graphic pipeline. The original algorithm has been slightly modified to work with the input structures because of the lack of Volume Textures in WebGL. Therefore, our algorithm simulates the 3D data by using a 2D tiling map of the slices of the volume maintaining the tri-linear interpolation, so there is not a real loss in quality because the interpolation is the same as the used in the GPU. Even thought a slight impact in performance could be generated for this interpolation, this is minimal and very difficult to perceive because the browsers are not capable of handle such fast events. This is so because the browsers are heavily dependent on several layers such as the shader compilers, hardware architecture, graphic drivers, etc. Our algorithm was designed to run entirely in the client (which is the novelty of our proposal). Some delays are obviously expected because of the network performance, and the interpreted nature of Javascript. Our implementation[9] does not evidence a real overhead for the server to present the data in 3D as occurs in [**93**], therefore allowing more clients to be connected simultaneously. At the same time, more powerful clients are required to handle this approximation.

The limitations in our method, even been WebGL compilant, stem from the fact that some browsers do not adequately provide powerful enough shader language implementations to even allow compilation of larger shader programs.

6. Conclusions and future work

Two different case studies (medical- and weather radar-imaging) presented here illustrate the capabilities of complex volume rendering visualization in Web browsers. Although many performance improvements and optimizations are still needed, the material discussed indicates that rendering volumetric data with Web standard technology is applicable to many other technical fields. Such an initiative also re-ignites interest for visualization functions implemented in the past for high-end desktop visualization applications. The integration of our implemented software in the Web follows the upcoming HTML5 standard, namely a Javascript API and the new WebGL context for the HTML5 canvas element (which gives the application a professional look). The implementation of the algorithm in declarative languages as X3DOM is planned.

The scope of the present article did not include the integration of different rendering styles. However, interactive and complex lighting integration are promising ways to improve render quality. The use of multi-dimensional interactive transfer functions is also an promising direction to explore. The minor optimizations that we already applied in this work allow us to expect that mathematically-planned negotiation between speed performance and quality is a promising research field. An additional goal for minimization is the optimized handling of time-varying datasets using videos instead of images as render input. since video formats already minimize transmitted data by reducing temporal redundancy, it would be possible to diminish the bandwidth currently required, for example, for animated render of radar-scanned weather phenomena.

Another important evolution will be the integration of surface rendering within volume-rendered scenes in order to visualize, for example, segmented areas in medical images or terrain surfaces. Some tests have already been performed on desktop prototypes. This article lays the technical ground that would make the integration of surface render in volume-render (via WebGL) possible and reasonable.

[9] http://demos.vicomtech.org/volren

CHAPTER 8

Volume Visual Attention Maps (VVAM) in Ray-Casting Rendering

- Andoni Beristan[2]
- John Congote[1,2]
- Oscar Ruiz[1]

[1] CAD CAM CAE laboratory, Universidad EAFIT
Carrera 49 No 7 Sur - 50, Medellín, Colombia
[2] Vicomtech Research Center
Donostia - San Sebastin, Spain

Context

Volume Visual Attention Maps (VVAM) in ray-casting rendering. Beristain, Andoni, John Congote, and Oscar Ruiz.. Studies in Health Technology and Informatics 173 (2012): 53. PMID: 22356956 [PubMed - indexed for MEDLINE], JOURNAL

Volume visualization of medical images present a difficult field of study because the best model of visualization is subjective and depends from the user and the objective. This work present a methodology which interacts with the user and generates a personalize model of visualization for this kind of data.

This research has received the supported of the CAD/CAM/CAE Laboratory at EAFIT University and the Colombian Council for Science and Technology COLCIENCIAS.

As co-authors of such publication, we give our permission for this material to appear in this document. We are ready to provide any additional information on the subject, as needed.

Prof. Dr. Eng. Oscar E. Ruiz
oruiz@eafit.edu.co
Coordinator CAD CAM CAE Laboratory
EAFIT University, Medellin, COLOMBIA

Dr. Andoni Beristain
aberistain@vicomtech.org
eHealth & Biomedical Applications
Vicomtech, San Sebastian, SPAIN

Abstract

This paper presents an extension visual attention maps for volume data visualization, where eye fixation points become rays in the 3D space, and the visual attention map becomes a volume. This Volume Visual Attention Map (**VVAM**) is used to interactively enhance a ray-casting based direct volume rendering **DVR** visualization. The practical application of this idea into the biomedical image visualization field is explored for interactive visualization.

1. Introduction

Dense volume data visualization on conventional 2D monitors is a complex task due to the superposition of many data layers in the view plane. Many different techniques have been proposed in order to enhance or select different regions or data value intervals in the image, such as direct volume rendering (**DVR**) techniques and the related transfer functions, but it is still an open problem. In the medical field, professionals have more patients and image data, but less time for analysis, so techniques which help them increase the image review quality and efficiency are becoming increasingly relevant.

Visual Attention Maps[**116**], **VAM** from now on, usually visualized as heat maps, depict the focus of the users attention graphically using different values and colors for each location depending on the time the user spent looking at it. This structure is generally constrained to a plane, since the eye tracking process is usually related to the attention on a screen, and limited by the eye tracking system [**117**].

This paper presents an extension of **VAM** for volume data visualization, the Volume Visual Attention Map (**VVAM**), where **VAM** become 3D volumes (see Figure 8.1), as a process to support and enhance the online volume data visualization/analysis procedure in a ray-casting based **DVR**. A software prototype has been developed in order to test the feasibility of the approach, which is able to compute the **VVAM**, and use it to enhance a ray-casting based **DVR** method for medical volume image visualization, in real-time (60 fps), without any noticeable delay.

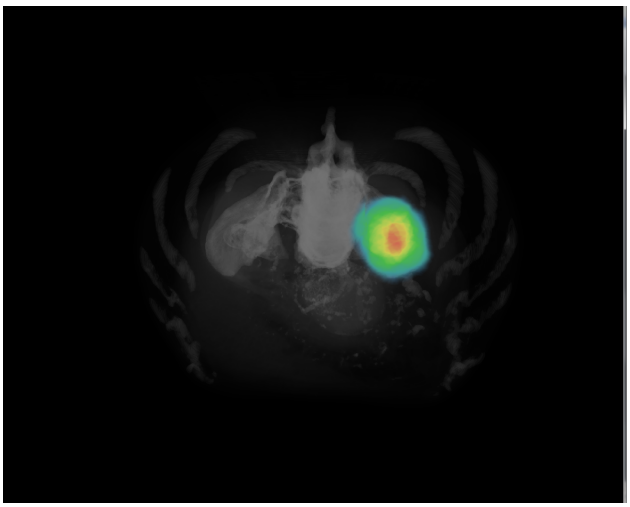


FIGURE 8.1. VVAM on medical image volume

2. Materials

Our prototype comprises several components:

- A commercial, point-of-regard based, non-intrusive eye tracker. For a technical understanding of the eye tracking techniques, the reader is referred to [**117**].
- A **GPU** based ray-casting **DVR** implementation previously presented in [**106**].
- The **VVAM** module implementation in **GPU**, integrated into the **GPU** ray-casting implementation, so that it performs in real time.

Our system allows using the eye tracking interface to interact with the volume visualization in two different ways: on one hand manipulating the virtual camera (zoom and rotate), and on the other hand by computing the volume visual attention map, which in turn guides the real-time **DVR** process to enhance the most salient regions.

3. Methods

When viewing a scene, the eyes perform two kinds of operations called saccades and fixations [**117**]. Fixations are relatively stable eye focus points and correspond to human cognitive processes where we extract visual information that we process. Saccades on the other hand are unconscious rapid eye motions with no relationship to any cognitive process, and usually correspond to the eye motion between consecutive fixations. Therefore, only fixations are considered for the **VVAM** computation. In a conventional VAM a fixation is defined by a 2D coordinate, $p(x, y)$ and the time amount spent t, $Fix_2D(p, t)$.

In **VVAM** we define a fixation as a Gaussian cylinder that crosses the volume data using a projection of the 2D fixation coordinates onto the volume data perspective view. The value of each point inside the cylinder corresponds to a 1D Gaussian function on the distance to the cylinder centerline multiplied by the normalized fixation time. Therefore, it is defined by the 3D coordinates of its centerline ends, $pIn(x, y, z)$ and $pOut(x, y, z)$ the radius, r, the fixation time, t, and the 1D Gaussian function parameters, G, that is, $Fix_3D(pIn, pIut, r, G, t)$.

3D attentional maps were previously introduced by [**118**]. Although, the authors use them for offline attention analysis, and their visualization method was not well suited for ray-casting DVR. As opposed to theirs, our definition naturally deals with the specific characteristics of ray-casting based volume data visualization. Ray-casting represents a blending of many layers, instead of depicting the closest layer only. We simulate this fact by using a volume crossing cylinder. The addition of a Gaussian function, not considered by [**118**], corrects minor accuracy errors in the eye tracking procedure. It also takes into account that eyes do not necessarily focus on only one point, but on a region, and that points closer to the center point have more relevance for the user.

The **VVAM** computation follows the next steps: identifying fixations in 2D, then obtaining the fixation cylinder input and output coordinates, and finally adding the fixation cylinder contribution to the **VVAM**.

There are many different kinds of **VVAM**. For our system we chose the time normalized version, without a forgetting factor. This method allows time varying region selection, and it is more convenient to avoid implementation value overflows and to reduce the computation cost of the fixation cylinder values. After adding each fixation, time measures are normalized, and the whole **VVAM** values are updated. Both, the fixation cylinder computation and **VVAM** update procedures are time consuming, so taking advantage of our **GPU** based ray-casting implementation, we have implemented the **VVAM** computation into an OpenGL fragment shader.

The **VVAM** is provided to the ray-casting fragment shader so that the volume visualization is adapted to the users attention focus, modifying the values provided by the original opacity and color transfer functions. Volume regions with higher **VVAM** values look more opaque and more color saturated, while lower valued regions have a reduced opacity, and are less saturated are represented with higher detail level, using a denser sampling of the volume set along the ray-casting ray (see Figure 8.2). Alternative enhancements are also possible.

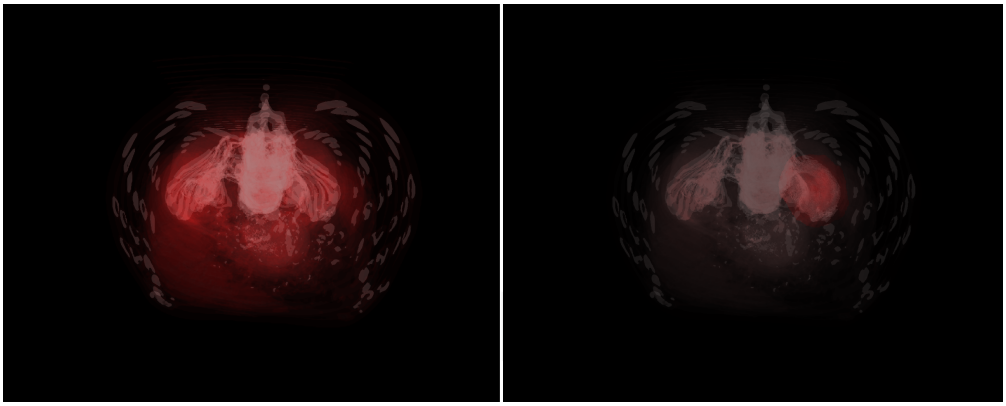


FIGURE 8.2. Left: No VVAM based enhancement. Right: VVAM enhanced visualization.

Figure 8.2 on the left depicts a low detail level **DVR**, where the **VVAM** is still empty, while the picture on the right shows the same perspective of the volume, where the user has focused on a region, forming an ellipsoidal shape. The low quality of the non-focused region has been exaggerated, so that the difference is clearly noticeable in the pictures. Nevertheless, the difference is clearly perceptible on any conventional computer monitor. And Figure 8.1 shows a heat map based **VVAM** representation which corresponds to the **VVAM** in Figure 8.2, right.

Eye tracking is also used to modify the model viewing perspective. The virtual camera can orbit around the volume data and zoom in/out. Currently, the keyboard is used to set the specific camera action to perform, while eye-gaze is used to set the action location (context) and to trigger it.

The interaction sequence consists of using the rotation and zoom controls to look at the regions of interest for the medical professional from different perspectives, so that as time goes by, those volume regions of interest are highlighted and the surrounding data fades away progressively.

In [**119**] an alternative way to use eye-gaze for enhanced ray-casting based DVR is proposed. In that paper the volume data is constantly rotating around its center and the user is asked to keep the focus of attention on the region of interest while this rotation takes place. Afterwards, the visualization parameters are automatically tuned to highlight the regions which were the focus of attention. We consider that our system has several advantages over that one. First of all, our process is completely interactive and online as opposed to theirs, because the user perceives a real-time feedback with the enhancement of the current selection. In addition, we consider that focusing on a 3D region while the volume is rotating is much more difficult than making the user select several view planes, using eye interaction too, and focusing on each of these static views, and even more difficult if the user wants to select several non-connected regions simultaneously.

4. Conclusions & Discussion

We have presented an adaptation of the Visual Attention Maps to the interactive ray-casting **DVR** visualization, which includes the definition of the **VAM** for 3D volumes, denoted as Volume Visual Attention Maps (**VVAM**), and its online usage to enhance or highlight the volume data regions of interest. We have also implemented a prototype to test the feasibility of the approach and its suitability in the medical image visualization field.

We consider that this new technique can increase the effectiveness and efficiency of ray-casting **DVR** visualization, especially in the medical image visualization field. Effectiveness can be improved in two ways: firstly because **VVAM** can be used to track which image regions have been reviewed by the medical professional, and which remain unexplored, to avoid skipping any detail; and secondly because it has a

didactic potential, to teach future medical professionals the right way to review images. Efficiency comes in terms of time, as eye based virtual camera manipulation can be quicker and more natural than using a mouse, and because this natural way to enhance complex regions automatically permits focusing in what is really important for the professional. There is a final benefit in the usability and naturalness of the approach. Our approach is intuitive, since it is grounded in the common behaviour of humans when looking at a scene, which is to focus on what it is really important to the user. In addition, our approach provides the user with real-time feedback of the focus region, which helps in the selection refinement. These two characteristics make the interface natural, in the sense that it has a flat learning curve and a low cognitive load.

Currently the eye based camera interaction requires the use of the keyboard, but we are working on using voice commands for this control. In the same sense, even although this system has been designed for eye based interaction, it could also be used with conventional keyboard and mouse interaction, if the eye tracking device is not available. In this particular case, the concept of **VVAM** would be more related to a complex shape 3D selection technique guided by the mouse cursor position.

Finally, we are considering the migration of this system to a web based application, following the current trend in cloud computing and thin client based solutions for medical image visualization as it is part of our roadmap [**120**].

CHAPTER 9

MEDX3DOM: MEDX3D for X3DOM

- John Congote[1,2]

[1] CAD CAM CAE laboratory, Universidad EAFIT
Carrera 49 No 7 Sur - 50, Medellín, Colombia
[2] Vicomtech Research Center
Donostia - San Sebastián, Spain

Context

Medx3dom: Medx3d for x3dom. Congote, J. (2012, August). 17th International Conference on 3D Web Technology (pp. 179-179). ACM. ISBN 978-1-4503-1432-9. doi: 10.1145/2338714.2338746. CONFERENCE

The Web3D consortium develops standard for the visualization of medical images and a great collaboration between the consortium and the research group for volume visualization integrated by Vicomtech and CAD/CAM/CAE Laboratory was stablish. This work present an integration of the methodologies developed for volume visualization in the standard of MEDX3D of the Web3D.

This work was partially supported by the Basque Government's EMAITEK research programme and CAD/CAM/CAE Laboratory at EAFIT University and the Colombian Council for Science and Technology –COLCIENCIAS–. Also we want to thanks Yvonne Jung for their support and help with the integration of the code into X3DOM project, and Luis Kabongo for his help in the volume styles algorithms.

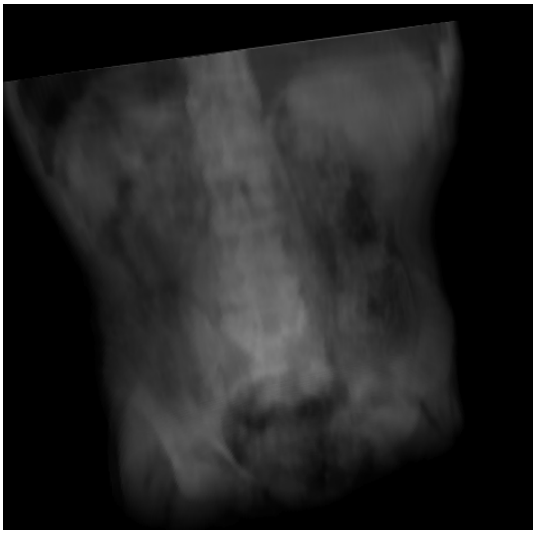


FIGURE 9.1. X-ray style volume rendering with *MEDX3DOM*

Abstract

We present an implementation of *MEDX3DOM* a *MEDX3D* standard implemented into the *X3DOM* framework. The article present the report of a work in progress of the implementation identifying the critical sections to be migrated into the new architecture, and possible extensions of the standard on the Web environment. Results for the early implementation are shown, where the visualization of medical datasets with advance direct volume rendering algorithms is obtain under the X3DOM architecture with interactive framerates and good image quality. An example of the HTML5/X3DOM document is presented and its result is shown. Preliminary results are discussed and future work is presented.

1. Introduction

MEDX3DOM is a work in progress implementation to support advanced medical visualization functionality on the Web without plugins. It is based on the ISO standard *MEDX3D* (Ref. [**121, 122**]) and implemented into *X3DOM* (Refs. [**103, 123, 124**]). *MEDX3DOM* is based on the previous work for volume visualization in WebGL (Refs. [**120, 125**]). *MEDX3DOM* reads medical data in diverse formats such as *DICOM*, *NRRD* and Raw formats. The output is defined by a collection of volume visualization styles for Direct Volume Rendering (*DVR*) algorithms.

DVR algorithms were initially presented by Blinn and Kajiya (Ref. [**126, 127**]) allowing the visualization of all inner characteristics of volumes at once. A comparison between different *DVR* algorithms, such as Texture Mapping, Raycasting, Splatting or Shear Warp, was presented [**128**]. Raycasting is one of the best algorithms for *DVR* which was proposed by Levoy (Ref. [**89**]). Its main idea is to calculate the volume rendering integral along rays launched from the view window into the volume. Thus, in this method, the color and opacity of each pixel in the final image are evaluated by accumulating the contribution of discrete samples of the ray that intersects the volume. Futher developments in raycasting for accelerated hardware using *OpenGL* were collected by Hadwiger (Ref. [**90**]).

The Khronos Group released the *WebGL 1.0* Specification in March 2011. It is a JavaScript binding of *OpenGLES 2.0 API* and allows a direct access *GPU* graphical parallel computation from a web-page. Calls to the *API* are relatively simple and its implementation does not require the installation of external plug-ins, allowing an easy deployment of multi-platform and multi-device applications. Browser based

3D middleware frameworks have been presented using *WebGL* as a rendering backend such as GLGE[1], SpiderGL (Ref. [**129, 130**]), SceneJS[2], WebView3D[3], XTK[4]. *XML3D* (Ref. [**131**]) and *X3DOM* (Ref. [**103**]) present a declarative model for 3D visualization on Web using *WebGL* as one of the backends.

Declarative programming is a model of data and logic representation focused on the description of the model more than the control flow. Markup languages are one of the declarative models which are extensively used in the web, e.g. *HTML*, *XML*, *SVG*, *MATHML*, etc. *X3D*[5] is a declarative *XML*-based markup lenguage for the representation of 3D data. *MEDX3D* is an extension of *X3D* supporting the advanced medical visualization functionality.

Medical visualization is a challenging scientific field since the interpretation of images leads to clinical intervention. Therefore, quality and fast interactive response are important features in this domain. Remarkable advances have occurred in medical imaging technology and recent applications supports the possibility of sharing imaging data online across clinical and research centers and among clinicians and patients. Image formats such as *DICOM* or *NRRD* allow to store medical volume data acquired by diverse techniques such as computer tomography (*CT*), magnetic resonance imaging (*MRI*), diffusion magnetic resonance imaging (*dMRI*), positron emission tomography (*PET*), ultrasound and others. A formalized and standardized model of visualization for medical imaging allows a correct interpretation of the images in heterogeneous architectures like Internet.

The contribution of this article is the implementation of the *MEDX3D* standard using the *X3DOM* architecture, allowing the advanced medical visualization functions to be used in heterogeneous architectures. Also the generation a common framework for medical visualization which can be used, formalized, extended and tested in several devices and platforms.

The article is organized as follows. Section 2 presents the work related to this article, including a description of volume rendering techniques, visualization of medical images. The methodology of the developed work is explained in Section 3. Then, the results accomplished are presented in Section 4, and finally, Section 5 states future developments.

2. Related Work

2.1. Visualization of Medical Images. Advanced visualization of medical images require the use of algorithms with high computer processing demand. Client/Server architectures allow the distribution of computer processing in several models. They can be classified as Fat Client, Thin Client and Hybrid. Each one of them has specific characteristics.

2.1.1. *Fat Client.* In the Fat Client model, all the processes are made in the client, freeing the server for data storage and communication only. This approximation depends heavily of the processing power of the client and the visualization quality or interactivity of the application is not guaranteed. Masseroli (Ref. [**132**]) presents an architecture based on Java Applets in which the server is used for storage and communication only, the visualization of medical images is based on 2D slices of the datasets and basic tools for editing are implemented. Hamza (Ref. [**133**]) presents an architecture for the visualization of medical images in 3D as a set of surfaces with transparency using X3D. Cantor Rivera (Ref. [**134**]) presents a visualization framework based on the libraries of *VTK*[6] for reading images and extracting the geometry, and uses WebGL for the client rendering.

[1] www.glge.org
[2] www.scenejs.org
[3] www.arivis.com/en/arivis-WebView/arivis-WebView-3D
[4] www.github.com/xtk/X/wiki
[5] www.web3d.org/x3d/
[6] www.vtk.org

2.1.2. *Thin Client.* Thin client allows the use of lightweight clients for visualization, but all the computational process is performed in the server. Scalability of the system is very difficult to achieve because for each new client using the service the server load increase. Nevertheless, the use of this model allows the use of low power devices and the visualization is the same for all the clients. Poliakov (Refs.[**135, 136**]) presents a model of Web Service visualization for medical imaging where the render is made on the server and send to a *JAVA* client application through the network. Blazona (Refs. [**137, 138**]) presents an architecture where the client uses *X3D* web plugins for the final visualization but the server side is responsible to generate the models that will be visualized. Smelyanskiy (Ref. [**139**]) presents a set of medical visualization algorithms optimized for parallel architectures and distributed computing.

2.1.3. *Hybrid.* Hybrid methods share the processing capabilities of the server and the client, but, since communication is required, network latency can generate several problems for the generation of the integrated images. Yoo (Ref. [**140**]) presents a hybrid method which uses video streaming for the visualization of volume models integrated with geometry data in the client. Seamless visualization of volume data and geometry can be found in the work of Mahmoudi and Settapat (Refs. [**141, 142, 143**]), these works uses the client's 3D capabilities but uses server processing for the heavy computational process and the extraction of specific visualizations. These models also allow the implementation of advanced tools of image processing and collaborative visualization. A very interesting approximation for hybrid models was presented by Jomier (Ref. [**144**]), in which the visualization of large datasets can be performed and different backends for rendering, depending on the client capabilities, are used.

2.2. MEDX3D. *MEDX3D* (Ref. [**121**]) is an extension of the *X3D* ISO standard to support advanced medical visualization functionality. The standard presents a set of tags which allow to read different volume formats such as *DICOM*, *NRRD* or Raw data. Also, the standard defines volume styles visualizations which allow to render with different non-photorealistic models, enhancing specific regions of the volume which normally represent interesting regions to be visualized, such as edges, blobs and occlusions. Several applications on Web are used in medical fields (Ref. [**108**]). One of the applications is medical training simulations (Ref. [**145**]). In this work a simultaneous visualization and haptic interaction was presented. Extensions for the *MEDX3D* standard are proposed by Fried and Ullrich (Refs. [**146, 147**]).

3. Methodology

MEDX3DOM implements the standard *MEDX3D* in *X3DOM*. The implementation is based on the generation of the nodes for two components: Texturing3D and VolumeRendering. The actual status of the implementation declares all nodes for *X3DOM* architecture but only the some basic functionality are implemented.

Texturing3D nodeset define the tags to work with the information of 3D data, specific readers for file formats such as *DICOM*, *NRRD* and raw data are part of the standard. Other kind of data sources could be implemented like *WADO*, which is an specification for a WebService to obtain *DICOM* data. 3D texture data is normally stored in a **Texture3D** structured which is part of common 3D APIs like *OpenGL*. This structure is not available in WebGL and a mapping between a Texture3D and a Texture2D was define in order to overcome this limitation. This mapping is defined as a shader function (Fig. 9.2).

Volume data is storage in atlas format. An example of medical volume dataset represented in this format is shown in figure 9.3. This model of representation also allows the storage of preprocessed data such as image gradients shown in figure 9.4, where each color magnitude represent the direction of the gradient vector for each axis.

Given the flexibility of the *X3DOM* architecture the transfer functions can be store as images, an example of a two dimensional transfer function is presented in figure 9.5. Its also possible to use a video

```
vec4 cTexture3D(sampler2D uData,vec3 volpos)
{
  float s1,s2;
  float dx1,dy1;
  float dx2,dy2;

  vec2 texpos1,texpos2;

  s1 = floor(volpos.z*numberOfSlices);
  s2 = s1+1.0;

  dx1 = fract(s1/slicesOverX);
  dy1 = floor(s1/slicesOverY)/slicesOverY;

  dx2 = fract(s2/slicesOverX);
  dy2 = floor(s2/slicesOverY)/slicesOverY;

  texpos1.x = dx1+(volpos.x/slicesOverX);
  texpos1.y = dy1+(volpos.y/slicesOverY);

  texpos2.x = dx2+(volpos.x/slicesOverX);
  texpos2.y = dy2+(volpos.y/slicesOverY);

  return mix( texture2D(uData,texpos1),
    texture2D(uData,texpos2),
    (volpos.z*numberOfSlices)-s1);
}
```

FIGURE 9.2. GLSL Fragment shader function to map a volume coordinate to a texture 2D atlas, the variables $numberOfSlices$, $slicesOverX$, $slicesOverY$ are uniforms and represent the number of slices of the volume, the rows and columns.

texture for transfer functions or volume data, allowing the animation of the models without a maintaining the interactivity of the visualization and showing the image in a dynamic way (Fig. 9.6).

The implementation of the MEDX3D standard into X3DOM allows the declaration of volume visualization in X3D and visualized the render in browser without the use of plugins. The files follow the standard MEDX3D but the image texture is defined as a **Texture2D** structure. The file format is presented in the figure 9.7, were an extraction of and HTML5 file is presented were a volume is rendered in a webpage.

4. Results

We present an early implementation of the *MEDX3DOM* proposal, which allows visualization of medical volume datasets following the guidelines of the *MEDX3D* standard. The implementation is partially integrated with *X3DOM* using the *WebGL* backend and showing some basic styles of volume rendering.

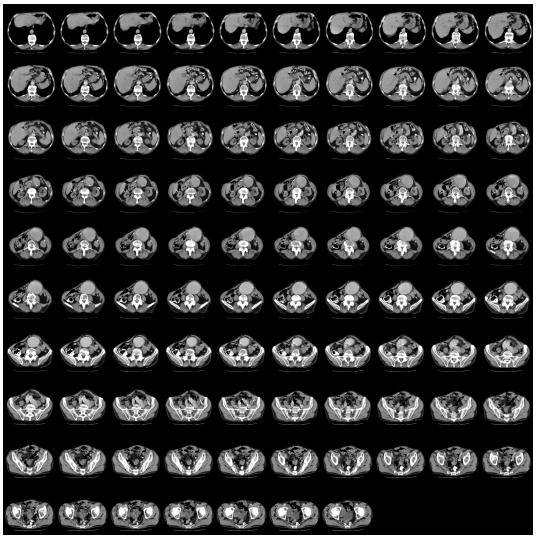


FIGURE 9.3. Texture atlas with the volume data for the aorta dataset

FIGURE 9.4. Texture atlas with the volume gradient data for the aorta dataset, enhanced color for better visualization.

Data reading is implemented using the previous model of image atlas, but different test show the easy integration of *NRRD* and raw data readers into Javascript and *X3DOM*. *DICOM* is a very extensive and complicated file format and the implementation of a reader in Javascript its not recommended. But compiler translation technologies like *emscripten*[7] allows the generation of Javascript code from C or C++ languages, some test show the correct implementation for image formats as *TGA* or *JPEG2000*, the translation of *DICOM* format could be possible and some libraries had been initially tested such as *DCMTK*.

Volume storage in the client is made with a **Texture2D** structure, the limit of the structure in the *WebGL* implementations restricts the size of the volume to be rendered. Some alternatives can be implemented such

[7]www.github.com/kripken/emscripten

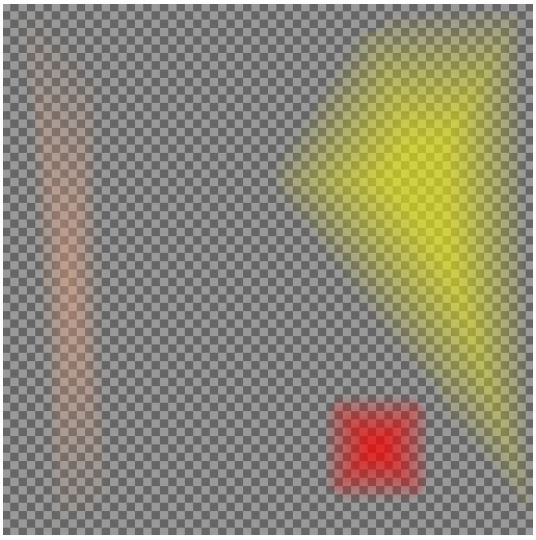


FIGURE 9.5. 2D transfer function image, background squares represent are presented to visualize the transparency value.

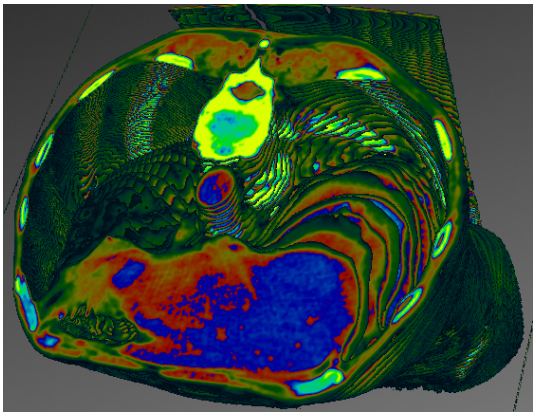


FIGURE 9.6. Volume rendering with a color-full transfer function.

as Level of Detail for volumes or Compressed Textures. Nevertheless this restriction is hard to solve and the impact in the visualization is critical in low graphics memory devices.

Volume Styles are not fully implemented, and only some simple styles are presented. The render pipeline of the Styles follow the implementation of *H3D*. It's expected that the other styles could be easily integrated into the architecture.

5. Future Work

Optimization models for the raycasting algorithm such as empty space skipping, pre-multiplied transfer functions, ray jitter can be integrated into the rendering pipeline to enhance the quality of the render. Other visualization models such as Multi-Planar Reconstruction (*MPR*) are going to be integrated in *MEDX3DOM*.

Vector fields are another representation for medical visualization used specially for *dMRI* datasets. This representation need preprocessing models to be visualized know as Flow Visualization (Ref. [**148**]). Linear Integral Convolution (*LIC*) is a methodology to visualize this kind of data and will be integrated in the implementation.

```
<html>
<head> ... </head>
<body>
<h1>Volume Rendering</h1>
<X3D xmlns='http://www.web3d.org/specifications/x3d-namespace'
 showStat='true' showLog='true' width='500px' height='500px'>
  <Scene>
    <Background skyColor='0 0 0'/>
    <Viewpoint description='Default' zNear='0.0001' zFar='100'/>
    <Transform>
      <VolumeData id='volume' dimensions='4.0 4.0 4.0'>
        <ImageTexture containerField='voxels'
         url='aorta4096.png'/>
        <OpacityMapVolumeStyle> </OpacityMapVolumeStyle>
      </VolumeData>
    </Transform>
  </Scene>
</X3D>
</body>
</html>
```

FIGURE 9.7. HTML/X3DOM file fragment to visualize a volume using *MEDX3DOM*, Texture image defined in atlas format

Hybrid visualization of volumes and geometry is important for the visualization of medical dataset where the volume is registered with other geometry, e.g. Brain Volumes and Tractography (Ref. [**149**]), Head Volumes and Dental Surfaces, etc. This hybrid visualization is going to be integrated in *MEDX3DOM*.

Transfer functions are a fundamental tool for the correct visualization of volumes. Prototypes of automatic generation of transfer functions by visual clues, know as Visual Volume Attention Maps (*VVAM*) (Ref. [**150**]), has been developed and the methodologies are going to be tested in *MEDX3DOM*.

Use of the volume rendering technology can be used for other datasets besides medical images, such as geovisualization of weather data (Ref. [**151**]), ocean data, underground structures or mechanical components. Some of this datasets are going to tested in the implementation.

Part 4

Hybrid Rendering of Surfaces and Volumes

CHAPTER 10

Real-time volume rendering and tractography visualization on the WEB

- John Congote[1,2]
- Esther Novo [1]
- Luis Kabongo[1]
- Daniel Ginsburg[3,5]
- Stephan Gerhard[4]
- Rudolph Pienaar[3,5]
- Oscar Ruiz[2]

[1] CAD CAM CAE laboratory, Universidad EAFIT
Carrera 49 No 7 Sur - 50, Medellín, Colombia
[2] Vicomtech Research Center
Donostia - San Sebastián, Spain
[3]Children's Hospital Boston, United States
[4]Institute of Neuroinformatics
Uni/ETH Zurich, Switzerland
[5]Harvard Medical School
Boston, United States

Context

Realtime Volume Rendering and Tractography Visualization on the Web. J. Congote and E. Novo and L. Kabongo and D. Ginsburg and S. Gerhard and R. Pienaar and O. Ruiz. Journal of WSCG. v.20, n.2, 2012. ISSN 1213-6972. pages 81-88. WoS ISI and Tompson Reuters. url http://wscg.zcu.cz/wscg2012/ JOURNAL

The visualization of medical images is an active field of research and the implementation of the volume rendering methodologies developed in during the Ph.D., results in several collaboration aggredments with other institutions. This work present the results for the visualization of Medical Images from the Children Hospital and the Institute of Neuroinformatics using the Volume Rendering methodology researched in Vicomtech and CAD/CAM/CAE Laboratory.

This work was partially supported by CAD/CAM/CAE Laboratory at EAFIT University and the Colombian Council for Science and Technology -COLCIENCIAS-

As co-authors of such publication, we give our permission for this material to appear in this document. We are ready to provide any additional information on the subject, as needed.

Prof. Dr. Eng. Oscar E. Ruiz
oruiz@eafit.edu.co
Coordinator CAD CAM CAE Laboratory
EAFIT University, Medellin, COLOMBIA

Eng. Esther Novo
enovo@vicomtech.org
eHealth & Biomedical Applications
Vicomtech, San Sebastian, SPAIN

Dr. Luis Kabongo
jbarandiaran@vicomtech.org
eHealth & Biomedical Applic
Vicomtech, San Sebastian, SPAIN

Dan Ginsburg
dginsburg@upsamplesoftware.com
Children's Hospital
Boston, USA

Stephan Gerhard
connectome@unidesign.ch
Institute of Neuroinformatics
Uni/ETH Zurich, Switzerland

Dr. Rudolph Pienaar
Rudolph.Pienaar@childrens.harvard.edu
Harvard Medical School
Boston, USA

Abstract

In the field of computer graphics, *Volume Rendering* techniques allow the visualization of 3D datasets, and specifically, Volume Ray-Casting renders images from volumetric datasets, typically used in some scientific areas, such as medical imaging. This article aims to describe the development of a combined visualization of *tractography* and *volume rendering* of brain T1 MRI images in an integrated way. An innovative web viewer for interactive visualization of neuro-imaging data has been developed based on *WebGL*. This recently developed standard enables the clients to use the web viewer on a wide range of devices, with the only requirement of a compliant web-browser. As the majority of the rendering tasks take place in the client machine, the effect of bottlenecks and server overloading are minimized. The web application presented is able to compete with desktop tools, even supporting high graphical demands and facing challenges regarding performance and scalability. The developed software modules are available as open source code and include MRI volume data and tractography generated by the Diffusion Toolkit, and connectivity data from the Connectome Mapping Toolkit. Our contribution for the Volume Web Viewer implements early ray termination step according to the tractography depthmap, combining volume images and estimated white matter fibers. Furthermore, the depthmap system extension can be used for visualization of other types of data, where geometric and volume elements are displayed simultaneously.

1. Introduction

Three-dimensional data can be found in several scientific fields, coming from simulation, sampling or modeling processes. Regarding the biomedical scope, several scanning techniques, such as magnetic resonance (MRI) or computerized tomography (CT), are used for storing body imaging samples as volumetric datasets formed by groups of parallel slices, where the term volumetric dataset refers to a scalar field. These datasets are usually visualized in three dimensions in order to facilitate specialists to interpret information.

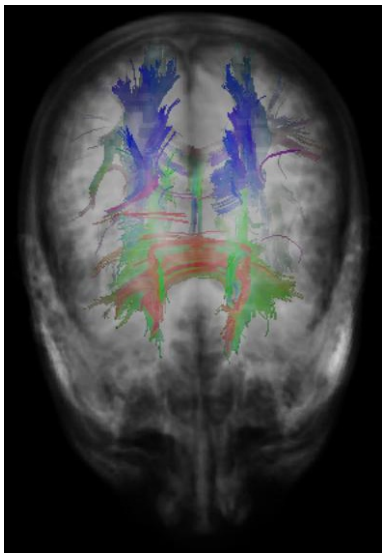


FIGURE 10.1. Combined visualization of volume rendering and tractography information on the web

Visualization of medical volumetric datasets can suitably be performed by the use of *Direct Volume Rendering* algorithms. These methods show important characteristics of datasets, even though rendering is not usually photo-realistic.The problem addressed in this paper is the visualization of tractography information obtained from dMRI (diffusion MRI) together with volume data corresponding to MRI or CT images.

In order to represent the volumetric datasets, volume rendering techniques allow the visualization of all inner characteristics of volumes at once, by projecting data into 2D images, according to the corresponding position of a virtual camera. The main idea of the ray-casting algorithm is to launch rays from the camera into the volume, calculating the volume rendering integral along the rays. Thus, in this method, the colour and opacity of each pixel in the final image is evaluated by launching a ray in the scene from the view position, sampling the volume at discrete points along the ray and accumulating the contribution of each sample.

Our contribution is an implementation of a web rendering system for medical images, which integrates volume rendering and geometric objects within a compliant WebGL browser, based on the volume ray casting algorithm and built on previous developments [**152**]. Due to the technology limitations of Webgl, the improvements developed allow us to create a web application for combined visualization of volume rendering and tractography, as shown in Figure 10.1, being able to compete with desktop tools, supporting high graphical demands and facing challenges regarding performance and scalability.

The article is organized as follows. Section 2 presents the work related to this article, including a description of volume rendering techniques, visualization of medical images and geometry intersection. The methodology of the developed work is explained in Section 3. Then, the results accomplished are presented, and finally, Section 5 states the conclusions and future developments.

2. Related Work

2.1. Volume Rendering. In computer graphics, Ray Casting is a well known direct volume rendering technique that was designed by Kajiya and Herzen [**153**] as one of the initial developments in this area. Traditionally, three dimensional objects have been created by using surface representations, drawing geometric primitives that create polygonal meshes [**154**], hence provoking the loss of information from one dimension. Further developments [**155**] accomplished the mathematical modeling of the ray casting process, based on the light's behaviour equations. Thus, the volume rendering integral was defined. A comparative between different direct volume rendering algorithms, such as Texture Mapping, Ray Casting, Splatting or Shear Warp, was presented [**97**]. Ray casting is a flexible algorithm that allows the implementation of acceleration methods, such as Empty Space Skipping [**156**] or *Early Ray Termination*. Early ray termination is an optimization process that establishes certain limitations in the volume, so that the samples encountered after them do not contribute to the value of the pixel.

Ray casting suitably fits GPUs' operating mode [**157**], because of the independence of each ray that is launched to the scene, making this algorithm highly parallelizable and allowing the exploitation of GPUs' parallel architecture. For GPU ray casting, the volume element is stored in the GPU memory as a 3D texture and a fragment shader program is used in order to implement the ray casting algorithm.

A quality evaluation model was developed for comparing the different Direct Volume Rendering techniques [**158**]. These methods handle a higher amount of data than surface rendering techniques, therefore, the complexity of the algorithms is increased, as well as the necessary rendering time [**159**]. Optimized volume rendering methods avoid empty spaces by introducing a volume proxy geometry [**160**].

Web 3D Rendering. The use of the recently released WebGL standard [**96**] leads to new methods for web 3D visualization, where most part of the computational processes are performed in vertex and fragment shaders that run on the GPU hardware. WebGL is a software library that enables HTML5-based browsers to identify clients' graphics hardware. HTML5, the latest Internet standard propose, provides native elements for audio and video. WebGL consists of a low-level imperative graphic programming API based on OpenGLES 2.0 for Javascript that enables flexibility and exploits the characteristics of advanced graphics cards. Due to the constant improvement of the performance of Javascript interpreters, the management of scene elements behaves similarly to the ones obtained by using natively compiled languages. Moreover, some WebGL extensions have been implemented in order to achieve a friendly interaction, such as SpiderGL [**161**].

Several standards and proprietary solutions are currently being developed in order to fulfil the necessity of moving 3D visualization into the web [**111**], such as X3D, a standard derived from VRML that stores 3D information in a scenegraph format using XML (Extensible Markup Language). This model has been implemented in a declarative form, as an extension of HTML; X3DOM presents a framework for integrating X3D nodes into HTML5 DOM content [**103**] and other alternatives have also been developed, e.g. XML3D [**162**]. Finally, there is a standardization for X3D in the MedX3D volume rendering model [**163, 164**].

2.2. Visualization of Medical Images. Medical visualization is a challenging scientific field because interpretation of images may lead to clinical intervention. Therefore, quality and fast interactive response are important features in this domain. Remarkable advances have occurred in medical imaging technology and applications in the past few years, supporting the possibility of sharing imaging data online across clinical and research centres and among clinicians and patients. The development of these kind of applications is influenced by connectivity, security and resources' heterogeneity concerns.

On-server rendering can be considered a partial solution for Medical Imaging [**165**]. Moreover, several web implementations for volumetric visualization have already been presented [**163**], although many of these solutions require third party systems to allow visualization or their scalability is limited by the rendering server.

As medical volumetric imaging requires high fidelity and high performance, several rendering algorithms have been analyzed, leading to thread- and data-parallel implementations of ray casting [**166**]. Thus, architectural trends of three modern commodity parallel architectures are exploited: multi-core, GPU, and Intel Larrabee. Other approaches describe the development of web-based 3D reconstruction and visualization frameworks for medical data [**167**]. Such applications based on X3D technology allow extending cross-platform, inter-application data transfer ability. Several applications have been implemented using web 3D rendering techniques, for example, evaluation systems at the educational level [**108**] or medical training simulations [**145**].

dMRI. Diffusion Magnetic Resonance Imaging (dMRI) relies on the visualization of water diffusion using data from MRI. Diverse methodologies have been presented over the last years and can be classified into two categories: Image based and Object based techniques. The first methodology divides the space in voxels and the assigned colour represents the principal diffusion direction [**168**]. However, tracks can not be easily identified since no segmentation of the visualization is performed, and therefore direction information is difficult to observe since voxel colour mapping is not one-to-one, *i.e.*, different directions might be represented by the same colour. Otherwise, in object based methodologies, objects, such as ellipsoids and lines, are used together with colour mapping in order to enhance visualization and give a direction sense to the representation.

Visualization of brain white matter cortical tracks is one of the most important applications of dMRI, since it allows to non-invasively visualize white matter anatomy, and detecting of anomalies [**169, 170**]. Tractography, which refers specifically to the representation of the white matter tracks based on the water diffusion information, employs lines to represent the diffusion direction and to visualize the white matter paths. In general, lines are generated using randomly distributed seed points; together with the principal diffusion information and a prescribed interval of time, the different paths are generated. However, this representation becomes dependent on the amount and location of seed points to correctly visualize tracks [**171**] because erroneous connections might be produced between tracks due to the existing error in data. Incorrect visualization of branching of tracks is another drawback, since only one path is generated per each seed point.

Probabilistic methodologies have been proposed [**171**] to represent branching of white matter tracks, in which secondary seed points are included in regions in which branching is assumed. Therefore, a denser visualization is performed in those regions. An algorithm was proposed for path visualization [**172**], in which the different global paths are simplified by one simple curve, clustering the different paths and then using average curves to obtain one simple curve that summarizes each cluster.

2.3. Geometry Intersection. The application described in this article requires representing volume rendering and tractography together, *i.e.*, both volumetric and polygonal data have to be displayed in the same scene. There are several models for combining polygonal geometry and volume rendering. Some methods identify the intersection between rays launched in the volume rendering process and geometry [**173**]. This technique can be optimized by creating octrees for dividing the geometric space and prove intersections correctly.
Other models try to achieve a correct visibility order for the intersections between volume and geometry [**90**]. Geometry has to be rendered in the first place to correctly look at the intersections of the geometry and the volume. Besides, parts that are occluded by the geometry should not contribute to the final image, not performing any ray casting at all. In order to achieve this feature, rays should terminate when they hit a polygonal object, accordingly modifying the ray length image if a polygonal object is closer to the view point than the initial ray length.

3. Methodology

In our project, the results of the Connectome Mapper are directly loaded in the browser using WebGL and JavaScript. The FreeSurfer cortical surface reconstruction binary files are loaded and processed in JavaScript and converted to WebGL vertex buffer objects for rendering. The surfaces are overlaid with per-vertex curvature values computed during the FreeSurfer processing stream. The tractography data is likewise parsed in the JavaScript code and rendered as line primitives coloured based on direction. Finally, the structural network itself is converted to JSON (JavaScript Object Notation) as an offline preprocess and loaded into the browser using JavaScript. The networks are visualized in 3D along with the fiber tracts and volumes enabling exploration of connectivity information in real-time.
The work described in this paper has been developed using volume ray casting, a widely used algorithm for generation of 2D representations from three dimensional volumetric datasets. The obtained images are 2-dimensional matrices $I : [1, h] \times [1, w] \rightarrow \mathbb{R}^4$ (w: width and h: height, both in pixels). Each pixel is represented by a colour expressed by a four-tuple of red, green, blue and alpha real-valued components, $(R, G, B, A \in [0, 1])$.
An entire volume is represented by a 3-dimensional array of real values $V : [1, H] \times [1, W] \times [1, D] \rightarrow [0, 1]$ (H: Height, W: Width, D: Depth of the represented volume, all of them in positive integer coordinates). Therefore, $V(x, y, z) \in [0, 1]$. The projection model used in this work is called pin-hole camera [**114**]. The pin-hole camera technique uses intrinsic $K \in M_{3\times4}$ and extrinsic $R \in M_{4\times4}$ real-valued matrices in order to project every 3D point $p \in \mathbb{P}^3$ onto a 2D point $p' \in \mathbb{P}^2$.
The volume ray casting algorithm defines the colour for each pixel (i, j) in the image, which is also known as projection screen, I, according to the values of a scalar field $V(x, y, z)$. This scalar field is associated to the points (x, y, z) reached by rays that are originated at a certain pixel or camera, represented as C in Figure 10.2. A cuboid geometry is generated with coordinates $(0, 0, 0)$ to $(1, 1, 1)$. This cube represents the boundary established for the volumetric dataset. Each ray intersects with the cuboid volume V at points $p_{(i,j)}(x, y, z)$ and $q_{(i,j)}(x, y, z)$, which represent the input and output coordinates of the ray into and out from the volume, respectively.
Then, each obtained ray pq is equi-parametrically sampled. For every sampled point $s(x, y, z)$ over the ray, an approximation of the scalar field $V(s)$ is calculated, commonly by using trilinear interpolation. The sampled points also influence the colour of the originating pixel, due to the use of a composition function (Equations 43-46), where the accumulated colour A_{rgb} is the colour of the point s in the volume V, and A_a is the transparency component of the pixel, which has a value of 1 at the end of the rendering process. Given a certain set of coordinates (x, y, z) in the volume and a ray step k, V_a is the scalar value of the volume V, V_{rgb} is the colour defined by the given transfer function V_a, S represents the sampled values over the ray and O_f, L_f are the general Opacity and Light factors.

$$S_a = V_a \times O_f \times \left(\frac{1}{s}\right) \tag{43}$$

$$S_{rgb} = V_{rgb} \times S_a \times L_f \tag{44}$$

$$A^k_{rgb} = A^{k-1}_{rgb} + \left(1 - A^{k-1}_a\right) \times S_{rgb} \tag{45}$$

$$A^k_a = A^{k-1}_a + S_a \tag{46}$$

In the ray casting process performed in this work, geometry G is formed by a set of segment lines L (although G could also be represented as a set of points P or triangles T). Each segment L is defined by two points in the space. Lines are projected through projection matrices onto a different image, where the values of colour (r, g, b, a) and depth $(depth)$ are defined for each pixel (x, y).

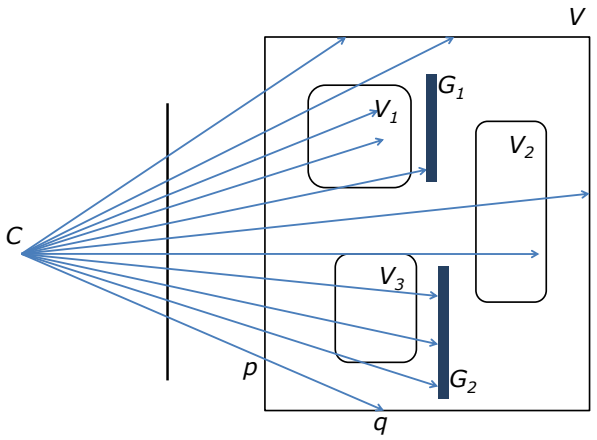


FIGURE 10.2. 2D representation of the ray casting algorithm performance (types of ray termination)

Each pq ray traverses the cuboid volume V, where both volume elements V_i and geometries G_i are rendered in the same process by modifying the early ray termination method, as depicted in Figure 10.2. This technique checks the alpha value for each sample of the transparency colour of the ray. If the value V_a is equal to 1, which means that the ray has reached the final colour, the remaining steps of the ray are not evaluated. Rays might terminate due to several reasons: when encountering a very dense volume (such as V_1 in fig. 10.2), when intersecting with a geometric element (*e.g.* with G_1) or when exiting the boundary cube, at point q.
The early ray termination model is also used to check the length of the ray and compare it to the depthmap of the figure. In conclusion, a projection of the geometry is obtained, as well as the colour and depth for each pixel in the image. This information can be compared to the length of the ray, terminating the ray when the alpha value is 1 or when the depth is equal to the geometry depth.

4. Results

This section describes the accomplished implementation of a real-time web viewer for both direct volume rendering and tractography visualization. This work is based on the WebGL standard and performs the ray casting algorithm with an early ray termination optimization.

4.1. Tractography. The Connectome Mapper [**174**] is a publicly available software that provides a pipeline to automatically generate structural networks from raw dMRI data of the brain. Gray and white matter segmentations are obtained by processing T1 MPRAGE MRI using the Freesurfer set of tools. The Diffusion Toolkit is used later for reconstruction. A deterministic streamline algorithm is used to obtain tractography, by generating fiber tracts of the same subject. For cortical and subcortical regions of interest, a parcellation is performed. Finally, these datasets are coregistered and a network is generated by weighting the connectivity between regions based on the fiber tracts [**175**].

4.2. Data Processing and Volume Interpolation. For the developed work, all the slices that correspond to a certain volume are composed into a single image, as shown in Figure 10.3. This image is generated by placing slices in a matrix configuration as a preprocessing step of the rendering algorithm. The size of the image stored in GPU memory could range from 4096×4096 on a PC (which can contain up to 256^3 volume) to $1024x1024$ on other devices (which can contain up to $128 \times 128 \times 64$). The screen resolutions being reduced on mobile devices it seems reasonable to scale down or even crop the volumes original dimensions in order to match the maximum GPU available memory.

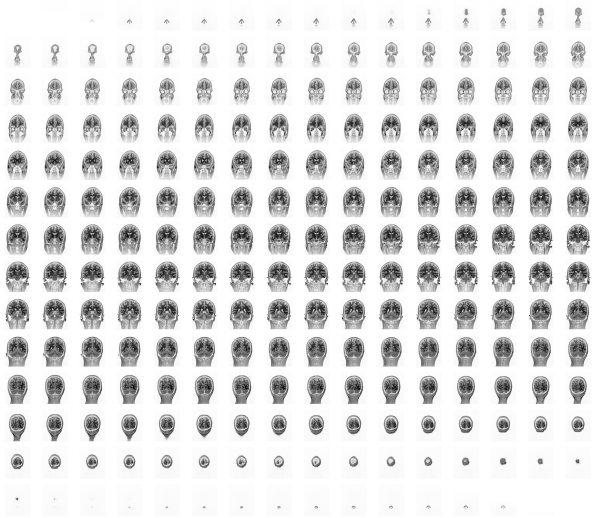


FIGURE 10.3. Brain dataset in mosaic form, read by the shader

In medical imaging, the sample bit depth is usually higher than 8 bits per pixel. This is a drawback that has to be handled for the development of web applications, where commonly supported formats are limited to 8 bits per sample. In the described experiment, information from medical datasets was reduced to 8 bits per sample.

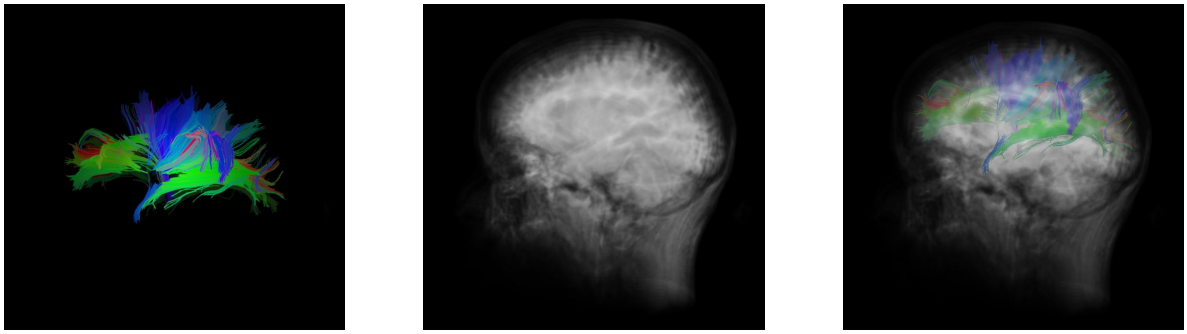
Identification of Ray Coordinates. According to the ray casting algorithm, the displayed colours of the boundary cuboid geometry represent the coordinates at each point (x, y, z). Coordinates are stored as r, g, b colour components for each pixel. Then, the cube can be rendered in the scene from the desired view point. In order to achieve volume visualization, several steps are followed in the rendering process. First of all, the rendering of the colour cube is performed according to the depth function change.
Taking this into account, rays are defined for each point of the cube, starting at the front faces, where the virtual camera is located, and ending at the back region. The colour of every point of the cube represents the exact coordinates of the ray for each pixel in the image. The colour information is stored as 24 bit

RGB values. The range of values that can be represented may seem small or imprecise for large images, but colour interpolation provides precision enough for ray coordinates. The depth information is stored in different buffers in order to obtain the corresponding depth value for each ray. Finally, the geometry is rendered and the colour and depth buffers are stored to be processed in the volume shader.

4.3. Visualization. The previously presented GPU implementation of volume rendering based on WebGL was used to develop a real-time online tractography and volume rendering viewer, accordingly to Table 10.1, proving this standard to be a valid technology for real-time interactive applications on the web. The results shown in the table below were accomplished when interacting with the web viewer from several computers, using the same web browser (*Chrome*) and the same number of steps, 50. For every graphic card tested, the application can be completely considered to have a real-time behaviour.

Graphic card model	Frame rate
NVidia GeForce GTX480	60 fps
NVidia GeForce GTX285	60 fps
NVidia 9600GT	28 fps
NVidia Quadro FX 3800M	20 fps
NVidia Quadro FX 880M	15 fps

TABLE 10.1. Performance of the developed viewer for different graphic cards, using Chrome as web browser, the number of steps equal to 50



(a) Tractography (i) (b) Volume Rendering (i) (c) Combined visualization (i)

FIGURE 10.4. Tractography, volume rendered image of brain T1 MPRAGE MRI and combined visualization on the web

In the developed work, the web viewer shows tractography information obtained from dMRI in the first place, represented in Figure 4(a). These organized fiber tracks in the white matter connect various cortical regions to each other. The tractography is represented using WebGL line primitives, where each fiber track is rendered by a set of points. The colour is assigned based on the absolute value of the unit vector pointing in the direction from the start point to the end point of the tract. The length value of each tract is stored in a per-vertex attribute together with the position and colour. The minimum tract length value is placed in a uniform variable in the vertex shader. The vertex shader determines whether the tract is longer than the minimum length to render. The entire tractrography set for the brain is efficiently rendered using a single draw call with one vertex buffer object. Thus, no dynamic geometry generation is performed in JavaScript.

Direct volume rendering of MRI data (Figures 4(b)) is developed simultaneously with the tractography. The volume renderer loads the MRI dataset from the server into a tiled 2D texture. Then, ray-tracing is performed in the shader in order to obtain the volume rendering. This implementation of a volume rendering system for the Web is based on the Volume Ray-Casting algorithm. Since the algorithm is implemented in WebGL, the reached visualization speed is similar to native applications, due to the use of the same accelerated graphic pipeline. The algorithm simulates 3D data by using a 2D tiling map of the slices from the volume maintaining trilinear interpolation and runs entirely in the client.
In the developed Web viewer, shown in Figure 10.5, the tractography and the volume rendering from brain MRI data can be represented separate or simultaneously, as depicted in Figures 4(c). Several features can be modified at runtime, by adjusting the provided sliders. Tractography's position can be changed according to the three main axes and fiber tracks can be seen more clearly by reducing the volume opacity. Finally, the minimum tract length can also be modified.

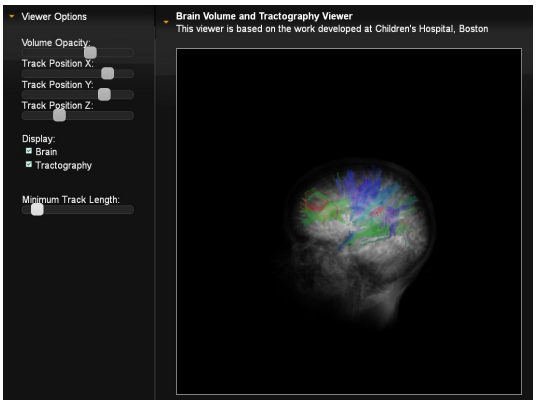


FIGURE 10.5. Volume rendering and tractography web viewer (sliders available for configuration)

5. Conclusions and Future Work

This paper describes the successful implementation of remote visualization of medical images based on WebGL[1]. Interaction with remote medical images was limited by many technical requirements, but the emergence of recent standards such as WebGL and HTML5 allow the development of applications that enable clients to access images without downloading them, maintaining data in a secure server and being able to perform functions, *e.g.* registration, segmentation, *etc.*, in a web context. These technologies empower web browsers to handle 3D graphics naturally. Thus, modern browsers support a wide range of applications, from simple rendering of two dimensional images to complex manipulation of 3D models.
The achieved visualization of volume rendering and tractography on the web, used for the implementation the presented viewer (shown in Figure 10.5), has demonstrated the capabilities of complex volume rendering visualization in web browsers, as well as the potential of WebGL for interactive visualization of neuroimaging data. Combined representation of volume rendering of brain T1 MRI images and tractography in real time has been accomplished. The main strength of the WebGL standard used here is the ability to provide efficient access to GPU rendering hardware with no special client-side software requirements, except for a compatible browser. Thus, this platform has great potential for imaging tools, particularly those providing web-based interfaces for automatic pipelining of image data processing.

[1] `http://www.volumerc.org/demos/brainviewer/webgl/brain_viewer/brain_viewer.html`

In the work explained herein, the early ray termination algorithm was modified in order to combine volume and geometric elements in a seamless way. Thus, the developed software modules, which are available as open source code, successfully implement early ray termination step according to the tractography depthmap, performing a combination between volume images and estimated white matter fibers.

CHAPTER 11

Visualization of Flow Fields in the Web Platform

- Mauricio Aristizabal [1,2]
- John Congote [1,2]
- Alvaro Segura [2]
- Aitor Moreno [2]
- Harbil Arregui [2]
- Oscar Ruiz [1]

[1] CAD CAM CAE laboratory, Universidad EAFIT
Carrera 49 No 7 Sur - 50, Medellín, Colombia
[2] Vicomtech Research Center
Donostia - San Sebastin, Spain

Context

Visualization of Flow Fields in the Web Platform. Mauricio Aristizabal and John Edgar Congote and Alvaro Segura and Aitor Moreno and Harbil Arriegui and Oscar E. Ruiz. Journal of WSCG. v.20, n.3, 2012. ISSN 1213-6972. pages 189-196. WoS ISI and Tompson Reuters. url= http://wscg.zcu.cz/wscg2012/ JOURNAL

The volume datasets provides different kind of data, normally scalar values, but vector fields are also presented for volume dataset. This work present a methodology for the extraction of geometry information from vector field for further integration in the volume rendering pipeline.

This work was partially supported by the Basque Government's ETORTEK Project (ITSASEUSII) research program and CAD/CAM/CAE Laboratory at EAFIT University and the Colombian Council for Science and Technology COLCIENCIAS

As co-authors of such publication, we give our permission for this material to appear in this document. We are ready to provide any additional information on the subject, as needed.

Prof. Dr. Eng. Oscar E. Ruiz
oruiz@eafit.edu.co
Coordinator CAD CAM CAE Laboratory
EAFIT University, Medellin, COLOMBIA

Dr. Aitor Moreno
amoreno@vicomtech.org

Vicomtech, San Sebastian, SPAIN

Eng. Harbil Arregui
harregui@vicomtech.org

Vicomtech, San Sebastian, SPAIN

Eng. Alvaro Segura
asegura@vicomtech.org

Vicomtech, San Sebastian, SPAIN

Mauricio Aristizabal
maristizabal@vicomtech.org

Vicomtech, San Sebastian, SPAIN

Abstract

Visualization of vector fields plays an important role in research activities nowadays. Web applications allow a fast, multi-platform and multi-device access to data, which results in the need of optimized applications to be implemented in both high-performance and low-performance devices. Point trajectory calculation procedures usually perform repeated calculations due to the fact that several points might lie over the same trajectory. This paper presents a new methodology to calculate point trajectories over highly-dense and uniformly-distributed grid of points in which the trajectories are forced to lie over the points in the grid. Its advantages rely on a highly parallel computing architecture implementation and in the reduction of the computational effort to calculate the stream paths since unnecessary calculations are avoided, reusing data through iterations. As case study, the visualization of oceanic currents through in the web platform is presented and analyzed, using WebGL as the parallel computing architecture and the rendering Application Programming Interface.

1. Introduction

Vector field visualization has an important role in the automotive and aero-spatial industries, maritime transport, engineering activities and others. It allows the detection of particularities of the field such as vortexes or eddies in flow fields, but also permits exploring the entire field behavior, determining e.g., stream paths.

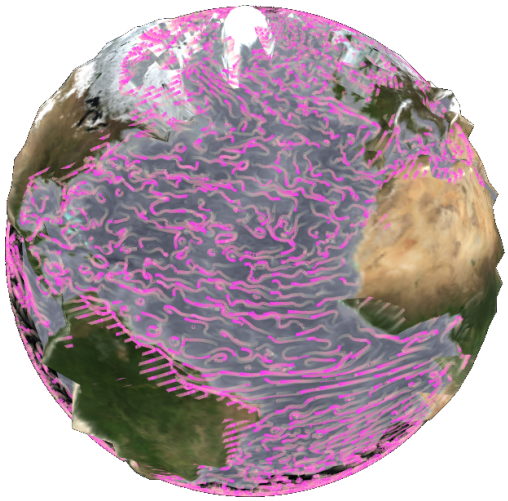


FIGURE 11.1. flow visualization of Atlantic ocean currents in WebGL. Hierarchical integration was used to reduce the total number of iterations required to calculate the paths.

Particularly, ocean flow visualization is important in maritime transport and climate prediction, since the movement of huge water masses can produce temperature variations of the wind currents, and, as a result, its visualization becomes determinant to represent the ocean's behaviour.

With the growth of a multiverse of devices, development of multi-platform applications has become a common goal for developers. The web is being established as a universal platform in order to unify the development and execution of applications. However, challenges arise since applications must be optimized in order to be implemented as well as on high-performance as on low-performance devices.
The contribution of this paper is a new methodology to calculate point trajectories of a highly dense grid of points over n-dimensional vector fields, in which the trajectories are forced to pass over the grid points

(Figure 11.1). This allows to implement a hierarchical integration procedure [**176**], which takes in advance previous calculated data in order to avoid repetitive and unnecessary calculations, and reduces the complexity of the algorithm from linear to logarithmic. The procedure is suitable to be implemented over highly parallel computing architectures due to independent calculations and the number of computations to be performed. As a result, we employ WebGL as the parallel computing engine to calculate the iterations, using images in order to store the data through the iterations.
Different from other procedures, in which the calculation of the trajectories is performed for each point in particular, our methodology allows to merge its calculation for all the points in which the field is discretized. Therefore, the number of unnecessary computations is critically reduced.
This paper is organized as follows: Section 2 presents the related work. Section 3 exposes the methodology in which the contribution of this work is explained. Section 4 presents a case of study in oceanic currents and finally the conclusions of our work are presented in section 5.

Glossary

API	Application Programming Interface
CUDA	Compute Unified Device Architecture
PL	Piecewise Linear
SIMD	Single Instruction Multiple Data
GLSL	Graphic Library Shading Language
WebGL	Web Graphic Library
LIC	Line Integral Convolution
GPU	Graphics Processing Unit
FBO	FrameBufer Object
Shader	Instructions to be performed in the GPU

2. Literature Review

2.1. Flow Visualization. A great amount of methodologies to visualize vector fields (flow fields) has been developed among the last decades. Geometric-based approaches draw icons on the screen whose characteristics represent the behavior of the flow (as velocity magnitude, vorticity, etc). Examples of these methodologies are arrow grids [**177**], streamlines [**178**] and streaklines [**179**]. However, as these are discrete approaches, the placement of each object is critical to detect the flow's anomalies (such as vortexes or eddies), and therefore, data preprocessing is needed to perform an illustrative flow visualization. An up-to-date survey on geometric-based approaches is presented by [**180**].
However, in terms of calculating those trajectories for determined points in the field, the procedures usually compute for each point the integrals, and, as a result, the procedures are computationally expensive for highly dense data sets.
On the other hand, texture-based approaches represent both a more dense and a more accurate visualization, which can easily deal with the flow's feature representation as a dense and semi-continuous (instead of sparse and discrete) flow visualization is produced. A deep survey in the topic on texture-based flow visualization techniques is presented by [**181**].
An animated flow visualization technique in which a noise image is bended out by the vector field, and then blended with a number of background images is presented by [**182**]. Then, in [**183**] the images are mapped to a curved surface, in which the transformed image visualizes the superficial flow.
Line Integral Convolution (LIC), presented by [**184**], is a widely implemented texture-based flow visualization procedure. It convolves the associated texture-pixels (texels) with some noise field (usually a white noise image) over the trajectory of each texel in some vector field. This methodology has been extended to represent animated [**185**], 3D [**186**] and time varying [**187, 186**] flow fields.

An acceleration scheme for integration-based flow visualization techniques is presented by [**176**]. The optimization relies on the fact that the integral curves (such as LIC) are hierarchically constructed using previously calculated data, and, therefore, avoid unnecessary calculations. As a result, the computational effort is reduced, compared to serial integration techniques, from $O(N)$ to $O(\log N)$, where N refers to the number of steps to calculate the integrals. Its implementation is performed on the CUDA architecture, which allows a GPU-based parallel computing scheme, and therefore the computation time is critically reduced. However, it requires, additionally to the graphic APIs, the CUDA API in order to reuse data, and hence, execute the procedure.

2.2. WebGL literature review. The Khronos Group released the WebGL 1.0 Specification in 2011. It is a JavaScript binding of OpenGL ES 2.0 API and allows a direct access to GPU graphical parallel computation from a web-page. Calls to the API are relatively simple and its implementation does not require the installation of external plug-ins, allowing an easy deployment of multi-platform and multi-device applications. However, only images can be transfered between rendering procedures using FBOs.
Several WebGL implementations of different applications have been done such as volume rendering, presented by [**188**] or visualization of biological data, presented by [**189**]. However, for the best of our knowledge, no other implementation that regards to LIC flow visualization on WebGL has been found in the literature or in the Web.

2.3. Conclusion of the Literature Review. WebGL implementations allow to perform applications for heterogeneous architectures in a wide range of devices from low-capacity smart phones to high-performance workstations, without any external requirement of plug-ins or applets. As a result, optimized applications must be developed. In response to that, this work optimizes the calculation of point trajectories in n-dimensional vector fields over highly dense set of points, forcing the trajectories to lie over the points in the set. As a consequence, previously calculated data can be reused using hierarchical integration, avoiding unnecessary calculations and reducing the complexity of the algorithm.

3. Methodology

The problem that we address in this work is: given a set of points and a vector field that exists for all of these points, the goal is to find the finite trajectory that each point will reproduce for a certain period of time.
Normal calculation of point trajectories in n-dimensional vector fields, requires to perform numerical integration for each particular point in order to reproduce the paths. In the case of a dense set of points, the procedures suffers from unnecessary step calculations of the integrals, since several points in the field might lie over the same trajectory of others. Hence, some portions of the paths might be shared. Figure 11.2 illustrates this situation.

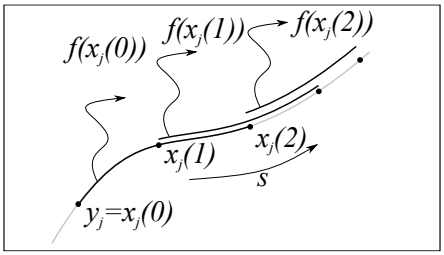


FIGURE 11.2. Trajectory overlapping in several point paths.

In order to avoid repeated computations, we propose a methodology to calculate trajectories of highly dense grid of points, in which the paths are forced to lie over the points in the grid, i.e., the paths are generated as

a PL topological connection between a set of points that approximates the trajectory. With this, hierarchical integration [**176**] is employed to iteratively compute the paths and reuse data through the iterations.

3.1. Hierarchical Integration. Since line integration over n-dimensional vector fields suffers from repeated calculations, hierarchical integration [**176**] only calculates the necessary steps and then iteratively grows the integrals reusing the data. This reduces the computational complexity of the algorithm from $O(N)$, using serial integration, to $O(\log N)$. The procedure is summarized as follows.
For an arbitrary point in the field $y \in Y$ with $Y \subseteq \mathbb{R}^n$, let us define $f : Y \to \mathbb{R}^m$, as an arbitrary line integral bounded by its trajectory c_y. Consider its discrete approximation as described in equation 47.

$$f(y) = \int_{c_y} w(x(s))ds \approx \sum_{i=1}^{t} w(x(i * \Delta s))\Delta s \tag{47}$$

where t is the maximum number of steps required to reproduce c_y with Δs the step size. $x(0) = y$ is the starting point of the trajectory to be evaluated and w is the function to be integrated. The integration procedure is performed for all points $y \in Y$ in parallel.

We assume that $\Delta s = 1$, $\forall y \in Y$ and therefore $f(y) \approx \sum_{i=1}^{t} w(x(i))$. The algorithm starts with the calculation of the first integration step for all the points in the field. Namely,

$$f_0(y) = w(x(1)) \tag{48}$$

It is required to store the last evaluated point $x(1)$ over the growing trajectory and the partial value of the integral for all the points y in order to reuse them in the following steps to build the integral. With this, the next action is to update the value of the integral, using the sum of the previously calculated step at y and the step evaluated at its end point ($x(1)$). Namely,

$$f_1(y) = f_0(x(0)) + f_0(x(1)) \tag{49}$$

In this case, the end point of $f_1(x(0))$ is $x(2)$ as the calculation evaluates $f_0(x(1))$. Therefore, the next iteration must evaluate f_1 at $x(0)$ and $x(2)$ in order to grow the integral. In general, the k'th iteration of the procedure is calculated as follows:

$$f_k(y) = f_{k-1}(x(0)) + f_{k-1}(x(end)) \tag{50}$$

It is important to remark that each iteration of this procedure evaluates two times the integration steps evaluated in the previous iteration. As a result, the total number of integration steps t is a power of two, and the hierarchical iterations required to achieve this evaluations is reduced by a logarithmic scale, i.e., $k = \log_2 t$. Also notice that the evaluation of the vector field is preformed only once, in the calculation of the first step, which avoids the an unnecessary evaluation of the vector field, which are computationally demanding for complex vector fields. Figure 11.3 illustrates the procedure up to four hierarchical steps.

3.2. Stream Path Calculation. In order to perform the visualization of a vector field using trajectory paths, lets assume a homogeneously distributed set of points

$$Y = \{y,\ z \in \mathbb{R}^n | y - z = \Delta y, \quad \Delta y \text{ is constant } \forall z \text{ adjacent to } y\} \tag{51}$$

and a n-dimensional vector field $F : \mathbb{R}^n \to \mathbb{R}^n$. The goal is to calculate for each point $y \in Y$, the PL approximation of the trajectory that the point will describe according to F, defined by the topological connection of a particular set of points $A_y \subset Y$. Figure 11.4 illustrates the approximation.

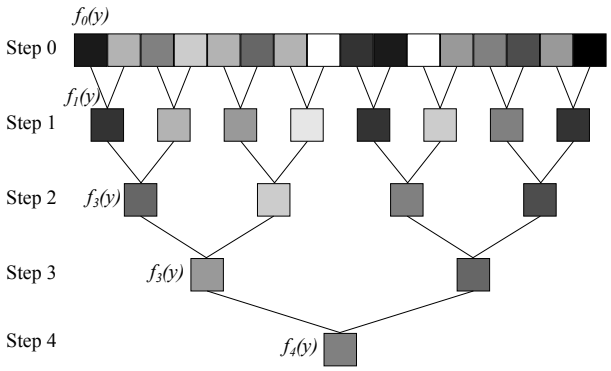


FIGURE 11.3. Exponential growth of hierarchical integration methodology. At step 3, the procedure evaluates 8 serial integration steps, meanwhile at step 4 it evaluates 16 serial integration steps.

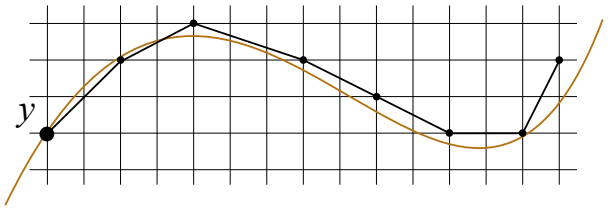


FIGURE 11.4. PL approximation of the trajectory by the procedure.

The trajectory c_y of an arbitrary point y in the field is defined as

$$c_y = x_y(s) = \int_l F(x_y(s))ds \tag{52}$$

where l represents a determined length of integration.
Using hierarchical integration, for each point in the field the first step of the PL trajectory is calculated, this is, the corresponding end point of the first step of the integral is computed using a local approximation of the point in the set of points.

$$x_y(0) = y \tag{53}$$

$$y' = x_y(0) + \gamma F(x_y(0)) \tag{54}$$

$$x_y(1) = \underset{x_y}{arg\,\min}(Y - y') \tag{55}$$

where y' is the first iteration result of the Euler integration procedure, γ is a transformation parameter to adjust the step given by the vector field to the local separation of the set of points and $x_y(1)$ is defined as the closest point in Y that approximates y'. The value of $x_y(1)$ is then associated (and stored) to y. The set A_y contains the reference to the points of the trajectory that describes y, and therefore for equation 54, A_y is defined as:

$$A_y^0 = \{x_y(1)\} \tag{56}$$

Similarly to the hierarchical integration procedure, the next steps are performed to join the calculated steps in order to grow the trajectories. Therefore, for each point y, its computed trajectory is joint with its last

point's trajectory, this is, for the step in equation 54.

$$A_y^1 = A_y^0 \cup A_{x_y(1)}^0 = \{x_y(1), x_y(2)\} \tag{57}$$

Note that each iteration of the procedure will increase the number of points in the trajectory by a power of two. Therefore, the growth of the paths is exponential. In general, the k'th iteration is calculated as

$$A_y^k = A_y^{k-1} \cup A_{x_y(k^2)}^{k-1} = \{x_y(1), x_y(2), \ldots, x_y((k+1)^2)\} \tag{58}$$

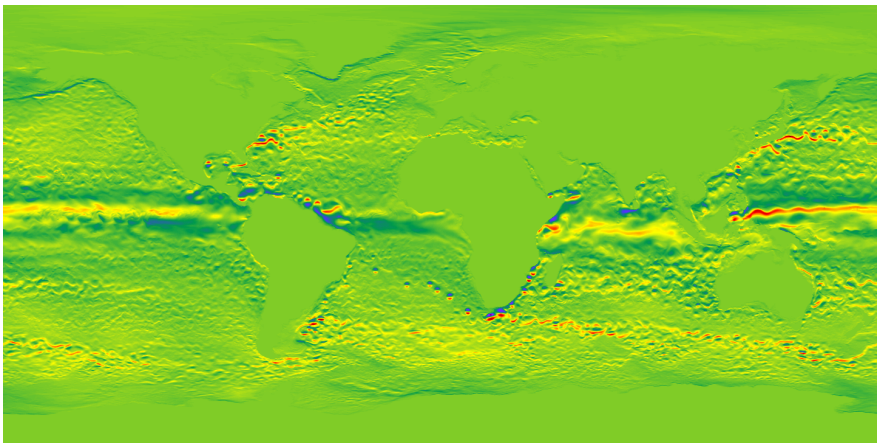
The accuracy of the procedure is strongly determined by the discretization (density) of Y, since it is directly related to the step size in the integration procedure, i.e., the approximation of the first step end-point is determinant. In order to increase the accuracy of the procedure, the computation of the first step can be calculated with e.g., a 4th order Runge Kutta numerical integration, however, it might significantly increase the computation time of the procedure if the computation time of the vector field function is relevantly high.

3.3. Time-Varying Data. In terms of unsteady flow fields, i.e., time-varying vector fields, the generation of the trajectories might seem difficult. In that case, as proposed in [**176**], time is considered another dimension of the vector field. Therefore, the set of points is formed with the position coordinates of the points and discretized time steps, producing an $n+1$-dimensional grid of points.
It is also determinant for the accuracy of the procedure that the density of the discretization set is high, in order to increase the precision of the approximated trajectories.

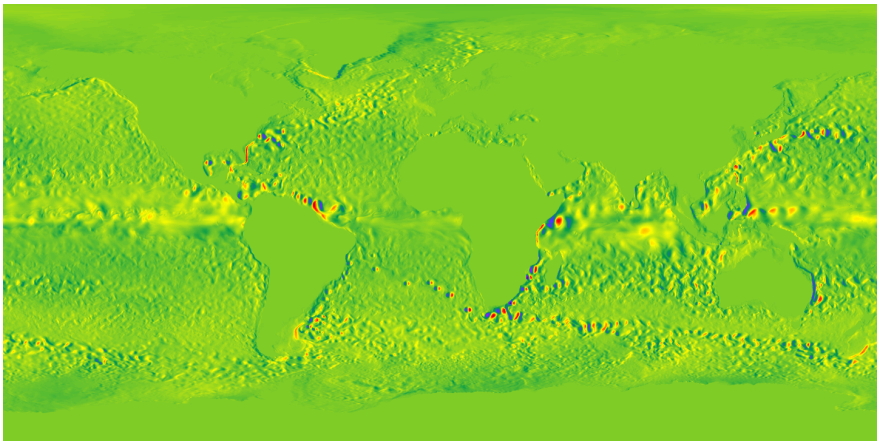
3.4. Animation. Dynamic scenes are demanding in most of the visualization procedures. We consider in this section two kinds of dynamic scenes. A first kind of procedures refers to when the vector field is steady, i.e., it remains constant through the time. In this case, the goal is to visualize the motion of the particle all across the field.
Since the paths for all the points in the field are calculated, the representation of the particle's trajectory through the frames is simple. Consider a point y and its approximated trajectory given by the set of points A_y. Notice, as described in section 3.2, that the first point of the set $A_y\,[1]$, i.e., $x_y(1)$, represents the next point in which y will lie in a determined period of time. As a result, at a posterior frame, the displayed trajectory should be $A_{x_y(1)}$.
The second type of procedure refers when vector field is varying with the time. Complementary to the animation stated before, a second kind of dynamic scene is comprised since it is also important to visualize the changes that a trajectory suffers in the time. In the case of time varying data, as in the steady case, all the points have an associated trajectory. In order to animate the change of one trajectory, from one frame to another, the trajectory that will be represented refers to the one of the point with the same point coordinate, but the next time coordinate. i.e., $A_{y,t+\Delta t}$.

4. Case Study

In this section the visualization of 2D oceanic currents using the proposed methodology is performed. The implementation has been done in WebGL, so the methodology's parallel computing capabilities are fully used. WebGL offers the possibility to use the rendering procedure to calculate images (textures) through Framebuffer Objects, and then use those rendered textures as input images for other rendering procedures. As a consequence, for this implementation we associate the pixels of an image to the points in the field, and therefore, the rendering procedure is used to compute the different hierarchical iterations, which are stored in the color values of the pixels. Finally, the trajectories are hierarchically constructed. The implementation was performed on an Intel Core2Quad 2.33 GHz with 4 GB of RAM and with a nVidia GeForce 480.



(a)



(b)

FIGURE 11.5. Global information. Gray-scale of the oceanic currents magnitude in the (a) longitudinal and (b) latitudinal directions, of the 1^{st} January of 2010. Images from the NASA's ECCO2 project.

4.1. Implementation. For a $w \times h$ grid of points (w and h being its width and height respectively in number of elements), images of size $w \times h$ in pixels are used, in which a particular pixel (i, j) is associated with the point (i, j) in the grid. Since for each particular pixel, a four component vector is associated, i.e., a vector of red, green, blue and alpha values, each value can be associated as a particular position of another pixel. This is, if the value of a pixel is r, then its associated pixel coordinates are given by

$$i = r \bmod w \tag{59}$$

$$j = \frac{r - i}{w} \tag{60}$$

where mod represents the remainder of the division of r by w. As a result, if for each hierarchical integration, only the last point of the trajectory is to be stored, then one image can store four hierarchical iterations.

For the zero'th hierarchical iteration, and the image I to store its calculation, the value of a pixel (i, j) is given by

$$i_0 = i + kF_i(i, j) \tag{61}$$

$$j_0 = j + kF_j(i, j) \tag{62}$$

(63)

where the parameter '0' refers to the hierarchical step 0, k represents the scaling factor of the vector field, and $F_i(i, j)$ represents the component of the vector field over the direction of i, evaluated at the point (i, j). The vector field used in this case study is shown in figures 5(a) for the direction of i and 5(b) for the direction of j.
In general, the k'th step is calculated as follows,

$$i_{next} = I(i, j, k-1) \bmod w \tag{64}$$

$$j_{next} = \frac{I(i, j, k-1) - i_{next}}{w} \tag{65}$$

$$I(i, j, k) = I(i_{next}, j_{next}, k-1) \tag{66}$$

In the case that k is greater than four, then more images are used to store the values of the hierarchical iterations.
With that, all the desired steps are stored in the necessary images. In order to build the trajectories from those images, a generic line, formed by k^2 points, is required. Each point in the trajectory needs to have an associated value, that refers to its order in the trajectory, i.e., the first point in the trajectory has an index 0, the second point the value 1 and so on. With this index associated to each point of the trajectory of the point y, the position of each point is calculated as follows.

$$while(1)\ do \tag{67}$$

$$HL = floor(\log_2(index)) \tag{68}$$

$$index = index - 2^{HL} \tag{69}$$

$$y = evalHL(HL, y) \tag{70}$$

$$if(index < 0.5) \tag{71}$$

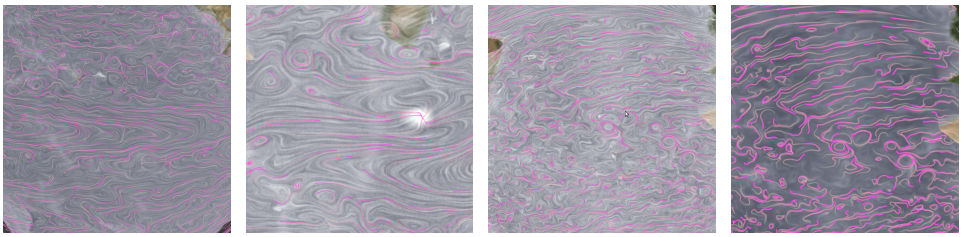
$$y_{end} = y \tag{72}$$

$$Finish \tag{73}$$

(74)

Where HL is the hierarchical level that needs to be evaluated and the function $evalHL()$ returns the new position of y a point, for a particular hierarchical level.

4.2. Results. A general grid of 2048×2048 points is used as the world's discretization. The vector field information was acquired from the NASA's ECCO2 project (see figure 11.5), in which high-resolution data is available. Only six hierarchical levels, i.e., $2^6 = 64$ points in the trajectory are used for each point in the field, as a result only 2 images are required to calculate the trajectories.
The time needed to compute all the hierarchical levels (from 0 to 6) was 3 ms. The visualization was performed to 10000 equally-distributed points all over the field, which means that 64000 points need to be transformed by the trajectory computation. The computation time of those trajectories was 670 ms (Final results are shown in Figure 11.6).
In order to compare the visualization with the methodology, the LIC visualization of the vector field using the methodology that we proposed in [**148**] was used as a background image. It shows that for this level of



(a) (b) (c) (d)

FIGURE 11.6. Comparison between different visualizations of the oceanic currents using 6 hierarchical steps.

discretization (2048×2048), the visualization is correct and accurate. However, sparse data might produce inaccurate results.

5. Conclusions and Future Work

In this paper, we have presented a novel methodology to calculate trajectories of points in highly dense point sets, in which the trajectories are formed as piecewise-linear connections of the points in the set. This allows to merge the calculation of the different trajectories, and use iteratively the data to avoid repeated and unnecessary calculations. The procedure is suitable to be implemented in parallel architectures such as OpenCL or CUDA, since the calculations of the integrals at any moment is independent from the calculation of the other points.

As a result, thanks to the use of hierarchical integration, the procedure reduces the computational complexity of the calculation of the trajectories from linear to logarithmic. The methodology deals with n-dimensional and time-varying data and animated visualization can be easily achieved due to the fact that the trajectories are calculated for all the points in the set.

Since the procedure performs an approximation of the trajectories using a piecewise-linear connection of the points in the set, the accuracy of the algorithm is strongly influenced by the discretization distance between the points, because this distance determines the minimum integration step to be used.

An active procedure in order to adjust the position of the points in the grid through the iterations, and then increase the accuracy of the calculated trajectories is proposed as future work of this article.

CHAPTER 12

Web Based Hybrid Volumetric Visualisation of Urban GIS data Integration of 4D Temperature and Wind Fields with LoD-2 CityGML models

- John Congote [1,2]
- Aitor Moreno [2]
- Luis Kabongo [2]
- Juan-Luis Pérez [3]
- Roberto San-José [3]
- Oscar Ruiz [2]

[1] CAD CAM CAE laboratory, Universidad EAFIT
Carrera 49 No 7 Sur - 50, Medellín, Colombia
[2] Vicomtech Research Center
Donostia - San Sebastin, Spain
[3]Environmental Software and Modelling Group
Technical University of Madrid (UPM), Spain

Context

Web based hybrid volumetric visualisation of urban GIS data. Congote, J., Moreno, A., Kabongo, L., Perez, J. L., San-Jose, R., Ruiz, O. published on-line: 24 October 2012. DOI: 10.1051/3u3d/201203001, Event 3u3D2012 Usage,Usability, and Utility of 3D City Models European COST Action TU0801 2012. EDP Sciences. CONFERENCE

This work represent the result of a practical and functional implementation of the research of this thesis, applying all the previous methodologies of surface extraction, volume visualization, vector field visualization, and standard formats into a unified architecture.

This work was supported by COST Action TU0801 “Semantic Enrichment of 3D City Models for Sustainable Urban Development”, the Basque Government’s ETORTEK research programme, the EAFIT University and the Colombian Council for Science and Technology (COLCIENCIAS). The authors thankfully acknowledge the computer resources, technical expertise and assistance provided by the Centro de Supercomputación y Visualización de Madrid (CESVIMA) and the Spanish Supercomputing Network. The authors also acknowledge the ESPAÑA VIRTUAL CENIT project funded by the Spanish Ministry of Science and Technology.

As co-authors of such publication, we give our permission for this material to appear in this document. We are ready to provide any additional information on the subject, as needed.

Prof. Dr. Eng. Oscar E. Ruiz
oruiz@eafit.edu.co
Coordinator CAD CAM CAE Laboratory
EAFIT University, Medellin, COLOMBIA

Dr. Aitor Moreno
amoreno@vicomtech.org

Vicomtech, San Sebastian, SPAIN

Dr. Luis Kabongo
lkabongo@vicomtech.org

Vicomtech, San Sebastian, SPAIN

Juan-Luis Pérez

Environmental Software and Modelling Group
Technical University of Madrid (UPM), SPAIN

Prof. Roberto San-Jos´

Environmental Software and Modelling Group
Technical University of Madrid (UPM), SPAIN

Abstract

City models visualisation, buildings, structures and volumetric information, is an important task in Computer Graphics and Urban Planning. The different formats and data sources involved in the visualisation make the development of applications a big challenge. We present a homogeneous web visualisation framework using X3DOM and M3DX3DOM for the visualisation of these urban objects. We present an integration of different declarative data sources, enabling the utilization of advanced visualisation algorithms to render the models. Its tested with a city model composed by buildings from the Madrid University Campus. Volumetric datasets coming from Air Quality Models and 2D layers wind. Results of visualisation of all the urban models in real time on the Web, achieving interactive rates. An HTML5 web interface is presented to the user, enabling real time modifications of visualisation parameters. This work also addresses usability issues of the Framework when time based dataset are presented to the user.

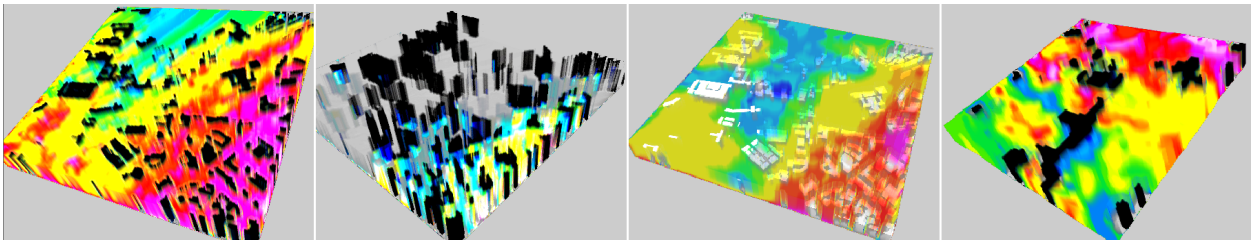


FIGURE 12.1. Different results of the proposed hybrid visualization web application, showing a temperature scalar field combined with georeferenced buildings.

1. Introduction

Scientific visualisation of city scale models, such as geometry, terrain, volumetric data and others are a big challenge in computer graphics. These models are represented in several markup languages such as CityGML, X3D, COLLADA, DEM for geometry. NetCDF, Raw data for volumetric datasets or special compressed as JPEG, GeoTIFF formats for images. Some of this formats are rarely used and only some specialized software can work with these formats. The Web with the new standards as XML or HTML5 allows the inclusion of different models into a common framework, but the restrictions presented in the Web architecture difficult the implementation of the algorithms necessary for the correct visualisation of this models.

Volume rendering is a model of scientific visualisation. The methodology is used in several areas like medical imaging, geo-visualisation, engineering, and almost any other field, which deals with n-Dimensional datasets. The method renders an image of a volume using the metaphor of passing light through semi-transparent objects. This grants the identification of internal structures, which are occluded by the external levels of the volume. The colour of the object is normally arbitrary and generated by a false colour model that is user assigned for each dataset and the region of interest of the volume.

Our contribution is the visualisation of air quality models, wind fields, buildings in a common framework using Declarative 3D technologies such as X3DOM and MEDX3DOM. Modification over the frameworks are proposed to work with these datasets. Also brief description of the volume rendering techniques. Then, the followed methodology to load AQM datasets into a volume rendering Web application and integrated with georeferenced geometry.

2. Related Work

2.1. Air Quality Information visualisation. Air quality visualization has special requirements respect to other kind of information. Large masses of spatial data have to be managed by the visualization system. 2D maps and temporal series are usually used for the visualization of the air quality data produced by the AQMs. These models generate spatial and temporal information, so most of the vector and scalar fields we wish to visualize vary with time. Animation of atmospheric data has been used to examine changes in the data over time. In case of wind data, they are visualized through vectors with a magnitude and a direction.

Air pollution and meteorology modelling are disciplines where 3D visualization is beginning to become very important subjects. The scientifics need tools to visualize and animate the data produced by their simulations. New visualization techniques allow to improve the level of understanding of the chemical and physical atmospheric process. Also, the employ of 3D displays and fast visualization tools help to navigate through the data.

2.2. Volume visualisation. In computer graphics, Ray-casting is a well known direct volume rendering technique that was designed by Kajiya and Herzen [**127**] as one of the initial developments in this area. Traditionally, three dimensional objects have been created by using surface representations, drawing geometric primitives that create polygonal meshes [**89**], hence provoking the loss of information from one dimension. Ray casting suitably fits GPUs' operating mode [**190**], because of the independence of each ray that is launched to the scene, making this algorithm highly parallelisable and allowing the exploitation of GPUs' parallel architecture. For GPU ray casting, the volume element is stored in the GPU memory as a 3D texture and a fragment shader program is used in order to implement the ray casting algorithm. visualisation of volumes on the web had been implemented and by Congote et al. [**120**].

2.3. Flow visualisation. A great amount of methodologies to visualize vector fields (flow fields) has been developed among the last decades. Geometric-based approaches draw icons on the screen whose characteristics represent the behaviour of the flow (as velocity magnitude, vorticity, etc). Examples of these methodologies are arrow grids [**177**], streamlines [**178**] and streaklines [**179**]. However, as these are discrete approaches, the placement of each object is critical to detect the flow's anomalies (such as vortexes or eddies), and therefore, data preprocessing is needed to perform an illustrative flow visualisation. An up-to-date survey on geometric-based approaches is presented by [**180**]. However, in terms of calculating those trajectories for determined points in the field, the procedures usually compute for each point the integrals, and, as a result, the procedures are computationally expensive for highly dense data sets. Visualisation of flows in the web had been implemented by Aristizabal et al. [**148**]

2.4. Hybrid visualisation. There are several models for combining polygonal geometry and volume rendering, normally this methods use a blending between the image results, making these approaches unfeasible for city models, where large sections of the model could be occluded, this problem can not be solved with these techniques. Some methods identify the intersection between rays launched in the volume rendering process and geometry [**173**]. Ray casting is a flexible algorithm that allows the implementation of acceleration methods, such as Empty Space Skipping [**102**] or *Early Ray Termination*. Early ray termination is an optimization process that establishes certain limitations in the volume, samples encountered after them do not contribute to the value of the pixel [**90**]. Geometry is rendered in the first place to correctly look at the intersections of the geometry and the volume. Besides, parts that are occluded by the geometry should not contribute to the final image, not performing any ray casting at all. In order to achieve this feature, rays should terminate when they hit a polygonal object, accordingly modifying the ray length image if a polygonal object is closer to the view point than the initial ray length. Hybrid visualisation on web had been proposed by Ginsburg et al. [**149**]

2.5. Web 3D Rendering and MEDX3D. The use of the recently released WebGL standard [**96**] leads to new methods for web 3D visualisation, where most part of the computational processes are performed in vertex and fragment shaders that run on the GPU hardware. WebGL is a software library that enables

HTML5-based browsers to identify clients' graphics hardware. HTML5, the latest Internet standard propose, provides native elements for audio and video. WebGL consists of a ow-level imperative graphic programming API based on OpenGLES 2.0 for Javascript that enables flexibility and exploits the characteristics of advanced graphics cards. Due to the constant improvement of the performance of Javascript interpreters, the management of scene elements behaves similarly to the ones obtained by using natively compiled languages. Moreover, some WebGL extensions have been implemented in order to achieve a friendly interaction, such as SpiderGL [**130**].Several standards and proprietary solutions are currently being developed in order to fulfill the necessity of moving 3D visualisation into the web [**111**], such as X3D, a standard derived from VRML that stores 3D information in a scenegraph format using XML (Extensible Markup Language). This model has been implemented in a declarative form, as an extension of HTML; X3DOM presents a framework for integrating X3D nodes into HTML5 DOM content [**103**] and other alternatives have also been developed, e.g. XML3D [**131**]. Finally, there is a standardization for X3D in the MedX3D volume rendering model [**121, 122**]. The MEDX3DOM proposal is presented by Congote [**125**] to visualize volumetric datasets in a declarative model.

3. Methodology

MEDX3DOM implements the standard *MEDX3D* in *X3DOM*. The implementation is based on the generation of the nodes for two components: *Texturing3D* and *VolumeRendering*. The actual status of the implementation declares all nodes for *X3DOM* architecture. Texturing3D nodeset define the tags to work with the information of 3D data, specific readers for file formats such as *DICOM*, *NRRD* and raw data are part of the standard. Other kind of data sources could be implemented like *WADO*, which is an specification for a WebService to obtain *DICOM* data or *NetCDF* to read scalar data. 3D texture data is normally stored in a **Texture3D** structured which is part of common 3D APIs like *OpenGL*. This structure is not available in WebGL and a mapping between a Texture3D and a Texture2D was define in order to overcome this limitation.

Volume data is storage in atlas format. An example for the volume dataset represented in this format is shown in figure 12.3. This model of representation also allows the storage of preprocessed data such as vector fields, where each color magnitude represent the direction of the gradient vector for each axis. Given the flexibility of the *X3DOM* architecture the transfer functions can be store as images or generated by the application.

The implementation of the MEDX3D standard into X3DOM allows the declaration of volume visualisation in X3D and visualized the render in browser without the use of plugins. The files follow the standard MEDX3D but the image texture is defined as a **Texture2D** structure. The file format is presented in the Figure 12.2, were an extraction of and HTML5 file is presented were a volume is rendered in a web page.

3.1. Flow Visualisation Style. A new style is proposed for the visualisation of vector field, the *StreamLineFlowStyle* allow the visualization of vector fields such as winds by transforming the vectors to a geometry representation know as stream lines. This lines are modeled as geometry internally for the MEDX3DOM extension. The visualisation of geometry and volumetric datasets in a common render pass is know as Hybrid visualisation. Other flow visualisation styles could be implemented where no geometry proxy is generated. Composition methods for visualisation of several volumes simultaneously should be implemented.

3.2. Hybrid Visualisation. The ray casting process has been improved to allow the integration of geometry. Each ray traverses the volume, where the volume dataset are represented. But the ray traversal will take into account any geometry inside the volume, so the ray casting termination condition has been modified. The termination condition is based on the alpha value for each sample of the transparency colour of the ray.

```
<html>
<head> ... </head>
<body>
<h1>Volume Rendering</h1>
<X3D xmlns='http://www.web3d.org/specifications/x3d-namespace'
  showStat='true' showLog='true' width='500px' height='500px'>
  <Scene>
    <Background skyColor='0 0 0'/>
    <Viewpoint description='Default' zNear='0.0001' zFar='100'/>
    <Transform>
     <Shape>
         <Inline url="buildings.x3d"> </Inline>
     </Shape>
     <VolumeData id='volume' dimensions='4.0 4.0 4.0'>
        <ImageTexture containerField='voxels'
         url='aqm/aqm_frame0.png'/>
<OpacityMapVolumeStyle>
<ImageTexture url='aqm/transferFunction.png'/>
</OpacityMapVolumeStyle>
      </VolumeData>
     <VolumeData id='volume' dimensions='4.0 4.0 4.0'>
        <ImageTexture containerField='voxels'
         url='aqm/aqm_wind_frame0.png'/>
<StreamLineFlowStyle >
</StreamLineFlowStyle>
      </VolumeData>
    </Transform>
  </Scene>
</X3D>
</body>
</html>
```

FIGURE 12.2. HTML/X3DOM proposed file fragment to visualize a the first frame of a AQM volumetric dataset using *MEDX3DOM*. The texture image is defined in atlas format (Figure 12.3).

If the value to 1, it means that the ray has reached the final colour, and therefore, the remaining steps of the ray are not evaluated. Our approach to take into account the geometry of the buildings is to provoke an early termination of the ray casting, detecting when the ray collides with a geometric element. In that case, the alpha value is set to 1, and the finalisation of the ray casting is triggered.

4. Results

In this work, our goal is to experiment with the AQM dataset and check if they can be loaded and rendered using volume rendering techniques. For such goals, we have selected some dataset provided by the Madrid Technical Univesity (UPM). In the following sections, we will describe the process to convert the AQM dataset into the required images for the volume rendering engine, enhancing the significant differences

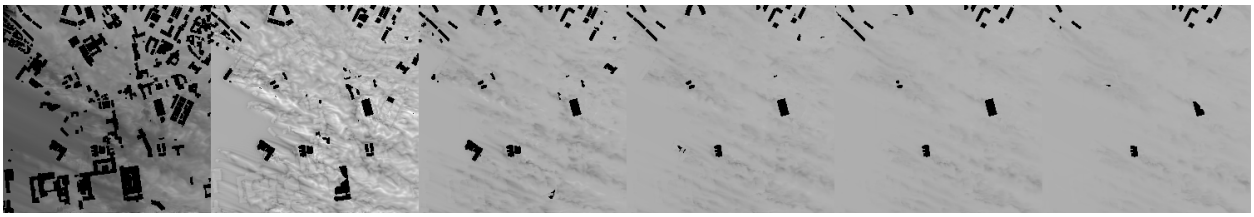


FIGURE 12.3. Texture atlas with the volume data for the AQM dataset

of such datasets to the classical volumetric datasets. In the last sections, the HTML interface for the Web application will be described, including the hardware and software platform used for the tests.

4.1. Dataset description and loading stage. Turbulent urban flow simulations require Computational Fluid Dynamics models. In this research the datasets have been obtained from large scale numerical experiments to simulate turbulent fluxes for urban areas with the EULAG (UCAR, US) [**191**] CFD model, which was modified to include an energy balance equation to obtain the urban fluxes. In the Figure 12.3 we can see a sequence of 2D map of the Potential Temperature (K) simulated by the EULAG model.

The dataset represents the temperature and winds fields over a region of Madrid, which includes some buildings. The coverage of the data is around 1 km $\times$ 1 km., with a spatial resolution of 4m, giving a 250 $\times$ 250 regular grid. Also, the dataset covers 100 m. in altitude with the same spatial resolution, resulting in a volumetric dataset of 250 $\times$ 250 $\times$ 25 values for each variable. Temperature is a scalar value and wind is a vector field with two components horizontal (u) and vertical (v). The simulation time period is 5.5 minutes with outputs every 30 seconds. The computational time of this experiment is about 72 hours of CPU time with 64 processors.

Each line of the data file contains the X, Y and Z coordinates and the value, using a specific value (-1.E+34) for the "non existing values" (black pixels in Figure 12.3, normally associated to the places where the buildings are located. Once the data has been loaded, individual images per Z level can be created and saved for further use. In this case, we have mapped the range 298K to 313 K to the 256 possible values of a byte.

4.2. Volumetric ray casting for AQM. Once the dataset has been converted to a set of Z-ordered images (representing the scalar 3D field), some extra steps have to be done to be successfully loaded by the volumetric ray casting techniques. Essentially, the AQM dataset are very similar to the common medical datasets (see Figure 12.3), but there are some differences. First, the typical AQM dataset are not exactly a cubic grid, finding that the length in the Z axis (altitude) is less than in XY space. Furthermore, the typology of geodata in the urban areas are biased, i.e., the most interesting data structures are near the ground level, leading to a quite limited number of Z-slices with interesting information. This effect can be seen in the Figure 8, where the first four Z-slices contains almost all the information of the dataset (composed of 25 levels), a 16% of the whole volume. It is worth to mention that the transfer functions are used to assign optical properties (colour and opacity) to the original volume data values in order to improve the visualisation of the internal parts of the volume data. In volume ray casting, the transfer functions are used to obtain the optical properties of the volume data at each ray step. These values are blended using the composition function. In general, two transfer functions are defined, the colour transfer function, which obtains an RGB value and the opacity transfer function, which obtains a transparency value.

4.3. HTML Interface description. We implemented a HTML5 user interface using jQuery (see Figure 12.4) to interact with the AQM datasets, providing standard functionality to navigate the datasets at

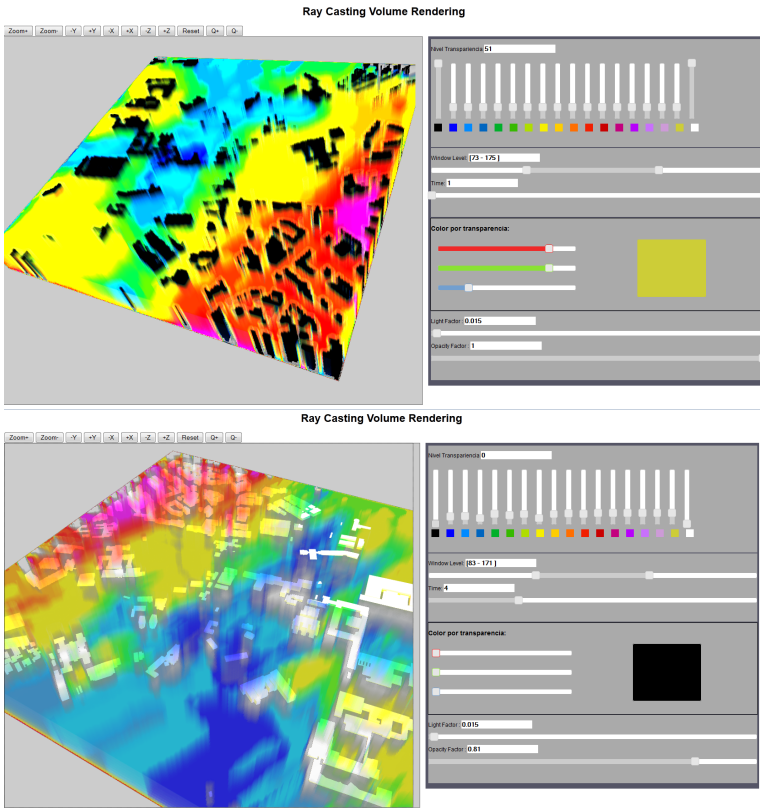


FIGURE 12.4. Different configuration of the HTML5 user interface and the achieved visual output.

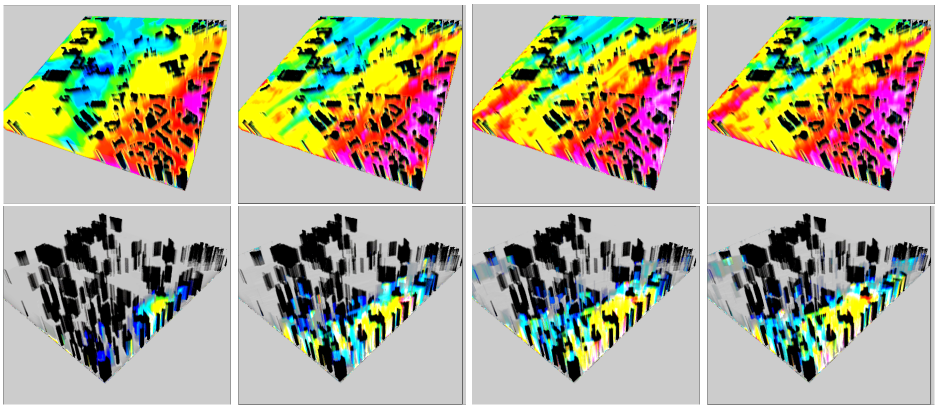


FIGURE 12.5. Two temporal series of the same datasets with different transfer function and window level configuration.

different level. First, we provide mouse interaction in the HTML5 canvas through X3Dom custom navigation techniques. Also, some buttons in the upper part of the web page can be used to navigate inside the volume (zooming and panning). The time navigation is provided through a slider, used to select the frame of the volume datasets (see Figures 12.5 and 12.6). The animation can be set to manual or automatic, iterating continously between the frames of the datasets.

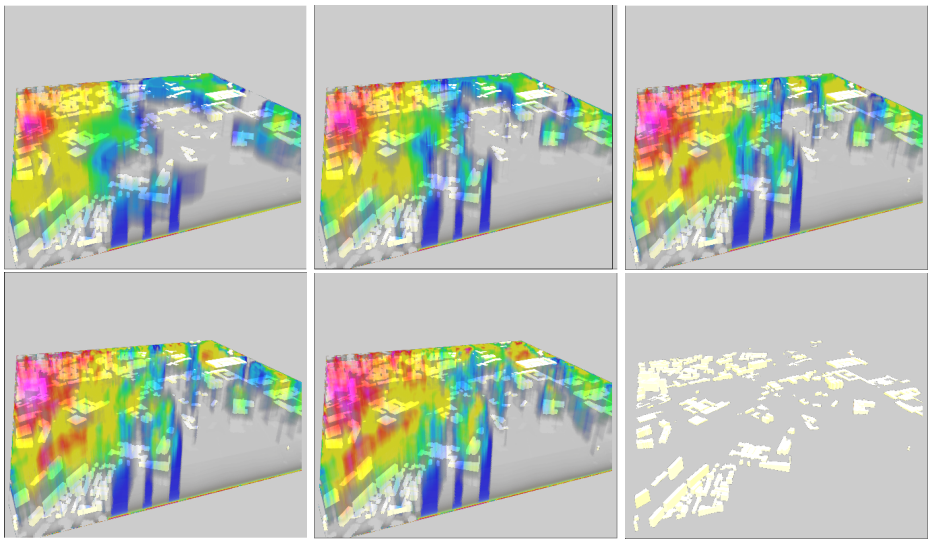


FIGURE 12.6. A set of frames of the hybrid visualisation of the volume and the buildings geometry (shown independently in the last subimage).

The Transfer Function configuration has been implemented combining a set of sliders, colours and the window level range configuration. There are 2 special sliders, fixed to the 0 and 255 values. The remaining 16 sliders maps the value range [1, 254] but it can be modified with the Window Level range editor, implemented as a range slider in jQuery. In the default settings, each sliders modifies the color and transparency of 16 values $(256/16)$. Reducing the window level to the range [64, 128], the 16 regular sliders will adjust the transfer function to that specific range, so each slider will map just 4 values ($(128-64)/16$). In the AQM dataset sample (Figure 12.5), the scalar values are defined in the 298 K to 313 K range (15 K), so each channel range is nearly 1 K with a full range window level. Choosing the [64, 128] window level will provide a visualisation of the range [302 K, 306 K], with a resolution of 0.25 K for each channel or slider.

The interface also displays two extra sliders, supporting global modifications of the light factor and the transparency. The global transparency is a handy way to control the overall transparency in all the channels. The light factor enhances the contrast of the colours, making more visible and noticeable the inner structures of the volume datasets.

4.4. Buildings Visualisation. The buildings model has been reconstructed from tagged SHP files, where each building corresponds with a 2D polygon, having their height specified as a feature in the DBF file. The reconstructed 3D model is equivalent to the LoD 2 proposed in the CityGML specification, although a X3D file was created to be later inlined in the X3D scene (see Figure 12.2).

Using the hybrid integration method for volume rendering allows us to visualise the buildings in the volume, as it can be seen in the Figures 12.6 and 12.7.

4.5. Wind Flow Visualisation. The wind information, provided as u and v components for each voxel in the dataset, were processed with the methods presented by Aristizabal et al. [**148**]. The result is a set of geometric streamlines, converted to X3D file format. The final step involved the integration of such X3D model as an inline node into the X3D scene (see Figure 12.2).

The integration of the streamlines with the volume dataset has been performed in the same way as the buildings, as they are not conceptually different, but some precautions had to be taken into account. A

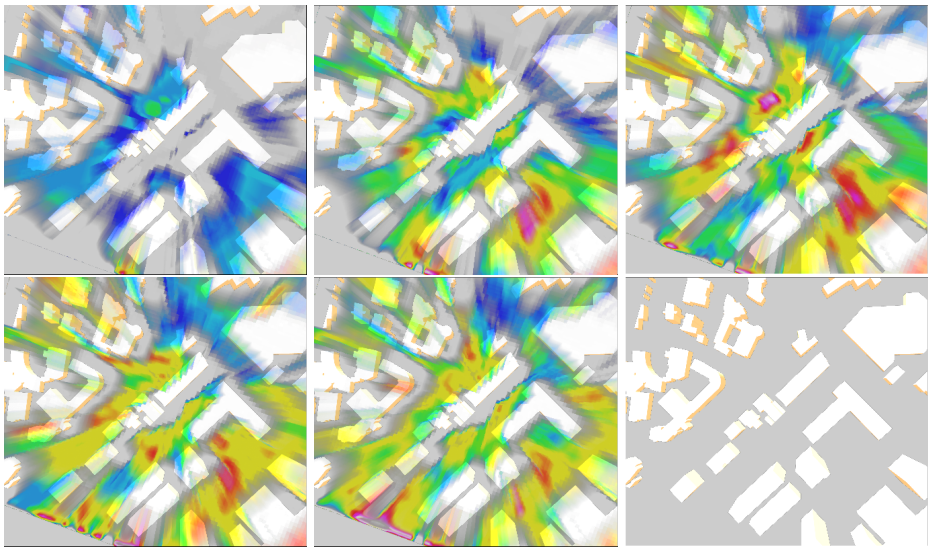


FIGURE 12.7. Top view of the volume dataset combined with the buildings geometry. The set of frames shows the temperature evolution of a small part of the urban area (shown independently in the last subimage).

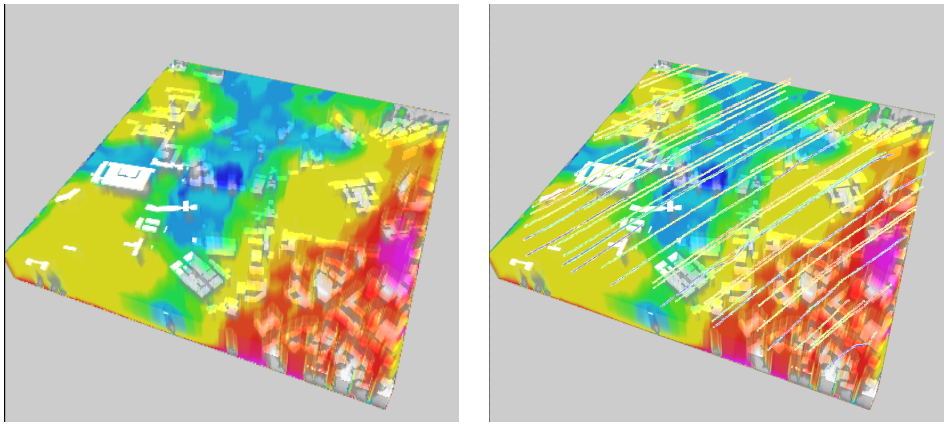


FIGURE 12.8. Hybrid visualisation of temperatures dataset (on the left, just the volume rendering and the buildings are shown) and wind information, reconstructed as colored geometric lines. For clarity purposes, only a small subset of wind flow lines have been added to the image (right).

full reconstruction of the complete vector field is so dense that the visualisation turns to be not usable and therefore, not practical. A simplification was performed to produce a smaller set of streamlines. Even with such reduction, we have reduced it even more to achieve better graphical results for the Figure 12.8.

4.6. Testing platform. The tests have been performed in an Intel Quad Core Q9400 processor, 4GB of RAM and a GeForce GTX 275, Windows 7 PRO 64 bits with the latest stable graphics drivers. Amongst all of the Web browsers with full implementation of WebGL standard, we selected Firefox 12.0 for the tests, although other browsers are known to work with the implementation like Chrome 19 and Opera 12.00 alpha. Both Chrome and Firefox, in default configuration, use Google's Angle library to translate WebGL's native

GLSL shaders to Microsoft's HLSL language and compile and run them through the DirectX subsystem. This procedure improves compatibility with lower-end hardware or older graphics drivers. Firefox also works if Angle is disabled; setting the native OpenGL subsystem as the rendering back-end. The Web application is served through a LightTPD Web server, installed and configured in the same computer for this experiment.

5. Conclusions and Future Work

This work has presented a Volume Rendering Web application to visualize 3D scalar and vector fields coming from AQM. An HTML5 interface has been provided to enable the users to configure the parameters of the visualisation and to navigate through the datasets (spatial and temporal navigation).

The integration of geometrical models (like buildings or wind streamlines) and the volume dataset helps to contextualize the data, helping to analyze visually the results of the AQM simulations.

In this paper a hybrid visualization has been presented, combining temperature (scalar field), wind flow (vector field) and the LoD 2 buildings existing in the zone. The results show that such integration of georeferenced datasets, provided through an interactive and high-performance Web visualization module, enhances the global understanding of the datasets.

This work has also presented how visual analysis of temporal datasets can be performed interactively. Using the same settings for the transfer function and window level, the fast loading times and the visualization performance of the models allows the creation of smooth animations, presented in this work as sequential frames.

This work has been focused on the rendering of temperature datasets, but the AQM can produce simulation for a wide variety of pollutants. For this multivariate scenario, it will be important to address the mechanisms to allow the selection and the merging criteria of the different pollutants, as it would depend on the users to give more priority to one pollutant than others.

For such operations, the key point is how users can deal with the heterogeneity of the multivariate scenarios. Transfer functions are a fundamental tool for the correct visualisation of such volumes. Some of the future work can be oriented to the generation of transfer functions by visual clues, know as Visual Volume Attention Maps (*VVAM*) [**150**]. Some preliminary results has been already researched and these methodologies are going to be tested in *MEDX3DOM*.

Conclusions

The main result of this thesis is the test of all their modifications, improvements, and extensions of algorithms and methodologies into real world problems, providing a reliable set of new tools for the resolution of established problems in different fields. Also, the collaboration with standardization consortium and other research groups, proves the usability of the developments.

The models generated by the combined method of full-body and face reconstruction, shows human recognizable avatars.

A formal methodology for the selection of parameters for surface reconstruction algorithms significantly improves the quality of the results of the algorithm.

The modification of the Marching Cubes algorithm allows the use of this algorithm for dataset which were untreatable.

Volume rendering algorithms in heterogeneous architectures and low level devices can be effectively used in telemedicine and remote diagnostics.

Hybrid visualization enrich the content of images allowing the presentation of more data in the same image.

Bibliography

[1] John Congote, Iñigo Barandiaran, Javier Barandiaran, Tomas Montserrat, Julien Quelen, Christian Ferrán, Pere J. Mindan, Olga Mur, Francesc Tarrés, and O. Ruiz. Real-time depth map generation architecture for 3d videoconferencing. In *3DTV-Conference: The True Vision - Capture, Transmissionand Display of 3D Video (3DTV-CON), 2010*, pages 1–4, Tampere, Finland, June 2010.

[2] Minglun Gong, Ruigang Yang, Liang Wang, and Mingwei Gong. A performance study on different cost aggregation approaches used in real-time stereo matching. *Int. J. Comput. Vision*, 75:283–296, November 2007.

[3] Kuk-Jin Yoon and In So Kweon. Adaptive support-weight approach for correspondence search. *IEEE Trans. Pattern Anal. Mach. Intell.*, 28(4):650, 2006.

[4] Zheng Gu, Xianyu Su, Yuankun Liu, and Qican Zhang. Local stereo matching with adaptive support-weight, rank transform and disparity calibration. *Pattern Recogn. Lett.*, 29:1230–1235, July 2008.

[5] Asmaa Hosni, Michael Bleyer, Margrit Gelautz, and Christoph Rhemann. Local stereo matching using geodesic support weights. In *Proceedings of the 16th IEEE Int. Conf. on Image Processing (ICIP)*, pages 2093–2096, New York, NY, USA, November 2009. Elsevier Science Inc.

[6] Liang Wang, Mingwei Gong, Minglun Gong, and Ruigang Yang. How far can we go with local optimization in real-time stereo matching. In *Proceedings of the Third International Symposium on 3D Data Processing, Visualization, and Transmission (3DPVT'06)*, 3DPVT '06, pages 129–136, Washington, DC, USA, 2006. IEEE Computer Society.

[7] Daniel Scharstein and Richard Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *Int. J. Comput. Vision*, 47(1-3):7–42, 2002.

[8] Sebastiano Battiato, Salvatore Curti, Marco La Cascia, Marcello Tortora, and Emiliano Scordato. Depth map generation by image classification. In *Electronic Imaging 2004*, pages 95–104. International Society for Optics and Photonics, 2004.

[9] S Battiato, A Capra, S Curti, and M La Cascia. 3d stereoscopic image pairs by depth-map generation. In *3D Data Processing, Visualization and Transmission, 2004. 3DPVT 2004. Proceedings. 2nd International Symposium on*, pages 124–131. IEEE, 2004.

[10] Daniel Scharstein and Richard Szeliski. High-accuracy stereo depth maps using structured light. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 1:195–202, June 2003.

[11] F. Tombari, S. Mattoccia, L. Di Stefano, and E. Addimanda. Classification and evaluation of cost aggregation methods for stereo correspondence. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8, June 2008.

[12] Douglas C Montgomery. *Design and analysis of experiments*. Wiley, 2008.

[13] Cuthbert Daniel. Use of half-normal plots in interpreting factorial two-level experiments. *Technometrics*, 1(4):311–341, 1959.

[14] Russell V Lenth. Response-surface methods in r, using rsm. *Journal of Statistical Software*, 32(7):1–17, 2009.

[15] A. Hoyos, J. Congote, I. Barandiaran, D. Acosta, and O. Ruiz. Statistical tuning of adaptive-weight depth map algorithm. In Real, Diaz-Pernil, Molina-Abril, Berciano, and Kropatsch, editors, *CAIP 2011. Computer Analysis of Images and Patterns*, Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), pages 563–572, Seville, Spain, Aug. 29-31 2011. Springer. Berlin - Heidelberg. Vol. 6855 LNCS. ISBN 978-3-642-23677-8.

[16] Marc Levoy Greg Turk. Zippered polygon meshes from range images. In *ACM SIGGRAPH. Computer Graphics Proceedings, Annual Conference Series*, pages 311–318, July 1994.

[17] Stefano Marras, Fabio Ganovelli, Paolo Cignoni, Riccardo Scateni, and Roberto Scopigno. Controlled and adaptive mesh zippering. In *GRAPP - International Conference in Computer Graphics Theory and Applications*, 2010.

[18] Chen Shen, James O'brien, and Jonathan Shewchuk. Interpolating and approximating implicit surfaces from polygon soup. In *ACM Transactions on Graphics*, pages 896–904. ACM Press, 2004.

[19] G. Stylianou and A. Lanitis. Image based 3d face reconstruction: A survey. *International Journal of Image and Graphics*, 9(2):217–250, 2009.

[20] W. Zhao, R. Chellappa, P. J. Phillips, and A. Rosenfeld. Face recognition: A literature survey. *ACM Comput. Surv.*, 35:399–458, December 2003.

[21] Thabo Beeler, Bernd Bickel, Paul Beardsley, Bob Sumner, and Markus Gross. High-quality single-shot capture of facial geometry. *ACM Trans. on Graphics (Proc. SIGGRAPH)*, 29(3), 2010.

[22] Gloria Haro and Montse Pardís. Shape from incomplete silhouettes based on the reprojection error. *Image Vision Comput.*, 28:1354–1368, September 2010.

[23] Frédéric Pighin and J. P. Lewis. Introduction. In *ACM SIGGRAPH 2005 Courses*, SIGGRAPH '05, New York, NY, USA, 2005. ACM.

[24] D. Onofrio, S. Tubaro, A. Rama, and F. Tarres. 3D Face Reconstruction with a four camera acquisition system. In *Int'l Workshop on Very Low Bit-Rate Video Coding*, 2005.

[25] Ph. Leclercq, J. Liu, A. Woodward, and P. Delmas. Which stereo matching algorithm for accurate 3d face creation. In Reinhard Klette and Jovisa Zunic, editors, *Combinatorial Image Analysis*, volume 3322 of *Lecture Notes in Computer Science*, pages 690–704. Springer Berlin - Heidelberg, 2005. 10.1007/978-3-540-30503-3_53.

[26] Oleg Alexander, Mike Rogers, William Lambeth, Matt Chiang, and Paul Debevec. The digital emily project: photoreal facial modeling and animation. In *ACM SIGGRAPH 2009 Courses*, SIGGRAPH '09, pages 12:1–12:15, New York, NY, USA, 2009. ACM.

[27] Daniel Scharstein and Richard Szeliski. High-accuracy stereo depth maps using structured light. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 1:195–202, June 2003.

[28] Joaquim Salvi, Sergio Fernandez, Tomislav Pribanic, and Xavier Llado. A state of the art in structured light patterns for surface profilometry. *Pattern Recognition*, 43(8):2666 – 2680, 2010.

[29] Filareti Tsalakanidou, Frank Forster, Sotiris Malassiotis, and Michael G. Strintzis. Real-time acquisition of depth and color images using structured light and its application to 3d face recognition. *Real-Time Imaging*, 11(5-6):358 – 369, 2005. Special Issue on Multi-Dimensional Image Processing.

[30] SY Chen, YF Li, and J. Zhang. Vision processing for realtime 3-D data acquisition based on coded structured light. *Image Processing, IEEE Transactions on*, 17(2):167–176, 2008.

[31] Song Zhang, Dale Royer, and Shing-Tung Yau. High-resolution, real-time-geometry video acquisition. In *ACM SIGGRAPH 2006 Sketches*, SIGGRAPH '06, New York, NY, USA, 2006. ACM.

[32] T. Weise, B. Leibe, and L. Van Gool. Fast 3d scanning with automatic motion compensation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR'07)*, June 2007.

[33] R. I. Hartley and A. Zisserman. *Multiple view geometry in computer vision*. Cambridge University Press, 2004.

[34] José I. Ronda, Antonio Valdés, and Guillermo Gallego. Line geometry and camera autocalibration. *J. Math. Imaging Vis.*, 32:193–214, October 2008.

[35] Petra Kramer, Fernando Boto, Diana Wald, Fabien Bessy, Celine Paloc, Carles Callol, A. Letamendia, Izaskun Ibarbia, O. Holgado, and J.M. Virto. Comparison of segmentation algorithms for the zebrafish heart in fluorescent microscopy images. In George Bebis, Richard Boyle, Bahram Parvin, Darko Koracin, Yoshinori Kuno, Junxian Wang, Renato Pajarola, Peter Lindstrom, Andre Hinkenjann, Miguel L. Encarnacao, Claudio T. Silva, and Daniel Coming, editors, *Advances in Visual Computing*, Lecture Notes in Computer Science (LNCS), pages 1041–1050, Las Vegas, Nevada, USA, December 2009. Springer.

[36] Daniel Scharstein and Richard Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *Int. J. Comput. Vision*, 47(1-3):7–42, 2002.

[37] Milton Chen. Leveraging the asymmetric sensitivity of eye contact for videoconference. In *CHI '02: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 49–56, New York, NY, USA, 2002. ACM.

[38] Tam T, Cafazzo JA, Seto E, Salenieks ME, and Rossos PG. Perception of eye contact in video teleconsultation. *Journal of telemedicine and telecare*, 13(1):9–35, February 2007.
[39] http://www.3dpresence.eu. 3dpresence.
[40] A. Vetro, S Yea, and A. Smolic. Towards a 3d video format for auto-stereoscopic displays. *Proc. Conference on Applications of Digital Image Processing*, 7073, 2008.
[41] Kuk-Jin Yoon and In So Kweon. Adaptive support-weight approach for correspondence search. *IEEE Trans. Pattern Anal. Mach. Intell.*, 28(4):650, 2006.
[42] Thanarat Horprasert, David Harwood, and Larry S. Davis. A statistical approach for real-time robust background subtraction and shadow detection. In *ICCV Frame-Rate WS*, 1999.
[43] Pietro Perona and Jitendra Malik. Scale-space and edge detection using anisotropic diffusion. Technical Report UCB/CSD-88-483, EECS Department, University of California, Berkeley, december 1988.
[44] http://www.gstreamer.net. Gstreamer.
[45] T. Tuytelaars and K. Mikolajczyk. Local invariant feature detectors: a survey. *Foundations and Trends® in Computer Graphics and Vision*, 3(3):177–280, 2008.
[46] C. Harris and M. Stephens. A combined corner and edge detector. In *Alvey vision conference*, volume 15, page 50. Manchester, UK, 1988.
[47] H.P. Moravec. Towards automatic visual obstacle avoidance. In *IJCAI*, 1977.
[48] K. Mikolajczyk and C. Schmid. Indexing based on scale invariant interest points. In *Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on*, volume 1, pages 525–531. IEEE, 2001.
[49] J. Shi and C. Tomasi. Good features to track. In *Computer Vision and Pattern Recognition, 1994. Proceedings CVPR'94., 1994 IEEE Computer Society Conference on*, pages 593–600. IEEE, 1994.
[50] B.D. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *Proceedings of the 7th international joint conference on Artificial intelligence*, 1981.
[51] D.G. Lowe. Object recognition from local scale-invariant features. In *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*, volume 2, pages 1150–1157. Ieee, 1999.
[52] T. Lindeberg. *Scale-space theory in computer vision*. Springer, 1993.
[53] P. Viola and M.J. Jones. Robust real-time face detection. *International journal of computer vision*, 57(2):137–154, 2004.
[54] E. Rosten, R. Porter, and T. Drummond. Faster and better: A machine learning approach to corner detection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 32(1):105–119, 2010.
[55] K. Mikolajczyk, T. Tuytelaars, C. Schmid, A. Zisserman, J. Matas, F. Schaffalitzky, T. Kadir, and L.V. Gool. A comparison of affine region detectors. *International journal of computer vision*, 65(1):43–72, 2005.
[56] M. Agrawal, K. Konolige, and M. Blas. Censure: Center surround extremas for realtime feature detection and matching. *Computer Vision–ECCV 2008*, pages 102–115, 2008.
[57] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.
[58] K. Mikolajczyk, T. Tuytelaars, C. Schmid, et al. Affine covariant features. *Collaborative work between: the Visual Geometry Group, Katholieke Universiteit Leuven, Inria Rhone-Alpes and the Center for Machine Perception*, 2007.
[59] Pemysl Krek. Flow reduction marching cubes algorithm. In *Proceedings of ICCVG 2004*, pages 100–106. Springer Verlag, 2005.
[60] Carlos Cadavid Oscar E. Ruiz, Miguel Granados. Fea-driven geometric modelling for meshless methods. In *Virtual Concept 2005*, pages 1–8, 2005.
[61] J. C. Carr, R. K. Beatson, J. B. Cherrie, T. J. Mitchell, W. R. Fright, B. C. McCallum, and T. R. Evans. Reconstruction and representation of 3d objects with radial basis functions. In *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 67–76, New York, NY, USA, 2001. ACM.
[62] Bryan S. Morse, Terry S. Yoo, Penny Rheingans, David T. Chen, and K. R. Subramanian. Interpolating implicit surfaces from scattered surface data using compactly supported radial basis functions. In *SIGGRAPH '05: ACM SIGGRAPH 2005 Courses*, page 78, New York, NY, USA, 2005. ACM.
[63] Sarah F. Frisken, Ronald N. Perry, Alyn P. Rockwood, and Thouis R. Jones. Adaptively sampled distance fields: a general representation of shape for computer graphics. In *SIGGRAPH '00: Proceedings of the 27th annual*

conference on Computer graphics and interactive techniques, pages 249–254, New York, NY, USA, 2000. ACM Press/Addison-Wesley Publishing Co.

[64] William E. Lorensen and Harvey E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. *SIGGRAPH Comput. Graph.*, 21(4):169–169, 1987.

[65] Timothy S. Newman and Hong Yi. A survey of the marching cubes algorithm. *Computers & Graphics*, 30(5):854–879, October 2006.

[66] E. Chernyaev. Marching cubes 33: Construction of topologically correct isosurfaces. Technical report, Technical Report CERN CN 95-17, 1995.

[67] T. Lewiner, H. Lopes, A. Vieira, and G. Tavares. Efficient implementation of marching cubes' cases with topological guarantees. *Journal of Graphics Tools*, 8(2):1–15, 2003.

[68] Gunther H. Weber, Oliver Kreylos, Terry J. Ligocki, John M. Shalf, Bernd Hamann, and Kenneth I. Joy. Extraction of crack-free isosurfaces from adaptive mesh refinement data. In *Data Visualization 2001 (Proceedings of VisSym '01)*, pages 25–34. Springer Verlag, 2001.

[69] Raj Shekhar, Elias Fayyad, Roni Yagel, and J. Fredrick Cornhill. Octree-based decimation of marching cubes surfaces. In *VIS '96: Proceedings of the 7th conference on Visualization '96*, pages 335–ff., Los Alamitos, CA, USA, 1996. IEEE Computer Society Press.

[70] Scott Schaefer and Joe Warren. Dual marching cubes: Primal contouring of dual grids. In *PG '04: Proceedings of the Computer Graphics and Applications, 12th Pacific Conference*, pages 70–76, Washington, DC, USA, 2004. IEEE Computer Society.

[71] Afonso Paiva, Helio Lopes, Thomas Lewiner, and Luiz Henrique de Figueiredo. Robust adaptive meshes for implicit surfaces. *SIBGRAPI*, 0:205–212, 2006.

[72] Akinori Kimura, Yasufumi Takama, Yu Yamazoe, Satoshi Tanaka, and Hiromi T. Tanaka. Parallel volume segmentation with tetrahedral adaptive grid. *ICPR*, 02:281–286, 2004.

[73] Laurent Balmelli, Christopher J. Morris, Gabriel Taubin, and Fausto Bernardini. Volume warping for adaptive isosurface extraction. In *Proceedings of the conference on Visualization 02*, pages 467–474. IEEE Computer Society, 2002.

[74] Bernardo Piquet Carneiroz, Cludio T. Silva Y, and Arie E. Kaufman. Tetra-cubes: An algorithm to generate 3d isosurfaces based upon tetrahedra. In *IX Brazilian symposium on computer, graphics, image processing and vision (SIBGRAPI 96)*, pages 205–10, 1996.

[75] De Bruin Vos, P. W. De Bruin, F. M. Vos, F. H. Post, S. F. Frisken-gibson, and A. M. Vossepoel. Improving triangle mesh quality with surfacenets. In *In MICCAI*, pages 804–813, 2000.

[76] M. P. Dubuisson and A. K. Jain. A modified hausdorff distance for object matching. In *Pattern Recognition, 1994. Vol. 1 - Conference A: Computer Vision & Image Processing., Proceedings of the 12th IAPR International Conference on*, volume 1, pages 566–568 vol.1, 1994.

[77] John Congote, Aitor Moreno, Iigo Barandiaran, Javier Barandiaran, and Oscar Ruiz. Adaptative cubical grid for isosurface extraction. In *4th International Conference on Computer Graphics Theory and Applications GRAPP-2009*, pages 21–26, Lisbon, Portugal, Feb 5-8 2009.

[78] Yongwei Miao, Jieqing Feng, and Qunsheng Peng. Curvature estimation of point-sampled surfaces and its applications. In *Computational Science and Its Applications ICCSA 2005*, pages 1023–1032. Springer Berlin / Heidelberg, 2005.

[79] Markus Gross and Hanspeter Pfister. *POINT-BASED GRAPHICS*. Elsevier, 2007.

[80] Nina Amenta, Marshall Bern, and Manolis Kamvysselis. A new voronoi-based surface reconstruction algorithm. In *SIGGRAPH '98: Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pages 415–421, New York, NY, USA, 1998. ACM.

[81] Nina Amenta, Sunghee Choi, and Ravi Krishna Kolluri. The power crust. In *SMA '01: Proceedings of the sixth ACM symposium on Solid modeling and applications*, pages 249–266, New York, NY, USA, 2001. ACM.

[82] Hugues Hoppe, Tony DeRose, Tom Duchamp, John McDonald, and Werner Stuetzle. Surface reconstruction from unorganized points. In *SIGGRAPH '92: Proceedings of the 19th annual conference on Computer graphics and interactive techniques*, pages 71–78, New York, NY, USA, 1992. ACM.

[83] Michael Kazhdan, Matthew Bolitho, and Hugues Hoppe. Poisson surface reconstruction. In *SGP '06: Proceedings of the fourth Eurographics symposium on Geometry processing*, pages 61–70, Aire-la-Ville, Switzerland, Switzerland, 2006. Eurographics Association.

[84] Evgeni V. Chernyaev. Marching cubes 33: Construction of topologically correct isosurfaces. Technical report, CERN, 1995.

[85] Thomas Lewiner, Hlio Lopes, Antnio Wilson Vieira, and Geovan Tavares. Efficient implementation of marching cubes' cases with topological guarantees. *journal of graphics, gpu, and game tools*, 8(2):1–15, 2003.

[86] John E. Congote, Aitor Moreno, Iigo Barandiaran, Javier Barandiaran, and Oscar Ruiz. Adaptative cubical grid for isosurface extraction. In *4th International Conference on Computer Graphics Theory and Applications GRAPP-2009*, pages 21–26, Lisbon, Portugal, February 2009.

[87] Keisuke Fujimoto, Toshio Moriya, and Yasuichi Nakayama. Surface reconstruction from high-density points using deformed grids. In *WSCG2008 Communication Papers Proceedings*, pages 117–120, Plzen - Bory, Czech Republic, 2008. University of West Bohemia.

[88] Alexander Hornung and Leif Kobbelt. Robust reconstruction of watertight 3d models from non-uniformly sampled point clouds without normal information. In *SGP '06: Proceedings of the fourth Eurographics symposium on Geometry processing*, pages 41–50, Aire-la-Ville, Switzerland, Switzerland, 2006. Eurographics Association.

[89] Marc Levoy. Display of surfaces from volume data. *IEEE Comput. Graph. Appl.*, 8:29–37, May 1988.

[90] Markus Hadwiger, Patric Ljung, Christof R. Salama, and Timo Ropinski. Advanced illumination techniques for gpu-based volume raycasting. In *ACM SIGGRAPH 2009 Courses*, pages 1–166. ACM, 2009.

[91] Jennis Meyer-Spradow, Timo Ropinski, Jörg Mensmann, and Klaus H. Hinrichs. Voreen: A rapid-prototyping environment for ray-casting-based volume visualizations. *IEEE Computer Graphics and Applications (Applications Department)*, 29(6):6–13, Nov./Dec. 2009.

[92] Thomas Fogal and Jens Kruger. Tuvok, an Architecture for Large Scale Volume Rendering. In Michael Dogget, Samuli Laine, and Warren Hunt, editors, *Proceedings of the 15th International Workshop on Vision, Modeling, and Visualization*, pages 57–66, June 2010.

[93] Seyyed Ehsan Mahmoudi, Alireza Akhondi-Asl, Roohollah Rahmani, Shahrooz Faghih-Roohi, Vahid Taimouri, Ahmad Sabouri, and Hamid Soltanian-Zadeh. Web-based interactive 2d/3d medical image processing and visualization software. *Computer Methods and Programs in Biomedicine*, In Press, Corrected Proof:–, 2009.

[94] Luis Kabongo, Ivn Maca, and Cline Paloc. Development of a commercial cross-platform dicom viewer based on open source software. In Professor Heinz U. Lemke, PhD; Professor Kiyonari Inamura, PhD; Professor Kunio Doi, PhD; Professor Michael W. Vannier, PhD; Professor Allan G. Farman, and DSc PhD, editors, *International Journal of Computer Assisted Radiology and Surgery; CARS 2009 Computer Assisted Radiology and Surgery Proceedings of the 23rd International Congress and Exhibition*, volume 4, pages S29–S30, Berlin, Germany, June 2009. International Foundation of Computer Assisted Radiology and Surgery, Springer.

[95] V. Sundaram, L. Zhao, C.X. Song, B. Benes, R. Veeramacheneni, and P. Kristof. Real-time Data Delivery and Remote Visualization through Multi-layer Interfaces. In *Grid Computing Environments Workshop, 2008. GCE'08*, pages 1–10, 2008.

[96] Chris Marrin. *WebGL Specification*. Khronos WebGL Working Group, 2011.

[97] M. Meißner, J. Huang, D. Bartz, K. Mueller, and R. Crawfis. A practical evaluation of popular volume rendering algorithms. In *Proceedings of the 2000 IEEE symposium on Volume visualization*, pages 81–90. Citeseer, 2000.

[98] Lee Alan Westover. *Splatting: a parallel, feed-forward volume rendering algorithm*. PhD thesis, Chapel Hill, NC, USA, 1991. UMI Order No. GAX92-08005.

[99] P. Lacroute and M. Levoy. Fast volume rendering using a shear-warp factorization of the viewing transformation. In *Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, SIGGRAPH '94, pages 451–458, New York, NY, USA, 1994. ACM.

[100] William Hibbard and David Santek. Interactivity is the key. In *Proceedings of the 1989 Chapel Hill workshop on Volume visualization*, VVS '89, pages 39–43, New York, NY, USA, 1989. ACM.

[101] Bui Tuong Phong. Illumination for computer generated pictures. *Commun. ACM*, 18(6):311–317, 1975.

[102] J. Kruger and R. Westermann. Acceleration techniques for gpu-based volume rendering. In *VIS '03: Proceedings of the 14th IEEE Visualization 2003 (VIS'03)*, page 38, Washington, DC, USA, 2003. IEEE Computer Society.

[103] Johannes Behr, Peter Eschler, Yvonne Jung, and Michael Zöllner. X3dom: a dom-based html5/x3d integration model. In *Proceedings of the 14th International Conference on 3D Web Technology*, Web3D '09, pages 127–135, New York, NY, USA, 2009. ACM.

[104] B. Blazona and Z. Mihajlovic. Visualization service based on web services. *Journal of Computing and Information Technology*, 15(4):339, 2007.

[105] John Congote, Iñigo Barandiaran, Javier Barandiaran, Tomas Montserrat, Julien Quelen, Christian Ferrán, Pere J. Mindan, Olga Mur, Francesc Tarrés, and O. Ruiz. Real-time depth map generation architecture for 3d videoconferencing. In *3DTV-Conference: The True Vision - Capture, Transmissionand Display of 3D Video (3DTV-CON), 2010*, pages 1–4, Tampere, Finland, June 2010.

[106] Ricardo Marques, Luís Paulo Santos, Peter Leškovský, and Céline Paloc. Gpu ray casting. In António Coelho, Ana Paula Cláudio, Frutuoso Silva, and Abel Gomes, editors, *17 Encontro Português de Computaçao Gráfica*, pages 83–91, Covilha, Portugal, October 2009. En Anexo.

[107] NW John, M. Aratow, J. Couch, D. Evestedt, AD Hudson, N. Polys, RF Puk, A. Ray, K. Victor, and Q. Wang. MedX3D: standards enabled desktop medical 3D. *Studies in health technology and informatics*, 132:189, 2008.

[108] Nigel W. John. The impact of web3d technologies on medical education and training. *Comput. Educ.*, 49(1):19–31, August 2007.

[109] Andrew V. Poliakov, Evan Albright, Kevin P. Hinshaw, David P. Corina, George Ojemann, Richard F. Martin, and James F. Brinkley. Server-based approach to web visualization of integrated three-dimensional brain imaging data. *Journal of the American Medical Informatics Association*, 12(2):140 – 151, 2005.

[110] S.K. Yoo, J. Key, K. Choi, and J. Jo. Web-Based Hybrid Visualization of Medical Images. *Lecture notes in computer science*, 3568:376, 2005.

[111] J. Behr and M. Alexa. Volume visualization in vrml. In *Proceedings of the sixth international conference on 3D Web technology*, pages 23–27. ACM New York, NY, USA, 2001.

[112] Kirk Riley, Yuyan Song, Martin Kraus, David S. Ebert, and Jason J. Levit. Visualization of structured nonuniform grids. *IEEE Computer Graphics and Applications*, 26:46–55, 2006.

[113] Jon Goenetxea, Aitor Moreno, Luis Unzueta, Andoni Galdós, and Alvaro Segura. Interactive and stereoscopic hybrid 3d viewer of radar data with gesture recognition. In Manuel Graña Romay, Emilio Corchado, and M. Teresa García-Sebastián, editors, *HAIS (1)*, volume 6076 of *Lecture Notes in Computer Science*, pages 213–220. Springer, 2010.

[114] Richard Hartley and Andrew Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, Cambridge, UK, second edition, 2003.

[115] Álvaro Segura, Aitor Moreno, Igor García, Naiara Aginako, Mikel Labayen, Jorge Posada, Jose Antonio Aranda, and Rubén García De Andoin. Visual processing of geographic and environmental information in the basque country: Two basque case studies. In Raffaele De Amicis, Radovan Stojanovic, and Giuseppe Conti, editors, *GeoSpatial Visual Analytics*, NATO Science for Peace and Security Series C: Environmental Security, pages 199–208. Springer Netherlands, October 2009.

[116] Agnieszka(Aga) Bojko. Informative or misleading? heatmaps deconstructed. In JulieA. Jacko, editor, *Human-Computer Interaction. New Trends*, volume 5610 of *Lecture Notes in Computer Science*, pages 30–39. Springer Berlin Heidelberg, 2009.

[117] Alex Poole and Linden J Ball. Eye tracking in hci and usability research. *Encyclopedia of Human-Computer Interaction, C. Ghaoui (ed.)*, 2006.

[118] Sophie Stellmach, Lennart Nacke, and Raimund Dachselt. 3d attentional maps: aggregated gaze visualizations in three-dimensional virtual environments. In *Proceedings of the International Conference on Advanced Visual Interfaces*, AVI '10, pages 345–348, New York, NY, USA, 2010. ACM.

[119] Aidong Lu, Ross Maciejewski, and David S. Ebert. Volume composition using eye tracking data. In *Proceedings of the Eighth Joint Eurographics / IEEE VGTC conference on Visualization*, EUROVIS'06, pages 115–122, Aire-la-Ville, Switzerland, Switzerland, 2006. Eurographics Association.

[120] John Congote, Alvaro Segura, Luis Kabongo, Aitor Moreno, Jorge Posada, and Oscar Ruiz. Interactive visualization of volumetric data with webgl in real-time. In *Proceedings of the 16th International Conference on 3D Web Technology*, Web3D '11, pages 137–146, New York, NY, USA, 2011. ACM.

[121] N W John, M Aratow, J Couch, D Evestedt, A D Hudson, N Polys, R F Puk, A Ray, K Victor, and Q Wang. Medx3d: Standards enabled desktop medical 3d. *Studies In Health Technology And Informatics*, 132:189–194, 2008.

[122] Nicholas Polys, Andrew Wood, and Patrick Shinpaugh. Cross-platform presentation of interactive volumetric imagery. Departmental Technical Report 1177, Virginia Tech, Advanced Research Computing, 2011.

[123] J. Behr, Y. Jung, J. Keil, T. Drevensek, M. Zoellner, P. Eschler, and D. Fellner. A scalable architecture for the html5/x3d integration model x3dom. In *Proceedings of the 15th International Conference on Web 3D Technology*, Web3D '10, pages 185–194, New York, NY, USA, 2010. ACM.

[124] Johannes Behr, Yvonne Jung, Timm Drevensek, and Andreas Aderhold. Dynamic and interactive aspects of x3dom. In *Proceedings of the 16th International Conference on 3D Web Technology*, Web3D '11, pages 81–87, New York, NY, USA, 2011. ACM.

[125] John Congote. Medx3dom: Medx3d for x3dom. In *To be pusblish in Proceedings of the 17th International Conference on 3D Web Technology*, Web3D '12, New York, NY, USA, 2012. ACM.

[126] James F. Blinn. Light reflection functions for simulation of clouds and dusty surfaces. *SIGGRAPH Comput. Graph.*, 16(3):21–29, July 1982.

[127] James T. Kajiya and Brian P Von Herzen. Ray tracing volume densities. *SIGGRAPH Comput. Graph.*, 18:165–174, January 1984.

[128] M. Meißner, J. Huang, D. Bartz, K. Mueller, and R. Crawfis. A practical evaluation of popular volume rendering algorithms. In *Proceedings of the 2000 IEEE symposium on Volume visualization*, pages 81–90. Citeseer, 2000.

[129] Marco Callieri, Raluca Mihaela Andrei, Marco Di Benedetto, Monica Zoppè, and Roberto Scopigno. Visualization methods for molecular studies on the web platform. In *Proceedings of the 15th International Conference on Web 3D Technology*, Web3D '10, pages 117–126, New York, NY, USA, 2010. ACM.

[130] Marco Di Benedetto, Federico Ponchio, Fabio Ganovelli, and Roberto Scopigno. Spidergl: a javascript 3d graphics library for next-generation www. In *Proceedings of the 15th International Conference on Web 3D Technology*, Web3D '10, pages 165–174, New York, NY, USA, 2010. ACM.

[131] Kristian Sons, Felix Klein, Dmitri Rubinstein, Sergiy Byelozyorov, and Philipp Slusallek. Xml3d: interactive 3d graphics for the web. In *Proceedings of the 15th International Conference on Web 3D Technology*, Web3D '10, pages 175–184, New York, NY, USA, 2010. ACM.

[132] M. Masseroli and F. Pinciroli. Web tools for effective retrieval, visualization, and evaluation of cardiology medical images and records. In *Proceedings of SPIE, the International Society for Optical Engineering*, volume 4311, pages 325–332. Society of Photo-Optical Instrumentation Engineers, 2001.

[133] F.G. Hamza-Lup and T. Thompson. Interactive 3d user interfaces for exploring neuroanatomy.

[134] Diego Cantor-Rivera, Robert Bartha, and Terry Peters. Efficient 3d rendering for web-based medical imaging software: a proof of concept. In *SPIE, Proceedings of*, volume 7964, 2011.

[135] A.V. Poliakov, E. Albright, D. Corina, G. Ojemann, R.F. Martin, and J.F. Brinkley. Server-based approach to web visualization of integrated 3-d medical image data. In *Proceedings of the AMIA Symposium*, page 533. American Medical Informatics Association, 2001.

[136] A.V. Poliakov, E. Albright, K.P. Hinshaw, D.P. Corina, G. Ojemann, R.F. Martin, and J.F. Brinkley. Server-based approach to web visualization of integrated three-dimensional brain imaging data. *Journal of the American Medical Informatics Association*, 12(2):140–151, 2005.

[137] B. Blažona and Ž. Mihajlović. Introduction to the visualization service based on web services. In *The 32nd EUROMICRO Conference on Software Engineering and Advanced Applications (SEAA)*, 2006.

[138] Bojan Blazona and Zeljka Mihajlovic. Visualization service based on web services. *29th International Conference on*, pages 673–678, 2007.

[139] Mikhail Smelyanskiy, David Holmes, Jatin Chhugani, Alan Larson, Douglas M. Carmean, Dennis Hanson, Pradeep Dubey, Kurt Augustine, Daehyun Kim, Alan Kyker, Victor W. Lee, Anthony D. Nguyen, Larry Seiler, and Richard Robb. Mapping high-fidelity volume rendering for medical imaging to cpu, gpu and many-core architectures. *IEEE Transactions on Visualization and Computer Graphics*, 15:1563–1570, November 2009.

[140] Sun K. Yoo, Jaehong Key, Kuiwon Choi, and Jinho Jo. Web-based hybrid visualization of medical images. In *Proceedings of the 4th international conference on Image and Video Retrieval*, CIVR'05, pages 376–384, Berlin, Heidelberg, 2005. Springer-Verlag.

[141] Seyyed Ehsan Mahmoudi, Alireza Akhondi-Asl, Roohollah Rahmani, Shahrooz Faghih-Roohi, Vahid Taimouri, Ahmad Sabouri, and Hamid Soltanian-Zadeh. Web-based interactive 2d/3d medical image processing and visualization software. *Comput. Methods Prog. Biomed.*, 98(2):172–182, May 2010.

[142] S. Settapat, T. Achalakul, and M. Ohkura. Web-based 3d visualization and interaction of medical data using web3d. In *SICE Annual Conference 2010, Proceedings of*, pages 2986–2991. IEEE, 2010.

[143] Sittapong Settapat, Tiranee Achalakul, and Michiko Ohkura. Web-based 3d medical image visualization framework for biomedical engineering education. *Computer Applications in Engineering Education*, 2011.

[144] Julien Jomier, Sebastien Jourdain, Utkarsh Ayachit, and Charles Marion. Remote visualization of large datasets with midas and paraviewweb. In *Proceedings of the 16th International Conference on 3D Web Technology*, Web3D '11, pages 147–150, New York, NY, USA, 2011. ACM.

[145] Yvonne Jung, Ruth Recker, Manuel Olbrich, and Ulrich Bockholt. Using x3d for medical training simulations. In *Proceedings of the 13th international symposium on 3D web technology*, Web3D '08, pages 43–51, New York, NY, USA, 2008. ACM.

[146] E. Fried, Y. Geng, S. Ullrich, D. Kneer, O. Grottke, R. Rossaint, T.M. Deserno, and T. Kuhlen. An xml-based approach of medical data organization for segmentation and simulation. 2010.

[147] S. Ullrich, T. Kuhlen, NF Polys, D. Evestedt, M. Aratow, and NW John. Quantizing the void: extending web3d for space-filling haptic meshes. *Studies in health technology and informatics*, 163:670, 2011.

[148] Mauricio Aristizabal, John Congote, Alvaro Segura, Aitor Moreno, Harbil Arriegui, and O. Ruiz. Hardware-accelerated web visualization of vector fields. case study in oceanic currents. In Robert S. Laramee Paul Richard, Martin Kraus and Jos Braz, editors, *IVAPP-2012. International Conference on Computer Vision Theory and Applications*, pages 759–763, Rome, Italy, February 2012. INSTICC, SciTePress.

[149] Daniel Ginsburg, Stephan Gerhard, John Edgar Congote, and Rudolph Pienaar. Realtime visualization of the connectome in the browser using webgl. *Frontiers in Neuroinformatics*, October 2011.

[150] Andoni Beristain, John Congote, and Oscar Ruiz. Volume visual attention maps (vvam) in ray-casting rendering. *Studies in Health Technology and Informatics*, 173:53–7, 2012.

[151] Aitor Moreno, John Congote, Oscar Ruiz, Juan Luis Prez, and Roberto San Jos. Web based volume rendering of air quality 3d datasets at the city scale. In *COST TU0801 Workshop: 3D Issues in Environmental and Urban Systems, Madrid (Spain)*, April 2012.

[152] John Congote, Alvaro Segura, Luis Kabongo, Aitor Moreno, Jorge Posada, and Oscar Ruiz. Interactive visualization of volumetric data with webgl in real-time. In *Proceedings of the 16th International Conference on 3D Web Technology*, Web3D '11, pages 137–146, New York, NY, USA, 2011. ACM.

[153] James T. Kajiya and Brian P Von Herzen. Ray tracing volume densities. *SIGGRAPH Comput. Graph.*, 18:165–174, January 1984.

[154] Marc Levoy. Display of surfaces from volume data. *IEEE Comput. Graph. Appl.*, 8:29–37, May 1988.

[155] Robert A. Drebin, Loren Carpenter, and Pat Hanrahan. Volume rendering. In *Proceedings of the 15th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '88, pages 65–74, New York, NY, USA, 1988. ACM.

[156] J. Kruger and R. Westermann. Acceleration techniques for gpu-based volume rendering. In *Proceedings of the 14th IEEE Visualization 2003 (VIS'03)*, VIS '03, pages 38–, Washington, DC, USA, 2003. IEEE Computer Society.

[157] Henning Scharsach. Advanced gpu raycasting. *Proceedings of CESCG*, 5:67–76, 2005.

[158] Christian Boucheny, Georges-Pierre Bonneau, Jacques Droulez, Guillaume Thibault, and Stéphane Ploix. A perceptive evaluation of volume rendering techniques. In *Proceedings of the 4th symposium on Applied perception in graphics and visualization*, APGV '07, pages 83–90, New York, NY, USA, 2007. ACM.

[159] S. Bruckner. *Efficient Volume Visualization of Large Medical Datasets: Concepts and Algorithms*. VDM Verlag, 2008.

[160] Jörg Mensmann, Timo Ropinski, and Klaus Hinrichs. ccelerating volume raycasting using occlusion frustums. In *IEEE/EG Volume and Point-Based Graphics*, pages 147–154, 2008.

[161] Marco Di Benedetto, Federico Ponchio, Fabio Ganovelli, and Roberto Scopigno. Spidergl: a javascript 3d graphics library for next-generation www. In *Proceedings of the 15th International Conference on Web 3D Technology*, Web3D '10, pages 165–174, New York, NY, USA, 2010. ACM.

[162] Kristian Sons, Felix Klein, Dmitri Rubinstein, Sergiy Byelozyorov, and Philipp Slusallek. Xml3d: interactive 3d graphics for the web. In *Proceedings of the 15th International Conference on Web 3D Technology*, Web3D '10, pages 175–184, New York, NY, USA, 2010. ACM.

[163] N W John, M Aratow, J Couch, D Evestedt, A D Hudson, N Polys, R F Puk, A Ray, K Victor, and Q Wang. Medx3d: Standards enabled desktop medical 3d. *Studies In Health Technology And Informatics*, 132:189–194, 2008.

[164] Nicholas Polys, Andrew Wood, and Patrick Shinpaugh. Cross-platform presentation of interactive volumetric imagery. Departmental Technical Report 1177, Virginia Tech, Advanced Research Computing, 2011.

[165] Bojan Blazona and Zeljka Mihajlovic. Visualization service based on web services. *29th International Conference on*, pages 673–678, 2007.

[166] Mikhail Smelyanskiy, David Holmes, Jatin Chhugani, Alan Larson, Douglas M. Carmean, Dennis Hanson, Pradeep Dubey, Kurt Augustine, Daehyun Kim, Alan Kyker, Victor W. Lee, Anthony D. Nguyen, Larry Seiler, and Richard Robb. Mapping high-fidelity volume rendering for medical imaging to cpu, gpu and many-core architectures. *IEEE Transactions on Visualization and Computer Graphics*, 15:1563–1570, November 2009.

[167] S. Settapat, T. Achalakul, and M. Ohkura. Web-based 3d visualization and interaction of medical data using web3d. In *SICE Annual Conference 2010, Proceedings of*, pages 2986–2991. IEEE, 2010.

[168] Y. Masutani, S. Aoki, O. Abe, N. Hayashi, and K. Otomo. Mr diffusion tensor imaging: recent advance and new techniques for diffusion tensor visualization. *European Journal of Radiology*, 46(1):53–66, 2003.

[169] P.G.P. Nucifora, R. Verma, S.K. Lee, and E.R. Melhem. Diffusion-tensor mr imaging and tractography: Exploring brain microstructure and connectivity. *Radiology*, 245(2):367–384, 2007.

[170] A.J. Golby, G. Kindlmann, I. Norton, A. Yarmarkovich, S. Pieper, and R. Kikinis. Interactive diffusion tensor tractography visualization for neurosurgical planning. *Neurosurgery*, 68(2):496, 2011.

[171] H.H. Ehricke, U. Klose, and W. Grodd. Visualizing mr diffusion tensor fields by dynamic fiber tracking and uncertainty mapping. *Computers & Graphics*, 30(2):255–264, 2006.

[172] N. Ratnarajah, A. Simmons, O. Davydov, and A. Hojjat. A novel white matter fibre tracking algorithm using probabilistic tractography and average curves. *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2010*, pages 666–673, 2010.

[173] Marcelo Rodrigo Maciel Silva, Isabel Harb Manssour, and Carla Maria Dal Sasso Freitas. Optimizing combined volume and surface data ray casting. In *WSCG*, 2000.

[174] S. Gerhard, A. Daducci, A. Lemkaddem, R. Meuli, J.P. Thiran, and P. Hagmann. The connectome viewer toolkit: an open source framework to manage, analyze, and visualize connectomes. *Frontiers in Neuroinformatics*, 5, 2011.

[175] Daniel Ginsburg, Stephan Gerhard, John Edgar Congote, and Rudolph Pienaar. Realtime visualization of the connectome in the browser using webgl. *Frontiers in Neuroinformatics*, October 2011.

[176] M. Hlawatsch, F. Sadlo, and D. Weiskopf. Hierarchical line integration. *Visualization and Computer Graphics, IEEE Transactions on*, (99):1–1, 2011.

[177] R.V. Klassen and S.J. Harrington. Shadowed hedgehogs: A technique for visualizing 2d slices of 3d vector fields. In *Proceedings of the 2nd conference on Visualization'91*, pages 148–153. IEEE Computer Society Press, 1991.

[178] D.N. Kenwright and G.D. Mallinson. A 3-d streamline tracking algorithm using dual stream functions. In *Proceedings of the 3rd conference on Visualization'92*, pages 62–68. IEEE Computer Society Press, 1992.

[179] D.A. Lane. Ufat: a particle tracer for time-dependent flow fields. In *Proceedings of the conference on Visualization'94*, pages 257–264. IEEE Computer Society Press, 1994.

[180] T. McLoughlin, R.S. Laramee, R. Peikert, F.H. Post, and M. Chen. Over two decades of integration-based, geometric flow visualization. In *Computer Graphics Forum*, volume 29-6, pages 1807–1829. Wiley Online Library, 2010.

[181] R.S. Laramee, H. Hauser, H. Doleisch, B. Vrolijk, F.H. Post, and D. Weiskopf. The state of the art in flow visualization: Dense and texture-based techniques. In *Computer Graphics Forum*, volume 23, pages 203–221. Wiley Online Library, 2004.

[182] J.J. Van Wijk. Image based flow visualization. In *ACM Transactions on Graphics (TOG)*, volume 21, pages 745–754. ACM, 2002.

[183] J.J. Van Wijk. Image based flow visualization for curved surfaces. In *Proceedings of the 14th IEEE Visualization 2003 (VIS'03)*, page 17. IEEE Computer Society, 2003.

[184] B. Cabral and L.C. Leedom. Imaging vector fields using line integral convolution. In *Proceedings of the 20th annual conference on Computer graphics and interactive techniques*, pages 263–270. ACM, 1993.

[185] L.K. Forssell and S.D. Cohen. Using line integral convolution for flow visualization: Curvilinear grids, variable-speed animation, and unsteady flows. *Visualization and Computer Graphics, IEEE Transactions on*, 1(2):133–141, 1995.

[186] Z. Liu and R.J. Moorhead II. Visualizing time-varying three-dimensional flow fields using accelerated uflic. In *The 11th International Symposium on Flow Visualization*, pages 9–12. Citeseer, 2004.

[187] Z. Liu and R.J. Moorhead. Accelerated unsteady flow line integral convolution. *IEEE Transactions on Visualization and Computer Graphics*, pages 113–125, 2005.

[188] J. Congote, A. Segura, L. Kabongo, A. Moreno, J. Posada, and O. Ruiz. Interactive visualization of volumetric data with webgl in real-time. In *Proceedings of the 16th International Conference on 3D Web Technology*, pages 137–146. ACM, 2011.

[189] M. Callieri, R.M. Andrei, M. Di Benedetto, M. Zoppè, and R. Scopigno. Visualization methods for molecular studies on the web platform. In *Proceedings of the 15th International Conference on Web 3D Technology*, pages 117–126. ACM, 2010.

[190] H. Scharsach. Advanced GPU raycasting. *Proceedings of CESCG*, 5:67–76, 2005.

[191] P. K. Smolarkiewicz and L. G. Margolin. On forward-in-time differencing for fluids an eulerian semi lagrangian nonhydrostatic model for stratified flows. volume 35, pages 127–152, 1997.