



UNIVERSITY OF CAGLIARI

PHD SCHOOL OF MATHEMATICS
AND COMPUTER SCIENCE

XXVII CYCLE

COURSE IN COMPUTER SCIENCE

**Scalable Exploration of
Highly Detailed and Annotated 3D Models**

INF / 01

Author : Marcos Balsa RODRIGUEZ

Supervisors : Prof. Riccardo SCATENI
Dr. Enrico GOBETTI

2013 - 2014

Dedicado a mi familia y amigos
Dedicated to my family and friends

Abstract

With the widespread availability of mobile graphics terminals and WebGL-enabled browsers, 3D graphics over the Internet is thriving. Thanks to recent advances in 3D acquisition and modeling systems, high-quality 3D models are becoming increasingly common, and are now potentially available for ubiquitous exploration.

In current 3D repositories, such as Blend Swap, 3D Café or Archive3D, 3D models available for download are mostly presented through a few user-selected static images. Online exploration is limited to simple orbiting and/or low-fidelity explorations of simplified models, since photo-realistic rendering quality of complex synthetic environments is still hardly achievable within the real-time constraints of interactive applications, especially on low-powered mobile devices or script-based Internet browsers.

Moreover, navigating inside 3D environments, especially on the now pervasive touch devices, is a non-trivial task, and usability is consistently improved by employing assisted navigation controls. In addition, 3D annotations are often used in order to integrate and enhance the visual information by providing spatially coherent contextual information, typically at the expense of introducing visual cluttering.

In this thesis, we focus on efficient representations for interactive exploration and understanding of highly detailed 3D meshes on common 3D platforms. For this purpose, we present several approaches exploiting constraints on the data representation for improving the streaming and rendering performance, and camera movement constraints in order to provide scalable navigation methods for interactive exploration of complex 3D environments.

Furthermore, we study visualization and interaction techniques to improve the exploration and understanding of complex 3D models by exploiting guided motion control techniques to aid the user in discovering contextual information while avoiding cluttering the visualization.

We demonstrate the effectiveness and scalability of our approaches both in large screen museum installations and in mobile devices, by performing interactive exploration of models ranging from $9M$ triangles to $940M$ triangles.

Keywords: Computer Graphics, Real-time Rendering, Massive Model Rendering, Level-of-detail, Interaction Techniques.

“Simplicity, carried to an extreme, becomes elegance.”

Jon Franklin

Contents

List of Figures	xiii
List of Tables	xv
Acknowledgments	xvii
Preface	xvii
1 Introduction	1
1.1 Background and Motivation	1
1.2 Objectives	3
1.3 Achievements	3
1.4 Organization	4
I Background and Motivation	7
2 Application Domain	11
2.1 Introduction	11
2.2 Requirements	13
2.3 Discussion	19
2.4 Bibliographical Notes	20
3 Previous Work	21
3.1 Introduction	21
3.2 Scalable Visualization of Complex 3D Models	22
3.3 Interactive Exploration of Complex 3D Models	25
3.4 Information Discovery on Complex 3D Models	28
3.5 Discussion	29
3.6 Bibliographical Notes	30
4 Work Plan	33
4.1 Research Goals	33
4.2 Our approach	35
II Compact Representations for Complex 3D Models	41
5 Compression-domain Seamless Multiresolution Visualization of Gigantic Meshes on Mobile Devices	45
5.1 Introduction	45

5.2	Method Overview	48
5.3	Building the multiresolution structure	49
5.4	Server	55
5.5	Client architecture description	55
5.6	Implementation and Results	60
5.7	Discussion	63
5.8	Bibliographical Notes	65
6	Adaptive Quad Patches: An Adaptive Regular Structure for Web Distribution and Adaptive Rendering of 3D Models	67
6.1	Introduction	67
6.2	Method Overview	68
6.3	Surface Reconstruction, Parametrization and Quad Re-meshing	69
6.4	Quad-based Multiresolution Structure	70
6.5	Implementation and Results	77
6.6	Discussion	81
6.7	Bibliographical Notes	82
7	ExploreMaps: Efficient Construction of Panoramic View Graphs of Complex 3D Environments	83
7.1	Introduction	83
7.2	Creating the ExploreMaps graph	84
7.3	Efficient GPU Implementation	89
7.4	Implementation and Results	96
7.5	Discussion	98
7.6	Bibliographical Notes	100
III	Assisted Exploration of Complex 3D Models	101
8	HuMoRS: Huge models Mobile Rendering System	105
8.1	Introduction	105
8.2	System overview	107
8.3	User interaction	109
8.4	Implementation and Results	113
8.5	Discussion	119
8.6	Bibliographical Notes	119
9	IsoCam: Interactive Visual Exploration of Massive Cultural Heritage Models on Large Projection Setups	121
9.1	Introduction	122
9.2	Overview	123
9.3	Camera control	126
9.4	Image-based navigation and points of interest	130
9.5	Device mapping	131
9.6	Extending support to light field displays	133
9.7	Scalability	138
9.8	Implementation and Results	141
9.9	Discussion	148
9.10	Bibliographical Notes	149
10	ExploreMaps: Ubiquitous Exploration of Panoramic View Graphs of Complex 3D Environments	151
10.1	Introduction	151

10.2 Browsing Explore Maps	152
10.3 Implementation and Results	155
10.4 Discussion	156
10.5 Bibliographical Notes	157
IV Beyond Visual Replication	159
11 Adaptive Recommendations for Enhanced non-linear Exploration of Annotated 3D Objects	163
11.1 Introduction	163
11.2 Overview	165
11.3 The recommendation engine	168
11.4 User interface	172
11.5 Scalability	175
11.6 Implementation and User Study	176
11.7 Discussion	180
11.8 Bibliographical Notes	181
V Conclusions	183
12 Summary and Conclusions	187
12.1 Conclusions	187
12.2 Future Work	189
12.3 Bibliographical Notes	190
Bibliography	193
Curriculum Vitae	211

List of Figures

Background and Motivation	9
2.1 Dragon statue	14
2.2 Imposing scale of the David statue	14
2.3 Highly detailed 3D models	15
2.4 Photo-realistic rendering of a hotel room	15
2.5 David statue by Michelangelo	16
2.6 Museums and exhibitions	16
2.7 Cluttered presentation	17
2.8 Various display setups	18
2.9 Annotated 3D models	18
2.10 Various annotation links	19
Compact Representations	43
5.1 CATP overview	47
5.2 Sequence of diamond configurations	50
5.3 Tetrahedra merging	52
5.4 Geometry quantization	52
5.5 Vertex snapping	53
5.6 Detail of David’s eye interactively rendered on a iPad	59
5.7 St. Matthew and David on a 3rd generation iPad and a iPhone 4	61
6.1 The AQP pipeline	69
6.2 Reconstruction steps	71
6.3 Rampant model	71
6.4 Multiresolution structure	73
6.5 Seamless point dequantization	74
6.6 LOD seamless tessellation	75
6.7 Models rendered with the adaptive quad patches method	78
6.8 WebGL implementation running in Chrome	79
7.1 ExploreMaps pipeline	84
7.2 Finding probe positions	85
7.3 Optimizing probe positions	86
7.4 Resulting probes	87
7.5 Connecting probes	90
7.6 GPU algorithm	90
7.7 Discontinuity detection	91
7.8 Exploring the geometry	93
7.9 Path optimization	95
7.10 Connecting probes	96

Assisted Exploration	103
8.1 Remote inspection	106
8.2 Client-server architecture	107
8.3 Detail of a model interactively rendered on a Nexus 4 smartphone	108
8.4 Auto-centering and Interaction	109
8.5 Interaction states	110
8.6 Views selection process	112
8.7 Various levels of detail of a statue	114
8.8 User study performed on a Nexus 7 tablet	116
8.9 Performance comparison: Timings	118
9.1 Museum exhibition	122
9.2 Method overview	124
9.3 IsoCam	128
9.4 Temporal smoothing	129
9.5 Radial view selector	131
9.6 Context-based selection	131
9.7 Multi-Touch gestures	132
9.8 Natural immersive exploration of the David 0.25mm model (1GTriangle) on a 35MPixel light field display	134
9.9 Light field display concept	135
9.10 Light field display spatial resolution	136
9.11 Automatic hot-spot placement	137
9.12 Application setup	141
9.13 User interface evaluation	142
9.14 Performance evaluation	145
9.15 Qualitative evaluation	146
10.1 Mobile web-based exploration	152
10.2 Graph optimization	153
10.3 WebGL viewer	154
10.4 Browsing results	155
Beyond Visual Replication	161
11.1 System overview	165
11.2 Overlaid information	169
11.3 State Machine	170
11.4 Large projection setup	173
11.5 Suggestions and overlays	174

List of Tables

Background and Motivation	9
4.1 Method classification	36
Compact Representations	43
4.2 Scalable rendering approaches	43
5.1 CATP encoding bit rates	62
6.1 Adaptive Quad Patches processing results	79
7.1 Explore Maps pre-processing results	97
Assisted Exploration	103
7.2 Interactive exploration methods	103
8.1 Test device characteristics	113
8.2 Test devices performance	115
Beyond Visual Replication	161
10.1 Annotated 3D model exploration methods	161
11.1 Results of NASA task load index questionnaire	179
11.2 Method classification	185

Preface

THIS thesis represents a summary of the work done from 2012 to 2014 at the Visual Computing group of CRS4 (Center for Advanced Studies, Research and Development in Sardinia) under the supervision of Enrico Gobetti, whom I really want to thank for trusting me and offering me the opportunity to be part of his research group and work with all the people there, which has been a great experience both professionally and personally.

The work in this thesis has been performed within the framework of the DIVA project (Data Intensive Visualization and Analysis), which is an Initial Training Network (ITN) funded by the EU within the 7th Framework Programme. It brings together six full partner institutions, namely, the University of Zurich (UZH), the CRS4, the University of Rostock, the Chalmers University of Technology, Diginext, and Holografika, and eight associate partners. Associate partners include Eyescale Software GmbH (EYE), Geomatics & Excellence (GEXCEL), Compagnia Generale di Riprese aeree (BLOM CGR), Centre d'Essais et de Recherche de l'ENTENTE (CEREN), Fraunhofer IGD, AIRBUS, NVIDIA GmbH, AMD. Both research centers and universities, as well as industry partners, are represented in this network aiming to exploit synergies.



During this time I also attended the PhD Program in Computer Science at the School of Mathematics and Computer Science of the University of Cagliari under the kind tutoring of Riccardo Scateni, who I would like to thank as well.

In particular I would like to thank all my colleagues at the CRS4 for their support and collaboration during the past years. Special thanks go to Fabio Bettio, Fabio Marton, Marco Agus, Gianni Pintore, Alex Tinti, Katia Brigaglia, Cinzia Sardu, Luca Pireddu, Alberto Jaspe, Roberto Combet, Ruggero Pintus, Antonio Zorcolo and all my other colleagues in the CRS4 which made my stay so joyful even on hard times. I would also like to express my appreciation for people with whom I had the pleasure of collaborating these years: Fabio Ganovelli, Marco Di Benedetto, Renato Pajarola, Claudio Mura, and all the people in the DIVA project.

Finally, I would like to express my most special thanks to my closest family and friends, with special mention to my parents and girlfriend, which have always shown their unconditional support even on the distance while tolerating my long absences.

The work presented in this thesis has been partially supported by the People Programme (Marie Curie Actions) of the European Union's Seventh Framework Programme FP7/2007-2013/ under REA grant agreement num. 290227.

Pula, Italy, February 2015

Marcos Balsa Rodríguez

CHAPTER

1

Introduction

The availability of highly detailed 3D content is growing at fast pace thanks to the rapid evolution of 3D acquisition techniques and 3D model creation techniques. At the same time, the proliferation of powerful portable devices (i.e., smartphones and tablets) and the high connection speeds available nowadays, provide new ways for exploring this information even in remote contexts. This imposes the challenges of efficiently handling and transmitting all this information, and being able to explore and analyze the information in meaningful ways. In this thesis, we study compact representations for the distribution and rendering of highly detailed 3D models on commonly available modern GPU architectures. Furthermore, we present methods to interactively explore this complex models by providing the user with tools for easy manipulation of the 3D objects and information exploration. This chapter outlines the motivation behind this research, summarizes research achievements, and describes the organization of the thesis.

1.1 Background and Motivation

THE widespread availability of mobile graphics terminals and WebGL-enabled browsers, has promoted the adoption of 3D graphics over the Internet as common multimedia content. Thanks to the rapid evolution of both 3D acquisition techniques and 3D model creation techniques, the availability of highly detailed 3D models is growing at fast pace. In addition, the ever-increasing 3D capabilities of mobile devices, together with pervasive high-speed connections being commonly available, open the door to a whole new world of opportunities to explore all this rich multimedia content even on the move.

In many domains, highly detailed 3D models are an important ingredient of the information flow that needs to be made available to the public, such as gaming, prototyping (e.g., remote inspection, marketing or analysis), quality control, or virtual shopping.

Current 3D repositories, such as Blend Swap, 3D Café or Archive3D, present the available 3D models through a few user-selected static images. Online exploration is limited to simple orbiting and/or low-fidelity exploration of simplified models, since photo-realistic rendering quality of complex synthetic environments is still hardly achievable within the real-time constraints of interactive applications, especially on low-powered mobile devices or script-based Internet browsers.

The visualization, in the context of complex 3D models, should be able to retain the high detail available in the 3D representation for the user to be able to appreciate the fine details (e.g., real 3d scanned object). Additionally, when exploring highly detailed 3D models, information can be found at multiple scales (e.g., from global shape to very fine details or even surface roughness

or damages). Thus, navigation techniques must provide seamless navigation between multiple scales without the user losing the spatial context. Annotated 3D models are commonly used for improving the user understanding of the 3D object being visualized. This additional information provides contextual information when the camera is moving around the model, amplifying the visual information. At the same time, annotations may also be used to help the user to retain spatial context (e.g., showing part-whole relations) during the exploration.

Current state-of-the-art methods have several limitations when coping with complex 3D models:

- **Limited complexity.** Many generic solutions have been presented for interactive visualization of massive 3D models on general “desktop” platforms [Yoon 08]. However, when targeting mobile platforms or scripted environments (e.g., Web browsers using Javascript) those techniques doesn’t scale well due to the limited resources available. In the case of web browsers, the use of a virtual machine imposes heavy constraints on CPU intensive processes, while on mobile devices the limited amount of memory resources and network bandwidth are the more restricting constraints, together with power consumption and limited computing performance.
- **Difficult interaction.** Exploring complex 3D models, with multi-scale information, requires the user to continuously move between global shape exploration and proximal navigation, when very close to the surface, in order to explore fine details. This kind of exploration requires either a good set of navigation tools providing enough free movement to cover all the interesting views of the model [Hach 13], or constraining camera movements allowing the user to easily move around the object while enforcing good viewpoints [Khan 05]. The approaches relying on camera movement decomposition (Rotation-Translation-Scaling) can be difficult for novice users. On the other hand, constrained camera movement solutions typically rely on a surface-sliding metaphor, presenting difficulties when handling disconnected surfaces, not uncommon in cultural heritage models, for instance. Thus, there is no solution providing a simple interface which is easy-to-use and provides enough freedom for the user to be able to explore the whole 3D model.
- **Limited understanding of information.** Current approaches for annotated 3D models have studied the integration of textual information in a 3D environment by presenting spatially coherent textual information attached to the models [Sonn 05, Jank 10], and more recently, bidirectional navigation between textual and 3D information [Gotz 07, Jank 12, Call 13]. All these methods typically show some indications or highlight the areas containing information, often introducing cluttering in the 3D view. In addition, precise clicking is commonly required to select and activate the information, forcing the user to change the interaction mode from navigation to selection. Both cluttering and interaction mode changes difficult the user to focus on the 3D virtual object itself.

When dealing with ubiquitous access to 3D and multimedia information, the challenge is to create new methods enabling the remote visualization of this large amount of data, while providing the means to examine and understand the objects being represented in its complexity (i.e., extracting information at various levels, not only visually). In order to achieve this goal, the visual performance of current visualization systems has to be improved, and paired with new interaction methods that allow the exploration of complex models that may contain associated information at different levels (i.e., multi-scale, context-based or location based).

1.2 Objectives

The main research objective of this thesis is to enable the interactive exploration and better understanding of complex 3D models using commonly available 3D platforms. Advancing the state-of-the-art in this area requires solving the following problems:

- **Improving Scalability of Visualization Methods for Complex 3D Models** Nowadays, large amounts of highly detailed 3D models are becoming increasingly available, so there is a clear need for specialized and efficient methods for visualizing these data-sets. Therefore, we need to study compact data structures that exploit the characteristics of current 3D platforms (i.e., Web browsers, mobile devices and desktop) for improving scalability both in the visualization and the streaming of the data.
- **Improving Navigation Techniques for Complex 3D Models** The exploration of complex 3D models with multi-scale information requires a simple and effective navigation interface that enables the user to explore and study both global shape and very fine details. Our plan is to study interactive methods which are easy to use and provide enough freedom, enabling the exploration of the whole virtual object, while, at the same time, helping the user to retain the spatial context.
- **Improving Understanding of Complex 3D Models** Providing effective 3D content presentation is particularly relevant when the goal is to allow people to appreciate, understand, and interact with intrinsically 3D virtual objects. In this context, it is important to go beyond visual replication, providing contextual information that integrates and enhances the 3D model view. We need, thus, to study new methods for presenting the user with additional spatially coherent information, while avoiding cluttering the 3D view or requiring the user to focus on the interaction or the contextual information instead of the 3D virtual object.

While some partial solutions for some of these problems have been proposed in the recent years, there is currently no existing single approach able to fulfill all these requirements.

1.3 Achievements

The research work carried out during this thesis has led to the following achievements:

- The development of a compact representation for general dense 3D mesh models [Bals 13c]. This method exploits the properties of conformal hierarchies of tetrahedra to produce a data structure which is adaptive, compact, and GPU friendly. Clipping the original triangulation at tetrahedra level allows us to create a local barycentric parametrization of the geometry, providing a compact representation which can be directly decoded in GPU. Further compression for network streaming is obtained on top of the compact GPU-friendly representation by exploiting local data coherence.
- The introduction of a compact image-based encoding for complex 3D models which are quad-parametrizable [Gobb 12]. This method exploits the constraint of quad-parametrizable meshes for producing a fully regular compact multiresolution image-based representation suitable for storage, distribution, and real-time rendering of highly detailed 3D models on modern commodity and web platforms.

- The introduction of a compact image-based representation for supporting complex environments [Di B 14]. We constrain the possible camera positions to an optimized set of precomputed views providing full coverage of the scene. Thanks to constraining camera positions to a fixed set, we can overcome real-time rendering limitations by precomputing a set of panoramic views for all possible camera positions using off-line photo-realistic renderers.
- The development of a scalable method for natural exploration of extremely detailed surface models [Bals 14a]. We extended the classical *Trackball* method with automatic pivoting and added context-based point-of-interest selection to obtain a user interface for inspecting complex objects which is general, predictable, robust and intuitive. Furthermore, this user interface can be implemented in a wide range of configurations ranging from small screen mobile devices to non co-located large screen setups [Bals 15].
- The introduction of an interactive camera controller which provides collision-free and automatic smooth transition from orbital to proximal navigation [Mart 14]. The method exploits a distance-field representation of the 3D object to support the exploration of disconnected surfaces. Furthermore, by decoupling view position and view direction computation we provide a smooth navigation while maintaining *good* view directions.
- The introduction of a graph-based method for exploring complex 3D environments [Di B 14]. In order to support interactive exploration of complex 3D environments, we rely on a graph-based representation of the scene composed of a set of panoramic view positions and a set of arcs between neighboring view positions.
- The development of a new framework for enhanced exploration of annotated 3D models [Bals 15]. The method integrates a stochastic adaptive recommendation system based on a structured spatial information representation, centered around annotated viewpoints, with a walk-up-and-use user interface that provides unobtrusive guidance towards interesting view points.

1.4 Organization

This thesis is organized to show in a natural and coherent order all the results obtained. For that purpose, we have defined three parts, each one covering our approaches in order to cope with current limitations.

Part I: Background and Motivation. In this first part, we describe and analyze the problem domain, and present the requirements arisen from this analysis. Also in this first part, we provide a brief overview of previous related work and identify current limitations. Finally, we detail our research goals for dealing with current limitations in order to satisfy the requirements.

Part II: Compact Representations for Complex 3D Models. Introduces our approaches for improving scalable visualization of complex 3D models on common 3D platforms. We exploit scene characteristics to design compact data structures and efficient algorithms for distribution and rendering tailored for supporting platforms with constrained resources.

Part III: Assisted Exploration of Complex 3D Models. Addresses the problem of exploring complex 3D models containing information at multiple scales. For that purpose, we study camera motion constraints and image-assisted navigation aiming to help the user during the exploration.

Part IV: Beyond Visual Replication. In most application domains visual replication of real objects is not enough when the goal is for the user to appreciate, understand, and analyze the 3D model. This part describes our approach for integrating additional information in complex 3D models and helping the user both during exploration and information discovering.

Part V: Conclusions. This last part summarizes the work presented in this thesis and presents some avenues for future work.

Part I

Background and Motivation

With the increasingly widespread introduction of mobile terminals and WebGL enabled browsers, 3D graphics is becoming common multimedia content. Furthermore, the rapid evolution of 3D acquisition and modeling techniques is promoting high resolution 3D models to become increasingly common, and potentially available for ubiquitous exploration.

Many application domains can benefit from this flow of highly detailed 3D models both for public access, or for collaborative inspection and analysis (e.g., virtual shopping, collaborative design, automatic and remote quality control).

For that purpose, new efficient methods for distributing and rendering complex 3D models are required. Moreover, the intrinsic complexity of those models requires new interaction methods that help the user during the exploration. At the same time, in order to exploit the possibilities of this complexity, new methods for integrating contextual information should be proposed, aiming for better comprehension of the visual information.

In this first part, we will present a detailed study of the application domain, and a brief overview of the state-of-the-art on closely related topics. Also in this part, we will present and discuss our approach for dealing with current limitations.

Application Domain

In this chapter we analyze the needs of interactive exploration and understanding of complex 3D models in common 3D platforms. Taking as a representative example the domain of cultural heritage, we present a detailed study of requirements arisen from analysis of related work and discussions with domain experts.

2.1 Introduction

As it already happened with photography and audio/video, the creation of 3D content is becoming more and more affordable in terms of time, user skills and, consequently, economic investment. In the last few years, 3D scanning systems have become commodity components. At the same time, the rapid evolution and proliferation of low-cost graphics hardware has made advanced 3D modeling available to a variety of user. As content becomes easier to create and cheaper to host, more companies and individuals are building virtual worlds (e.g., Second Life hosts 270 terabytes of user-generated content in 2009 [Lab 09], and this is growing by approximately 100% every year).

With the increasingly widespread introduction of mobile terminals and WebGL enabled browsers, 3D graphics over the Internet is expected to attract a lot of additional attention. Still, unlike what has happened for standard media, which have converged high quality compressed formats specifically designed for storage and streaming, essentially based on the same small set of concepts, distributing and rendering non-trivial 3D models, especially on low-cost or mobile platforms, is still challenging. Detailed 3D models are heavy, non-trivial to render, and are experienced in a highly non-linear interactive way. These characteristics impose fast incremental loading and reasonable compression, GPU accelerated rendering methods, and adaptive view-dependent culling techniques. While a lot of generic solutions have been presented for general “desktop” platforms [Yoon 08], there is now an increasing interest for techniques tuned for lightweight, interpreted, and scripted environments (e.g., Web browsers).

Many application domains can benefit from this flow of highly detailed 3D models both for public access, or for collaborative inspection and analysis (e.g., virtual shopping, collaborative design, automatic and remote quality control). In particular, cultural heritage valorization and cultural tourism are some of the sectors which are benefiting from this evolution, as new technologies provide means to cover the pre-visit (documentation), visit (immersion) and post-visit (emotional possession) phases [Econ 11, Rodr 12]. Nowadays, it is possible to explore new ways of accessing massive sources of information which were previously only accessible at Museums, or not even accessible but stored in some warehouse because of the lack of space for exhibition. Through 3D digitization, large amounts of high resolution 3D objects are becoming available, thus requiring new methods to provide public access to these databases.

In cultural heritage, specially, an accurate visualization of the 3D model allows the user to be able to appreciate the fine details present in the real artifact. Multi-scale information is specially relevant in cultural heritage, where the macro-scale provides global shape and function information, while micro-scale gives information on the nature of the object (e.g., material), its manufacturing process (e.g., carvings), or even details on the degradation process (e.g., burned parts). When navigating through multiple scales, specially when getting very close to the model surface in order to appreciate fine details, it is very common for the inexperienced user to lose spatial context. Therefore, navigation techniques providing seamless transition between macro-scale and micro-scale exploration are required, while guided navigation is desirable in order to aid the user in keeping visual context during the exploration.

Moreover, understanding complex 3D models typically requires the support of 3D annotations to amplify and enhance the visual information. Additional information, in the form of textual annotations, overlaid images, or any other multimedia content, is generally used to provide contextual information to the user aiming to explain or extend the visual information. Depending on the context, this information can be of many types, including historic or manufacturing details (i.e., cultural heritage), material description (i.e., prototype design), or descriptive annotations (i.e., virtual training applications). There is so a need for effective 3D annotation techniques which help the user to explore and understand the knowledge enclosed in complex 3D models.

In the next section we present and discuss the requirements arisen from our study of the application domain, which involved experts from the cultural heritage domain. Next Chapter 3 provides a brief overview of previous work related to our problem domain, and Chapter 4 will describe our approach for coping with current limitations.

2.2 Requirements

In order to define our goals, we have started with a detailed analysis of the application domain. While our approach is of general use, our work has been motivated by the Mont'e Prama project, a collaborative effort between our center and the *Soprintendenza per i Beni Archeologici per le Province di Cagliari ed Oristano* (*ArcheoCAOR*, the government department responsible for the archaeological heritage in South Sardinia), which aims to digitally document, archive, and present to the public the large and unique collection of pre-historic statues from the Mont'e Prama complex, including larger-than-life human figures and small models of prehistoric buildings. The project covers aspects ranging from 3D digitization to visual exploration.

During the analysis, we involved a group of domain experts in a participatory design process with the goal of collecting the detailed requirements of the application domain; the expert's group included two archaeologists from *ArcheoCAOR*, two restoration experts from *CRCBC*, and one museum curator from *Museo Archeologico Nazionale di Cagliari*. Thus, we have been able to involve domain experts in the definition of the requirements, including particular requirements for the cultural heritage domain.

We have grouped the requirements derived from our analysis of the problem, and meetings with domain experts, in three blocks: visualization, exploration, and information presentation.

2.2.1 Visualization Requirements

R1. High-resolution details (magnified micro-structure). Thanks to recent 3D acquisition techniques, highly detailed 3D representations of real objects can be produced. This means that these complex 3D models present information at multiple scales (i.e., global shape and fine surface details). Even the finest material micro-structure carries valuable information (e.g., on the carving process, or giving hints on the deterioration process, see Fig. 2.1). For instance, the Mont'e Prama statues have millimeter-sized carvings, and thus require sub-millimetric model precision. This carving information should be clearly perceivable at all scales, and should be magnified for close inspection.

R2. Large-scale visualization (real-world-sized macro-structure). There is a wide range of models to cover for the various application domains, ranging from small pieces (e.g., screws in the case of quality control applications), to big objects (e.g., a car or a ship in the case of collaborative design). In



Figure 2.1: Dragon statue. Left: Notice the amount of micro-details that are present on the surface of this model. Center: Geometry of the 3D scanned reconstruction; Right: Colored 3D rendering of the reconstructed 3D model. Courtesy of Augmented Vision department at DFKI.

cultural heritage, in particular, there are larger-than-life human statues (see Fig. 2.2), for instance, which were constructed at imposing scale on purpose, and this macro-structure information should be immediately conveyed to the visitor through a real-scale (or larger-than-real) presentation. In order to cover those use cases we need to support large (wall-sized) displays.



Figure 2.2: Imposing scale of the David statue. In this photograph of the David statue by Michelangelo can be observed the imposing scale of the statue in contrast with human size. Courtesy of Wikipedia.

R3. Efficient storage and distribution. Our focus is on ubiquitous exploration of complex 3D models, so due to the inherently high resolution of this kind of models, there is typically a large amount of data to be stored, streamed through network, and rendered on a variety of 3D platforms. Thus the 3D representations used for distribution and rendering must be compact and support efficient streaming of the data, in order to permit remote exploration (see Fig. 2.3). At the same time, compact representations should minimize

the space required for the storage of large amounts of highly detailed 3D models, which would otherwise require vast amounts of storage.

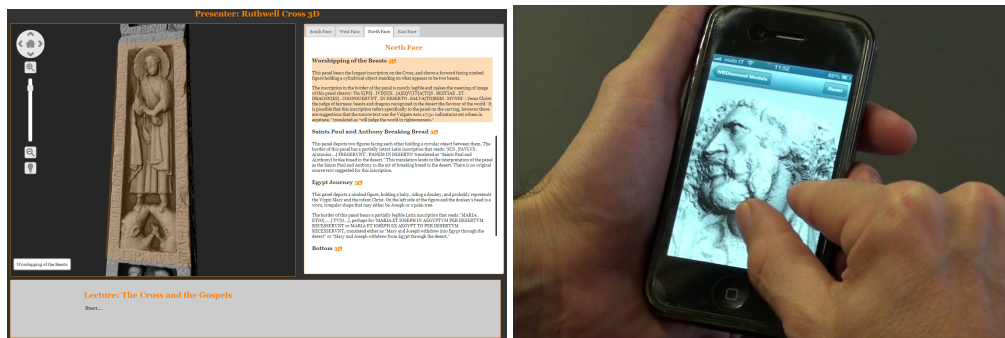


Figure 2.3: Highly detailed 3D models. Left: Browsing a 3D model of the Ruthwell Cross in a Web browser (Courtesy of Marco Callieri); Right: Browsing a 3D model of St. Matthew by Michelangelo on an iPhone. A considerable amount of data need to be transmitted in order to display the high resolution of these 3D models.

R4. Complex environment support. Some use cases does have the need for complex environments, both including scenes composed of many different complex 3D objects (e.g., a virtual museum), or scenes requiring complex lighting simulation (e.g., building prototyping, see Fig. 2.4).



Figure 2.4: Photo-realistic rendering of a hotel room. Several hours are required to generate images with this quality. Courtesy of CGRendering.com.

2.2.2 Exploration Requirements

R5. Seamless interactive exploration and zooming (macro and micro-structure). Comprehension of complex 3D models impose the capability to seamlessly move from macro-structure analysis, providing information on function and context, and micro-structure analysis, providing information on nature, shape and signification of decorations. Camera navigation should thus support both global object motion and proximal navigation (panning

over a surface to inspect details). The control modes should be active with real-time feedback, in order to provide the sense of control, and support smooth and seamless object inspection, going back and forth from shape inspection to detail inspection in natural way (see Fig. 2.5).



Figure 2.5: David statue by Michelangelo. Left: Face detail of the statue; Right: Closer detail of an eye. When navigating such complex models it is easy to lose the spatial context.

R6. Fast learning curve and assisted navigation. A good number of use cases in this application domain target non-technical users. Thus, the user interface must be simple and fast to learn, while providing unobtrusive guidance in complex interaction operations. In the case of a museum installation, for instance, where *walk-up-and-use* interfaces are expected, the visitors' experience could be easily frustrated if the proposed interaction paradigm does not allow them to immediately explore the content, through a natural user interface with an extremely short learning curve. Moreover, since museums must manage large amounts of visitors, long training times and/or guided training are not affordable (see Fig. 2.6 left).



Figure 2.6: Museums and exhibitions. Museums and exhibitions are typically crowded with visitors aiming to explore its contents. Left: Photo from the New Acropolis Museum of Athens. Courtesy of Wikipedia; Right: Photograph of the International Trade Fair of Sardinia showing an installation where there is one controlling interface, but multiple users can share the visualization experience.

- R7. Single user control, multi-user fruition.** In collaborative inspections, such as guided museum visits or prototype presentations, it is important to maximize the reach of the visual information to the public. In the case of museums, visitor experience tends to be personal, self-motivated, self-paced, and exploratory. At the same time, the support of multi-user viewing enables visitors to benefit from other people's experiences in terms of user interaction behavior and object exploration, also encouraging collaboration and discussions (see Fig. 2.6 right). While a single-user interface is considered appropriate, the physical setup should comfortably support at least small groups of observers.
- R8. Focus on the 3D virtual object (avoid occlusion from people and/or interaction widgets).** The 3D object being visualized is the important information as a general rule (e.g., collaborative design of a car). Thus, it should not be obstructed by other people or general clutter (e.g., interaction widgets), see Fig. 2.7. Specially in the case of a Museum, the visitor focus should thus be guided to the presentation medium, instead of concentrating in the user interface or in searching for information points (i.e., hot spots, textual labels).



Figure 2.7: Cluttered presentation. The display is heavily occluded by people in front of it, thus difficulting the user to focus on the information. Courtesy of GestureTek technology.

- R9. User interface and display flexibility.** There is a wide range of possible setups that can be of use in this application domain. In the context of collaborative design, or museum exhibitions (i.e., visit phase(immersion)), for instance, large displays should be supported in order to provide a better coverage of large objects, or for supporting multiple observers. On the other side, mobile devices and web browsers provide an interesting platform for promoting virtual shop products, or for covering the pre-visit (documentation) and post-visit (emotional possession) stages of a museum visit. Thus, we are facing a very wide range of display sizes (i.e., mobile devices, display walls) and a variety of user interface setups (i.e., co-located user interfaces

for mobile applications, or non co-located touch interfaces paired with large projection displays for museum exhibitions), see Fig. 2.8.

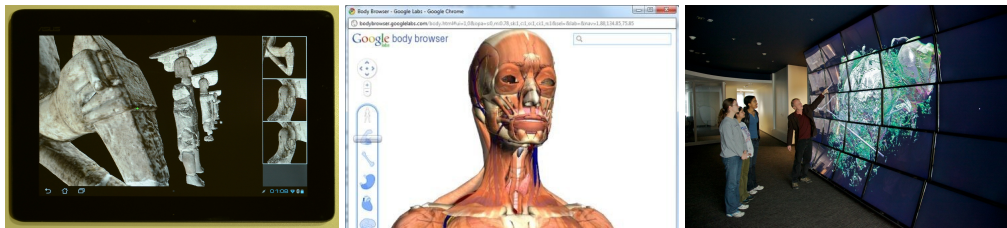


Figure 2.8: Various display setups. Left: A mobile device; Center: A web browser (Courtesy of Google Body Browser); Right: A 3D Wall at Northwestern University (Courtesy of Luc Renambot).

2.2.3 Information Presentation Requirements

R10. Annotation system. When displaying complex 3D models, additional information is needed to explain or describe visual information (i.e., integrated in the global context). A 3D annotation system is thus required in order to provide means for integrating additional information that will be presented to the user (see 2.9). In the context of cultural heritage, it is of particular interest to integrate additional information, both for giving interesting information to casual visitors, and for archaeologists and museum curators to document and analyze the artifacts (i.e., degradation status, historical notes, restoration details).

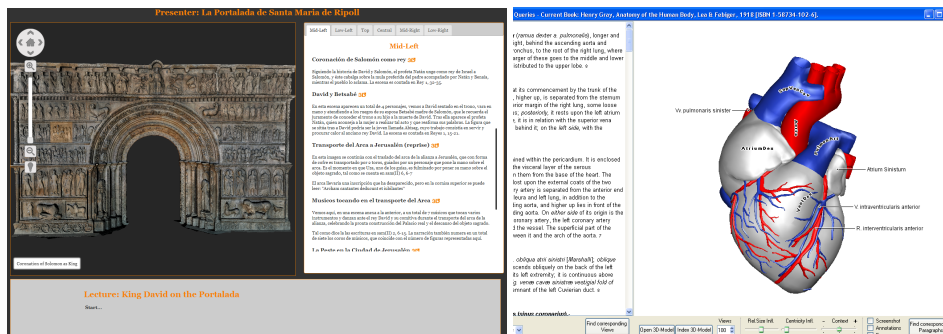


Figure 2.9: Annotated 3D models. Left: Browsing the Portal of the Ripoll Monastery. Courtesy of Marco Callieri; Right: Browsing the 3D representation of a heart (Courtesy of Timo Götzelmann).

R11. Spatially coherent information. The additional information must integrate and enhance the visual information being presented to the user. Thus, it is relevant to provide spatial correspondences between visual and additional information in order to emphasize the contextual relations. This spatial links may be tightly linked to the visual information (i.e., overlaid reconstructions, see 2.10 right), or just correlate high level information with respect to parts

or details in this region (see 2.10 left). We need to provide means to define and present these links.

R12. Information semantics. Many use cases arise for exploiting complex 3D models in common application domains, where additional information can be subject to different semantics. For that purpose, we need to define a flexible annotation system which enables the authoring system to define a wide range of semantic dependencies both on spatial relations (e.g., tightly coupled to view positions or not, see Fig. 2.10), and between related annotations (i.e., defining a hierarchy of annotations describing a presentation order going from global information to particular details, thus providing a coherent flow of information).

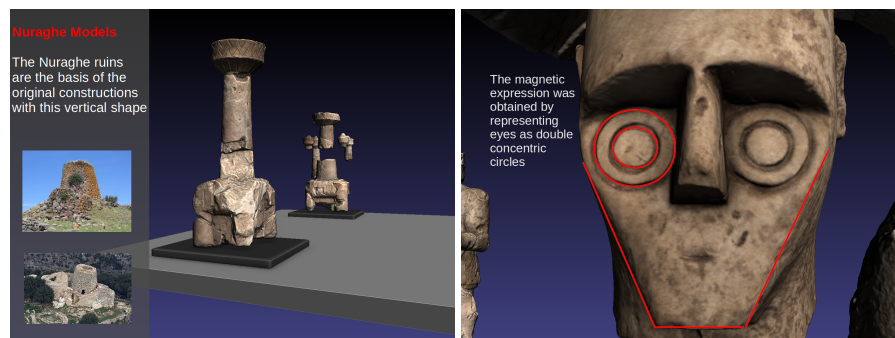


Figure 2.10: Various annotation links. Left: Additional information with little spatial constraints; Right: Overlaid information showing a tight spatial link between 3D information and additional information.

R13. Information authoring. In order to provide a rich annotation system, at least textual and visual information (drawings, images) should be supported. The authoring should be simple enough for non-technical users such as museum curators or archaeologists without particular training.

2.3 Discussion

In this chapter we have described the application domain, and provided a detailed analysis of the requirements arisen from our study of the problem domain, and discussions with domain experts. These requirements can be broadly categorized into *Visualization requirements* of 3D data, which dictate the need of seamless rendering at multiple scales and interactive rates of very detailed models on a variety of platforms, *Exploration requirements*, which impose easy to learn navigation methods that scale from large display installations to mobile settings, and *Information presentation requirements*, that impose the need of going beyond pure visual replication to incorporate both 3D models and annotations. Even

though these requirements arise from the analysis of a particular application domain (presentation of cultural heritage objects), most of them are applicable in a variety of domains where there is a need of exploring intrinsically 3D objects in an interactive way.

The requirements presented in this chapter impose severe constraints on the technical solutions. In particular, there is a need for scalable rendering solutions of 3D objects capable to meet real-time constraints in both local and remote settings, controlling bandwidth use at all levels of the distribution and rendering pipeline. Moreover, such a rendering solution should be driven by camera control interfaces that are flexible and usable with minimal training in a variety of settings. Finally, in most applications, both the rendering system and the user interface system should be capable to present not only real-looking 3D objects, but also associated information.

The state-of-the-art solutions proposed so far are discussed in the next chapter.

2.4 Bibliographical Notes

Most of the contents of this chapter were taken from papers [[Mart 14](#)] and [[Bals 15](#)], in which we discuss two novel systems for the interactive exploration and understanding of complex 3D models.

CHAPTER

3

Previous Work

This chapter provides a brief overview of the previous work on technological areas which are closely related to our problem domain, including visualization, interaction and information presentation approaches.

3.1 Introduction

IN order to satisfy the various requirements (**R1-R13**) discussed in Sec. 2.2, we need to address a number of technological limitations. In this chapter, we will present and briefly discuss current approaches which are closely related to our problem domain.

In particular, we will study techniques related to scalable visualization of complex 3D models, for supporting rendering highly detailed 3D models on common 3D platforms, including low-profile platforms (i.e., mobile/web platforms). Moreover, we will analyze current approaches for interactive exploration of complex 3D models in order to help the user keep the spatial context even when navigating from global shape to close surface inspection.

Our goals include not only visual exploration, but also better understanding of complex 3D models. Thus, we will also study current techniques for integrating additional information and presenting it to the user in order to amplify visual information.

We have distributed the related work in three main topics, which provides a loose mapping to the requirement classification proposed in the previous chapter: scalable visualization (Sec. 3.2), interactive exploration (Sec. 3.3), and information discovery (Sec. 3.4).

3.2 Scalable Visualization of Complex 3D Models

In order to improve the scalability of complex 3D model rendering (R1, R3) we have to deal with a number of computer graphics topics which have been long studied. Here we'll discuss the approaches most closely related to the work presented in this thesis. For further details, the reader may refer to well established surveys on massive model rendering [Yoon 08], image-based rendering [Shum 07], and mobile graphics [Capi 08].

3.2.1 Compact Mesh Models for Distribution and Rendering.

Although existing solutions have been demonstrated to be efficient on “desktop” platforms, only a few examples exist for rendering light 3D models on portable platforms (e.g., MeshPad [ISTI 12] for meshes or PCL [Mari 12] for points). Much of the work in model distribution has focused so far on compression of mesh structures rather than adaptive view-dependent streaming. MPEG-4 is a reference work in the field [Jova 08]. Classic methods for view-dependent LOD and progressive streaming of arbitrary meshes were built on top of fine-grained updates based on edge collapses or vertex clustering [Xia 96, Hopp 97, Lueb 97]. Many compression and streaming formats for the web have been built upon them [Magl 10, Blum 11, Nieb 10]. These methods, however, are CPU-bound and spend a great deal of rendering time computing a view-dependent triangulation prior to rendering, making their implementation in a mobile setting particularly challenging. With the increasing raw power of GPUs, currently higher-performance methods typically reduce the per-primitive workload by pre-assembling optimized surface patches [Cign 04, Yoon 04, Cign 05, Borg 05, Gobb 04a, Gobb 04b, Gosw 13], or introduce techniques for performing view-dependent refinement within geometry shaders [Hu 10].

These methods are proved very effective in terms of rendering speed, but still require coding of non-trivial data structures and techniques for decompression, leading to potential problems in script-based web implementations, or doesn't scale well on platforms with limited memory and computation resources (i.e., mobile platforms).

3.2.2 Quad-Parametrization and Re-meshing.

One approach commonly used for exploiting mesh characteristics for designing new rendering methods consists on reparametrizing the input mesh into a more suitable format. Representing complex two-manifold models as a collection of quads requires a parametrization of input models (refer to [Shef 06] for a survey).

The simplest approach is single-disk parametrization [Floa 05], which, however, can be applied only to genus-0 meshes and leads to high distortions unless the mesh has almost zero Gaussian curvature everywhere. These approaches [Lee 98, Prau 03, Khod 03, Schr 04, Krae 04] rely on a base mesh for the parametrization, using a triangle-based domain.

3.2.3 Details and Adaptive Mesh Refinement on GPU.

Many solutions have been proposed for dealing with highly detailed 3D models which rely on a coarse base representation on top of which details are added later in the rendering pipeline. An approach to the problem of rendering generalized displacement mapped surfaces by GPU ray-casting was proposed in [Oliv 00, Wang 03, Wang 04]. Other generalizations involve replacing the orthogonal displacement with inverse perspective [Babo 06], replacing the texture plane with a quadric [Manu 05], and handling self shadowing in general meshes [Poli 05]. The evolution of graphics hardware has allowed many surface tessellation approaches to migrate to the GPU, including subdivision surfaces [Shiu 05], NURBS patches [Guth 05], constrained urban models [Cign 07], and procedural detail [Boub 05, Boub 08]. This makes it possible to generate geometric details directly in the vertex shader.

3.2.4 Mesh Compression.

Compressed graphics data potentially enable platforms with very constrained resources, i.e., mobile devices, to better utilize the limited storage space and bandwidth at all levels of the pipeline. Many mesh compression algorithms offer good performance in compression ratio for both topology and vertex attributes. MPEG-4 [Jova 09] is a reference work in the field, and includes 3D mesh coding (3DMC) algorithms based on topological surgery algorithm [Taub 98b] and progressive forest split [Taub 98a]. State-of-the-art topology coders [Ross 01] are capable to achieve the theoretical minimum of 1.62 bpt (bits/triangle), approximately 3.24 bpv (bits/vertex). The decoding processes are however rather complicated and do not construct structures suitable for fast direct rendering. We focus, instead, in computing a representation for geometry that reduces the bandwidth required to transmit it to the graphics subsystem. This is achieved by constructing, for each mesh fragment, compressed primitive-topology representations that ensure high vertex coherence, as well as reducing vertex data size. For topology, Chhugani et al. [Chhu 07] presented an algorithm tailored for hardware decompression with 8 bpt (16bpv) by maintaining a cache coherent triangle sequence, and Meyer et al. [Meye 12] proposed a coding technique reaching 5 bpt (10

bpv), which, however, requires CUDA for decompression. Similarly to Chhugani et al. [Chhu 07], we sort topology and vertex data after computing a cache-coherent rendering sequence, using, however, a generalized strip optimized for the post-transform vertex cache rather than a triangle list. Hardware-compatible vertex data compression is typically achieved in this context by attribute quantization. Since global position quantization [Calv 02, Purn 05, Lee 09] provides poor rate-distortion performance for large meshes, recent efforts have concentrated on local quantization techniques [Lee 10], which, however, lead to cracks for multiresolution meshes.

3.2.5 Image-Based Rendering.

While in recent years, research efforts have produced systems capable of rendering moderately complex environments on the web and/or mobile devices [Magl 10, Nieb 10], real-time constraints limit achievable quality to moderate geometric complexity, simple shading models and/or baked illumination.

In order to cope with real-time limitations on common 3D platforms, we focus on supporting photo-realistic views of complex scenes through pre-computation (R4). Using image-based techniques to remove limitations on scene complexity and rendering quality for interactive applications, as well as to improve application usability is an old idea, that dates back at least to the branching movies of the 80s [Lipp 80] and the navigable videos and environment maps of the 90s (e.g., [Chen 95, Kimb 01]). More recently, these approaches have flourished in the context of applications that exploit camera pose (location, orientation, and field of view) and sparse 3D scene information to create new interfaces for exploring physical places by browsing large collections of photographs or videos [Snav 06, Vinc 07, Kopf 10, Tomp 12, Sank 12]. While much of earlier research has focused either on authored paths or on pre-acquired large photo/video collections, with an emphasis on view interpolation, image-based rendering from captured samples, or interfaces for navigation among large sets of precomputed images, we focus instead on how to efficiently and automatically *create* a set of representative views and connections starting from a given 3D environment, and on how to increase the sense of presence during constrained navigation.

In our approach in Chapter 7, we restrict the possible camera positions (but not orientations and fields of view), so we can side-step the complex problem of computing pixel-accurate viewpoint interpolations in general shading environments [Sinh 12]. Our method is therefore applicable to scenes including effects such as participating media, reflections, and refractions.

3.2.6 View Selection

When aiming to generate an image-based representation of a 3D model (see previous Sec. 3.2.5) arises the question of what are the *good* views of a 3D object. This question has been addressed by many researchers in perception, computer vision and computer graphics. Placing viewpoints to guarantee a complete and accurate sampling is an extensively studied computer vision problem [Scot 03]. Reconstructing a “virtual” object is a different problem, since there are no physical constraints on the field-of-view, position, orientation, and motion of the virtual sensors, and acquisition errors are limited to the discretization done by rasterization. View selection in this context has mostly been studied in image-based rendering literature. Most techniques use a fixed set of views [Rapp 98, Andu 07], leading to gaps or holes for non-trivial objects. Fleishman et al. [Flei 00] and Vazquez et al. [Vazq 02] introduced adaptive techniques for covering polygonal models, but require the prior definition of a “walking zone” for bounding camera positions.

3.3 Interactive Exploration of Complex 3D Models

When exploring complex, i.e., highly detailed, 3D models, with information that is to be found at multiple scales, it is easy for the inexperienced user to lose spatial context (R5, R6). For that reason, interaction methods which allow natural exploration of the whole 3D model from a wide range of distances (i.e., global shape, close surface inspection) are required. In addition, in order to support ubiquitous exploration of 3D models we need to consider a wide range of display and user interface configurations, including mobile devices and Web browsers, but also large display installations (R2, R7, R9). In this section we will discuss the most relevant approaches that deal with these problems.

3.3.1 Motion Control for Virtual Exploration

In the context of visualization of massive complicated scenes, users require interactive control to effectively explore the data (R5). Most of the work in this area is connected to camera/object motion control [Chri 09, Jank 13]. Variations of the virtual trackball [Chen 88, Shoe 92, Henr 04], which decompose motion into pan, zoom, and orbit, are the most commonly employed approaches. In order to solve the *lost-in-space problem* and avoid collisions with the environment, Fitzmaurice et al. [Fitz 08] have proposed the Safe Navigation Technique, which, however, requires explicit positioning of rotation pivot, and thus needs precise pointing. Moreover, motion decomposition and pivot positioning can be difficult for novice

users. For this reason, a number of authors have proposed techniques for effective object inspection that constrain viewpoint by using precomputed/authored camera properties [Hans 97, Burt 02, Burt 06], increasing authoring time and limiting effective view selection. Proximal surface navigation methods constrain the camera to stay in a region around the object and with a specific orientation with respect to the visible surface [Khan 05, McCr 09, Moer 12, Mart 12b]. These methods, following a hovercraft metaphor, slide a target point on the surface and compute camera positions so as to look at this point from a constant distance and at a good orientation. Surface/depth smoothing and algorithmic solutions are used to reduce jerkiness and deal with singularities and out-of-view-field collisions. Moreover, disconnected surfaces are difficult to handle with this approaches.

3.3.2 Image-assisted Exploration

Using views to help navigate within a 3D data-set is often implemented with thumbnail-bars. At any given time, one image of the data-set can be selected by the user as current focus. For a 3D navigation application [Lipp 80, Snav 06], where images are linked to viewpoints, selecting the focus image drives the setup of the virtual camera. Often, these images are also linked to additional information, which is displayed when the user reaches selects them, as an alternative to the usage of hot-spots [Andu 12, Beso 08]. The organization of the images in this kind of tools can be considered a challenging problem in many CH scenarios, since simple grid layout approaches do not scale up well enough with the number of images. A trend consists in clustering images hierarchically, according to some kind of image semantic, like combining time and space [Ryu 10], spatial image-distances [Eps 07, Jang 09], or a mixture of them [Mota 08] to automatically, or interactively [Girg 09, Cram 09] compute image-clusters. Most of these works strive to identify good clustering for images, rather than good way to dynamically present and explore the clustered data-set. Goetzelmann et al. [Gotz 07], presented a system to link textual information and 3D models, where links on the text are associated to predefined points of view, and the definition of a point of view activates a mechanism proposing contextual information.

3.3.3 Small Screen Interfaces for 3D Model Exploration

Most of the systems described in the previous Sec. 3.3.1 require that the user has direct control over the visualization space, but this solution can be ineffective when the visualization area is very small (R9), like it happens in smartphones or when explore model-details (i.e., very close views). To solve this problem,

Declé and Hachet [Decl 09] proposed an indirect method based on strokes for moving 3D objects in a touch screen mobile phone, while Kratz et al. [Krat 10] introduced an extension of the virtual trackball metaphor, which is typically restricted to a half sphere and single-sided interaction, to actually use a full sphere, by employing the “iPhone Sandwich” hardware extension which allows for simultaneous front-and-back touch input. However, latter solutions are suited for small scale models and they employ fixed center of rotation in the barycenter. An automatic pivoting method has been recently presented by Trindade and Raboso [Trin 11], but the method requires access to depth buffer, and it is not easily customizable to all mobile systems. Furthermore, their method computes the rotation center as intersection of the viewing vector with the object surface, and it suffers from discontinuities when complex models with sharp features are considered.

3.3.4 Multi-touch Interfaces for 3D Model Exploration

In recent years, researchers started to combine multi-point input technology with interactive 3D graphics. Hancock et al. [Hanc 09, Hanc 07] proposed different approaches using direct and indirect multi-touch interaction techniques for tabletop surfaces to manipulate 2D and 3D content. Martinet et al. [Mart 12a] have studied the effect of DOF separation in a 3D manipulation task on a direct multi-touch display. In their work, Reisman et al. [Reis 09] proposed co-location constraints for direct-touch object manipulation. The Cubtile has been designed by de la Riviere et al. [Rivi 08], where users interact with 3D content from an indirect multi-touch box-shaped device. Recently, direct touch interfaces have been created for interacting with visualizations in application domains ranging from astrophysics [Fu 10] to oceanography [Butk 11], fluid mechanics [Klei 12], and medicine [Lund 11]. In particular, Yu et al. [Yu 10] proposed a general direct-touch technique allowing users to explore 3D data representations and visualization spaces. Recent work on multi-touch surfaces has also focused on indirect interaction, which is not guided by co-location concerns, and seems particularly interesting for interactive 3D visualization [Mosc 08]. In particular, Knoedel and Hachet [Knoe 11] showed that direct-touch shortens completion times, but indirect interaction improves efficiency and precision, and this is particularly true for 3D visualizations. All these systems are targeted to scientific visualization of complex data, where non-trivial interaction may be required, thus not being suitable for use cases like a museum setting with mostly naive users.

3.3.5 Large Displays and Dual Display Setups

In order to support interactive large display installations (R2) dual display setups are typically used. Recently, a number of visualization systems have been presented exploiting two display surfaces. In general, these systems employ an horizontal interactive touch table, and a large display wall visualizing the data, sometimes combining them in a continuous curved display [Weis 10b, Wimm 10]. A two-display volumetric exploration system was proposed by Coffey et al. [Coff 12], which however employs a World-In-Miniature metaphor, simultaneously displaying a large-scale detailed data visualization and an interactive miniature.

3.3.6 3D Stereoscopic Rendering with Light Field Displays

Recent advances in 3D displays provide an interesting platform for exhibition installations where highly detailed 3D models can be presented exploiting stereoscopic viewing, thus providing a much more immersive experience (R2, R7). Light field displays provide unrestricted stereoscopic viewing and parallax effects without special glasses or head tracking. They are intrinsically multi-user and can be built by using high-resolution displays or, alternatively, multi-projector systems with parallax barriers or lenticular screens. The light field display hardware employed for this work is manufactured by Holografika (see www.holografika.com) and is commercially available. It uses a specially arranged projector array, driven by a cluster of PCs, and a holographic screen. Large, multi-view light field displays require generating multiple images, one for each available perspective. In state-of-the-art rendering methods for such displays, multiple center of projection (MCOP) geometries [Jones 07] and adaptive sampling [Agus 08] are exploited to fit with the display geometry and the finite angular resolution of light beams. One particular characteristic of these displays is the varying resolution depending on the projection depth, which is typically optimal on the screen plane.

3.4 Information Discovery on Complex 3D Models

In the context of complex 3D models there is typically a large amount of information to be found all around the object (R10, R11). Even with guided navigation methods, that help the user to retain the spatial context while providing smooth natural navigation at multiple scales, there arises the problem of information presentation (R8, R12, R13).

3.4.1 Contextual Information Representation and Presentation

Visual displays can be categorized into different types based on the relation between the representation and its referent and the complexity of the information represented [Hega 11]. Our work falls in the category of visual-spatial displays that dynamically mix 3D representation with associated overlays. We do not focus on designing navigation aids or displays for specific tasks (e.g., location awareness), but, rather, simply on providing flexible means to unobtrusively guide the user towards “interesting” nearby locations and to present contextual information. The majority of data representations for context-aware systems focus on general representations for data interconnections, rather than on interconnections between structured information and associated objects [Bolc 07].

Most works on information visualization [Liu 14] concentrate on data analysis, extrapolating results and presenting them using graphical representations tailored for better human comprehension, while we focus on techniques for enhancing 3D object exploration. Riedl et al. [Ried 06] propose narrative mediation as alternative to branching stories in order to provide non-linear narrative generation, but there is no mention to handling spatial relations.

Using linked multimedia information to enhance the presentation of complex data has been long studied, mostly focusing on guided tours [Fara 97], text disposition and readability [Sonn 05, Jank 10], usability of interaction paradigms [Bowm 03, Poly 11], and the integration of interconnected text and 3D model information with bidirectional navigation [Gotz 07, Jank 12, Call 13]. All these methods require precise picking to navigate through the information, thus presenting problems when targeting non co-located interaction setups (e.g., large projection displays), and often introduce clutter in the 3D view to display the pickable regions. An alternative to picking are methods that use postures or gestures to trigger visualization of contextual information, e.g., in the form of contextual menus [Isen 12].

3.5 Discussion

Interactive exploration of massive models has been well-studied during past years, but current methods doesn't scale well under very constrained resources, such as happens in mobile platforms where memory and processing resources are typically very limited. On scripted environments, like Web browsers, the CPU performance is heavily burdened thus probably preferring to move much of the processing to the GPU, specially on desktop platforms with powerful GPU hardware.

Current compression techniques are capable of achieving very good ratios at the expense of costly CPU or GPU decoding, thus not being suitable for platforms with limited computing capabilities. In order to cope both with limited memory resources and computing capabilities, compact data representations are required in order to better utilize the available resources by maintaining a compromise between compression ratio and decoding cost.

When dealing with scenes with complex lighting, real-time constraints limit the visual quality that can be achieved, in particular on limited platforms such as Web browsers or mobile devices. Thus, image-based techniques have to be exploited in order to precompute the visual information for interactive exploration.

Current interaction techniques are typically not suitable for mobile devices with small screens, where direct control is difficult due to small interaction space and finger occlusions. Other techniques designed for small screens are just suitable for small scale models, with a fixed center of rotation, thus not being able to provide the user with means to explore the multi-scale information present on complex 3D models. Scalable interaction methods that can be adapted to a wide range of display sizes supporting smooth multi-scale navigation have not yet been proposed.

In order to facilitate the exploration of highly detailed 3D models image-assisted methods and guided navigation are typically used. At the same time, contextual information helps to amplify the visual representation providing the user with better comprehension of the 3D virtual object. Current methods for contextual information presentation are not well-suited for small scale displays or non co-located interaction on large displays since they require precise picking, and typically introduce cluttering in the visualization. Therefore, non obtrusive and scalable methods for presenting contextual information to the user should be studied.

In this thesis, our focus is ubiquitous exploration of highly detailed and annotated 3D models. For that purpose, we will tackle memory and computing limitations on current platforms. We will also study interaction methods for exploring complex 3D models on a wide range of display configurations, as well as methods to aid the user during the exploration, both integrating guiding and contextual information components.

Next Chapter 4 will present our approach for dealing with current limitations.

3.6 Bibliographical Notes

Most of the contents of this chapter were taken from papers [[Gobb 12](#), [Bals 13b](#), [Bals 14a](#), [Mart 12b](#), [Mart 14](#), [Di B 14](#), [Bals 15](#)], in which we propose solutions to

the limitations of current state-of-the-art.

CHAPTER

4

Work Plan

In the previous chapters we have defined and analyzed our application domain. Moreover, we have also presented a brief overview of the state-of-the-art covering results that deal with our problem domain, and identified current limitations in terms of scalable rendering solutions, camera control interfaces, and information presentation techniques. In this chapter, we will present our research goals aiming to provide solutions for coping with current limitations. The basic idea underlying our approach is to impose constraints on kind of objects handled, annotation structure, and camera control techniques to propose efficient specific solutions. In particular, in terms of data representations and rendering techniques, we will explore methods for compact multiresolution encoding dense triangulated meshes, even more compact encodings for simpler objects that support an isometric quad parametrization, and pure image-based pre-rendered representations for 3D environments that can be appreciated from a small number of point of views. In terms of camera control, we will explore methods that guide the user by appropriately removing degrees of freedom: from simple automatic centering of virtual trackball, to surface-following methods, to constrained visits on a graph of views. In terms of information presentation, we will explore both a graph-based representation of 3D object annotations, and how constrained camera controllers can be exploited to guide the users towards interesting places and automatically show the related information.

4.1 Research Goals

After a detailed analysis of state-of-the-art (see Chapter 3), we have identified a number of limitations on current methods that need to be addressed in order to satisfy the requirements presented in Sec. 2.2.

In our analysis of the problem domain, we aimed at covering most common use cases. Being many the possible applications of highly detailed 3D models in our current society, it implies a wide range of requirements that need to be satisfied. There is no single solution able to cope with all the requirements and current limitations. Thus, we plan to explore a set of approaches tackling with

subsets of current limitations. The combination of such approaches should be able to cover common use cases.

Our main global objective is ubiquitous exploration of complex 3D models on common 3D platforms. For that purpose, we have defined and classified our research goals as follows:

- **Improving Scalability of Visualization Methods for Complex 3D Models.**

In order to support ubiquitous visualization of highly detailed 3D models we need to deal with distribution and rendering issues. First, large amounts of data must be transmitted for remote rendering, with current bandwidth limitations being specially tight on mobile and wireless networks (**R3**). Second, the client side will receive and process this data for rendering, which will require being able to display in real-time big amounts of geometry (i.e., triangles) (**R1**). Thus, we need compact data representations that not only can be efficiently streamed through networks while minimizing bandwidth usage, but also are efficient for being processed in the client for rendering. In addition, adaptive rendering techniques must be used in order to provide real-time performance, reducing the amount of data transmitted and rendered by exploiting view-dependent constraints (see Sec. 3.2.1). Current 3D platforms include, nowadays, mobile and web platforms (**R9**), which impose further limitations on memory resources and computation power. For that purpose, we must consider these platforms characteristics when designing our approaches. In particular, real-time visualization of scenes with complex lighting on low-profile platforms is one very challenging issue (**R4**), since current solutions for real-time visualization of massive 3D models have only been demonstrated on low-profile platforms with light 3D models (see Sec. 3.2.1). For coping with those limitations, we plan to exploit scene characteristics to design specific methods and algorithms for efficient distribution and rendering of complex 3D models. By imposing constraints on the supported 3D models, we can develop compact efficient data structures tailored for network streaming and real-time rendering on common 3D platforms, including low-profile platforms (i.e., mobile/web platforms). In addition, camera movement constraints will be exploited for enabling image-based rendering techniques to overcome real-time constraints when dealing with scenes with complex lighting.

- **Improving Navigation Techniques for Complex 3D Models.** In complex 3D models, it is common for information to be found at multiple scales, such as global shape or very fine details on the surface of the object. In order to allow the user to explore all this information, navigation techniques must

support both global inspection and proximal surface navigation (R5). In order to avoid the user losing spacial context, guided navigation methods must be used to help the user during navigation. For that purpose, we plan to introduce constraints to the camera motion, slightly limiting movement freedom in the environment aiming to help the user focus on the 3D virtual object while ensuring “good views” of the scene (R6). Interaction with large displays (R2) imposes a number of difficulties mostly due to their size, thus rendering impractical co-located setups (i.e., touch screens). In order to support multi-user visualization (R7), while avoiding cluttering due to other users occluding the display (R8), we plan to design interaction techniques supporting non co-located setups (R9).

- **Improving Understanding of Complex 3D Models.** Providing effective 3D content presentation is particularly relevant when the goal is to allow people to appreciate, understand, and interact with intrinsically 3D virtual objects. In this context, we plan to design a navigation-oriented interface relying on guided navigation for information discovery (R6). Overlaid information should be displayed associated to spatial and semantic constraints (R10, R11, R12), while avoiding cluttering the visualization with obtrusive indications such as hot spots (R8). Moreover, we plan to use an underlying recommendation system for determining what information to be proposed to the user, and exploiting guided navigation techniques to guide the user towards the information, instead of relying on obtrusive indications, such as hot spots or thumbnails. The recommendation system should let the authors define the spatial and semantic constraints associated with the additional information in a flexible and simple way, without requiring complex procedures in order to support non-expert users (R12, R13).

Table 4.1 shows a classification of the work presented in this thesis based on constraints on the input 3D model, and constraints on the camera movement.

4.2 Our approach

In this section we will briefly introduce our approach towards ubiquitous exploration of highly detailed and annotated 3D models.

Table 4.1 provides a classification of the methods that will be presented in this thesis based on their generality, and camera movement freedom. The horizontal axis classification is based on generality with respect to 3D model support: the approaches in the leftmost column can be applied to general 3D meshes; the approaches in the central column are suited for annotated 3D models or 3D

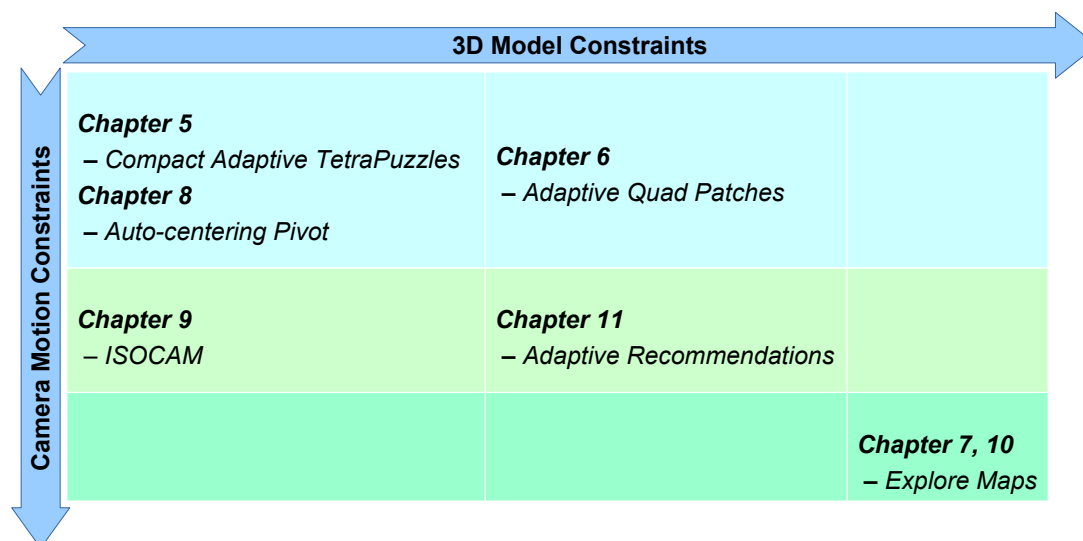


Table 4.1: Method classification. This chart classifies the methods presented in this thesis in function of constraints on the input 3D model, and constraints on camera motion.

models which can be parametrized as quad patches; finally, the ExploreMaps approach in the rightmost column requires models that can be described with a small number of points of view. On the other hand, the vertical axis classification depends on camera freedom: the top row contains approaches providing free camera movement; in the middle row there is the IsoCam approach which limits camera movement to the set of isosurfaces surrounding the 3D model, and our adaptive recommendation system that uses an attraction force to guide the user during the exploration. The ExploreMaps approach, in the bottom row, is again the most constrained technique since navigation is limited to a fixed set of points of view and a number of connecting paths.

The remaining chapters will present the approaches listed below:

Chapter 5: Compression-domain Seamless Multiresolution Visualization of Gigantic Meshes on Mobile Devices. Presents a method based on irregular triangulations which exploits the regularity of a hierarchical volumetric space partitioning to construct a compact and seamless multiresolution model (R1). We make use of the spatial partitioning for producing a local (i.e., to its containing node) parametrization of the geometry, which is clipped against its containing node bounds. This parametrization enables us to use local quantization techniques while minimizing the quantization error. The resulting compact representation can be directly decoded in the GPU with negligible performance impact, minimizing the required memory resources on the client. This approach is, thus, well suited for resource constrained platforms, such as mobile devices (R9). For network transmission,

we introduce further compression using a low-complexity coding approach, performing a prior data reordering in order to raise data coherency (R3).

Chapter 6: Adaptive Quad Patches: An Adaptive Regular Structure for Web Distribution and Adaptive Rendering of 3D Models. Introduces a compact representation for quad-parametrizable 3D models which produces a fully regular multiresolution (R1) image-based representation suitable for storage, distribution, and real-time rendering on modern commodity/web platforms. Thanks to the quad parametrization the final encoding of the 3D model results in a very compact and efficient collection of image tiles (R3). Those tiles can then be efficiently decoded in the GPU for rendering, minimizing the processing in the CPU. This approach is, thus, well suited for scripted environments such as Web browsers (R9).

Chapter 7: ExploreMaps: Efficient Construction of Panoramic View Graphs of Complex 3D Environments. Presents an image-based technique that enables interactive rendering of complex scenes on commodity platforms. To make that possible, we constrain camera movements to an optimized set of fixed panoramic view positions. This constraint allows us to overcome the real-time rendering limitations by precomputing all the possible views using off-line photo-realistic renderers (R4). The resulting 3D representation consists of a set of panoramic images (i.e., six images associated to the faces of a cube), and a set of panoramic videos for transitions, relying on standard image and video algorithms for network transmission (R3). Those features render this approach convenient for browsing scenes with complex lighting in low-profile platforms, such as mobile devices and Web browsers (R9).

Chapter 8: HuMoRS: Huge models Mobile Rendering System. Describes a method for exploring complex 3D models which provides an automatic centered pivot greatly simplifying the task of exploring multi-scale information. Aiming to help the user exploring multi-scale information (R5), we compute a natural interaction pivot from a stochastic sampling of the visible geometry. Thanks to this automatic pivot, the user can concentrate in the navigation (i.e. panning, rotating, and zooming) (R6) without having to rely on precise picking at any moment to define the pivot for rotating. Not relying on precise picking, makes this technique suitable for touch-based interaction on small screens (R9), where finger occlusion and reduced dimensions heavily penalize precision.

Chapter 9: IsoCam: Interactive Visual Exploration of Massive Cultural Heritage Models on Large Projection Setups. Introduces a new method that relies on a distance-field representation of the 3D model to provide

collision-free and seamless multi-scale navigation while ensuring *good* view directions. During multi-scale navigation inexperienced users easily lose the spatial context (i.e., when navigating very close to the surface). We tackle this problem by constraining the camera movement to follow the continuous set of isosurfaces defined by a distance-field representation of the 3D model. This constraint allows the user to explore the whole 3D model while always ensuring “good camera orientations” (R6) and provides smooth transition between global inspection (i.e., orbiting) and proximal navigation (i.e., surface hovering) (R5). Furthermore, we describe a system pairing a non co-located user interface with a large projection screen, for supporting large scale visualization (R2) and multi-user visualization (R7). Finally, we study how our constrained camera motion controller can be exploited to enhance 3D stereoscopic rendering on light field displays (R9). This kind of display provides multiple user stereoscopic 3D visualization (R7) without requiring 3D glasses, thus providing a much more immersive experience (R8).

Chapter 10: ExploreMaps: Ubiquitous Exploration of Panoramic View Graphs of Complex 3D Environments. Introduces our approach for browsing the Explore Maps generated with the method presented in Chapter 7. A graph-based representation of the scene is used for presenting the user with a selection of panoramic views that gives a full coverage of the scene. Both thumbnail-based navigation and closest-path selection can be used for traveling between graph-nodes (R6). The sense of immersion is enhanced by employing panoramic transition videos, thus providing free camera orientation during transitions. With this approach we can interactively explore scenes with complex lighting on commodity platforms (i.e., including mobile/web platforms) (R9).

Chapter 11: Adaptive Recommendations for Enhanced non-linear Exploration of Annotated 3D Objects. Introduces a novel approach for exploring complex 3D annotated models, integrating an stochastic adaptive recommendation system with a walk-up-and-use user interface that provides guidance towards interesting view positions while being minimally intrusive. In order to integrate and enhance the visual information (R10), we study unobtrusive methods for presenting the user with contextual information, while not cluttering the interface with indications or hot spots (R8). We rely on guided navigation techniques, introduced in previous chapters, for guiding the user towards the information by slightly applying an attraction force (R6). An stochastic recommendation system is in charge of determining the

information to be displayed depending on current camera parameters and a number of semantic and spatial constraints (**R12**), which are defined by an expert (**R13**).

Part II

Compact Representations for Complex 3D Models

Systems for rendering complex 3D models have to employ methods for filtering out as efficiently as possible the data that is not contributing to a particular image, and to adaptively, i.e., incrementally, load and render parts which are needed, efficiently using bandwidth and local resources by combining compression and data management methods.

In this part, we will present some approaches that exploit scene characteristics to design specific methods and algorithms for compression and adaptive rendering of complex 3D models.

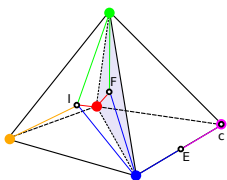
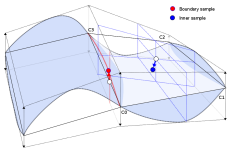
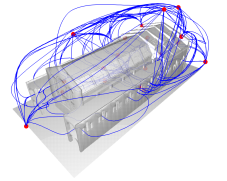
Model Type	Approach	Method	Published in	Features
Dense 3D meshes	<ul style="list-style-type: none"> - Tetrahedra hierarchy - Clipped geometry - Barycentric parametrization - Local quantization See Chapter 5		Web3D'13	<ul style="list-style-type: none"> - Compact GPU-friendly representation - 64bit per vertex encoding for position, normal and color - Proved efficient on mobile platforms
Parametrizable meshes	<ul style="list-style-type: none"> - Quad-parametrization - Image-based encoding See Chapter 6		Web3D'12	<ul style="list-style-type: none"> - Image-based encoding enables usage of standard image components for decoding on CPU - Most of the work moved onto the GPU - Compact encoding using less than 25bit per vertex for position, normal and color - Tailored for scripted environments, i.e., web platforms
Scenes with complex lighting	<ul style="list-style-type: none"> - Graph-based representation - Pre-computed visualization See Chapter 7		EG'14	<ul style="list-style-type: none"> - Interactive exploration of complex environments on low-profile platforms, i.e., mobile and web platforms - Enables support for complex lighting simulation in pre-processing

Table 4.2: Scalable rendering approaches.

CHAPTER

5

..... Compression-domain Seamless Multiresolu- tion Visualization of Gi- gantic Meshes on Mobile Devices

The need for interactively inspecting very large surface meshes, consisting of hundreds of millions of polygons, arises naturally in many application domains, including 3D scanning, geometric modeling, and numerical simulation. A number of solutions have been presented in previous years to compactly represent objects in a multiresolution way. In this chapter, we explore how to efficiently combine multiresolution with compression in order to cope with strong bandwidth and hardware capabilities limitations by presenting a compression-domain adaptive multiresolution rendering approach capable to scale from desktop GPU rendering to mobile graphics. The basic idea behind the proposed approach is to use, as in previous work, a regular conformal hierarchy of tetrahedra to spatially partition the input 3D model and to arrange mesh fragments at different resolution. In this approach, we create compact GPU-friendly representations of these fragments by constructing cache-coherent strips that index locally quantized vertex data, exploiting the bounding tetrahedron for creating a local barycentric parametrization of the geometry. For the first time, this approach supports local quantization in a fully adaptive seamless 3D mesh structure. For web distribution, further compression is obtained by exploiting local data coherence for entropy coding.

This compact representation minimizes resource requirements, thus being suitable for very constrained platforms (i.e., mobile devices).

5.1 Introduction

NETWORKED massive model renderers have to employ methods for filtering out as efficiently as possible the data that is not contributing to a

particular image and to adaptively load and render them, efficiently using bandwidth and local resources by combining compression and data management methods. A compressed continuous level-of-detail (LOD) model, i.e., a compact description of multiple representations of a single shape supporting the extraction of representations with varying accuracies in different regions, is the key element for providing the necessary degrees of freedom to achieve run-time adaptivity. The basic ingredients of a such a model are a *base mesh*, that defines the coarsest approximation to the 3D model surface, a set of compactly represented *updates*, that, when incrementally loaded and applied to the base mesh, provide variable resolution mesh-based representations, and a *dependency relation* among updates, which allows combining them to extract consistent intermediate representations. Different specialized multiresolution models, of various efficiency and generality, are obtained by mixing and matching different instances of all these ingredients.

In the most general (and common) case, the multiresolution model is based on a fully irregular approach in which the base mesh is an irregular triangulation with unrestricted connectivity, and updates are encoded as changes to regions of this triangulations. Because of their flexibility, fully irregular approaches are theoretically capable of producing the minimum complexity representation for a given error measure. However, this flexibility comes at a price. In particular, mesh connectivity, hierarchy, and dependencies must explicitly be encoded, and simplification and coarsening operations must handle arbitrary neighborhoods. By imposing constraints on mesh connectivity and update operations it is possible to devise classes of more restricted models that are less costly to store, transmit, render, and simpler to modify. This is because much of the information required for all these tasks becomes implicit, and often, because stricter bounds on the region of influence of each local modification can be defined.

The Compact Adaptive TetraPuzzle (CATP) method presented here builds on Adaptive TetraPuzzles (ATP) [Cign 04] by using a regular conformal hierarchy of tetrahedra to spatially partition the input 3D model and to arrange mesh fragments at different resolution in an implicit diamond graph.

In this work, however, tetrahedra not only partition but also clip the original triangulation. We can thus create compact GPU-friendly representations of each fragment by constructing cache-coherent strips that index compact interleaved quantized vertex data, exploiting the bounding tetrahedron for creating a local barycentric parametrization of the geometry. Appropriate boundary constraints are introduced in the splitting, simplification, and quantization steps to ensure that all conforming selective subdivisions of the hierarchy of tetrahedra lead to correctly matching surface fragments. Such an approach introduces for the first

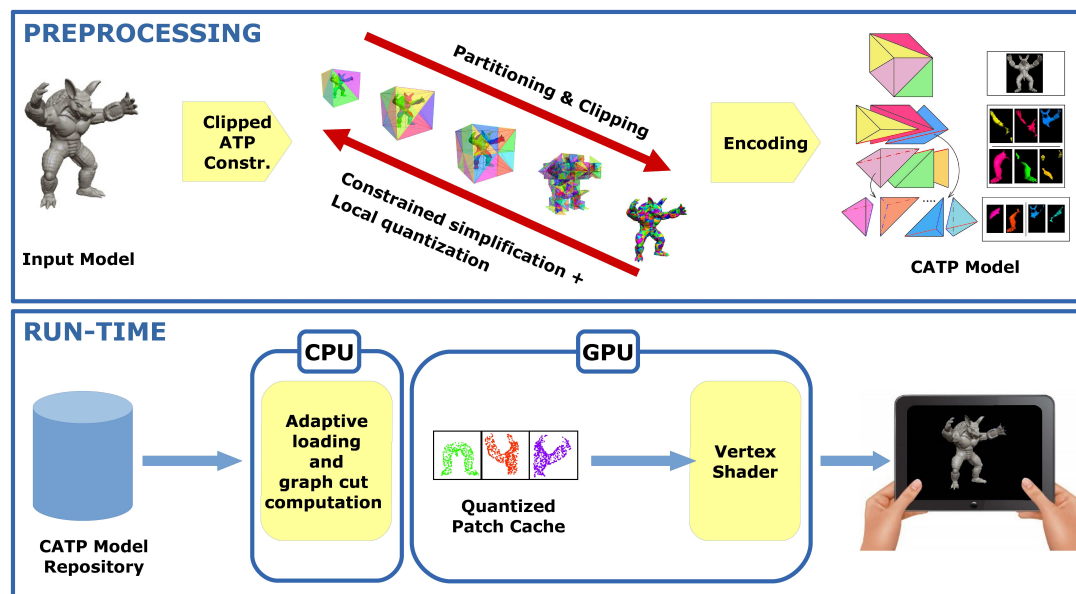


Figure 5.1: CATP overview. A regular conformal hierarchy of tetrahedra spatially partitions the input 3D model and arranges mesh fragments at different resolution in an implicit diamond graph. Tetrahedra not only partition but also clip the original triangulation, ensuring that each mesh fragment is fully contained within its bounding tetrahedron. Local quantization with appropriate boundary constraints can thus be used to compress the model. At run-time, clients maintain an adaptive local graph cut, and rendering is performed directed on compressed data, which is dequantized in the vertex shader.

time local quantization in a general adaptive 3D mesh structure.

For web distribution, further compression is obtained on top of the compact GPU-friendly representation by exploiting local data coherence using a low-complexity coding approach based on a wavelet transformation followed by entropy coding of coefficients.

At run-time, mobile viewer applications adaptively refine a local multiresolution model by managing a local GPU cache and asynchronously loading on-demand from a web server the required fragments. CPU and GPU cooperate for decompression, and a shaded rendering of colored meshes is performed at interactive speed directly from an intermediate compact representation that uses only 8bytes/vertex, therefore coping with both memory and bandwidth limitations. Keeping data compact is of particular importance in resource constrained platforms, e.g., current mobile devices, with extremely large screen resolutions, which dictate large rendering working sets, but limited main memory sizes. For instance, the current iPad generation sports a 3Mpixel display, but has a RAM capacity of only 1GB. Moreover, by decoding compressed data on-the-fly on graphics hardware, we can not only reduce local memory consumption and minimize GPU bandwidth usage, but also power consumption, thanks to the reduced memory access and data transmission through the system bus.

As highlighted in the overview of related work (Sec. 3.2.1), while certain other approaches share some of our method's properties, they typically do not meet the capability to rapidly generate adaptive seamless meshes by rendering from a very compact representation.

The efficiency of the approach has been successfully evaluated with a number of large models, including a massive 1G triangle colored model of Michelangelo's David (Sec. 5.6).

5.2 Method Overview

Starting from a high-resolution triangle mesh, we build, using a parallel out-of-core process, a hierarchical multiresolution structure based upon a conformal tetrahedra partitioning of the model's bounding box, similarly to what is proposed by the *ATP* approach [Cign 04].

The leaves of the multiresolution structure contain the full resolution original model while inner nodes contain simplified representations of the geometry with approximately half of the number of triangles contained into the children. The building process is performed off-line by iteratively inserting triangles from the input mesh into the hierarchical structure, which is recursively refined in order to maintain a maximum triangle count in the leaves. Then, coarser representations are built bottom-up by recursively merging children nodes, with a maximum triangle count and a representation error threshold as constraints.

A two-stage compression schema is used to transform input data to a compact representation suitable for GPU rendering and then further compressing this structure to reduce network traffic. The tetrahedral structure is exploited to encode vertex positions with barycentric coordinates. Tetrahedral barycentric coordinates express the position of a vertex inside a tetrahedron, as the combination of its four corners. These coordinates can then be quantized locally for each tetrahedra, thus minimizing the quantization error. To produce a conforming mesh, input triangles, differently from *ATP*, need to be clipped against the tetrahedra faces, thus each tetrahedron geometry is fully self contained. The continuity among adjacent tetrahedra is ensured when quantizing positions, by the fact that barycentric coordinates of vertices lying on tetrahedra faces are expressed as a combination of only the three corners defining that face, which are the same among neighboring tetrahedra in a conformal hierarchy. Normal and colors are also encoded to produce a compact quantized representation which is further compressed for storage and streaming (see Sec. 5.3).

At rendering time, multiple clients can access the data through a server farm where the multiresolution models are stored (see Sec. 5.4). On the client, an

adaptive rendering approach incrementally updates the representation that best fits the current point of view and retrieves the required data from the server in an incremental fashion. Differently from ATP, where the multiresolution structure was encoded by six binary trees of disjoint tetrahedra, we base our run-time structure on *diamonds*. Each diamond is composed by the set of all the tetrahedra sharing their longest edge. Using a diamond based structure, see Weiss and De Floriani [Weis 10a], dependencies are implicitly encoded into the hierarchy, and refinement is interruptible, producing a conforming mesh also when children data is not available. This feature perfectly fits in remote visualization applications, where is common to experience data fetch delays due to bandwidth limitations. The working set is kept to a fixed small size by keeping data directly in a compact GPU format suitable for direct rendering through specific *shaders* that do the decoding directly in the *Vertex Shader* (see Sec. 5.5).

5.3 Building the multiresolution structure

The construction process (see Fig. 5.1) of the multiresolution structure starts from a *triangle soup*, i.e., a flat list of triangles with direct vertex information, together with a list of boundary vertices. The process is composed of two main phases: a first one where the data-set is partitioned into a tetrahedra hierarchy, and a second phase where data is simplified in a bottom-up fashion to build inner node representations.

5.3.1 Tetrahedra Partitioning

The first phase starts with the insertion of the input triangles into the root diamond, which is generated by partitioning the bounding box of the mesh into six tetrahedra sharing a major box diagonal as their longest edge. Thereupon, in a top-down manner, input triangles are inserted into the tetrahedra hierarchy, which is recursively refined in order to maintain a maximum triangle count per leaf node. In contrast with ATP [Cign 04], when a new triangle is to be inserted, the triangle is clipped against each of the leaf nodes it overlaps. The generated triangles are then inserted into the corresponding node. In such a way, each tetrahedron fully contains its geometry. Whenever the number of triangles contained in a node exceeds a given limit, the tetrahedron σ is split by the plane passing through the midpoint of its longest edge and the opposite edge in σ . Then, the triangles contained in the tetrahedron are redistributed among the two children tetrahedra. Towards guaranteeing the conformality of the resulting tetrahedra

mesh after this splitting, all the tetrahedra belonging to the same diamond of σ , are split at the same time.

After all the triangles have been inserted into the hierarchy, leaves contain the original geometry. Nonetheless, due to quantization errors, this representation slightly deviates from the input mesh. In order to be able to reproduce the original geometry, leaf nodes are further refined, recursively splitting triangles and inserting them into children tetrahedra until the quantization error is below a user defined threshold, generally a fraction of the average edge length. After this first phase, the original mesh is represented by the leaf nodes, while inner nodes are empty.

Triangle clipping requires special care to avoid producing disjoint vertices due to precision error on shared geometry (i.e., when splitting an edge shared by two neighbor tetrahedra). Before performing the clipping, the edge vertices are sorted according to a simple less operator which take into account also the plane orientation, thus obtaining a repeatable procedure.

This pre-processing builds a diamond hierarchy where each diamond consists of a set of tetrahedra. The diamond hierarchy contains three types of diamonds formed by 6, 4 and 8 tetrahedra, and contain respectively 8, 6 and 10 vertices [Weis 10a], see Fig. 5.2. These three diamond configurations occur repeatedly in sequence during hierarchy traversals. Diamonds and tetrahedra corners are expressed on an integer grid, where diamonds are uniquely identified by their center, and tetrahedra are identified by their level and center integer coordinates. It is possible through a look-up table to go from a diamond to its children, parents, and constituent tetrahedra. Thus, we can avoid to store pointers in the pre-processing and also in the run-time structure.

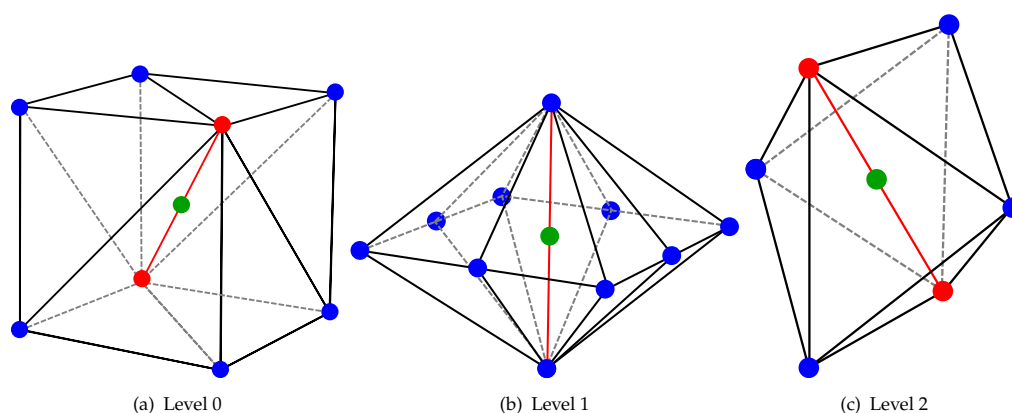


Figure 5.2: Sequence of diamond configurations. There are three different diamond configurations that occur iteratively during the hierarchy refinement.

5.3.2 Simplification

In a second phase, the hierarchy is traversed bottom-up in order to generate inner nodes simplified geometry representation. For this purpose, for each inner node, the geometry contained in its four child nodes is simplified by fulfilling the maximum triangle count per node and a given representation error threshold. Each diamond is simplified independently from the other diamonds, with the constraint that vertices lying on boundary faces are left unchanged to maintain continuity among neighboring diamonds of the same level and adjacent levels of resolution.

There are three different cases to take into account: (a) inner boundary vertices, i.e., vertices near the inner faces of the diamond children, connecting with triangles from other tetrahedra inside the diamond; (b) outer boundary indices, i.e., vertices near the faces of the diamond, connecting with triangles of other diamonds or tetrahedra; (c) inner vertices: all other vertices. Vertices of type (a) can only be simplified against other (a) vertices in order to maintain the same representation in all neighboring tetrahedra sharing these vertices, so the re-parametrization of this vertices to barycentric coordinates remains the same. Vertices of type (b) are fixed and cannot be simplified to ensure continuity with neighbor diamonds. Inner vertices of type (c) can be simplified in any manner. In brief, since triangles are clipped to their containing tetrahedra, in contrast with ATP, during the simplification the only constraint is not to modify: external edges on the faces of the diamond that are shared with other diamonds, and internal edges on the faces of tetrahedra that are shared among two tetrahedra in the diamond. The latter is required to ensure that edge vertices have the same coordinates in both tetrahedra when re-parametrized into barycentric coordinates, see Fig. 5.3(a). Because diamond topology changes at each level, there are no vertices that remain locked up to the root and all the data can be simplified to get a uniformly sampled model.

5.3.3 Barycentric parametrization and quantization

Before emitting tetrahedron data its geometry is reparametrized into barycentric coordinates. Each inner point is expressed as a linear combination of the 4 tetrahedron corners, while points lying on tetrahedron faces are expressed as combination of only the three corners defining the face. The latter, ensures continuity between neighboring tetrahedra since the points on shared faces will be defined as combination of the same three vertices, see Fig. 5.3(b). We chose this representation because it offers a compact representation that can be locally quantized without producing cracks between adjacent tetrahedra. However, due

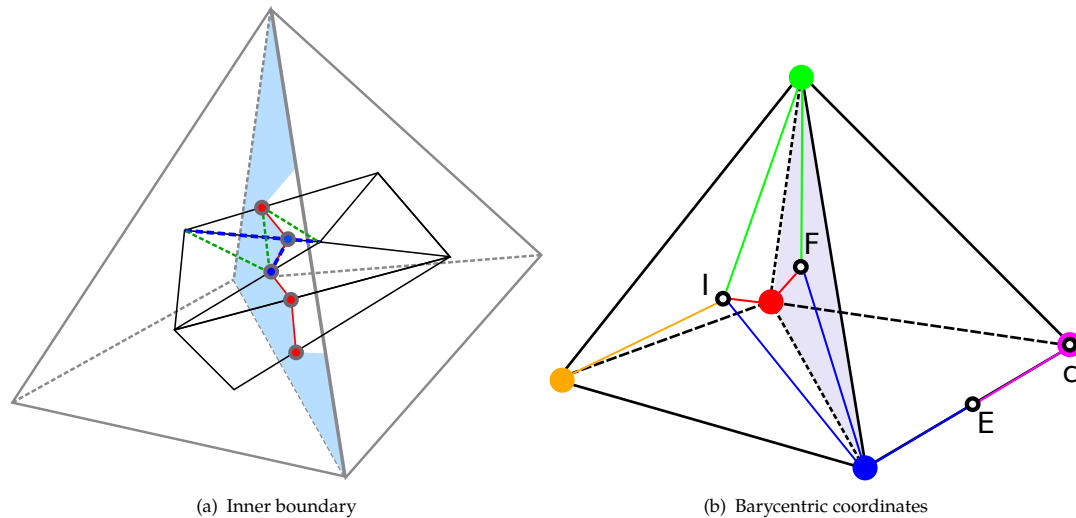


Figure 5.3: Tetrahedra merging.

a) Geometry simplification. An edge shared between two neighboring tetrahedra should be simplified only over the plane defined by the face connecting the tetrahedra. This way, the barycentric coordinates of the vertices will depend only on the coordinates of the vertices defining face and so will be common for both tetrahedra. The blue edge disappears when blue vertices are merged. Green edges appear during the re-triangulation of the clipped geometry. **b) Barycentric coordinates.** Four examples of vertices with their positions with respect to the tetrahedron: I inner, F on face, E on Edge, C on corner. For each vertex, we show the corners which provide non null barycentric coordinate contribution.

to precision error, boundary points could be placed not exactly on the faces, thus producing different values on different tetrahedra, once data is quantized, see Fig. 5.4.

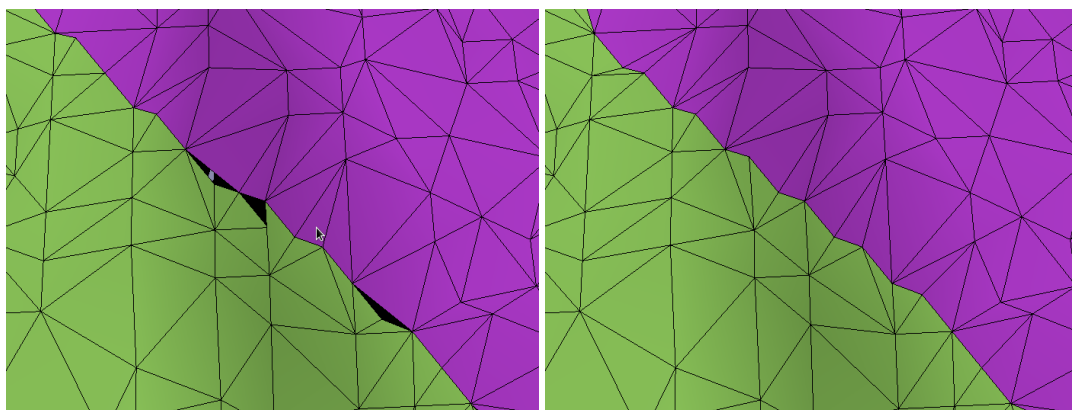


Figure 5.4: Geometry quantization. Projecting boundary vertices on tetrahedron faces permit to solve mesh discontinuity among adjacent tetrahedra, which are due to quantization of slightly different values.

Hence, to ensure consistent quantizations, we subdivide the points in 4 cases depending on their position with respect to the containing tetrahedra: near a corner, near an edge, near one face, and inner point. In the first case the point is represented with only 1 not null coefficient, 2 coefficients for the second case

expressing linear interpolation among 2 points, 3 in the third case for barycentric coordinates over a 3D triangle, and finally 4 for inner points. These cases are checked in this order, and with decreasing epsilon, to be sure to behave in the same manner on adjacent tetrahedra for the same points, see Fig. 5.5.

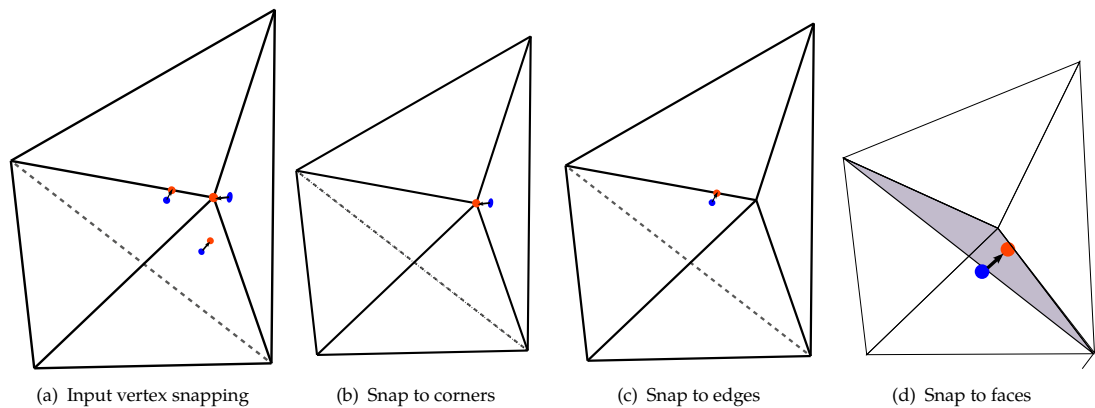


Figure 5.5: Vertex snapping. (a) Shows sample vertices that must be snapped onto (b) corners; (c) edges or (d) faces.

The quantized geometry is stripified in a cache coherent way to maximize the number of cache hits of transformed vertices. Thereafter, vertices are reordered to match the order of appearance in the strip. This sorting permits to reflect the spatial position in the memory layout, thus keeping similar values (i.e., neighboring vertices) near each other, which will be useful for the compression step.

5.3.4 Compression

The quantized vertex coordinates are encoded together with normals and colors into a compact 64bit representation suitable for direct rendering, where 3 bytes are used for position, 2 bytes for normal and 3 bytes for color. Position is parametrized with 4 barycentric coordinates, but only 3 components are required since the four components sum to one. This 64bit/vertex encoding provides an extremely compact aligned representation that can be efficiently accessed on current GPUs, which typically require vertex data to be aligned on 32 bit boundaries. For a colored representation, 64bit/vertex is thus an optimal size. Normals are encoded using the *octahedron normal vector* approach [Meye 10], which maps unit vectors to two parametric coordinates. This encoding consists in projecting the normals onto the octahedron by normalizing them with the 1-norm. The octahedron is unwrapped to a square and the $[u, v]$ parameters in the plane are quantized to 8 bits, leading to sub-degree precision [Meye 10]. Decompression is numerically

stable and requires only few basic operations which can be executed in the vertex shader.

This compact representation is well suited for rendering through a simple shader (see Sec.5.5), but higher compression rates can be achieved using a low-complexity codec applied to both vertex data and topology. In order to maximize data correlation for the entropy coding, color is transformed to the YCoCg reversible format [Malv 08], and all of the vertex attributes (i.e., position, normal and color) are deinterleaved and separated into their components and stored as a sequence of streams, to which strip indices are also appended (i.e., $[P_0^0 \dots P_n^0], [P_0^1 \dots P_n^1], [P_0^2 \dots P_n^2], [N_0^u \dots N_n^u], [N_0^v \dots N_n^v], [Y_0 \dots Y_n], [Co_0 \dots Co_n], [Cg_0 \dots Cg_n], [Io \dots Im]$). Each stream is then transformed using a reversible n-bit to n-bit wavelet based on the Haar wavelet transform in order to reduce entropy [Sene 04]. This approach uniformly treats topology and vertex data, and generalizes the usual linear prediction methods typically applied to vertex positions. The low-pass coefficients produced by the wavelet transformation are iteratively filtered by the same wavelet until we remain with a single (root) approximation coefficient. The resulting approximation and detail coefficients are then transformed with a range codec: integers are arithmetically coded using a single symbol for the value 0, while other values are encoded using exponent, mantissa and sign, with different context for each encoded bit. This is a variation of the symbol encoding method used in the FFV1 Video Codec [Mart 79]. During decompression all the steps are undone in reverse order.

In the course of the building process we also maintain a temporary version of the simplified data stored uncompressed on an external memory data repository, which is used to build coarser level simplifications without accumulating quantization errors.

5.3.5 Parallel processing

Simplification and encoding can be easily parallelized, being each diamond independent from the others. After the recursive subdivision, the simplification starts from the leaves and goes up to the root. A master process takes care of assembling each diamond of the current level, fetching corresponding tetrahedra geometry from the uncompressed data repository, and assigning it to a worker process. On the worker process, inner node diamonds get their geometry simplified, encoded and compressed. For leaf nodes, only encoding and compression are performed. After this job the worker sends the compressed diamond geometry back to the master node. After all diamonds of the current level are processed, the master starts to process the next coarser level and, level by level, the data-set is simplified

up to the root. At that point, since children nodes are no more needed, their uncompressed representations can be discarded.

5.4 Server

On the server side, data for each model is stored in separated databases. In the pursuit of scalability, the server acts as a repository of data with zero processing overhead. An abstraction layer handles communication processes through different transport protocols, such as HTTP or direct connection through TCP. A simple module for Apache2 is in charge of handling HTTP requests, which relies upon a local database to efficiently locate the requested data. Berkeley DB is used for storage, accessing and caching data in the server side due to its open source license and its matureness as embeddable database. Berkeley DB provides an efficient and scalable transactional database engine with high reliability and availability, able to handle up to terabytes of data. It also allows configuration of per-process replicated cache and shared index memory among different database instances. Altogether, provides an scalable architecture with reduced memory load for servers when dealing with hundreds of clients in parallel. On the other side, Apache2 is a mature and open source server which provides an efficient, secure and extensible architecture for developing HTTP services. Its scalable and multi-threaded architecture together with features like persistent server processes and load balancing are essential to the performance of our application. A custom Apache module implements a connection-less protocol based on HTTP which receives queries composed of database name and node identifier. This module extracts the query parameters, retrieves the corresponding data from the DB, and sends back either node's data or an empty message if it is not present. This architecture relies on mature components that have been widely tested and provide good scalability and performance when dealing with thousands of clients.

5.5 Client architecture description

Although the method is directly applicable for desktop platforms, we have focused our tests on embedded devices such as Apple iPhone/iPad or Android devices in general, since they provide a very promising platform for remote exploration of high quality 3D content. On those platforms there is support for OpenGL ES, the specification for embedded devices, which is slowly converging with its desktop counterpart. Currently, the most commonly used version on those platforms is OpenGL ES 2.0, although version 3.0 is already present on

most recent hardware. OpenGL specifications for Embedded Devices have been defined to be a fully functional subset of its desktop counterparts where only the more general functionality has been included in order to minimize circuitry complexity and energy consumption.

Older devices support only the ES 1.1 version, which offers a lighter version of OpenGL 1.5, where immediate mode has been suppressed together with complex primitives such as quads or polygons. The functionality include Vertex Buffer Objects (VBO) and Vertex Arrays to feed the GPU with geometric primitives. Most current devices do support the ES 2.0 version, based upon the 2.0 desktop specification, where the whole fixed pipeline functionality has been removed in favor of the shader based pipeline, where *Vertex* and *Fragment* shaders must be provided giving more flexibility. The GLSL specification for ES has also been modified adding control for data precision. On most recent devices there is already support for ES 3.0, where occlusion queries, transform feedback and multiple render targets have been included in the specification. There is also finally a standard compression format ETC2/EAC, and support for 32bit integer, and 3D textures, among a lot more of newly supported functionality. This concrete version shows a clear converging tendency towards desktop OpenGL 4.x.

Embedded GPUs typically focus on high efficiency and low power consumption, although nowadays they are able to offer decent computational power in comparison with desktop GPUs. The Adreno GPU integrated in Qualcomm processors, among other GPUs used in current mobile devices, use Tile Based Rendering (TBR). Only once all the primitives have been submitted the driver splits the geometry into tiles which are then rendered using a small amount of in-core memory. The PowerVR SGX5XX, used in the various iPhone/iPad series and some high-class Android devices, go a bit further and use Tile Based Deferred Rendering (TBDR), which delays fragment operations until occlusion tests have been processed avoiding expensive calculations for occluded fragments. This architecture, widely used in embedded GPUs, penalizes reading back from the frame buffer since it requires waiting for all the tiles to be written prior to reading. In general, current generation of embedded GPUs provide really good performance together with an efficient energy consumption; although the continuously increasing display resolution makes the fragment load to penalize heavily the rendering performance (i.e, iPad 3 resolution of 2,048 by 1,536 uses a PowerVR SGX543MP4 with 16 unified shader units to render this massive amount of fragments).

Taking into account these architecture constraints, the rendering engine has been designed to minimize fragment processing while feeding the GPU with large geometry batches using cache optimized indexed triangle strips.

5.5.1 Adaptive view-dependent representation

Each frame, depending on the viewing parameters and a given fixed screen space tolerance, the client performs an adaptive rendering of the multiresolution model. For this purpose, the client relies on a hierarchical multiresolution representation of the model that is incrementally refined depending on the navigation. Initially starting with a coarse representation of the whole model, the hierarchy is traversed for each render view point in order to determine the available working set. The traversal algorithm takes into account diverse parameters: the viewing position, the available GPU resources, the current CPU usage level, and the required network bandwidth. Differently from ATP [Cign 04] the refinement is performed on a diamond basis. The viewer maintains the multiresolution structure as a map of diamonds, each of them identified by its center integer coordinates. For each diamond, on creation, there are available through a small look-up table its parents, children and tetrahedra indices. Each of the tetrahedra indices corresponds to an entry in the cache containing the compact representation of the fragment geometry. To each diamond we associate a view dependent priority which is the projected average edge length if the diamond is visible, or zero otherwise. A diamond is refined if its priority is higher than a user selected pixel tolerance. Refinement of a diamond stops if it is not visible, if it should be refined but children data is still not available, or if it fulfills the viewing constraints. Diamond based refinement is capable of producing a conformal tetrahedral mesh when each diamond is split only if its parent diamonds are already present in the graph. Such a refinement has the valuable benefit of being interruptible, hence we can use memory, triangle and time budgets to limit the used resources and to avoid locks, thus permitting interactive performance. We update the multiresolution structure cut using two diamond heaps: the refinement one, which is sorted with decreasing priority, and contains the leaves of the cut, and the coarsening heap which contains the parents of the leaves, with increasing priority. At each frame, instead of traversing all the hierarchy from the root, we update the priority of each diamond on the two heaps, then we refine the top of the refinement heap until achieving the desired error threshold, or one of the budget constraints is reached. Once over a new frame we also coarsen the top of the coarsening heap to release resources. The two heaps are properly updated for each refinement and coarsening operation.

In RAM memory, we maintain the cache of tetrahedra compact geometries, which are indexed through the diamond graph. The cache implements a LRU policy that maximizes the reuse of nodes while enforcing a resource usage below a given limit. The compact format permits to directly map data as Vertex Buffer

Objects, which are ready to be sent to the GPU. This compact representation also permits to perform ray-casting without needing a decompression step. Ray-casting is used to identify the touch point over the model for interaction purposes. Each tetrahedron also contains a small hierarchical tree of bounding boxes, computed just after loading, which is used to improve ray-casting performance. LRU fragments are kept in the cache as long as they are referenced by the diamond graph. After a coarsening operation, when a fragment is no more referenced, it goes toward the end of the LRU and is discarded as soon as new resources are needed.

5.5.2 Multi-threaded data access layer

The retrieval of data is performed through an asynchronous data access layer which encapsulates the data fetching mechanism and avoids blocking the application when the requested data is not yet available. The main thread, in charge of performing the hierarchy traversal for determining the working set, asks the cache for the nodes required for the current view position. If the requested data is available, the node is returned and so the traversal continues until the best available representation is reached; otherwise, a new request for this node is enqueued and the traversal stops since this is the best available representation. Another thread is responsible of fetching the requested data, contained in a priority queue. Depending on an available bandwidth estimation, a given number of requests is sent to the server, while the remaining requests are ignored. Since request priority corresponds to the node's projected error, coarser nodes are always requested first. On each frame, the request queue is cleared and filled again with the nodes needed for that frame, and so will be served at some point only after coarser nodes are available. This thread also handles incoming data and performs the decompression from the entropy coded version to the compact GPU representation, proceeding with the reverse sequence described in the pre-processing phase. Entropy decoding, then per component backward wavelet transform, and finally conversion from YCoCg to RGB. After this decompression, data is stored in an interleaved array of 8 bytes per vertex with 3 bytes for barycentric coordinates, 2 bytes for the octahedron normals, and 3 bytes for the RGB color.

5.5.3 Rendering process

Before rendering a simple shader is activated. The visible tetrahedra of the current cut are traversed by a visitor which takes care of managing a cache on GPU of Vertex Buffer Objects (VBO). The size of the GPU cache is smaller than



Figure 5.6: Detail of David's eye interactively rendered on a iPad. This 1Gtriangles model is colored using post-restoration color data. Note how our compression preserves extremely high quality details in shape, normal, and colors.

the size of the CPU one, thus more memory remains for CPU data, limiting the need of requesting and decoding multiple times data that exited from the limited GPU resources. When a node is visited, if it is not present in the cache, a corresponding VBO is created and inserted into GPU cache and rendered, otherwise only rendering is performed. Rendering consists in binding the buffer, setting up the vertex attribute pointers and drawing the optimized stitched strip sequence present in the geometry indices. For alignment purposes, we address vertex attributes as two 4-bytes words, and let the shader separate the position, normal, and color components.

The shader must transform data expressed in local barycentric coordinates. The transformation is given by this simple equation $\mathbf{v} = \|\mathbf{c}_0\mathbf{c}_1\mathbf{c}_2\mathbf{c}_3\| \cdot |\mathbf{v}_b|$, where \mathbf{c}_i represent the corner i th while \mathbf{v}_b is the vector of the 4 barycentric coordinates. Thus the 4 corners can be replaced by a matrix, which is post-multiplied to the model view matrix. Therefore, rendering from barycentric coordinates is not causing extra per-vertex cost with respect to using Cartesian coordinates. Since color is already in the RGB24 format, the only extra work that needs to be performed is the decoding of normals from the two quantized octahedral map coordinates. From the quantized coordinates remapped into $[-1, 1]$ we

compute $n_z = 1.0 - |u| - |v|$. Then if $n_z > 0$ we are on the upper side of the octahedron and $n_{xy} = uv$, otherwise we are on the lower part and we need to revert the n_{xy} components according to these equations: $n_x = (1 - n_y) \cdot \text{sign}(u)$ and $n_y = (1 - n_x) \cdot \text{sign}(v)$, see [Meye 10] for further details. Attribute decoding cost is thus negligible with respect to the other work performed by the shader (in particular, transformation, projection, and shading). Fig. 5.6 illustrates the quality of rendering that can be achieved using compressed data.

5.5.4 Graphical User Interface

On the iOS platform, we have taken advantage of the Cocoa Touch UI framework to design a simple Graphical User Interface (GUI) composed of a Model List Widget and OpenGL Rendering Layer. End users can easily browse and select the desired model through the Model List Widget and interact with the OpenGL Rendering Layer through standard multi-touch gestures. It is possible to rotate the model about its bounding box by moving a single finger on the screen, move the model with two fingers or zoom it in and out by performing a pinch gesture.

Interaction is also possible through an alternative “target-based” approach, with which a single quick tap by the user selects a target point which is attached to the model. This target point, rendered on screen as a small colored sphere, allows the user to easily rotate the model about by moving a single finger on the screen. By tapping the target again instead, it will trigger a smooth animation that moves the camera from its current position toward the target’s position. The target sphere can be deactivated anytime by tapping outside of the model.

5.6 Implementation and Results

We have implemented a prototype hardware and software system based on the design previously discussed in this chapter.

Using this method, we developed a framework composed of a C++ pre-processor, a client iOS app, and a HTTP server. Several tests were performed on pre-processing and rendering of very large models. Here we present results relative to two 3D large models from the Digital Michelangelo Repository of Stanford University with 0.25mm resolution: the David statue with 940 M triangles, and the St. Matthew statue with 374 M triangles. The David model is enriched with the color signal acquired after restoration and blended with geometry with the algorithm proposed by [Pint 11], while the St. Matthew has a precomputed ambient occlusion gray scale per vertex.

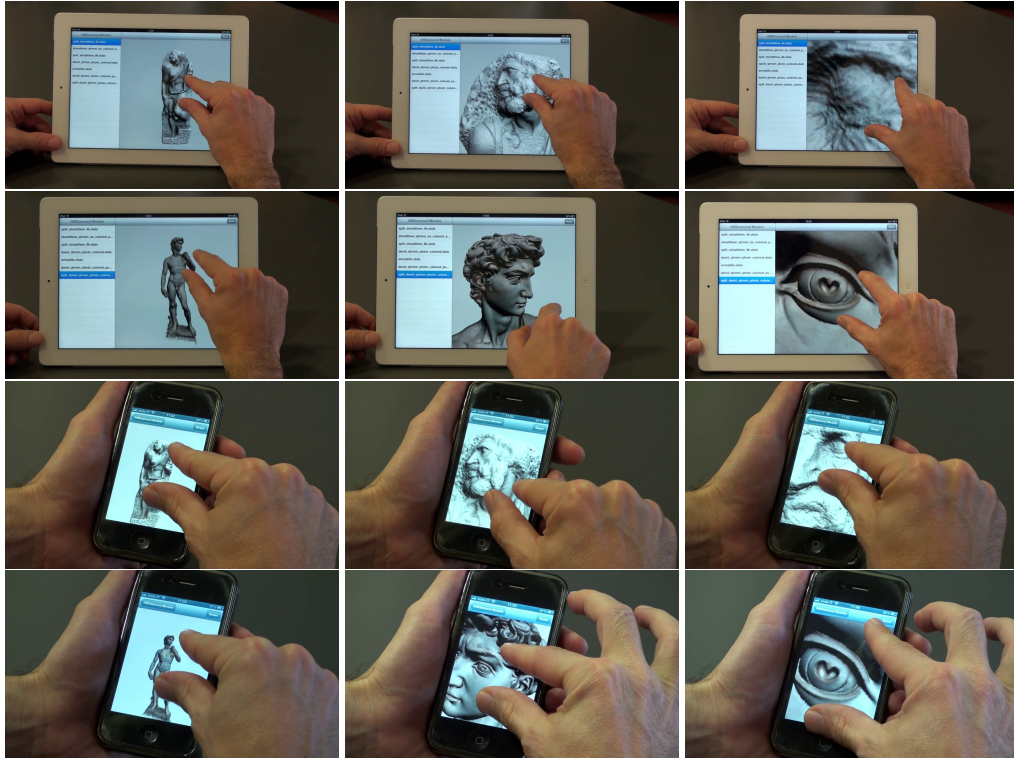


Figure 5.7: St. Matthew and David on a 3rd generation iPad and a iPhone 4. Representative frames from the accompanying video illustrating the interactive remote exploration of the colored David (1Gtri) and St. Matthew (374Mtri) datasets. The average frame rate is 37fps on the iPad and 10fps on the iPhone. Triangle throughputs vary from 30Mtri/s on iPad to 2.8Mtri/s on iPhone.

5.6.1 Dataset Construction Performance

The pre-processor has been implemented using C++ and the OpenMPI high performance message passing library. Each model has been processed using a single off-the-shelf PC with Linux 3.0.6 (Gentoo distribution) and an Intel(R) Core(TM) i7 CPU 960 @ 3.20GHz with 24GB RAM. We constructed all multiresolution structures with a target maximum leaf size of 8000 triangles/tetrahedron and a leaf quantization tolerance of 0.25mm.

Processing of the David statue took about 10h45m on 8 cores, while about 4h15m for the St Matthew, which correspond to roughly 24k triangles/second.

The data compression rate is 49.1 bits/vertex for the David model and 45.1 bits/vertex for the St. Matthew model, including the mesh topology information, see Table 5.1. The David's color information is heavier than St. Matthews, since the latter is only a low frequency ambient occlusion component. Instead, normals of the St. Matthew require more bits due to the roughness of the surface with respect to the David. Color encoding is loss-less, and normal error induced by octahedron encoding is sub-degree, and this represents the limit of this method when using 8 bits per component. Our wavelet transformation and entropy

coding step produces a compression of about 2.3x for vertex data and 3.5x for mesh topology with respect to our compact GPU friendly representation.

	David bpv	St Matthew bpv
Position	10.9	10.9
Color	9.1	3.8
Normal	8.6	9.7
Mesh Topology	20.5	20.7
Total	49.1	45.1

Table 5.1: CATP encoding bit rates. Bits per vertex subdivided per position, color, normal and mesh topology for the two processed models.

5.6.2 Rendering performance

The client was implemented on iOS 6 using C++, OpenGL and Objective-C++/C++. We evaluated the rendering performance of the technique on a number of inspection sequences on an iPhone 4 and on a 3rd generation iPad. The iPhone has a 1Ghz Apple A4 processor with 512 MB RAM, a PowerVR SGX535 GPU and a screen resolution of 640 x 960 pixels, while the iPad has a 1Ghz Dual-core Apple A5X processor with 1GB RAM, a PowerVR SGX543MP4 GPU and a screen resolution of 2048 x 1536 pixels. The two devices were chosen as representative extreme cases. The iPad has the currently largest screen, while the iPhone is an “old generation” phone with average specs. It must be considered that the current generation of mobile devices, such as the Apple iPhone 5, Samsung Galaxy S3 and Note 2, have technical specifications similar or even higher than the iPad 3’s, with a sensibly smaller screen resolution. We can thus expect much better results on these newer generation devices.

The quantitative results presented here in details were collected during interactive inspections with pixel tolerance 3 of the David and St. Matthew models. The sessions were designed to be representative of typical mesh inspection tasks and to heavily stress the system, and includes rotations and rapid changes from overall views to extreme close-ups. The qualitative performance of our adaptive renderer is also illustrated in an accompanying video, that shows live recordings of the analyzed sequences. Representative frames are shown in Fig. 5.7.

On the 3rd generation iPad we are able to render models with an average throughput of 30 Mtriangles/second, with an average rendering frame-rate of 37 fps, which eventually drops to 15 for full refined views, when the number of triangles reaches the 2 Mtriangles maximum triangle budget. As expected, the iPhone 4 got slightly worse results in terms of interactivity, with a throughput of 2.8 Mtriangles/second, with an average frame-rate of 10 fps, and a worst case of 2.8 fps for views with maximum of 1 M Triangle budget. As demonstrated in the

video, performance is perfectly adequate for interactive inspection tasks. The quality of representation is extremely high. An example is presented in Fig. 5.6.

5.6.3 Streaming performance

The latency time needed to fully refine the data at the application start-up and to refine the model during the exploration, is probably one of the most critical issues that a mobile device need to deal with. Of course, this time is independent from the rendering thread but only depends on the network bandwidth. The multiresolution structure along with the output-sensitive technique adopted, allow the client to only need a working set which depends on the screen resolution. Hence, the latency time to download the current working set is proportional to the maximum resolution of the mobile device. We have measured performance with a wireless connection of a Linksys WAP 200 802.11 b/g access point 54 Mbps, as well as with UMTS/HSPA connections. The wireless network was shared among many clients, and we measured its peak performance to be 17 Mbps.

With the iPad, at start-up we need to load about 14.5MB to see the whole David statue in full screen (1.1Mtri), and 19.9 MB to see the St. Matthew (1.8Mtri). Our application performs data fetching asynchronously in a separate thread to avoid delaying interactive rendering. We measured data fetching speed to be of about 4.8Mbps on the wireless network. We are thus able to use about 35% of the available bandwidth. Full refinement takes about 30s for both statues. Due to progressive refinement, after a couple of seconds, however, the statues are already visible with a reasonable quality. On the UMTS/HSPA, the data fetching speed was measured to be about 3.3Mbps, for a full refinement latency of about 45s. The iPhone4 is about 1.5x slower, which is mostly due to the lower CPU performance, which leads to increased decoding time.

5.7 Discussion

We have presented an architecture capable of distributing and rendering gigantic 3D triangle meshes on low-powered platforms, proving its performance on common hand-held devices. Our architecture exploits the properties of conformal hierarchies of tetrahedra to produce a data structure which is adaptive, compact, and GPU friendly. By combining CPU and GPU compression technology with our multiresolution data representation, we are able to incrementally transfer, locally store and render extremely detailed models on hardware-constrained platforms with unprecedented performance.

The original Adaptive TetraPuzzles approach was tailored for desktop systems which typically have enough memory resources and local data storage, as well as fast network connection. On mobile devices, where memory resources are much more limited, it is not always possible to maintain in memory the working set of data needed to render the visible part of the model. This is particularly relevant when exploring very fine details of the model where more data is needed to satisfy the visual quality constraints. Nonetheless, due to power consumption constraints and driver limitations, network bandwidth on mobile devices is usually way below desktop platforms. Thanks to our compact GPU-friendly representation we can cope with the limitations on bandwidth and memory resources present in mobile platforms, since the rendering can be performed directly from our compact representation.

Besides improving the proof-of-concept implementation, we plan to extend the presented approach in a number of ways. In particular, we are currently incorporating occlusion culling techniques, useful for data-sets with a high depth complexity, and we plan to introduce more sophisticated shading/shadowing techniques.

This enabling technology is intended to be a high performance building block for mobile 3D graphics. A major application area of massive model rendering is cultural heritage, where highly detailed representations are required to reproduce the unique aura of real objects. We also plan to better integrate this technology with web infrastructures by providing an implementation running in WebGL.

Advantages. Our approach improves on the well-known ATP, supporting general dense 3D meshes and providing an efficient method for distribution and rendering complex 3D models on resource-limited platforms. The compact GPU-friendly representation allows both minimizing bandwidth and memory usage, and power-consumption thanks to this reduced data traffic. Bigger portions of the 3D model can be stored in local memory enabling high resolution 3D model exploration on devices with reduced memory resources, and at the same time the cache module is better exploited reducing bandwidth usage when re-visiting parts of the model.

Limitations. The regularity of the subdivision structure does not adapt to geometric complexity. In addition, the mesh simplification approach repeatedly merges nearby surface points based on error minimization considerations. The method, thus, performs best for highly tessellated surfaces that are otherwise relatively smooth and topologically simple, since it becomes difficult, in other

cases, to derive good “average” merged properties and to generate subdivision with a reduced amount of boundary vertices/patch.

Scalability. The method poses minimal constraints on the hardware, and just requires standard vertex shaders for data decoding. In terms of computation cost, the decoding in the vertex shader adds only a matrix multiplication for vertex position dequantization, and a few basic operations for recovering the normal vector. On all tested platforms, we achieved maximum performance when data is stored in an interleaved array of 8 bytes per vertex.

5.8 Bibliographical Notes

The major part of the content of this chapter is based on paper [Bals 13c] where we presented an extension to the well-known Adaptive TetraPuzzles [Cign 04] with a compression-domain representation that reduces both the size of data to be transmitted and the memory occupancy both in system and video memory. Introduction and Discussion are based on paper [Bals 13b], which discusses the method presented in this chapter and the Adaptive Quad Patches approach presented in Chapter 6.

Adaptive Quad Patches: An Adaptive Regular Structure for Web Dis- tribution and Adaptive Rendering of 3D Models

The scheme presented in the previous chapter provides an efficient solution for general triangulated meshes, providing fast incremental loading and reasonable compression, GPU accelerated rendering methods, and adaptive view-dependent culling techniques.

There is now, however, an increasing interest for techniques tuned for lightweight, interpreted, and scripted environments. The limitations of such platforms impose additional constraints on 3D streaming formats, which should be based as much as possible upon preexisting components in order to avoid the overhead of coding complex decoders and data structures in non-optimized programming environment, such as JavaScript.

In this chapter, we show how a much simpler representation can be obtained for particular kinds of objects, smooth and topologically simple, which comprise an important subset of the 3D models, e.g., in cultural heritage applications. We exploit these characteristics by proposing a solution based on an iso-parametric quad-parametrization of the 3D models, on top of which we construct a multiresolution structure. The resulting representation is extremely compact, and can be implemented on top of pre-existing libraries. Thus, this scheme is well-suited for scripted environments such as Web browsers, where the limited CPU performance for interpreted code can be overcome by exploiting efficient implementations for image decoding already present in the API.

6.1 Introduction

WE introduce a remote rendering approach in which a large class of textured geometric models, which can be parametrized into quads,

are converted into compact multiresolution representations suitable for storage, distribution, and real-time rendering on modern commodity/web platforms.

The Adaptive Quad Patches (AQP) method presented here employs a solution that encodes much of the shape and appearance of a model into a texture. This is also the goal of *geometry images* [Gu 02, Sand 03], which enable the powerful GPU rasterization architecture to process geometry in addition to images, and the networking component to rely on already existing and optimized libraries for compression and streaming of images. Geometry images focus on re-parametrizations of meshes onto regular grids, while we focus on developing a specific multiresolution structure on top of a re-parametrized model. Our quad-based parametrization leads in addition to a tighter texture packing and a simple handling of chart boundaries. We also adapt semi-uniform adaptive patch tessellation [Dyke 09] to handle collections of quad patches at different LODs with textured detail. Whereas previous adaptive GPU mesh refinement approaches are typically used to amplify coarse geometry, our end-to-end framework is designed to faithfully reproduce a resampled high-resolution model.

Our method extends and combines recent results in geometric processing, real-time rendering, and web programming. In particular: we exploit recent results on surface reconstruction and isometric parametrization to transform a point cloud into a two-manifold mesh whose parametrization domain is a small collection of 2D square regions; we encode the resulting parametrized mesh into a very compact multiresolution structure composed of variable resolution quad patches whose geometry and texture is stored in a tightly packed texture atlas; we adaptively stream and render variable resolution shape representations using a GPU-accelerated adaptive tessellation algorithm with negligible CPU overhead.

Our approach is scalable and enables interactive exploration of gigantic 3D mesh models on common hardware, including web browsers.

6.2 Method Overview

Our contribution is an unattended software pipeline for automatically converting a large variety of textured geometric models into compact multiresolution representations suitable for storage, distribution, and real-time rendering on modern commodity/web platforms. Fig. 6.1 illustrates the main components of our pipeline.

The pipeline takes as input a dense point sampling of the original model. This kind of sampled representation can be created from a large variety of models -

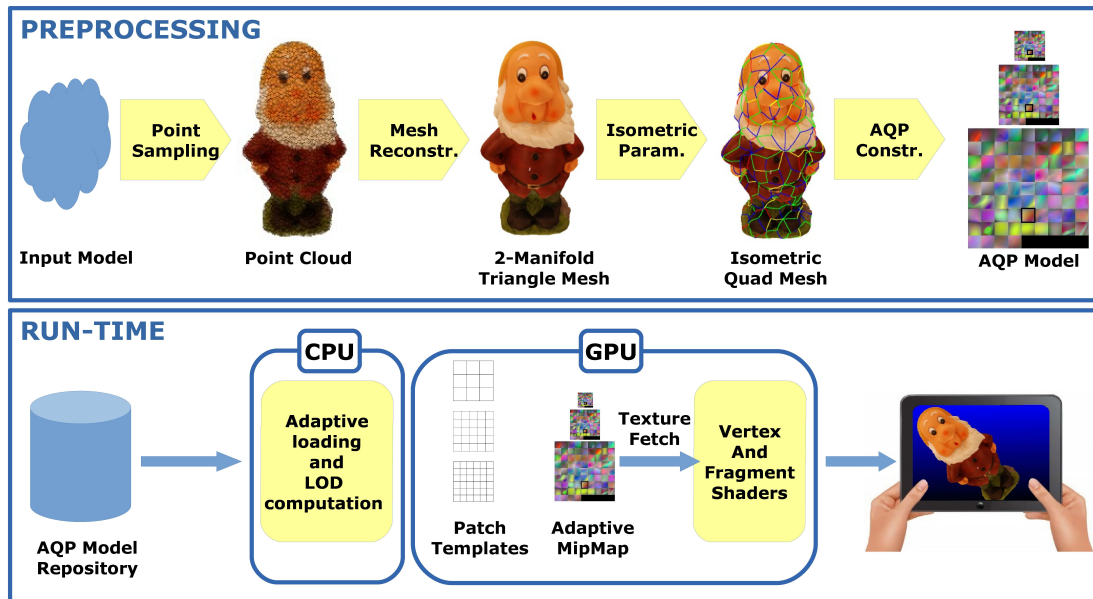


Figure 6.1: The AQP pipeline. We take as input renderable models and generate compact adaptive streamable representations.

point clouds, meshes, or parametric objects. A two-manifold triangular mesh is first fit to the point cloud using a surface reconstruction and topology cleaning step, and the resulting two-manifold mesh is parametrized (see Sec. 6.3). Our parametrization domain D consists in a small collection of almost isometric square patches. Since each of these patches can be sampled on a grid with lines parallel to its sides, storing 3D positions, normals and colors of the associated point on the mesh in a $N \times N$ square patch, the overall shape representation consists of M square patches of $N \times N$ samples. This regular structure is then encoded into a compact multiresolution structure composed of variable resolution quad patches assembled in 2D images. Geometry and texture are stored in a tightly packed multiresolution texture atlas, which can be streamed over the network for generating variable resolution shape representations using a GPU-accelerated adaptive tessellation algorithm (see Sec. 6.4). The resulting rendering subsystem has negligible CPU overhead and is heavily built on top of consolidated 2D image representations. It can thus be efficiently implemented both on conventional commodity platforms and on the newly emerging scripting platforms for the web. The various steps of our pipeline are detailed in the following sections.

6.3 Surface Reconstruction, Parametrization and Quad Re-meshing

We assume that the input to our pipeline is a point cloud or a tessellated mesh, possibly with artifacts such as non manifoldness.

The first phase of the method transforms the input in a clean manifold triangle mesh. As a first step, we use Poisson reconstruction [Kazh 06] to obtain a manifold and watertight version of the input mesh, which is saved in a streaming format. We then perform a single streaming pass over the generated triangle mesh, and discard from it the connected components with less than a prescribed number of triangles, to remove topological noise. It should be noted that these reconstruction and filtering steps may not be necessary if the input mesh is already two-manifold, or it may be replaced with other reconstruction or topological repair techniques.

The second phase consists of parametrizing the mesh on a simple quad-based domain. We first construct an almost isometric triangle mesh parametrization through abstract domains [Piet 10] (see Fig. 6.2.(b)), which maps the original mesh to a simplified parametrization domain made of equilateral triangles. The method works by applying local simplification operations to the input mesh, and remapping the triangles of the original region onto those of the simplified region. By iterating the simplification and remapping process, the algorithm ends with a small parametric domain consisting of a simple triangle mesh. This domain is in turn remapped into a collection of 2D square regions by adding a vertex in the barycenter of each triangle and building a quad for each edge (see Fig. 6.2.(c)). The produced parametrization exhibits very low isometric distortion, because it is globally optimized to preserve both areas and angles. In order to manage larger models than those handled by the original method [Piet 10], we have heavily reduced memory usage by employing a multiple-choice approach instead of a global queue to select the edge collapses during the simplification phase [Wu 02].

Once we have obtained the quad-based parametrization of the input model, we re-sample each quad, taking the samples from the original geometry. The sampling phase, which works on the point cloud representation used as input for the reconstruction step, associates to each quad a regular grid of samples (position, color, normals). This final collection of regular grids (see Fig. 6.2.(d)) is used as input for the multiresolution structure creation phase.

6.4 Quad-based Multiresolution Structure

The previous steps of the pipeline are able to produce a parametrized mesh made of a set of equally sized quad patches, each of them composed of $w \times w$ samples. Vertex, color and normal information are available for each sample.

In order to achieve adaptivity, we encode the resulting parametrized mesh into a very compact multiresolution structure composed of a collection of variable

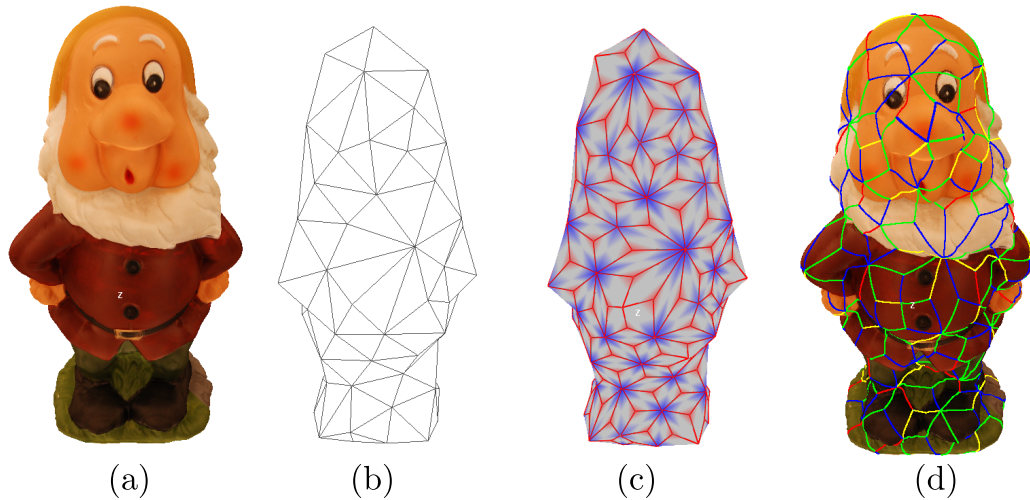


Figure 6.2: Reconstruction steps. The original model (a) is parametrized on a simple quasi isometric triangulation (b), which is in turn remapped to a collection of 2D square regions (c), used as a basis for re-sampling the model (d).



Figure 6.3: Rampant model. Example of rendering with patch color, level color and with original color.

resolution quad patches, whose geometry and texture is stored in a tightly packed texture atlas. At run time, we exploit this structure to rapidly distribute and generate seamless view-dependent multiresolution visualizations. These view-dependent representations are constructed by adaptively loading and combining patches at different resolutions, depending on viewing parameters. Surface continuity is guaranteed by carefully handling patch boundaries (see Fig. 6.3).

6.4.1 Quad Structure

The main advantage of our quad parametrization is that a complex surface can be compactly and efficiently stored by storing geometry in a tightly packed texture atlas. Since all quads have the same size, packing is trivial and very efficient.

Each image quad represents a square surface patch, and is made independent from the others by replicating in it the boundary vertices. The patch triangulation is implicit. A surface is thus represented by a 2D texture that contains a number of patches, arranged in a 2D grid. Because of the GPU maximum texture size limitation (generally 4K or 8K), a single large model can be split into a number of texture pyramids, each of them with the maximum resolution lower than the limit. A multiresolution representation is constructed from the high-resolution representation by building a texture mip-map through a filtering operation (see Fig. 6.4(a)). Each half resolution representation can be constructed by a simple average filter, with special care taken only for properly handling samples at quad boundaries (see Sec. 6.4.2). Inside a single pyramid, square patches are simply organized as a square 2D matrix of $N \times M$ patches (see Fig. 6.4(a)).

The geometry mipmap is enriched by parallel color and a normal mipmaps, which are based on the same concept of quad patch simplification. These two mipmaps are not constrained in our system to have the same resolution of the geometry. In general, they will have higher resolution with respect to the geometry one, allowing us to achieve the same effect of surface texturing with color and normal maps (which typically are at higher resolution than the geometry).

6.4.2 Pre-processing

For each pyramid we build three mipmap hierarchies: a geometry, a color and a normal mipmap. Processing of the three structures share some aspects: they start sampling the input data-set on a patch basis, and then build inner mipmap level, with a patch filter approach that maintains continuity among boundary samples of adjacent patches. Inner patch samples are simply averaged from 4 children samples, instead boundary samples are averaged without taking into consideration the 2 samples which do not belong to the boundary. The corner samples which are shared among 4 adjacent patches, are filtered with pure sub-sampling for the same reason, see figure 6.4(b). Operating without this special care would produce corresponding boundary samples with different contribution for adjacent patches, thus losing continuity.

We exploit the geometry patch structure to encode the positions as a map of 3D displacements with respect to the bi-linear interpolation of the patch corners at the corresponding u, v parametric coordinates. The corners of all the square patches are stored quantized at 16 bits per component, in a root file, which would correspond to the coarsest level of our multiresolution structure. All other levels, which represent displacements with increasing resolution with respect to the root, are stored quantized at 8 bits per component. Instead of using a global, per level,

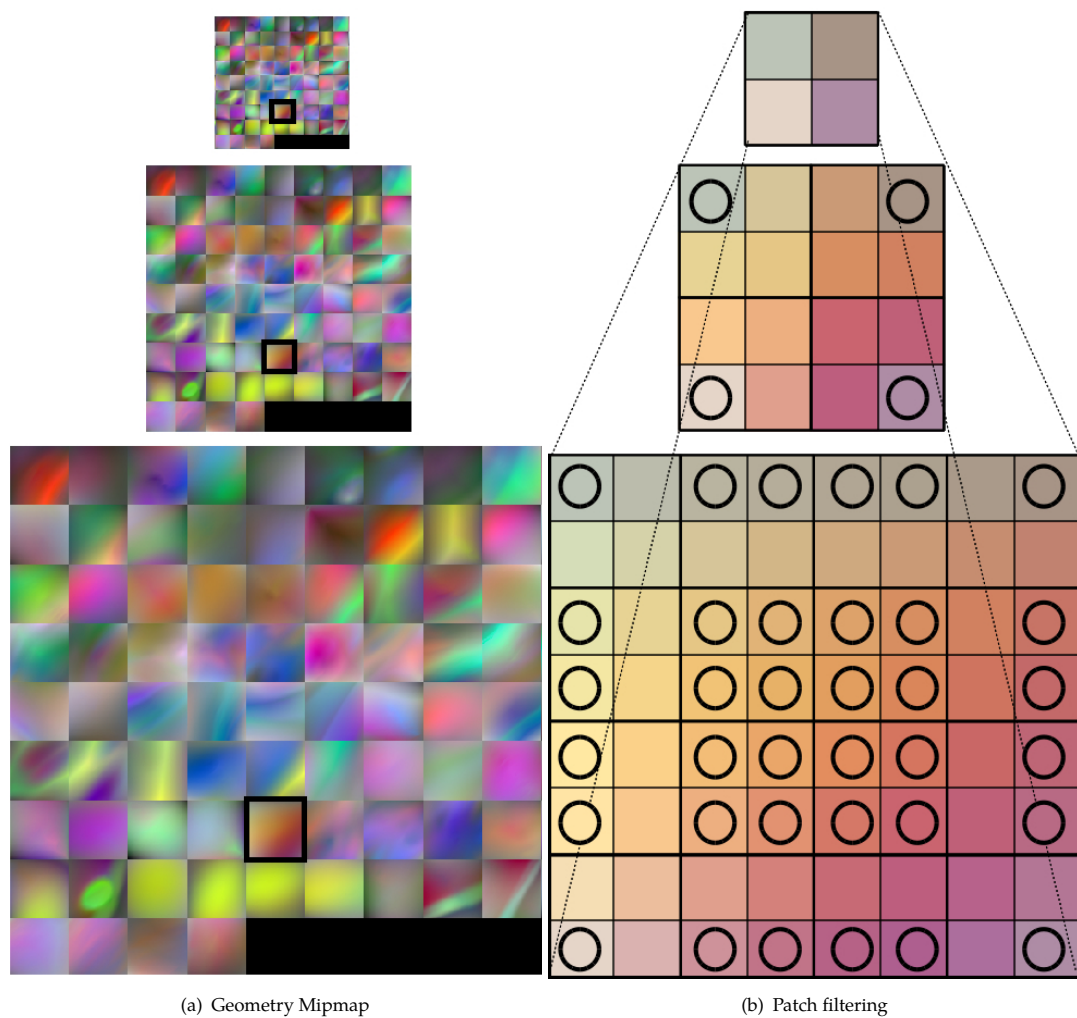


Figure 6.4: Multiresolution structure. In figure 6.4(a) there is an example of three levels of a geometry mipmap with 76 patches on a grid 9×9 , with highlighted the filtered patch of figure 6.4(b). Figure 6.4(b) shows three levels of a quad patch. Circles inside quads shows which samples contribute to the generation of the parent sample: one for the corner (sub-sampling), two for edges, and four for inner sample. Upper level correspond to the root patch representation with only the four corners.

uniform quantization range, which would introduce too many discretization artifacts, we decided to modulate the quantization range per generated vertex.

In a first step, quad quantization ranges are computed for each patch, by taking the minimum and the maximum differences between positions inside the patch and predictions obtained through bi-linear interpolation of corner positions. In order to avoid discontinuities caused by different per-patch quantization, we move quantization information to the patch corners. We thus determine for each patch corner the minimum of all the adjacent quad minimum values, and the maximum of all the maximum values. Using these corner values, the quantization range for a sample of a patch at parametric coordinates u,v is given by the bi-linear interpolation at u,v of all the for corner quantization factors. This

way, quantization on edges depends only by the two corners defining the edge, and thus is shared among the two adjacent patches, solving the quantization continuity problem.

Geometry is finally stored using PNG compression to avoid to introduce further artifacts due to possibly uncontrolled lossy compression. For colors and normals, instead, we can choose between storing them as PNG files or using DXT1 (for colors) and DXT5 (for normals) compression. Using these hardware-supported compressed formats is only possible when using our OpenGL renderer, since WebGL currently lacks support for them. Data is stored in separate files: each mipmap is subdivided by levels, and then the level is split into tiles if its width is bigger than a predefined value, 512 samples in the current implementation. This approach is useful to avoid to require a complete level at a single time, which surely would be too big for the finest level of details. Tile width is a multiple of the patch finest width to avoid to split a patch into separate files.

6.4.3 Adaptive seamless rendering

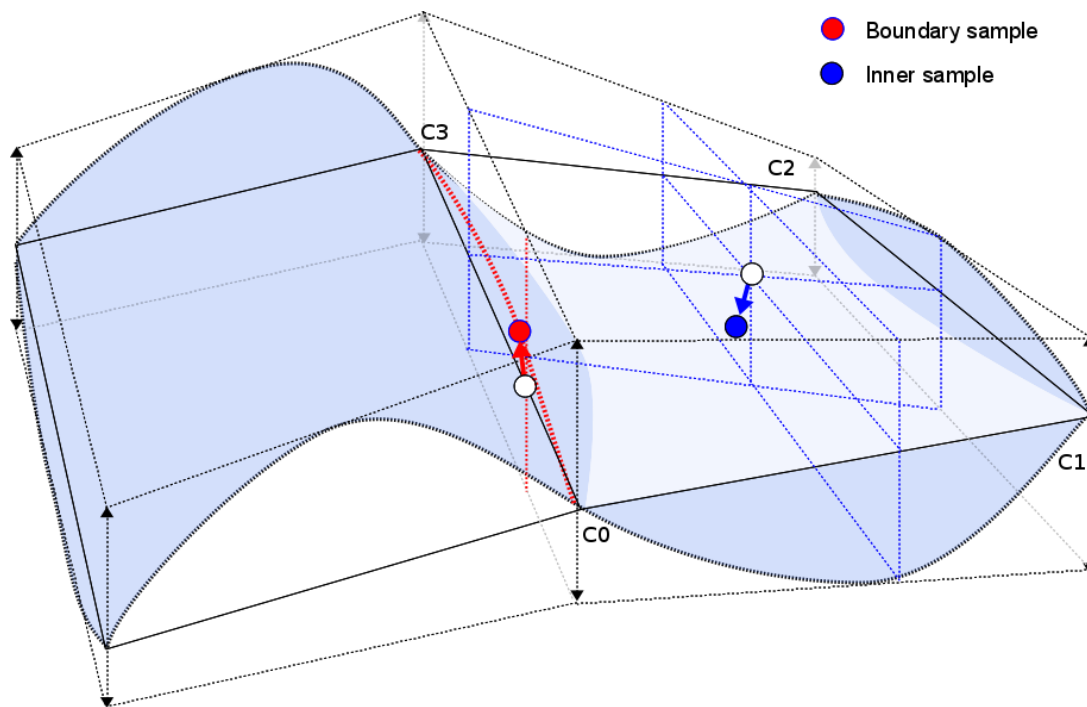


Figure 6.5: Seamless point dequantization. Vertices on the boundary of the two adjacent patches, like the red one, share the same dequantization values derived by the linear interpolation of the same two corners $C0, C3$. Inner vertices quantization min, max are derived from the bi-linear interpolation of the 4 corners min, max values. White circles show corners interpolations on the patch. Vertical arrows shows the corner min, max ranges, used for dequantization.

We adaptively stream and render variable resolution shape representations using a GPU-accelerated adaptive tessellation algorithm with negligible CPU

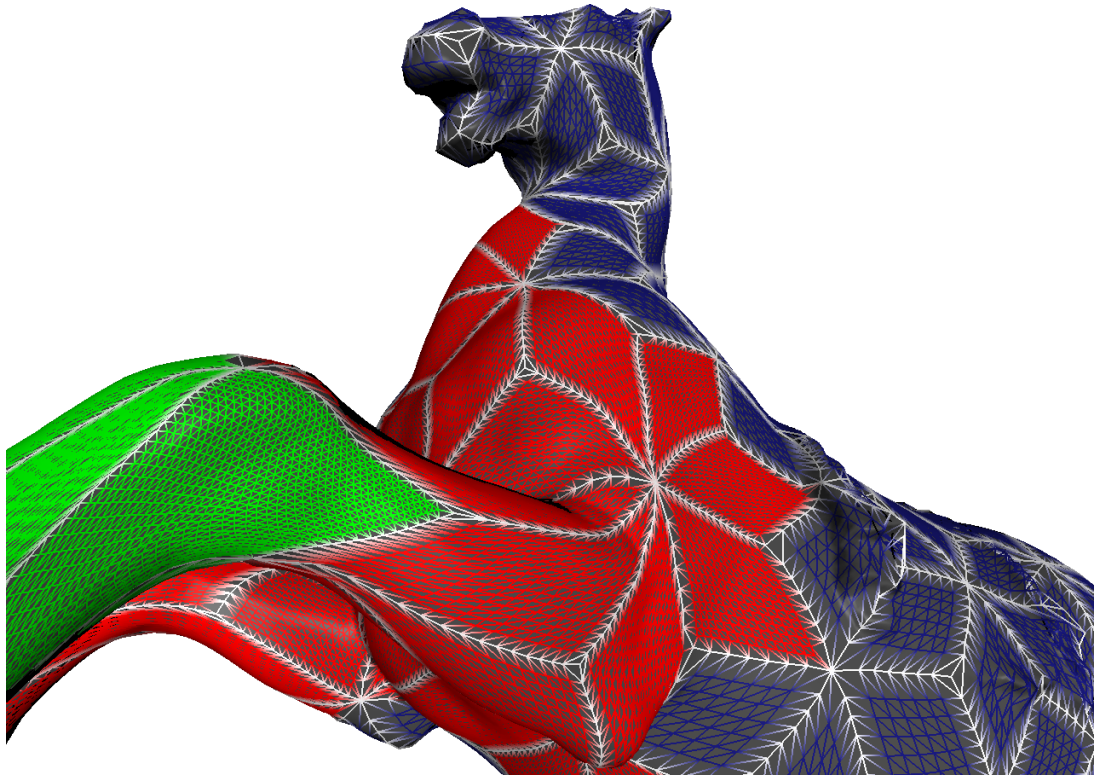


Figure 6.6: LOD seamless tessellation. Seamless tessellation among patches at different LODs: vertices are snapped on the edges at the edge LOD, which depends from the projected edge length on the screen.

overhead. Seamless rendering is substantially performed by the GPU through a vertex/fragment shader pair, leaving to the CPU only the tasks of selecting the proper level of details for each patch, and of querying missing data from a server. Adaptive tessellation of a coarse mesh could be done exploiting the geometry shader, but this GPU stage cannot output more than a certain number of primitives, (1024 in the original specification) thus limiting the subdivision levels. We preferred to use the instancing approach, creating during the initialization a small number of subdivision regular grids, containing the (u, v) parametric coordinates of the vertices and an index telling where the vertex resides (inside or on the boundary) relatively to the patch. We use $K = \log_2(w) - 1$ vertex buffer objects, (being w the linear size of a patch at maximum resolution) to tessellate the patches from a size of 4 linear samples to the maximum size w , with resolutions doubled for each level. Let's say that root is at level 0, first patch at level 1, and finest patch at level K . The renderer pre-allocates for each pyramid three texture mipmaps (geometry, color and normals) initialized only with the root data and which will contain the patches at various resolutions, once they will be available. At each frame the renderer selects the proper level of detail for each patch, if it is not available chooses the finest available level for it and posts a request for the tile containing the desired data (see Sec. 6.4.4). To produce a

continuous representation patches must match perfectly along the edges, so they must have the same level of resolution for each edge. The patch LOD evaluation first computes the desired LOD for each edge of the quad by projecting it to the screen and comparing it with the desired screen tolerance. Edge LOD cannot be higher than the minimum of the two finest level of available data of the two adjacent patches along this edge. The quad patch LOD is set to the maximum (finest) of the 4 edge LODs. A texture is filled at each frame with the 4 edge LODs for each quad.

In the draw procedure for each patch the tessellation corresponding to the selected quad LOD is drawn with a proper vertex/fragment shader pair. The tessellation vertices are triple with (u, v, e) where e represent the edge to which belongs that vertex (0, 1, 2, 3) or 4 for inner vertices. The vertex shader convert the (u, v) and quad patch id to the corresponding coordinates in the texture mipmap, where it can fetch the geometry displacement. When a vertex belongs to the inner part is simply a matter of scaling and translating (u, v) to fetch proper data. Instead when detecting edge vertices we need to handle them properly before fetching data, to be able to stitch together adjacent patches. Edge LOD is always coarser or equal than patch LOD. To get a seamless representation we snap boundary patch vertex parametric coordinates (u, v) at the edge resolution, which is the same for adjacent patches also if their quad LOD is different. The snap procedure identifies the edge id from the third component of the vertex and read the corresponding LOD value from the edge LOD texture at (quad, edge id) texture coordinates. Then snaps the vertex (u, v) parametric coordinates from current quad LOD to the edge LOD using following equations:

$$edgesize = 2^{edgelevel+1} - 1$$

$$\mathbf{uv} = \frac{round(\mathbf{uv} \cdot edgesize)}{edgesize}$$

Once modified (u, v) , and set LOD as the current edge LOD, the sampling procedure is the same as for inner vertices. The dequantization is performed with a scale factor that depends from (u, v) as highlighted in 6.4.2: we need to get the 4 quad corners quantization min and max values, and bi-linearly interpolate them. The resulting min,max pair is used to dequantize the vertex displacement. The quantization factors obtained in such a way permits to have the same values all over the edge between two adjacent patches, because they derives only from the interpolation of the two corners defining that edge. Corners min,max quantization factors are four pairs of 16 bit values stored for each quad of each levels into a static texture which is loaded at initialization and reused at each frame. The base quad position is given by the bi-linear interpolation of the 4 quad corners

at the possibly modified (u, v) coordinates. Then, if vertex is not one of the four corners, its value is offset by the vector found in the geometry texture mipmap at remapped uv coords, considering the quad offset and the quad size (see Fig. 6.5). The resulting rendering is seamless (see Fig. 6.6).

Color and normal (u, v) coordinates are found in a similar way, except for the snap step, which revealed to be not necessary for these attributes. Then these coordinates are passed to the fragment shader which takes care of properly sampling color and normal mipmaps to perform per pixel texturing and shading.

6.4.4 Adaptive streaming

Our compressed representation forms the basis of a scalable streaming system able to adapt to client characteristics and to exploit available network bandwidth.

The server component provides access to tile repositories, without differentiate among position, normal, or color components. From the server's point of view, a repository is just a database with a unique key for indexing a block of bytes containing an encoded bit-stream representing a compressed wavelet coefficient matrix. In order to increase server-side scalability, no processing is done in the server, whose only behavior is to return a block of bytes if present. This approach makes it possible to leverage existing database components instead of being forced to implement a specific storage manager. In this work, storage management is done through Berkeley DB, and data serving is done through an Apache2 server extended with an appropriate module.

The client implements streaming using asynchronous data fetching during rendering. During rendering, requests for missing patches are remapped to unique identifiers built from pyramid ids and tile ids and stored in a request queue. The priority is the difference between the desired patch LOD and the currently available one. At the end of the frame, only as many new requests as those allowed by the estimated network bandwidth are issued and managed by a separate network access thread, and the remaining ones are ignored.

A separate thread takes care of getting data from the server and possibly decompresses them as in the case of PNG tiles. When data becomes available it is inserted into the proper pyramid texture mipmap.

6.5 Implementation and Results

An experimental software library and viewer applications supporting the AQP technique have been implemented both using the OpenGL and WebGL environments. The OpenGL version, implemented in C++, works both on Linux and



Figure 6.7: Models rendered with the adaptive quad patches method. Top row shows the rendered models at pixel tolerance one. Bottom row shows the patch structure. Complexity ranges from 6.4 to 62.5Msamples.

Windows platforms, and can be also used as a Web browser plugin using QT 4.8. The WebGL version is written in JavaScript on top of the publicly available SpiderGL library [Di B 10].

We have extensively tested our system with a number of data-sets. In this paper, we discuss the results obtained with the models in Fig. 6.7, all coming from laser scanning acquisitions. The complexity of the input data-sets ranges from 8Mtriangles to 90Mtriangles (left to right). Our tests are focused on the WebGL version of our code, in order to evaluate the feasibility of the method for coping with scripted environment limitations.

6.5.1 Pre-processing and compression rates

Table 6.1 shows the processing results of the various data-sets, using the WebGL version of our code, which uses only PNG compression applied to delta encoded samples. It is clear that compression rates can be heavily improved by using DXT1 and DXT5 to compress attributes, but these compressed encodings are not widely supported in WebGL implementations. For this reason, the benchmarks presented in this paper use plain textures and PNG encoding for transport. Even with such a simple approach, the method is able to encode a sampled geometry in about 15bps for models with positions and normals, and about 24bps for

colored models. It should be noted that our objective is not to achieve state-of-the-art compression rates, but, rather, to propose a method supporting adaptive streaming, and variable resolution rendering with an easy implementation in a WebGL context.

Dataset	Input Tri	Out level	Patch Count	Output Samples	Geom. bps	Color bps	Normal bps
Dwarf	8.4M	7	300	6.4M	6.3	9.54	8.53
Shepherd	8.4M	7	300	6.4M	6.3	0	8.71
Horse	8.4M	7	300	6.4M	6.3	9.21	8.42
Head	94.4M	9	180	62.5M	6.3	0	8.40

Table 6.1: Adaptive Quad Patches processing results. Adaptive quad patches representations of the test models.

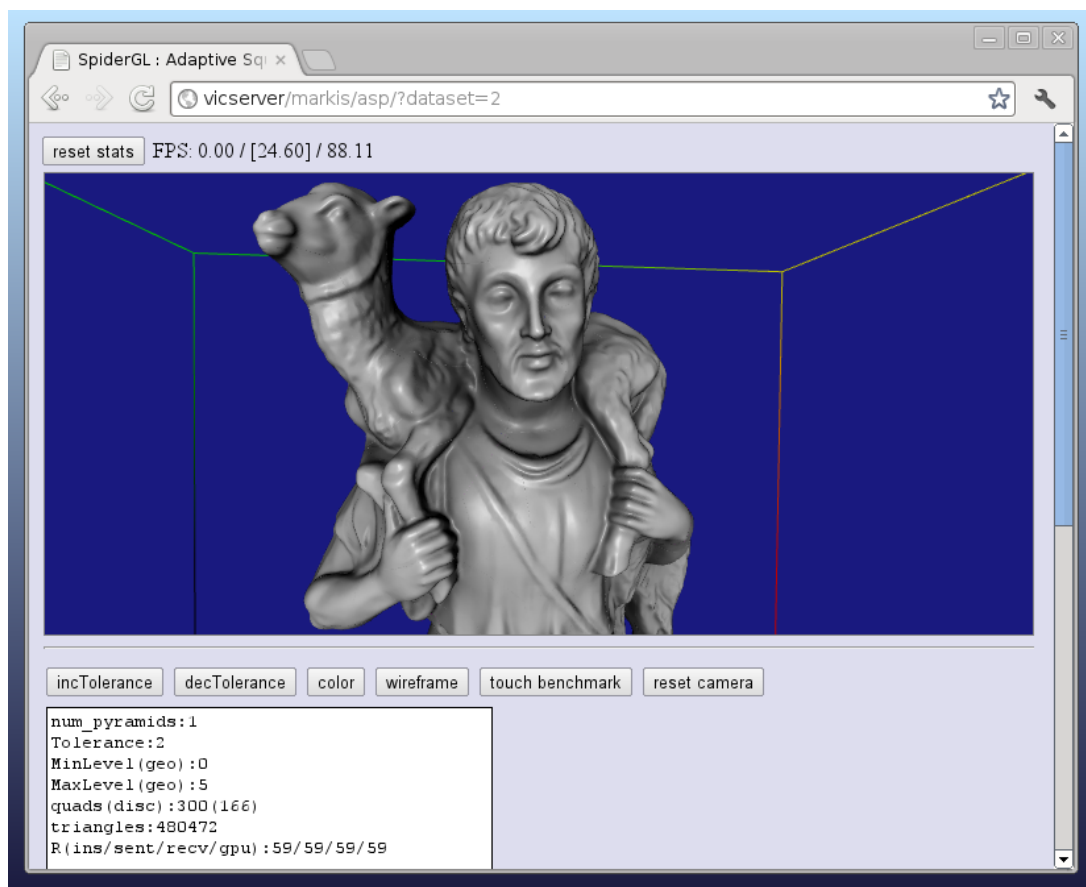


Figure 6.8: WebGL implementation running in Chrome. 3D content can be delivered in a HTML5 canvas. Models are incrementally loaded during rendering.

6.5.2 Adaptive rendering

The method has been implemented for both mobile clients using native code and web-based environments, using JavaScript and OpenGL. In order to guarantee full compatibility among platforms and minimum decoding overhead, we have

used plain PNG encoding for storage and transport of quad patches, achieving a compression rate of about 24bps for colored models. Rendering performance has been evaluated both for in-browser rendering (desktop machine) using WebGL and mobile rendering using the native implementation. Using a Chromium browser, on a 1.6 GHz laptop equipped with an NVidia Geforce GTX 260M with 1 GB video memory, we achieve a throughput of 34.2MTri/s, for an average frame-rate of 37fps for a 750x350 in-browser window and a tolerance of 1 pixel. Coming to mobile platforms, on a Acer Iconia Tab W501, with a AMD Radeon HD 6290 graphics adapter with 384MB DDR3, we achieve a peak throughput of 27.5MTri/s using the native C++/OpenGL implementation, which is perfectly adequate to guarantee real-time performance. The combination of incremental multiresolution refinement with compression also reduces network bandwidth, which was measured to range from 312Kbps for exploration of areas not previously seen, and peaks of 2.8Mbps at viewpoint discontinuities. Interactive mobile applications are thus possible both for wireless connections to typical ADSL lines, and for current mobile broadband network such as UMTS/HSPA. More detailed results, with a focus on the WebGL implementation, are available in the original paper [Gobb 12].

6.5.3 Network streaming

Extensive network tests have been performed on all test models, on an ADSL 8Mbit/s connection, on a mobile broadband connection, as well as on an intranet. Tests have been made for the viewer application under interactive control.

In an interactive setting, since rendering is progressive and, on average, viewpoint motion is smooth, only few new patches per frame need to be refined, and only data for patches not already cached are requested to the server. We have measured the bandwidth required by a client to provide a “no delay” experience in typical inspection sequences. We measured an average bit rate of 312Kbps for exploration of areas not previously seen, and peaks of 2.8Mbps at viewpoint discontinuities, i.e., when the application has to refine the model all the way to the new viewpoint and the refinement algorithm has to always push new patches to refine in the request queue because of non incremental update.

By introducing client-side or server-side bandwidth limitations, it is possible to reduce burden on network and server, making the system more scalable while maintaining a good interactive quality. Due to the reasonable compression rate and refinement efficiency, we have found that using the system on a 8 Mbps ADSL line produces nice interactive results. In that case, delays in case of rapid motion become visible, but with little detriment to interaction (e.g., only 2s are

needed to produce a fully refined model visualization from scratch). This also allows a single low-end server to manage a large number of clients.

6.5.4 Device compatibility considerations

There are a variety of GPU chipsets from different vendors present in current mobile devices. From those, the most extended ones include the PowerVR SGX 5xx GPUs present in Apple devices and many Android phones, and the Qualcomm Adreno family integrated in Android HTC devices and many of the latest high-end Android devices. The hardware present in those two GPU families fully support OpenGL ES 2.0 and, in their latest versions, OpenGL ES 3.0. Unfortunately, while on Android devices the drivers typically expose VTF capabilities, the drivers from iOS 4.x and on does not. Rendering this feature unavailable in iPhone/iPad/iPod devices. The latest T6xx version of the ARM Mali GPU, integrated in the Nexus 10, also includes support for VTF. A number of tablet devices, in addition, use chipsets with discrete or integrated GPUs originally designed for netbooks, which fully support VTfs. An example are chips from the AMD Radeon HD series used in a number of Windows-based tablets.

6.6 Discussion

We have introduced a remote rendering approach, which builds a compact image-based multiresolution representation suitable for efficient distribution and rendering of highly detailed 3D models on low-powered platforms and, in particular, scripted environments such as Web browsers.

In the future, we want to study the impact of more aggressive lossy compression on the error introduced in geometry reconstruction.

Advantages. The pipeline is fully automatic and targets densely tessellated models, such as those created by 3D scanning or modeling systems such as ZBrush. Our approach bridges the gap that currently exists from general-purpose meshes to rendering oriented structures based on real-time tessellation with normal/bump maps, which are typical of modern gaming platform but currently require considerable human effort to create. The simplicity of a regularly re-meshed representation has many benefits. In particular it reduces random memory accesses and eliminates the indirection and storage of per triangle vertex indices and per vertex texture coordinates. The resulting representation is compact, can be built on top of existing image representations, and is very well suited to streaming.

Due to the negligible run-time CPU overhead, real-time performance is achieved both on conventional GPU platforms using OpenGL, and on the emerging web-based environments based on WebGL. Promising applications of the technology range, thus, from the automatic creation of rapidly renderable objects for local and online games to the set-up of browsable 3D models repositories in the web.

Limitations. The proposed method is not general purpose, but targets meshes defining closed objects with large components (i.e., typical solid objects without fine topological details). As for other compressed streamable formats, we do not strive to exactly replicate the original geometry and color, but only to visually approximate them in a faithful way. As a result, and similarly to compressed video/image formats, our representation is lossy, and thus not applicable in situations where precise measures of the original geometry are required (e.g., CAD systems).

Scalability. This technique performs vertex displacement of a base surface using the information contained in the image-encoded representation, thus Vertex-Texture-Fetch is a required feature. Although nowadays most desktop and mobile GPUs already support texture access in the Vertex Shader (thanks to unified shader architectures), there are still some cases where it is not supported (i.e., old NVIDIA Tegra GPUs).

6.7 Bibliographical Notes

The major part of the content of this chapter was based on paper [Gobb 12], where we presented the Adaptive Quad Patches approach. Introduction and discussion are based on paper [Bals 13b], where we discussed both the approach presented in this chapter and the Compact Adaptive TetraPuzzles which is presented in Chapter 5.

.....

ExploreMaps: Efficient Construction of Panoramic View Graphs of Complex 3D Environments

In previous chapters, we have presented some approaches for interactive exploration of highly detailed 3D models on common 3D platforms. When dealing with scenes with complex lighting, on the other hand, real-time constraints impose hard limits on the achievable quality. The approach presented in this chapter addresses these limitations by introducing a simple graph representation named ExploreMaps, where nodes are nicely placed point of views, called probes, and arcs are smooth paths between neighboring probes. The basic idea is to ensure visual quality in all kinds of environments by presenting precomputed imagery rather than real-time-renderings. This chapter focuses on the problem of how to appropriately place these probes in arbitrary 3D environments and how to compute nice-looking path, while Chapter 10 will focus on the navigation interface approach than can be realized on top of the graph-based representation.

7.1 Introduction

RECENT research efforts have produced systems capable of rendering moderately complex environments [Yoon 08]. Besides of that, real-time constraints limit the quality that can be achieved to simple shading and/or baked illumination, specially on low performance platforms such as mobile and web environments.

In current 3D repositories, such as Blend Swap, 3D Café or Archive3D, 3D models available for download are mostly presented through a few user-selected static images, or simple orbiting of simplified versions of the original 3D models.

for interactive applications, exploration is typically limited to simplified models. This is particularly true on low-powered mobile devices or script-based Internet browsers.

In this work, we introduce an approach aimed at automatically providing a richer experience in presenting 3D models on low-powered platforms and web environments. The method builds on a novel efficient technique for transforming a generic renderable 3D scene into a simple graph representation, dubbed ExploreMaps, where nodes are nicely placed panoramic views, called probes, and arcs are smooth panoramic video paths connecting neighboring probes. Our GPU-accelerated unattended construction pipeline distributes probes so as to guarantee complete coverage of a generic scene, before clustering them using perceptual criteria, determining preferential viewing orientations, finding smooth good looking connection paths. Probe images and path videos are then computed with off-line photo-realistic renderers, overcoming real-time rendering limitations. At run-time, the graph is exploited both for generating visual scene indexes and movie previews, and for supporting interactive exploration through a low-DOF assisted navigation interface (see Chapter 10 for details). Due to negligible CPU/GPU usage, real-time performance is achieved on emerging WebGL environments even on low-powered mobile devices (see Fig. 7.1).

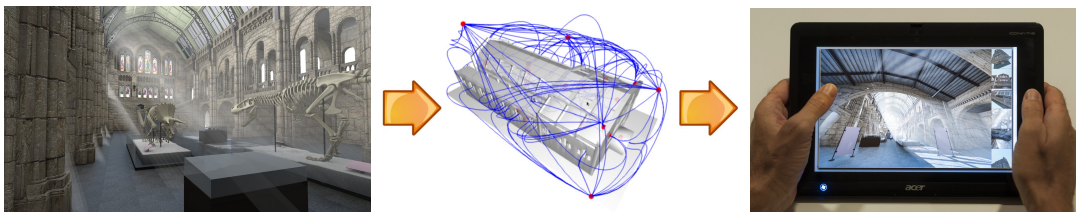


Figure 7.1: ExploreMaps pipeline. We automatically transform a generic renderable model (left) into a simple graph representation named ExploreMaps (center), where nodes are nicely placed point of views that cover the visible model surface and arcs are smooth paths between neighboring probes. The representation is exploited for providing visual indexes for the 3D scene and for supporting, even on low-powered mobile devices, interactive photo-realistic exploration based on precomputed imagery (right).

7.2 Creating the ExploreMaps graph

The input scene is assumed to have a geometry, used for view discovery, as well as a shaded representation, used for high quality rendering. The only assumptions made on the geometric representation is that its bounding box is known, that it can be rasterized producing depth and normal buffers, and (optionally) has a preferential up vector (+Y by default). The explicit definition of the up direction can be removed by employing an unsupervised upright direction solver for man-made objects [Fu 08, Jin 12]. The shaded representation, instead, is any

description of the same scene that can be given as input to an external high-quality renderer. In the simplest form, it consists of the geometric representation enriched with lights and material properties.

```

VirtualExploration() {
  v = PlaceFirstProbe();
  V += MaximizeSeenSurface(v);
  while (not IsComplete(V)) {
    v = PlaceNewProbe();
    V += MaximizeSeenSurface(v);
  }
}
    
```

Listing 7.1: Placing a set of probes to cover all visible surfaces

```

MaximizeSeenSurface(v){
  do{
    Snew = NewSurface(v);
    b = Barycenter(Snew);
    v = MoveTo(b);
  } while (not Converged());
  return v;
}
    
```

Listing 7.2: Moving a probe to maximize the new surface seen

7.2.1 Discovering the scene

We start by formally defining the problem of exploring a scene given only its bounding box and an abstract method for rasterizing the geometry. Let $\mathbf{v} = \{v_0, \dots, v_{n-1}\}$ be a set of n probes, S the entire input surface to be discovered, $S(v_i) \subseteq S$ the portion of surface *seen* by probe v_i , and $S(\mathbf{v}) = \bigcup_{v_i \in \mathbf{v}} S(v_i)$. With this notation, the exploration problem is posed as the problem of finding a set of probes \mathbf{v} such that $S \subseteq S(\mathbf{v})$. We say that a set of probes satisfying this condition is *complete*, which means it sees the whole input surface. Note that we do not have prior knowledge of S , which must be *discovered* by the algorithm.

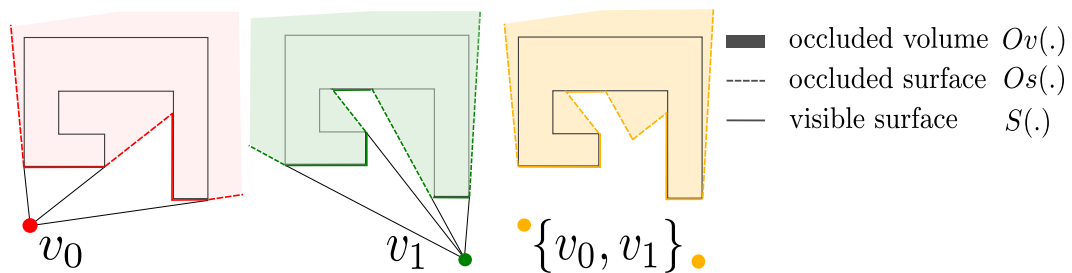


Figure 7.2: Finding probe positions. Occlusion surfaces and occlusion volumes.

7.2.1.1 Incremental algorithm

Our incremental algorithm, see Listing 7.1, starts with an empty set of probes and, at each iteration, adds a new probe and optimizes its position so that it sees a portion of the surface not visible from previous probes, until the coverage is complete¹. Fig. 7.2 illustrates a 2D example of an object, with two probes, the portion of surface seen, and the *occluded surfaces* $Os(v_i)$, i.e., the surfaces

¹Note that, while in this paper we focus on complete automation, the method could also start with user-defined set of probes, e.g., to force inclusion of semantically relevant/nice shots in the graph

joining the discontinuities of $S(v_i)$. The occluded volume for a probe, $Ov(v_i)$ is the (infinite) region of 3D space non visible from v_i and the occluded volume from a set of probes is the intersection of the single viewing volumes $Ov(\mathbf{v}) = \bigcap_{v_i \in \mathbf{v}} Ov(v_i)$. Finally the occluded surfaces of a set of probes, $Os(\mathbf{v})$ is the union of the single occluded surfaces intersected with the occluded volume: $Os(\mathbf{v}) = \bigcup_{v_i \in \mathbf{v}} Os(v_i) \cap Ov(\mathbf{v})$. The occluded surfaces are typically used as candidate locations to place the next probe [Wils 03] as a way to look “behind the corner”. It is easy to prove that if we found a set of probes \mathbf{v} such that the occluded surface is empty, it means that all the reachable surface is seen by some probe, that is: if $Os(\mathbf{v}) = \emptyset$ then $S(\mathbf{v}) = S$. Since we are interested only on the surface reachable from the outside, placing the first probe outside the scene bounding box is a sufficient initialization (PlaceFirstProbe()).

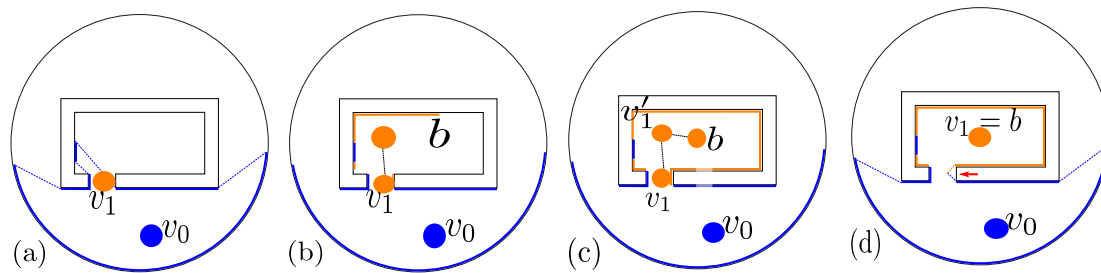


Figure 7.3: Optimizing probe positions. Illustration of the view optimization algorithm. (a) Probe v_1 is added on the occluded surface $Os(v_0)$; (b) the new surface portion seen by v_1 and its barycenter b are shown; (c) the probe is moved to b , and visible surface and barycenter are recomputed; (d) the probe ends up on the barycenter of the surface seen.

7.2.1.2 Finding a new probe position

The position for placing a new probe (PlaceNewProbe()) is chosen at random in the current occluded surface. This is a common choice for many view planning strategies, as it guarantees that some new portion of the surface that will be seen. We then start an iterative optimization algorithm, described in Listing 7.2. At each step, the algorithm tries to maximize the area of the surface seen by the new probe and unseen by previous ones. This is done by iteratively moving the probe position towards the barycenter of the surface visible from its current position (MoveTo(b)). If this is not possible because the segment connecting the current position and b intersects the surface, the probe is put halfway between the current position and the intersection point. This strategy, reminiscent of Lloyd relaxation [Lloy 82], ends when an equilibrium position is reached, i.e., when the barycenter of the surface seen is the same as current position itself (up to a small threshold). The approach implicitly assumes that the probe is at the interior of a closed surface, e.g., inside a building, since, otherwise, the location of the barycenter of the unseen surface would tend to be too close to the surface

itself, or even collapse onto it. We therefore artificially bound the scene with a spherical background object, used by the discovery algorithm, but ignored in the final panoramas. Figure 7.3 illustrates the positioning of a probe, from its placement on the occluded surface to the convergence of the algorithm. The situation depicted in the example is fairly common. We can see that a probe “entering” in a room tends to position itself at the room’s center. Note that not all the surface seen by the new probe during the first steps of the optimization will be visible also from the final position (see the portion indicated by a red arrow in Fig. 7.3.(d)).

7.2.2 Optimizing the set of probes

Our exploration algorithm finds a set of probes that globally guarantee the full coverage of the reachable surface (see Figure 7.4, left). While in principle we could build a graph over this set, we infer a new graph that also takes into account perceptual criteria, in order to enhance user experience. This is done by clustering the set of probes obtained by the exploration phase and replacing each cluster with a representative probe so that the resulting graph is almost equivalent in terms of coverage but *better* in terms of browsing.

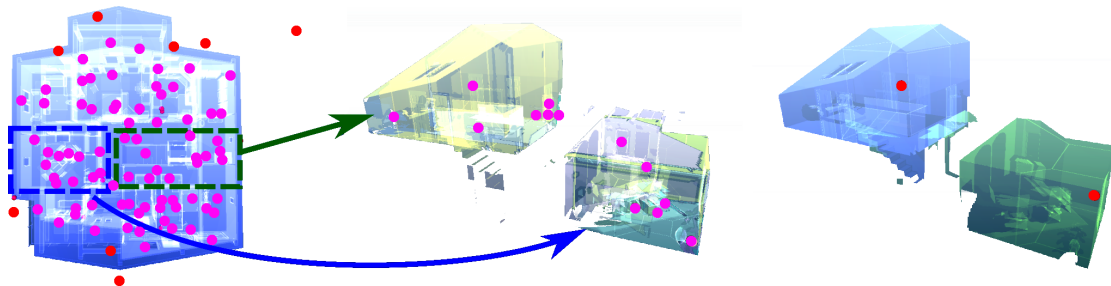


Figure 7.4: Resulting probes. Left: result of Virtual Exploration; middle: two clusters highlighted; right: synthesis of the clusters.

7.2.2.1 Probe Clustering

The clustering is performed by applying the Markov Cluster Algorithm (a.k.a. MCL [Van 08]), with default power and inflation parameters ($e = 2, r = 2$) to the coverage graph. MCL is a well known randomized algorithm that finds out *natural* clusters of nodes in a graph. The key idea is that when you take random walks starting from a node, it is more likely that you stay in the same cluster of the starting node than you jump to another cluster. MCL can use weights on the arcs (i.e., the weight determines how likely the arc is crossed in the random walk), and we assign the weights as the amount of overlap in visible surface between the two probes connected by the arc. This will tend to cluster together

probes that see the same area. The algorithm terminates when it reaches a steady state, obtaining a number of clusters $c_0 \dots c_m$. Two examples of these clusters are shown on Figure 7.4 (middle). In a second phase, we find the *synthesis* of each cluster c_i , i.e., a single new probe that sees most of the surface seen by all the probes in the cluster, while providing a significant panorama in terms of perceptual criteria.

7.2.2.2 Stability Criterion

Secord et al. [Seco 11] introduce criteria for assessing the quality of a view with the purpose of automatically defining the best point of view in a perceptual sense. However, there are important differences with our case, because we must support general object viewing, including exploration of indoor scenes, while the metric proposed in [Seco 11] is only concerned with the object-in-hand paradigm, assuming that the object silhouette is entirely visible. This means that attributes concerning the projected surface area, silhouette and semantic (see [Seco 11]) are hardly applicable in our case, and the linear goodness function they propose would reduce to:

$$G(v) = \alpha \text{MaxDepth}(v) + \beta \left(1 - \int H(z)^2 dz \right)$$

where *Maxdepth* is the maximum depth value, H is the normalized histogram of the depth values and α and β are $[0, 1]$ weights to combine the two metrics. However, it is easy to verify that these criteria alone could lead to unnatural positions. A clear example is a point of view “behind” a column, from where there may be large maximal depth and uniform depth distribution, but where the feeling is to hide behind a column and not exploring the scene. Therefore, we weight $G(v)$ with the *stability* $St(v)$ of the point of view v , leading to a goodness function

$$G_s(v) = G(v) \cdot St(v)$$

We say that a point of view is stable if the view does not change much with respect to the neighborhood. In the previous example, it is clear that a position behind a column (that is, near to an occluder) is not stable because a small displacement would unveil a large part of the scene. In order to formalize this idea, we define a distance function between two points of view v and w as $\Delta(v, w) = \max\{\phi(v, w), \phi(w, v)\}$, where $\phi(v, w) = \text{Diff}(P_w(S(v)), D(w))$, $S(v)$ is the 3D surface seen from v , $P_w(\cdot)$ is the projection of a 3D surface on w , $D(\cdot)$ is the depth map from a point of view and $\text{Diff}(\cdot, \cdot)$ is the number of different depth values in the comparison between two depth maps. We thus define the stability of a point as $St(v) = \max_{w \in N(v)} \{\Delta(v, w)\}$, where $N(v)$ is a set of point

near v . In our system we used the points distributed on the sphere centered at v with radius equal to the distance of the *near plane* used to produce the panoramas.

7.2.2.3 Probe synthesis

The synthesis step starts from the position of the probe within the cluster that has the largest coverage of the surface seen by all probes in the cluster. We then perform a local randomized search using a *simulated annealing* approach, which applies small random perturbations of a probe's position, with an objective function that combines the coverage and the perceptual metric using a weighted average. Upon convergence, using the same perceptual metric, we determine a small set of orientations for each probe that will serve both as preferred arrival orientations when the user move to the probe, and as the *look-at* orientation used when generating thumbnails to represent the probes in a navigation application. Best orientations for each view are selected by finding the dominant peaks in the goodness function with mean-shift clustering [Coma 02]. The up vector for the data-set, if present, is used to avoid upside-down views.

7.2.3 Connecting probes

Once we have the new set of probes, we have to define which probes must be connected with an arc and the geometric path that connects them. One simplistic choice would be to connect two probes if they are mutually visible from each other. This solution is definitely not acceptable, since it would drastically reduce navigation possibilities. For instance, see Fig. 7.5, a probe in a room may not necessarily directly see a probe out of the window, nonetheless the transition from the first to the second makes perfect sense. More generally, view planning does not even guarantee that nearby probes are in general mutually visible. A smarter choice would be connect probes that see a common point in the surface. Even though this strategy would perform much better, it is easy to see that a number of common cases break the same rule (see, e.g., the same example in Fig. 7.5). We have thus decided to *connect two probes if there exists a point in space that is visible from both of them*. which generally leads to a meaningful set of arcs (see Figure 7.5 right and accompanying video). The technique for finding smooth feasible paths is explained in Sec. 7.3.2.

7.3 Efficient GPU Implementation

The different phases of ExploreMap construction have been mapped to an efficient and robust GPU implementation, whose major components are probe placement

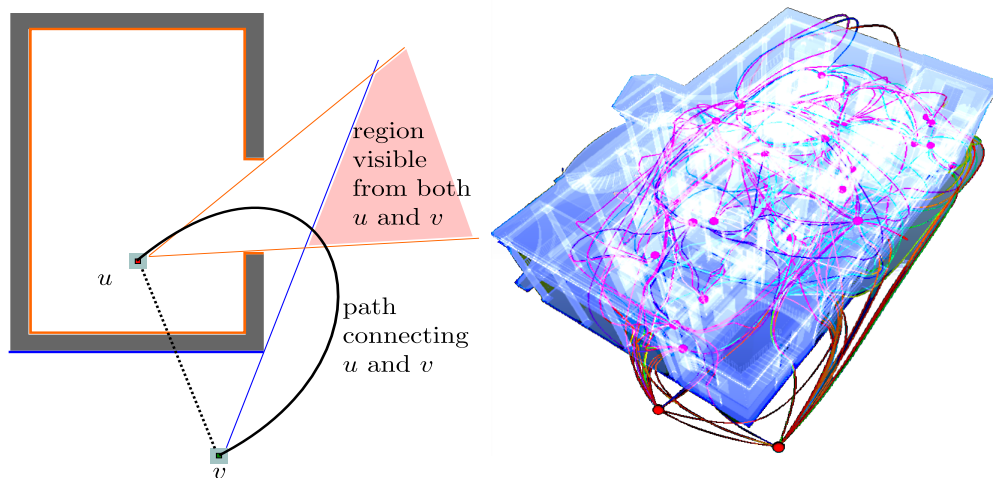


Figure 7.5: Connecting probes. Left: Two probes are connected if there exists a point in space that is visible from both of them. Right: the path graph found for the German house example.

(Sec. 7.3.1) and path generation (Sec. 7.3.2).

7.3.1 Probe placement

The structure of the GPU implementation of our exploration algorithm is illustrated Fig. 7.6. Although the GPU algorithm follows the scheme presented in Listing 7.1 and Listing 7.2, there are several technicalities that need to be addressed.

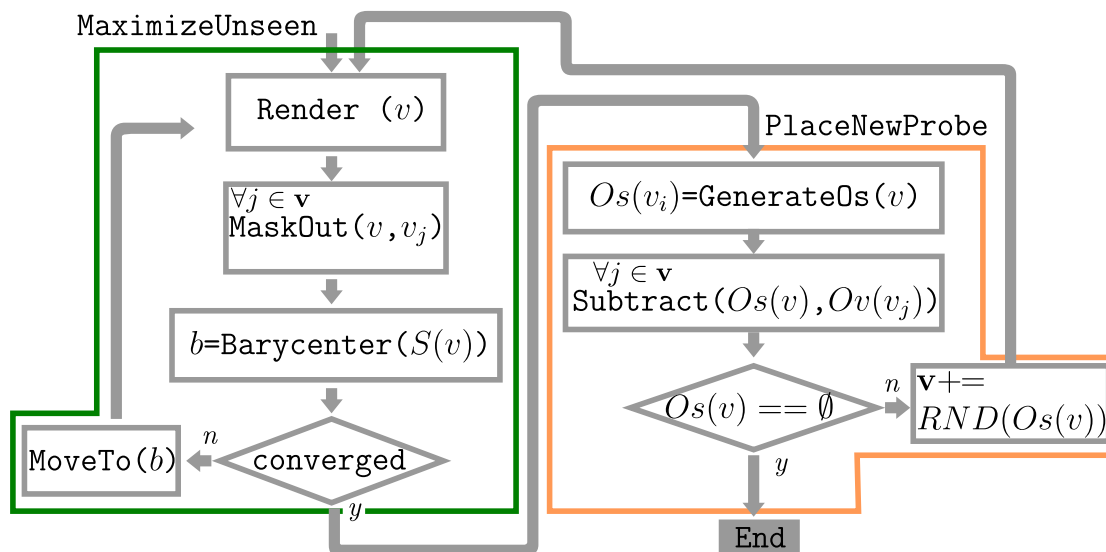


Figure 7.6: GPU algorithm. Flowchart of the GPU implementation of our Virtual Exploration algorithm.

A panoramic view is composed of 6 renderings from the probe position so as to create an environment cube map, and the surface $S(v)$ is the set of 3D points obtained by unprojecting all the fragments belonging to the renderings in

v . In the following, when we refer to operations on the (panoramic) views, the intended meaning is to all their 6 sub-views.

7.3.1.1 Render

The function $\text{Render}(v)$ performs the rendering of the scene for each view with a *render-to-texture* (RTT) approach, producing a *depth map* $D(v)$ and a *normal map* $N(v)$ of the acquired surface. These two maps are then analyzed with a full-screen quad pass to generate a *discontinuity* (or *jump*) map $J(v)$, a map of boolean values where *true* indicates the border of the occluded surface $O_s(v)$. We perform a full-screen pass to find out in parallel, for each acquired pixel, if it is on a discontinuity by testing if its projection in world space is more distant than a threshold τ_{pln} from the plane defined by its neighbor pixels and their respective normals. This is slightly more sophisticated than just testing the difference in the depth map or the distance in world space, for it takes into account the local orientation of the surface with respect to the view direction, as shown in Fig. 7.7.

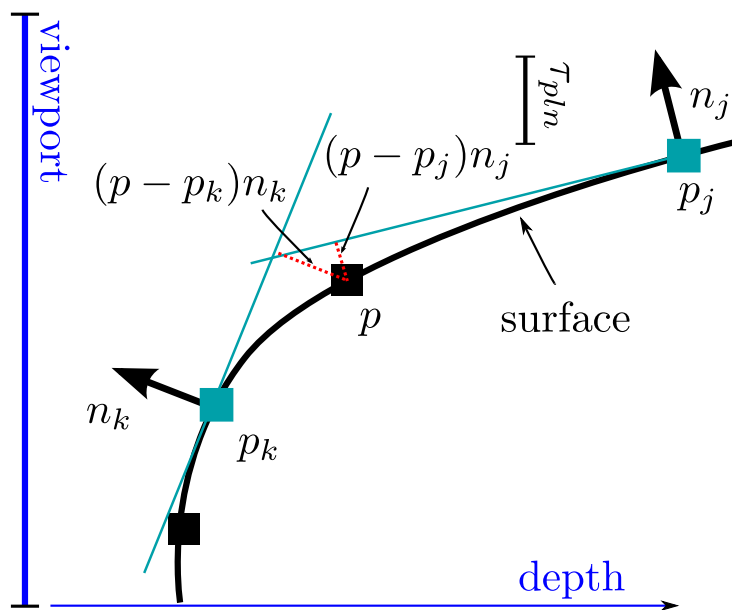


Figure 7.7: Discontinuity detection. The sample p is considered on a discontinuity if it is farther than τ_{pln} from any of the planes built with its adjacent samples positions and normals.

7.3.1.2 MaskOut

Once $D(v)$, $N(v)$ and $J(v)$ have been produced, we need to find out how much of the acquired scene surface is seen *only* by the new probe. This is done by using a shadow map-like approach, treating the surface acquired by all previous probes as a shadow caster on the new, shadow receiving probe. The test, referred to as $\text{MaskOut}(v, v_j)$, is asymmetric, meaning that it only determines which fragments of $S(v)$ will be lost in the comparison against v_j . This phase starts by

generating a *mask* map $M(v)$ where all pixels are set as valid, e.g., not masked. Then, with a series of full-screen passes, we try to invalidate the pixels belonging to already acquired surfaces. For this purpose, we use the depth map $D(v_j)$ of previous probes as a shadow buffer and perform the visibility test with the corresponding pixel of the current probe in $D(v)$. Note that since each view consists of 6 rendering, $\text{MaskOut}(v, v_j)$ would require, in a trivial implementation, 36 renderings to be completed. We optimize the operation by skipping non-intersecting frustum pairs.

7.3.1.3 Barycenter

To compute the barycenter of non-masked parts of the scene seen by the probe ($\text{Barycenter}(S(v))$), we consider $D(v)$ as a range map whose regular tessellation form the acquired scene surface. A full-scene pass is used to generate a *barycenter* map $B(v)$ that will contain, at each pixel (x, y) , the barycenter of a surface quad obtained tessellating the samples $(x + \delta_x, y + \delta_y)$ $\delta_x, \delta_y = 0, 1$ of $D(v)$. Note that only the newly seen surface must contribute to the barycenter computation, and quads over discontinuities must not be taken into account (they are no other than the occlusion surface of the view). To this purpose, we access the discontinuity map $J(v)$ and the mask map $M(v)$ to ignore the contribution of quads that contain at least one vertex classified as a discontinuity or masked out. At the end of this process, every pixel in $B(v)$ will contain its associated, area-weighted barycenter b_q , or zero if it has been rejected due to discontinuities or masking. Calculating the barycenter of the acquired surface then consists of summing all values in $B(v)$: this is accomplished by applying a parallel *reduction* on $B(v)$ with the sum operator, easily done by generating mipmap pyramids. The barycenters of the six views, thus contained in the top-level 1×1 pixel maps, are then read back on the CPU and summed to produce the final probe barycenter.

7.3.1.4 Converged

Convergence is tested using the relative length of the movement from the current probe position to its barycenter and variation of acquired area. When one of these quantities is below a given threshold (or a max number of iterations has been reached), the probe terminates its exploration.

7.3.1.5 MoveTo

As explained, moving the position p of the probe towards the barycenter b ($\text{MoveTo}(b)$) requires to test that the segment connecting p and b does not intersect the surface, i.e., they are in line of sight from each other. Luckily, this test

is trivial and only consists of projecting b on the depth map $D(v)$ and check if it is visible or in shadow.

7.3.1.6 Placing a new probe

The first step ($\text{GenerateOs}(v)$) consists of generating a sampling of the occluded surface $O_s(v)$. This is done by explicitly generating a tessellation of the occluded surface by means of a geometry shader. We issue the rendering of a regularly tessellated grid with vertices associated to samples in the map of discontinuities $J(v)$. If at least one of the is a jump, the geometry shader outputs all 3 vertices by reading the depth values from $D(v)$ and unprojecting them in world space, otherwise the input triangle is discarded. These surviving triangles are no other than a tessellation of the occluded surface $O_s(v)$. The output vertex stream is recorded by the transform feedback into a buffer that is then read back in CPU memory, where triangles are densely sampled into a texture $T(v)$. With the hardware used in our tests, this proved to be faster than using tessellation and geometry shaders to generate samples directly on the GPU. The second step, ($\text{Subtract}(O_s(v), O_v(v_j))$), consists of erasing all the samples outside the occluded volume, i.e., that are visible by at least one of the previous probes. Again, this is accomplished with a shadow map-like approach: we first test samples visibility using the depth map of previous probes, then use a geometry shader and the transform feedback to compact the samples list by eliminating visible ones. After reading back the samples contained in the transform feedback buffer, they are added to the occluded surface samples list L . The starting position of a new probe is then selected by randomly picking a sample from L . If L is empty, then the whole occlusion volume has been carved and the set of probes generated so far is *complete*.

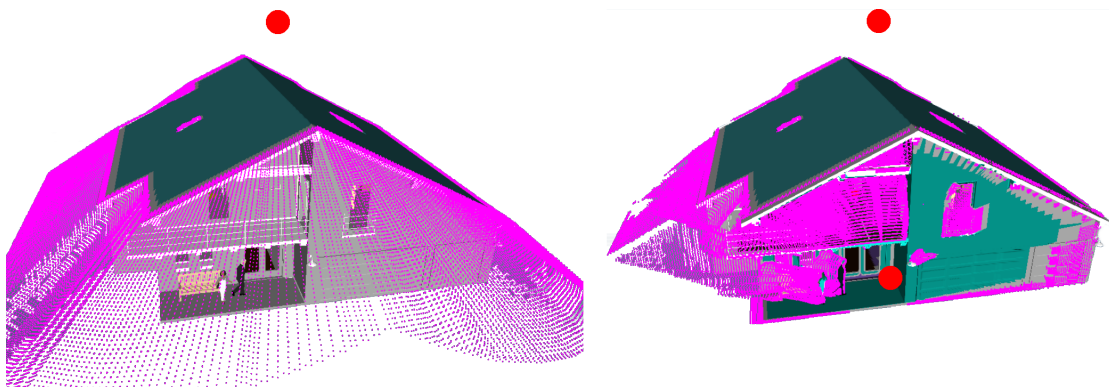


Figure 7.8: Exploring the geometry. Sampling of the occluded surface after the first probe is positioned (left, shown as a red sphere), and after 7 probes (right).

Figure 7.8 shows probe positions (red spheres), surface seen (dark green), and sampling of the occluded surface (magenta) at two steps of our virtual exploration algorithm.

```
bool FindPath(v,u){
    v_curr = v;
    do{
        SetCamerasFrom(v_curr);
        RenderSceneOffset();
        EnableOcclusionQuery();
        RenderOffset(DM(u));
        nfrag = FinishOcclusionQuery();
        if (nfrag > 0) {
            p = ClosestTo_u();
            if (p == u) v_curr = u;
            else v_curr += delta * Normalize(p-v_curr);
        }
    } while( (v_curr != v) && (v_curr != u));
    return (v_curr == u);
}
```

Listing 7.3: Finding a path between two probes.

7.3.2 Path generation

In order to check if there is a point that sees two probes v and u , and to define a path connecting them, we adopt the following GPU accelerated strategy (see Listing 7.3). From the probe v , we render the scene, then enable hardware occlusion queries and render a tessellation of the depth map of u ($DM(u)$), assigning to each vertex of the tessellation its distance from u . If at least a fragment of the tessellation is written in the buffer, the corresponding 3D point can be seen both from u (because part of the depth map tessellation) and from v (because the tessellation is rendered from v). We thus know that the two must be connected. In order to find a path, we can then move v by a fixed step towards the common point closest to u and iterate the algorithm, as shown in Fig. 7.9.

A very important detail is that each rendering of the scene must be done with an offset, in order to prevent the path being closer to the surface than the value of the near plane used when building panoramic views. We do this by using screen space impostors. Let r be value for the near plane. We first render the scene, then for each fragment we issue a quad with side r in world space, which will be used to ray-cast a sphere of the same radius in the fragment shader.

The path found with this algorithm could be already used for computing the transition videos. In order to make it smoother while preventing interpenetration with the scene without drastically changing the path, we finally optimize the

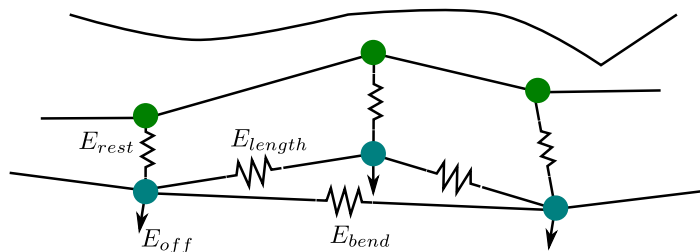


Figure 7.9: Path optimization. The path is smoothed by minimizing a set of energies: E_{rest} , E_{length} and E_{bend} are strings connecting path nodes and E_{off} is a repulsion force aiming to avoid interpenetration with the scene.

path by solving a energy minimization problem:

$$\min E_{rest} + E_{length} + E_{bend} + E_{off}$$

implemented with a mass-spring system. While the first three energy terms are simple springs connecting the path's nodes as shown in Fig. 7.9, the component E_{off} is introduced to prevent the optimized path to get too close to the surface, and it is calculated by rendering the scene from the node position, computing a repulsion force for each fragment (proportional to the inverse of square distance) and summing up to obtain a global vector force. It should be noted that we could further include a more sophisticated perceptual metric here, in order to optimize the panoramic images seen during the path. This is not currently included in our system, since we are focusing on short paths connecting probes that already meet quality constraints. Fig. 7.5 right shows the path graph found for the German house example.

7.3.3 Creating a data-set

Once we have probes and paths, we can assemble the final data-set, which consists of one panoramic image and one thumbnail image for each probe and one panoramic video for each path. These are created on a render farm using the shaded model and an external photo-realistic renderer, which renders all probes and video frames as cube maps, and all thumbnails as perspective images. Path timings are computed using a slow-in/slow-out travel strategy (in this paper, 4s for traversing the model in all presented results). Finally, an index file describing the graph structure is generated.

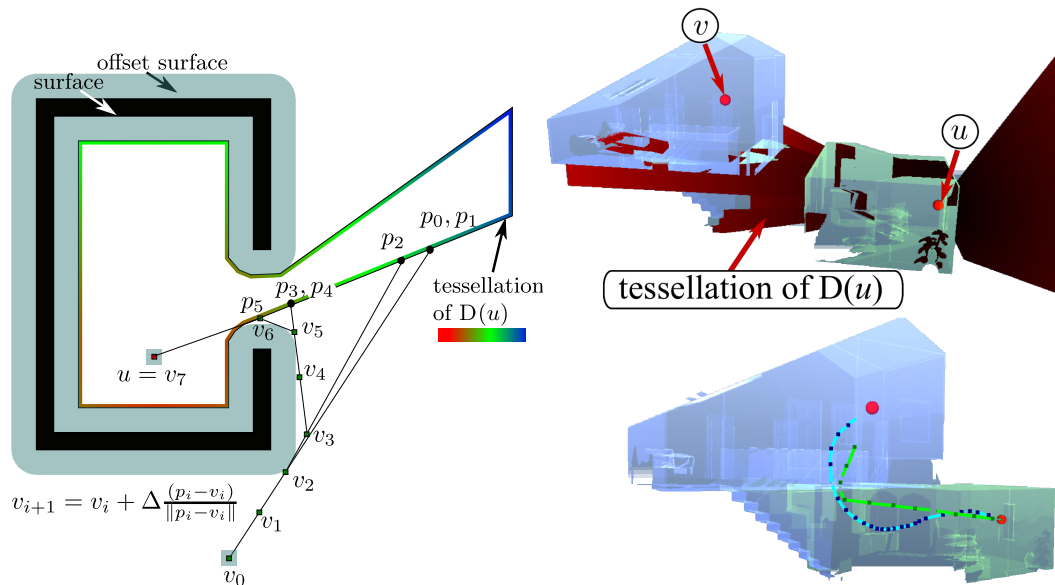


Figure 7.10: Connecting probes. (Left) Choosing the probes to connect with an arc. p is the closest point to u that is also visible from v , therefore the algorithm moves v towards p . The squares around v_i and u represent the near planes. (Right) A real case: on the left the tessellated depth map, on the right the path (green), and the smoothed path (blue).

7.3.4 Pre-fetching behavior

While the user is looking around and zooming, we pre-fetch transition videos of all paths leaving the probe. The current version loads all videos in parallel. Given the client-server implementation, it would also be possible to assign to each path leaving a probe a pre-fetching priority, based on prior choices of people having visited the data-set.

7.4 Implementation and Results

We have implemented a prototype system based on the design previously discussed in this chapter.

The proposed method has been used to develop a complete prototype system, using C++/OpenGL for pre-processing, Blender 2.68a as external rendering engine, and Apache2 for web serving. The client application, which is described together with our graph-based navigation approach in Chapter 10, has been written in JavaScript using WebGL and HTML5. It is able to deliver results in a HTML5 canvas running in WebGL-enabled browsers (Chrome 30 in this paper).

7.4.1 Test models

In order to evaluate our pipeline, we downloaded several models from public repositories (Trimble 3D Warehouse, Archive3D). These websites show a few

views of each model so that users can judge if they are interested in downloading it. Therefore, these sites are a perfect example of how the ExploreMaps could be used for a higher quality browsing of 3D models. Results discussed in this paper are for the models presented in Table 7.1, which have been selected to be representative of various model kinds, featuring complex illumination and/or geometry. In particular, Museum, Sponza, and Sibenik contain complex material and lighting descriptions, including global illumination and participating media, Neptune is a typical 3D scanning result, while the others feature difficult case for general planning methods (e.g., underground tunnels and complex multi-room environments).

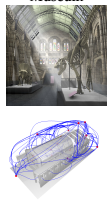
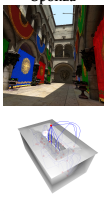
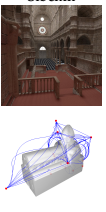
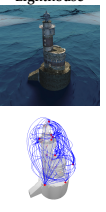
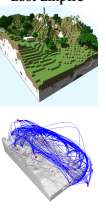
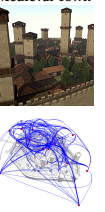
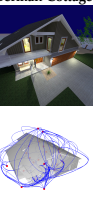
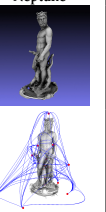
	Museum	Sponza	Sibenik	Lighthouse	Lost Empire	Medieval Town	German Cottage	Neptune
								
Input #tri	1,468,140	262,267	69,853	48,940	157,136	14,865	79,400	2,227,359
Output #probes #clusters #paths	70 17 127	36 10 29	92 21 58	57 17 81	74 25 206	78 30 222	140 23 102	79 19 93
Time (s) Exploration Clustering Synthesis Path Path smoothing Thumbn. Thumbn. pos Total	154 17 144 7 3,012 11 2 3,347	23 3 35 1 122 3 2 189	63 27 449 31 81 7 1 659	15 8 453 12 89 5 1 583	41 13 264 22 482 8 4 854	34 14 395 80 199 10 4 736	163 118 427 23 185 7 2 925	38 14 279 13 150 6 1 501
Storage (MB) Probes Paths	59 248	28 146	72 113	59 159	86 371	103 376	79 390	43 120

Table 7.1: Explore Maps pre-processing results. Selected input data-sets and associated processing statistics. Probes are stored as 6x1024x1024 JPEG images, while paths are stored as 6x256x256@25fps webm videos. The numbers are for the graph instance used for the accompanying video. We verified that the number of probes, clusters, and transitions are only marginally influenced by the randomized initialization by processing the same models 10 times with different initialization. The Hausdorff distance between the probes of all the pair of graphs is 0.9% of the diagonal of the bounding box, the maximum distance is 10% and the number of probes varies in an interval of 10% around the expected value.

7.4.2 Pre-processing

Table 7.1 shows the performance of the pre-processing phase for the test models on a PC with Windows 7 64-bit OS, Intel i7 @3GHz CPU, 12GB RAM, equipped with a NVIDIA GTX 670. First of all, we ran an experiment to prove that the number of probes, clusters, and transitions highly depend on the shape of the scene and are only marginally influenced by the randomized initialization. We processed the same model 10 times and then computed the Hausdorff distance between the probes of all the pair of graphs, obtaining that the average distance between two graphs for the same model is 0.9% of the diagonal of the bounding

box, the maximum distance is 10% and the number of probes varies in an interval of 10% around the expected value. As expected, for instance, a scene like Sponza needs less probes than the Medieval Town, because there are less separated spaces. The important thing, however, is that the processing time for each phase is linear with respect to the size of the elements involved. Thus, the time for Virtual Exploration is linear in the number of probes, the time for the synthesis is linear in the number of clusters, and the times for path determination and smoothing are linear in the number of paths. The number of paths is generally much higher for urban models, because it is more likely that two probes are visible from a common point, while it is smaller for closed environments like the German house. The synthesis task requires the acquisition and evaluation of nearby locations to estimate viewpoint stability, thus requiring more renderings. Also, finding and smoothing the paths are time consuming tasks, especially smoothing, which runs a local search optimization and requires 6 renderings for each evaluation of the objective function. However, note that our pipeline processed each of these models in times ranging from about 3 to a little less than an hour. It is interesting to observe that the biggest model in terms of polygon count was also one of the fastest to process. This happens because time depends on how complex the scene is, i.e., how many probes and paths do we need to see it all. Moreover, note that these phases may be easily distributed. More specifically, the synthesis only concerns a single cluster and path related operation involve pairs of probes. The dominant time of processing thus ends up being the rendering of panoramas and panoramic videos, which, as for movies, can be parallelized on a render cluster. In this paper, we used 32 8-core PCs, for rendering times ranging from 40 minutes to 7 hours/model.

7.5 Discussion

Our main contribution is an automated GPU-accelerated technique for transforming general renderable scenes into a panoramic view graph representation, exploited for creating automatic scene indexes and movie previews of complex scenes, as well as for supporting interactive exploration through a low-DOF assisted navigation interface (see Chapter 10). Real-time performance is achieved on WebGL-based environments even on low-powered mobile devices.

There are many potential applications for the ExploreMaps. In particular, we aim to open the way to a richer experience in presenting 3D models on dedicated web sites, no more limited to few still images or very constrained orbiting interfaces. Furthermore we can turn construction CAD into navigable previews for presentation to stakeholders/potential owners. Thanks to our

unattended processing pipeline, we envisage the implementation of a public web service allowing users to upload 3D models and make them browsable in WebGL-enabled browsers, making for 3D models what Flickr/YouTube made for images and videos.

Advantages. The proposed technique provides an automated pipeline for creating 3D explorable representations of general renderable scenes supporting complex lighting simulation. Limiting camera freedom allows overcoming real-time constraints and exploiting state-of-the-art rendering systems to precompute high quality renderings for interactive exploration. The scene can be interactively explored using image-based techniques and constrained navigation to inspect the static panoramic views and to navigate through connected points of view through a low-DOF interface.

Limitations. Admittedly, there are still a few limitations that need to be addressed. On the construction side, the sampling rate (that is, the viewport size of the cameras during the exploration phase), is fixed and inferred from the scene bounding box. For models where there are interesting details at very different scales (teapot in a stadium problem), the sampling rate should be made adaptive. Furthermore, shading information is not taken into account in the placement of probes, while it could be important for certain models, especially for the selection of best views. Currently, the proposed system considers a binary visibility model: if a surface exists, and is not fully transparent, it acts as a blocker. Semi-transparency handling could be included by considering the degree of opacity of a path during the path construction phase, penalizing paths that go through semitransparent surfaces with respect to paths going through free space. Domain-specific knowledge could also be incorporated in the system, e.g., to achieve human-scale exploration in architectural models. The incremental nature of our probe placement technique should also be exploited to let users optionally provide user-defined set of probes, e.g., to force inclusion of semantically relevant/nice shots in the graph.

Scalability. The visualization just requires correct panoramic view rendering, thus being suitable for most visualization platforms supporting texture mapping. The use of standard image and video data encoding gives a very wide range of configuration opportunities for tailoring quality and performance for the target platform.

7.6 Bibliographical Notes

Most of the contents of this chapter regarding the rendering of complex 3D models using image-based rendering is based on paper [Di B 14], where we presented an approach for rendering complex 3D scenes with complex illumination by computing a set of best views from the input model and letting the user navigate a graph-based representation of the scene through precomputed paths connecting the set of selected views. This is a joint work between CRS4 and ISTI-CNR. My contribution in this work is mainly on designing and implementing the navigation interface, which is presented in Chapter 10.

Part III

Assisted Exploration of Complex 3D Models

The exploration of complex 3D models, which typically contain information at multiple scales, requires the user to be able to explore the 3D model from a wide range of viewing distances. Thus, navigation interfaces must provide the means for covering both global shape inspection and proximal navigation for exploring fine details very close to the surface. Free movement in such complex environments may easily cause the user to lose the spatial context.

In this chapter, we present some approaches for interactive exploration of complex 3D models that exploit constraining the camera movement in order to provide guided navigation, allowing the user to concentrate on the 3D virtual object instead of the navigation itself.

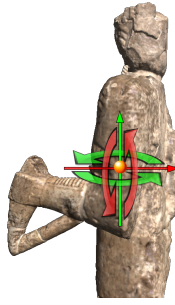
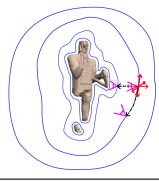
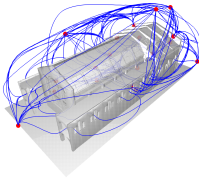
Movement Type	Approach	Method	Published in	Features
Free	<ul style="list-style-type: none"> - Automatic pivot placement - Pivot computed from stochastic sampling of visible geometry - Image-assisted exploration See Chapter 8		Web3D'14	<ul style="list-style-type: none"> - Natural automatic pivot placement - Support for disconnected surfaces - No precise clicking required - Tested on small screen devices and large display setups
Constrained to surface	<ul style="list-style-type: none"> - Isosurface navigation - Implicit distance-field representation - Image-assisted exploration See Chapter 9		JOCCH'14	<ul style="list-style-type: none"> - Smooth transition between orbital/proximal navigation - Natural guided navigation - Ensuring good views - Tested on large display setups
Graph-based	<ul style="list-style-type: none"> - Graph-based navigation - Image-assisted exploration See Chapter 10		EG'14	<ul style="list-style-type: none"> - Interactive exploration of complex environments on low-profile platforms, i.e., mobile and web platforms - Supports complex lighting simulation through precomputed visualization

Table 7.2: Interactive exploration methods.

CHAPTER

8

.....

HuMoRS: Huge models Mobile Rendering Sys- tem

Exploring complex 3D models imposes the need of a navigation interface which allows the user to navigate through the 3D model both from far views, for inferring the global shape and function of the object, and from very close views, to be able to analyze fine details that can be present on the surface of the object (i.e., carvings). In order to address the problem of complex 3D model exploration, we start with a free movement camera controller with automatic pivot definition in order to simplify the interaction. In particular, this greatly improves the usability on small screen devices where the task of pivot selection is difficult due to precise picking limitations. In addition, by automatically computing the interaction pivot, our approach provides smooth and continuous navigation between far and close views.

8.1 Introduction

THE increased availability and performance of mobile graphics terminals, including smartphones and tablets with high resolution screens and powerful GPUs, combined with the increased availability of high-speed mobile data connections, is opening the door to a variety of networked graphics applications. In this world, native apps or mobile sites coexist to reach the goal of providing us access to a wealth of multimedia information while we are on the move.

In the sector of cultural heritage, for instance, most applications of mobile devices are, however, focused on providing aids for navigation inside museal spaces [Fili 11, Rubi 13] or to substitute the audio guides with enriched 2D multimedia content, and they do not provide support for the inspection of 3D artworks, such as sculptures or bas-reliefs. In this context, for museum promotion,

it would be of great interest to provide tools for planning the visit of statue collections, allowing potential visitors to explore with the now ubiquitous tablets or smartphones the detailed 3D digital representation of artworks. To serve this goal, networked, user-friendly, flexible, scalable systems are required, and many challenges need to be addressed in parallel [Kuf11]. Of particular importance is the need to present information at the highest possible visual quality, in order to convey as much as possible the aura of the original artifact.



Figure 8.1: Remote inspection. Remote virtual exploration of a scene composed by several high resolution statues on a Asus TF201 tablet (left image), and on a LG Nexus 4 smartphone (right image). During interaction, models are adaptively downloaded from the network, and knowledge of the currently rendered scene is exploited to automatically center a rotation pivot for the camera controller and to propose context-dependent precomputed viewpoints.

In this chapter, we present a networked framework, dubbed *HuMoRS*, for streaming and interactive exploring huge highly detailed surface models on mobile platforms. In our approach, 3D models and associated information are stored in a web server and streamed in real-time on mobile devices (tablets and smartphones). The mobile client, implemented as an Android app, is controlled by a simple user interface, which combines an interactive camera controller, to incrementally explore the 3D model, with an interactive point-of-interest selector (see Fig. 8.1). During interaction, the camera controller exploits knowledge of the currently rendered scene to automatically center the trackball pivot and to propose context-dependent precomputed viewpoints in the neighborhood of the current view.

Our main contributions are the context-dependent camera controller and the point-of-interest selector, which are able to take context-based decisions based on an adaptive structure maintained on the mobile device and streamed from the web server. Our simple 3D camera controller extends the Two Axis Valuator Trackball [Chen 88, Zhao 11] with automatic pivoting, i.e, an automatic way to determine a good center of rotation based on the current view. In this way, we obtain a user interface for inspecting complex objects which is general, predictable, robust, scalable, smooth, and intuitive.

Our scalable implementation, relying on the approach presented in Chapter 5, supports giga-triangle-sized scenes composed by different models and hundreds of points-of-interest on mobile platforms, including middle-level smartphones and tablets. Moreover, we demonstrate the effectiveness of the proposed interface with a preliminary user-study comparing the time performances with respect to the most typical Virtual Trackball implementations, with and without pivot positioning.

8.2 System overview

We designed the HuMoRS system architecture by integrating the following components: a pre-processing component which builds multiresolution databases starting from high resolution triangle meshes, a web server for storing and streaming 3D models and associated information, and a mobile client integrating an adaptive multiresolution renderer and a simple and effective user-interface (see section 8.3).

8.2.1 Model preparation

In order to ensure real-time performance on large data-sets presented at high resolution on mobile devices, we rely on the visualization technique presented in Chapter 5, which builds a multiresolution representation of the 3D model suitable for adaptive streaming and rendering.

Additional information is stored in the server in order to enable image-based navigation. Through a manual authoring process the user defines a set of interesting views on the model, and a thumbnail of the viewpoint is generated and stored together with the associated view matrix.

8.2.2 Client-server architecture

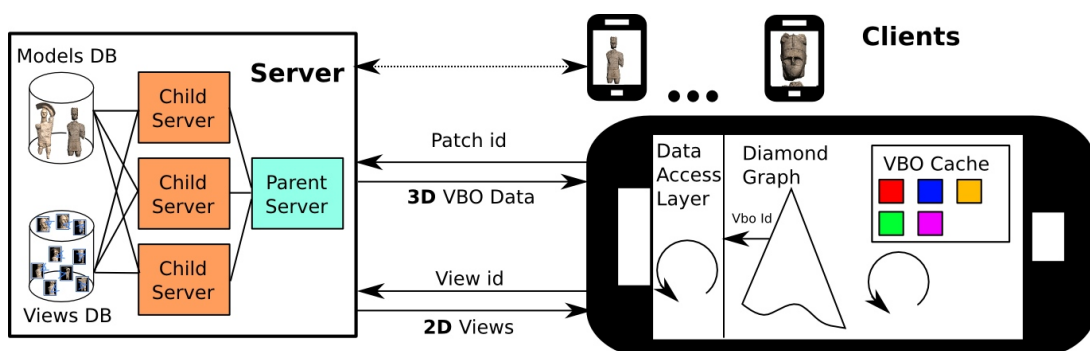


Figure 8.2: Client-server architecture.

The HuMoRS networked communication system is composed by the following components: a server for storing the models databases and related information, and a client for interactive rendering (see Fig. 8.2).

On the server side there is an Apache2 module for handling HTTP requests, which is built upon a local database for efficiently data retrieval. We use Berkeley DB for storage, accessing and caching data in the server side due to its open source license and its matureness as embeddable database. This database is exploited both for streaming the tetrahedra geometry for the rendering, and the views and associated snapshots for the image-based navigation.

On the client, an Android native application, the communication is handled through an abstraction layer relying on the CURL library for efficient communication under HTTP 1.1 protocol. Rendering is performed as described in Chapter 5, maintaining in RAM memory a cache of tetrahedra compact geometries which is adaptively refined in order to satisfy the visualization constraints (i.e., camera position and projected error threshold).

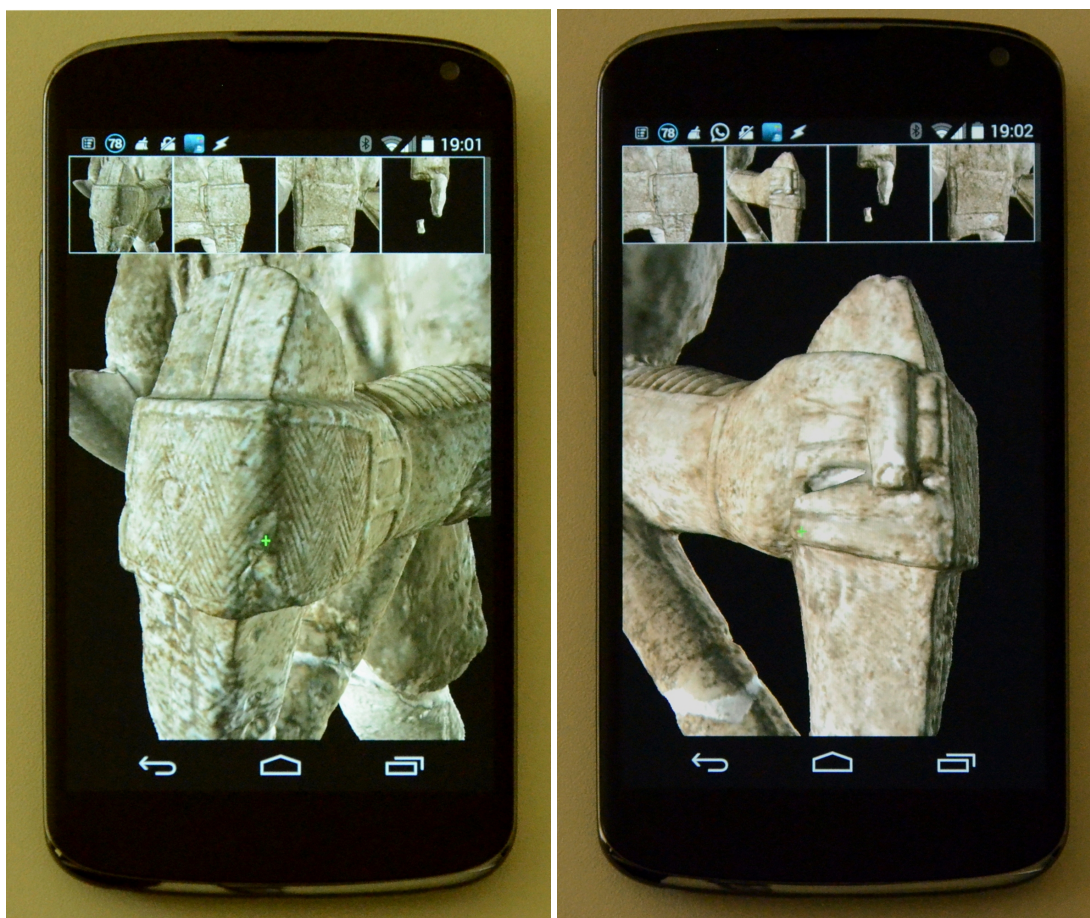


Figure 8.3: Detail of a model interactively rendered on a Nexus 4 smartphone. This 70Mtriangles model is colored using post-restoration color data. Note how our compression preserves extremely high quality details in shape, normal, and colors.

8.3 User interaction

The design of our method has taken into account requirements gathered from domain experts (see Sec. 2.2), as well as our analysis of related work (see Sec. 3).

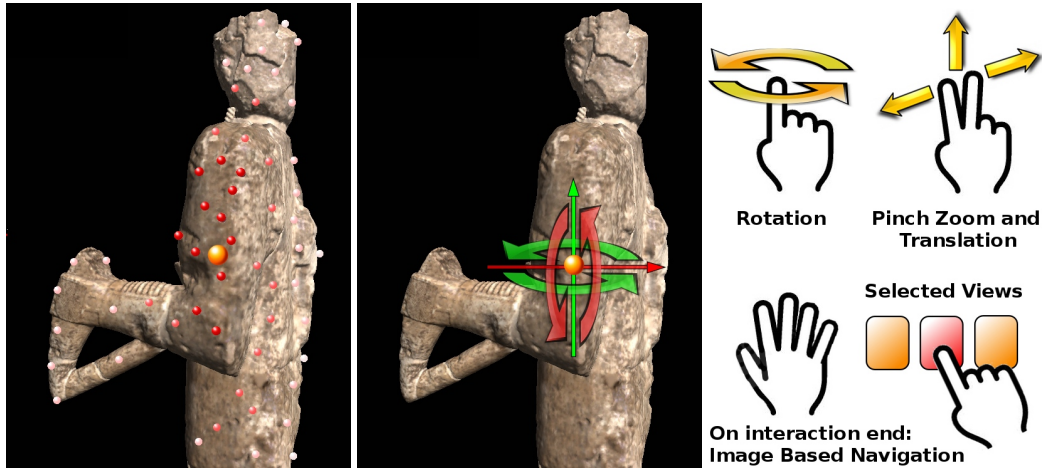


Figure 8.4: Auto-centering and Interaction. Left: automatic pivot computation performed by a weighted averaging of a uniform point sampling of the model in that moment observed by the camera (lighter samples have lower weights). Center: rotations are performed by dragging and mapping displacements according to the classical Virtual Trackball. Right: interaction gestures: one finger for rotation, two fingers for pinch zoom and translation. When interaction stops a set of precomputed views is presented to the user.

Since we deal with decorated and highly detailed cultural heritage objects, like statues and bas-reliefs, we had to take into account the fact that they present information at multiple scales (global shape and carvings), which could require sub-millimetric model precision. This carving information should be clearly perceivable at all scales, and should be magnified for close inspection. Furthermore, camera navigation should provide real-time feedback, in order to engage users providing them the sense of control, and support smooth and seamless object inspection, going back and forth from shape inspection to detail inspection. The user interface should thus also be perceived as simple and immediately usable. Given the limited size of display, the visualized object should not be obstructed by fingers and other interaction widgets, and finger movements should be limited in order to reduce user's effort during the exploration. Finally, since the application has to work in mobile settings, all context-dependent operations have to be designed so as to work on locally maintained structures, as querying the server for information would introduce too much latency on mobile networks.

To fulfill these requirements, we designed a user interface composed by a simple and effective metaphor (**ACeViT**: Auto-Centering Virtual Trackball) for interactive camera motion, and a context-based point-of-interest selector for moving the camera to precomputed viewpoints, see Fig. 8.4 and Fig. 8.5.

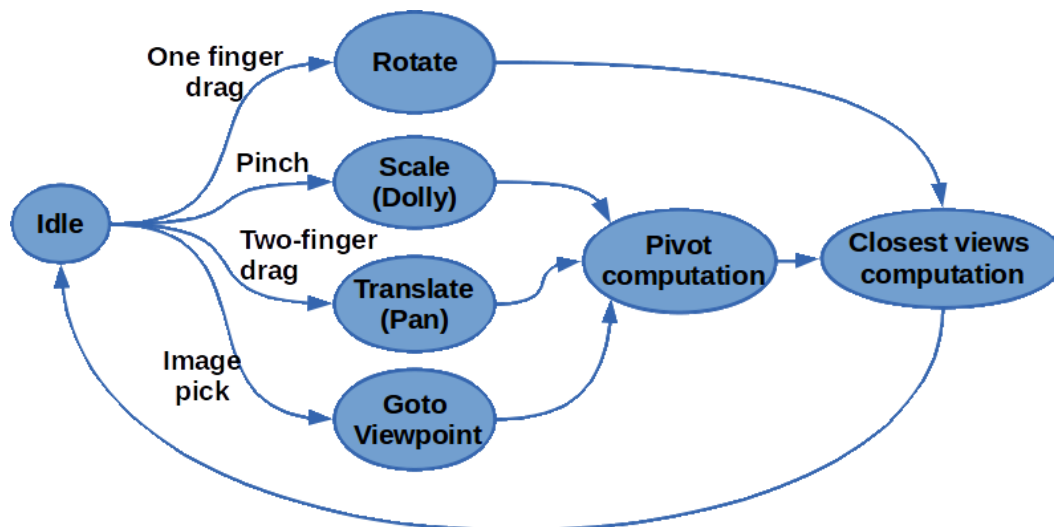


Figure 8.5: Interaction states. This diagram shows the various state transitions that compose the user interface.

8.3.1 Auto-Centering Virtual Trackball

During the last two decades many devices, interfaces and algorithms have been designed and proposed to map user input to virtual camera motions [Chri 09], aiming to maximize user performance and comfort during interactive explorations. The majority of navigation methods are devoted to typical desktop systems, in which user focus is directed to a monitor screen surface, and the input is given by employing 2D pointing devices, like mice, pads or joysticks. In these scenarios in which the work area size is comparable to the view area size, direct interaction methods are normally considered, and many exploration metaphors can be designed to fulfill the system requirements given by the data to explore and the user needs.

The widespread diffusion of touch devices, such as tablets or smartphones, has made people used to touch-based user interfaces based on direct manipulation of on-screen objects. 3D variations of the well-known 2D multi-touch RST technique, that allows the simultaneous control of Rotations, Scaling, and Translations from two finger inputs are becoming increasingly used as viewpoint manipulation technique in this context [Kurt 97, Jank 13]. While no real standard for 3D touch-based interaction exists [Keef 13], touch surfaces are now so common that people are encouraged to try to interact with them using typical 2D gestures, which is an important aspect for reducing training time. Even if the mapping between 2D and 3D motions is non-trivial and varies from a user interface to the next, users are encouraged to learn by trial and error while interacting.

It should be noted, however, that in the case of small mobile touch screens

standard co-located interaction techniques are not always effective due to occlusion problems since fingers can occlude the scene during motion, and simpler incremental techniques which reduce the user input need to be considered. It is therefore important to design the technique so as to avoid the need for precise co-location.

According to previous user analysis [Bade 05], the Two-Axis Valuator [Chen 88] appears to be the best incremental 3D rotation technique in terms of speed and satisfaction in standard settings, and it was considered as the base for our method. This interface maps 2D device motions to two axes of the view coordinate system, both orthogonal to the view-vector of the camera through a center of rotation [Shur 11]. Specifically, horizontal mouse movement Δu is mapped to rotations about the up-vector of the camera and vertical mouse movement Δv is mapped to a rotation about the vector perpendicular to the up-vector and the view-vector (see figure 8.4). Then diagonal device movements are mapped to a combined rotation about both axes. The center of rotation (pivot) can be defined in different ways: traditional trackballs fix the pivot as the center of the object bounding sphere, but many systems give the user the possibility to manually define the rotation center as a 3D point in the object surface. Both solutions can be tedious for users, since in the first case they are forced to many motion corrections, while in the latter they have to change operation mode when they want to select a new pivot position. In our method, users do not have to take care of controlling the rotation center, since it is automatically placed. Whenever a panning or rotation interaction ends (see Fig. 8.5), the pivot position is computed according to the current projection and viewing matrices (P, V) and a stochastic sampling $\Sigma = \{\mathbf{p}_1, \mathbf{p}_2, \dots, \mathbf{p}_N\}$ of current visible points on the surface. The following weighted sum is employed:

$$\mathbf{c} = \sum_i \gamma(u_i, v_i, w_i) \cdot \mathbf{p}_i \quad (8.1)$$

where $(u_i, v_i, w_i) = PV \cdot \mathbf{p}_i$ are the NDC coordinates of the projected point, and $\gamma(u_i, v_i, w_i) = \gamma(u_i) \cdot \gamma(v_i) \cdot \gamma(w_i)$ is a 3D separable Gaussian filter, giving the maximum weight to the current view matrix target position, and lower values when the samples are in the peripheral areas of the viewport. With respect to depth, the Gaussian is centered at the near plane, as to give to closest points the maximum contribution. The random sampling and the distance-based weighting ensure the avoidance of abrupt pivot changes when the user performs multiple small rotations. In order to suit different hardware settings, a variety of implementations can be considered. An efficient GPU-based screen-space implementation can be obtained by performing a stochastic sampling of the depth buffer and weighted accumulation of samples, by employing typical multi-pass

pyramid methods derived from image processing [Stre 06]. A model-based implementation instead can be applied to platforms where fragment processing and GPU feedback are not well supported or have performance issues (e.g., many mobile phones). In our implementation, we perform all computations in CPU and, since the models are rendered using patches (e.g., [Cign 04]), the sampling algorithm is applied to individual patches during hierarchy traversal and results are combined at run-time according to the renderer selection.

8.3.2 Image-based navigation

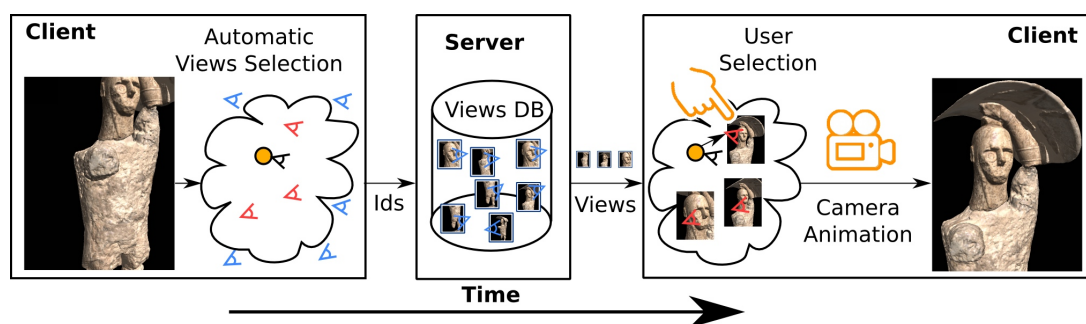


Figure 8.6: Views selection process. When the camera stops, the application identifies the best closest views and posts a request to the server which sends back the selected view thumbnails. Finally the user selects the desired view and an animation moves the camera to the requested position.

We provide the user also with a context-based guided navigation relying on nearest point-of-interest selection. This alternative control method allows the user to explore the object by traveling through a set of previously defined interesting views. On startup, the application loads a list of precomputed view points of the scene, which is organized into a KD-tree in order to provide fast searches. For each view point, a view matrix is stored together with the url of the corresponding view point thumbnail. Whenever an interaction ends, a list of best view candidates is computed and a separate thread is in charge of gathering the associated thumbnails to each view from the server. A small two level LRU cache in the client alleviates the latency of thumbnail retrieval for already visited viewpoints. The first cache is in charge of handling image requests to the server, containing various dozens of compressed images. The second level caches 2D textures, and must handle at least the number of simultaneous views that will be visible at the same time. Computation of best view set is performed in two steps. First, the closest views with respect to current view position are retrieved performing a Knn search. The resulting set of views is further filtered to select the views subset that best approximate the current view point. For that purpose, a uniform point sampling of the visible model is performed. The resulting point

Model	CPU	GPU	RAM	Screen
Nexus 7	Tegra 3 4x1.3Ghz	GForce ULP	1GB	7" 800x1280 0.9MPix
ASUS TF201	Tegra 3 4x1.3Ghz	GForce ULP	1GB	10.1" 1232x800 0.9MPix
Nexus 4	Snapdragon S4 Pro 4x1.5Ghz	Adreno 320	2GB	4.7" 768x1280 0.9MPix
Nexus 5	Snapdragon 800 4x1.5Ghz	Adreno 330	2GB	4.95" 1080x1920 2MPix
Asus TF701	Tegra 4 4x1.9Ghz	GeForce ULP	2GB	10.1" 2560x1600 4MPix

Table 8.1: Test device characteristics. Hardware characteristics of the devices used for testing, with processors from NVIDIA (Tegra) and Qualcomm (Snapdragon).

set is projected both from current view and from the candidate view in order to compute point-wise 2D distances. The views with lowest distances are then selected and presented to the user as a grid selection interface shown on the screen side, see Fig. 8.1.

The user can then navigate the model by clicking on the various thumbnails, thus starting an animation that will take the observer from its current position to the target view point, see Fig. 8.6.

The approach can be furthermore extended with static overlays, for authoring annotations and presenting heterogeneous enriched multimedia information, like required in typical cultural heritage applications [Mart 14].

8.4 Implementation and Results

We have implemented a prototype hardware and software system based on the design previously discussed in this chapter.

The **HuMoRS** system has been successfully tested in a variety of mobile platforms, in particular for the exploration of a set of 3D highly detailed models derived from the 3D scan acquisition of the statues of Mont'e Prama, ancient stone sculptures created by the Nuragic civilization of Sardinia, Italy, see Fig. 8.3, 8.7. The 3D models of these statues are highly detailed and often made of a few disconnected parts, posing important problems to navigation techniques. See Bettio et al. [Bett 13, Bett 14b] for details on the Mont'e Prama dataset.

8.4.1 System performance

The mobile client was implemented on Android 4.4 using C++, OpenGL ES 2.0 and Qt. The Qt library is in charge of handling UI events and GL context creation providing good portability for Android, Windows, Linux or iOS. We evaluated the rendering performance of the system on a number of inspection sequences on a variety of devices that represent both mid-class and top-class SoC(System on Chip) currently available in the market, see Table 8.1 for device characteristics.



Figure 8.7: Various levels of detail of a statue. Images correspond to Nexus 4, Asus TF201 and Nexus 7, respectively. Top: View of the whole statue. Bottom: A close-up of the statue showing small-scale details.

8.4.1.1 Rendering performance

Models were rendered using a target resolution of $0.3\text{tri}/\text{pixel}$ on screens with less than 2Mpix and $0.2\text{tri}/\text{pixel}$ when there are more than 2Mpix , leading to graph cuts of 150 nodes on average, with approximately $8K\text{tri}/\text{node}$. Navigation was interactive with negligible interaction delays for all datasets, with frame rates constantly above 20 fps, for the Tegra 4 and SnapDragon processors, while Tegra 3 handled above 10 fps. The multiresolution structure used for rendering is also exploited for object queries during interaction, proving successful, as camera transformation computation, and pivot calculation, in our camera controller always took below 10% of the frame time. The sessions were designed to be representative of typical mesh inspection tasks and to heavily stress the system, including rotations and rapid changes from overall views to extreme close-ups. The qualitative performance of our adaptive renderer is also illustrated in an accompanying video, that shows live recordings of the analyzed sequences. During the tests the average triangle count was $1\text{-}3\text{Mtris}$ per frame, depending on screen

Device	Rendering	Downloading	Decoding	Whole view refine	Close view refine
ASUS TF201	10 Mtri/s	11.2Mbps	1MB/s	4.6MB, 7s	37MB, 57s
Nexus 7	10 Mtri/s	11.2Mbps	1MB/s	4.6MB, 7s	37MB, 57s
Nexus 4	20 Mtri/s	18Mbps	1.2MB/s	4.6MB, 5s	37MB, 41s
Nexus 5	20 Mtri/s	20Mbps	1.4MB/s	7.4MB, 8s	45MB, 40s
Asus TF701	25 Mtri/s	24.8Mbps	1.5MB/s	8.2MB, 8s	65.8MB, 59s

Table 8.2: Test devices performance. Performance results for the various hardware configurations.

coverage and model refinement. The measured performance shown average frame rates of 10-30 fps on most devices, while Tegra 3 devices performance was about 5-15fps, see Table 8.2. When the user is interacting, the maximum triangle budget is adjusted to ensure a minimum of 10 fps. As demonstrated in the video, performance is perfectly adequate for interactive inspection tasks, while providing extremely high representation quality. An example is presented in Fig. 8.3.

We have also performed tests on a scene composed of 10 statues, ranging from 40Mtri to 70Mtri each, resulting in about 600Mtri, see Fig. 8.1. Performance was measured between 4-10 fps on Tegra 3 devices, and 10-20 fps on the other tested devices. Average triangle count per frame depends mostly on the statue in foreground, while other statues covering less than 1/4 of the viewport add no more than 60Ktri each, representing about 10-30% overhead.

All the measures correspond to a rendering viewport of 5/6th of the screen, excluding the area used for image-based navigation, ranging from 0.7Mpixels to 3.3Mpixels.

8.4.1.2 Streaming performance

The latency time needed to download the data at the application start-up and to refine the model during the exploration is the most critical bottleneck affecting mobile devices, and it is independent from the rendering thread only relying on the network bandwidth. Performance was measured on a wireless connection using a Linksys WAP200 802.11b/g AP 54 Mbps, obtaining a peak performance of 28Mbps under a heavily shared environment.

For a *full view*, where the whole statue fits in the rendering viewport (see Fig. 8.7 top-left), about 4.6-8.2MB (500Ktri-1.3Mtri) were required for full refinement, depending on the rendering viewport resolution. In the case of a *detail view*, a close view of a small part of the statue (see Fig. 8.7 center-bottom), 37-65.8MB (1.6-2.3Mtri) are needed (see Table 8.2 for further details).

Data fetching is performed asynchronously in a separate thread, so it doesn't interfere with interactive rendering performance. Data fetching over Wifi shows

download speeds about 11-26Mbps, representing 40-88% of the available bandwidth. The decoding performance varies among the various devices ranging from 0.9MB/s to 1.5MB/s, thus determining the maximum streaming throughput, see Table 8.2. Achieving a fully refined *full view* of the statue requires about 5-8s depending on the screen resolution and the device performance when decoding. For a *detail view*, between 40-59s are required for full refinement. Nonetheless, thanks to the progressive refinement, within just few seconds the statues can be inspected with a reasonable quality.

Under UMTS/HSPA connection, we measured an average data fetch speed of 2.5Mbps over a measured peak performance of 3.4Mbps. Full refinement of a *full view* took about 20-50s, and 120-140s for a *detail view*.

All this times correspond to a fresh start with no cached data. However, during a typical inspection of the model, when the user gets to a new close-up view most data from the coarse representation is already present, thus requiring less time for full refinement.

8.4.2 User study

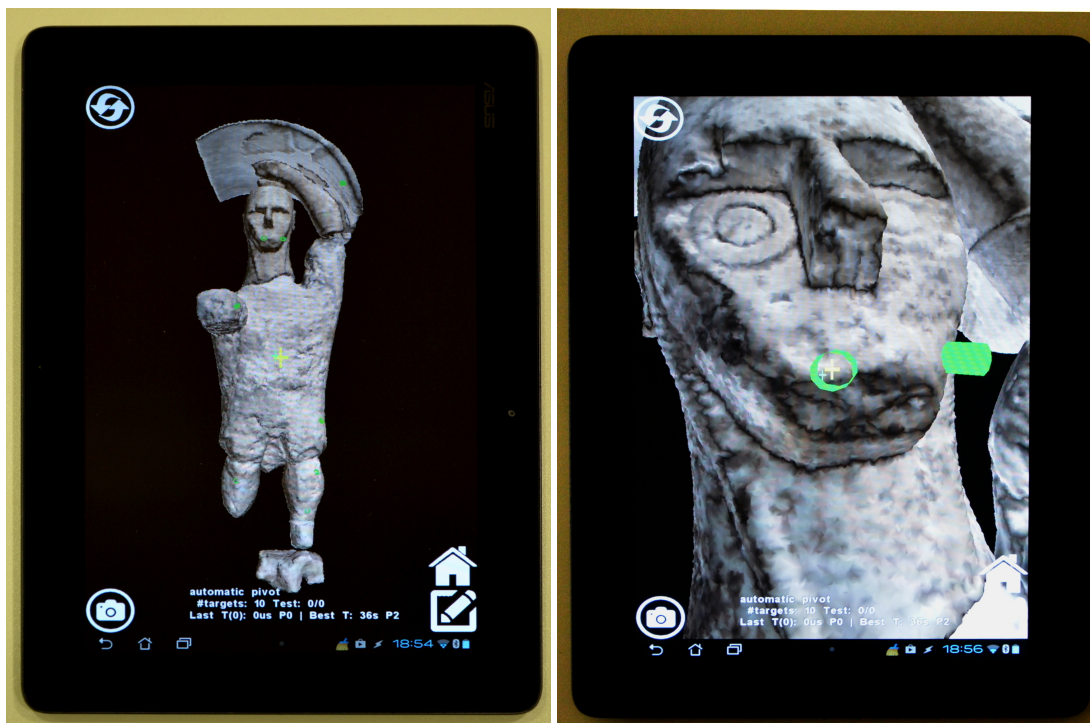


Figure 8.8: User study performed on a Nexus 7 tablet. ACeViT compared to Virtual TrackBall with fixed pivot and with manual pivot positioning. Left: test scene composed of a detailed surface model composed of 70Mtri. Right: tasks consisted of reaching and shooting specific positions and orientations indicated by green cylinders.

In order to test the effectiveness of the interaction method on mobile devices, we carried out also a preliminary performance evaluation of our auto-centering

Virtual Trackball (AceViT) with respect to the standard Two Axis Valuator Virtual Trackball with fixed pivot, and with manual selection of the pivot obtained by ray-casting against the scene on user request.

8.4.2.1 Setup

The user tests were performed on an ASUS Nexus 7, see Table 8.1. All the interfaces were implemented by using the typical RST approach: one finger to generate rotations, two fingers to perform zoom (by pinching) or pan (by dragging), continuous pressure with one finger to perform pivot update. As testing scene, we considered a scene composed of a Boxer statue (see Fig. 8.8) consisting of 70Mtri.

8.4.2.2 Tasks

The experiments consisted in letting users try the three different manipulation controllers (ACeViT, TrackBall with fixed pivot and TrackBall with manual pivot) in the context of a point-and-shoot interaction task [Bade 05]. Participants were asked to point-and-shoot a small set of green cylinders, which had to be shoot through in order to get a positive hit. By forcing the user to align the camera view direction with that of the cylinder axis, we obtained a task composed of a global approaching phase and a later local step for aligning camera with the cylinder. The cylinder radius was adjusted in order to avoid trivial alignments so the second step was hard to skip (see Fig. 8.8 left). When the targets were precisely pointed and aligned with respect to a cross viewfinder, users could shoot them by pressing a touch button (see Fig. 8.8 right).

8.4.2.3 Subjects

Ten subjects with ages ranging from 27 to 51 (mean 38.4 ± 7.9) were selected between the researchers of our research department. All persons involved had previous experience with 3D software systems and interfaces (particularly with Virtual Trackball interfaces).

8.4.2.4 Design

Subjects were proposed the interfaces in randomized order. After a brief training with the touch interface, the measured tests consisted of shooting 5 targets, randomly selected from a list of 10 potential candidates, in order to avoid any bias due to a-priori knowledge of target positions. For a complete testing session, users needed times ranging from 180 to 280 seconds. The times for hitting all tasks for each interface were measured and recorded.

8.4.2.5 Analysis

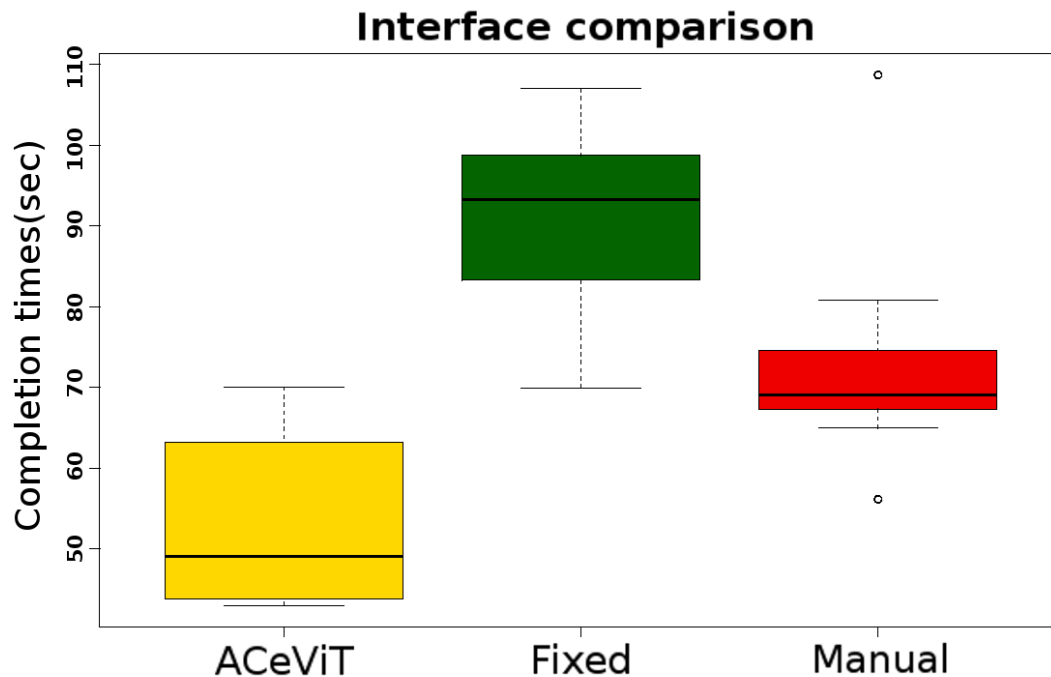


Figure 8.9: Performance comparison: Timings.

In summary, the complete test design consisted of 10 subjects, each one testing 3 camera controllers for a total of 30 time measurements. We performed a statistical analysis of completion times for the shooting tasks experiment. All computations were done by using the *R* package. Mean completion times were 53.68 ± 11.41 seconds for ACeViT, 73.64 ± 14.84 seconds for manual pivoting, and 90.95 ± 12.26 seconds for fixed pivoting. After performing a Shapiro-Wilk test for normality ($W = 0.948$, $p = 0.2101$), an analysis of variance revealed a significant effect of the interface ($F = 17.652$ and $p < 0.001$). Finally a post-hoc Tukey multiple comparisons of means revealed dramatic differences between ACeViT and fixed pivoting ($p < 0.001$), important differences between ACeViT and manual pivoting ($p < 0.01$), and significant differences between manual pivoting and fixed pivoting ($p = 0.029$). Fig. 8.9 shows the boxplots of the task completion times, as rendered by the *R* package. Statistical analysis and boxplots showed that automatic pivoting improves the time performances of shooting tasks. Moreover, by observing and listening think-aloud comments during the experiments, it appeared evident that users felt very comfortable with ACeViT since they could explore in intuitive manner the complex scene, and they could perform tasks with less motion corrections with respect to standard Virtual Trackball implementations, with fixed and manual pivoting.

8.5 Discussion

We have presented *HuMORS*, an interactive system for natural exploration of extremely detailed surface models on mobile devices. The system has been successfully tested in a variety of mobile platforms, in particular for the exploration of a set of 3D highly detailed models obtained with high resolution laser scanning of cultural heritage artworks. Furthermore, our auto centering camera controller has been compared with two consolidated Virtual Trackball implementations, collecting quantitative results from a series of tests on a mobile device involving 10 people. The camera controller appears to be intuitive and simple enough even for casual users who quickly understand how to browse complex models immediately. Our future plans concentrate on extending the system for supporting bidirectional connection between multiple multimedia types as well as narrative contents.

Advantages. The proposed technique provides natural exploration of complex 3D models, in particular highly detailed 3D models with multi-scale information, by providing an automatic rotation pivot. Automatic pivot selection allows the user to explore the 3D model going from global shape views to surface exploration in a continuous navigation without needing to perform precise picking (i.e., changing the interaction mode).

Limitations. Since the pivot is computed using an stochastic sampling of the visible surface, this technique is better suited for the exploration of highly detailed 3D models with fine surface details (e.g., 3D scanned objects).

Scalability. The pivot computation depends on a sampling of the visible surface of the 3D model, thus performance directly depends on the internal 3D object representation. When dealing with simple 3D models (i.e. few thousands of triangles) there is no need for complex data structures and brute-force approaches are perfectly feasible. Paired with efficient multiresolution data structures it has been proved both on high-end desktop systems and on mobile devices, exploring datasets composed of hundreds of millions of triangles.

8.6 Bibliographical Notes

Most of the content of this chapter is based on paper [Bals 14a]. In this paper, we presented a system for interactive exploration of huge 3D mesh models on mobile platforms which presents the user with a selection of close views aiming to facilitate the discovering of information on the various details of the 3D model.

.....

IsoCam: Interactive Visual Exploration of Massive Cultural Heritage Models on Large Projection Setups

The camera controller presented in previous Chapter 8 enables natural free exploration of complex 3D models. Although the method works and provides a relatively easy-to-use interface, novice users easily get lost when having too much freedom in 3D environments. This is particularly true in complex environments requiring multi-scale exploration. In this chapter, we propose a solution, dubbed *IsoCam*, which combines an object-aware interactive camera controller with an interactive point-of-interest selector and is implemented within a scalable implementation based on multiresolution structures shared between the rendering and user interaction subsystems. The collision-free camera controller automatically supports the smooth transition from orbiting to proximal navigation, by exploiting a distance-field representation of the 3D object. The point-of-interest selector exploits a specialized view similarity computation to propose a few nearby easily reachable interesting 3D views from a large database, move the camera to the user-selected point of interest, and provide extra information through overlaid annotations of the target view. Chapter 11 will further expand on how to efficiently combine camera controllers with information presentation to go beyond pure visual exploration. We also demonstrate how this constrained method can be adapted to ensure that nice views are presented in a variety of setups, including demanding light-field displays, which impose severe constraints on the placements of the displayed objects.

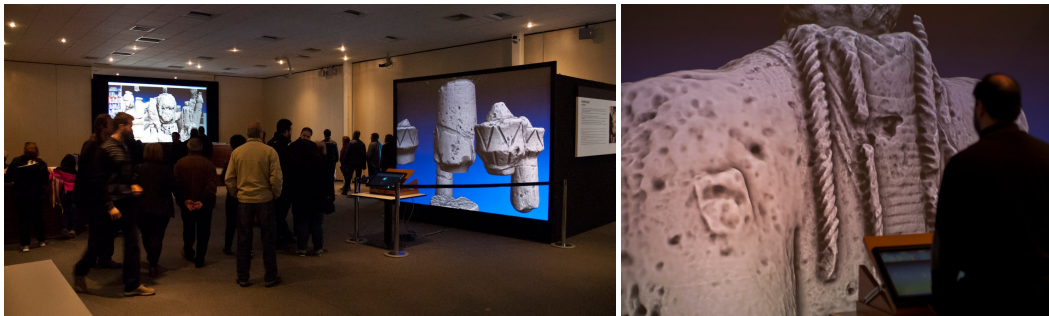


Figure 9.1: Museum exhibition. The method described in this work has been employed for the virtual stands in a large exhibition attended by approximately 3500 people, who had the possibility to freely inspect 38 highly detailed models. A larger setup is now installed in two permanent exhibitions at Archaeological Museums in Cagliari and Cabras (Italy).

9.1 Introduction

NOWADAYS, shape and material acquisition, as well as modeling techniques, are able to produce highly detailed and accurate 3D representations of cultural heritage artifacts. While this digitization process has a variety of applications, including archival, study, and restoration, visual communication is by large the most common utilization. Until recently, the most successful and widespread use of 3D reconstructions for exploration have been through mostly passive visual presentation modalities, such as videos or computer-generated animations. Interest is, however, now shifting towards more flexible active presentation modalities, such as virtual navigation systems, which let users directly drive navigation and inspection of 3D digital artifacts. These active presentation approaches are known to engage museum visitors and enhance the overall visit experience, which tends to be personal, self-motivated, self-paced, and exploratory [Falk 00]. In general, visitors do not want to be overloaded with instructional material, but to receive the relevant information, learn, and have an overall interesting experience. To serve this goal, user-friendly and flexible systems are needed, and many challenges need to be addressed in parallel [Kufli 11].

In our approach, 3D models and associated information are presented on a large projection surface (projection wall). We let users focus their attention exclusively on the large screen, while controlling the application through a touch-enabled surface placed at a suitable distance in front of it (see Fig. 9.1 right). In this setting, the display of the touch-enabled screen is used only to provide minimal help information on the application. We exploit this simple setup with an indirect user interface, dubbed *IsoCam*, which combines an object-aware interactive camera controller, to incrementally explore the 3D model, with an interactive point-of-interest selector. Our camera controller exploits a multiresolution distance-field representation of the 3D object, which allows it to be scalable and to smoothly

transition from orbiting to proximal navigation (see Sec. 9.3). The simple point-of-interest selector exploits a specialized view-similarity computation to propose a few nearby easily reachable interesting 3D views from a large database, move the camera to the user-selected point of interest, and provide extra information through overlaid annotations of the target view (see Sec. 9.4). A scalable implementation is realized on top of specialized multiresolution structures shared between rendering and user interaction subsystems.

Our system combines and extends a number of state-of-the-art results. The main novel contribution is an intuitive 3D camera controller, handling objects with multiple components while smoothly transitioning from orbiting to proximal navigation, integrated with a simple user interface for point-of-interest selection and overlaid information display. Thanks to view-adaptive structures, object-aware user-interface components are capable to support models composed of various billions of primitives and large numbers of points of interest, and have been used by thousands of inexperienced users in a real exhibition with a massive object collection (see Sec. 9.8).

9.2 Overview

The design of our method has taken into account requirements gathered from domain experts and described in Sec. 2.2. In Sec. 9.2.1 we provide a general overview of our approach, justifying the design decisions in relation to the requirements.

9.2.1 Approach

Requirements **R1–R13**, as well as our analysis of related work presented in Chapter 3, were taken as guidelines for our development process, which resulted in the definition of an approach based on indirect touch-based interactive control of a large projection surface through an assisted user-interface. The main components of our user-interface are an object-based assisted system for interactive camera motion tuned for both orbiting and proximal object inspection, and a context-based point-of-interest selector for moving the camera to precomputed viewpoints and display associated information in overlay. Multiresolution technology is used both in the user-interface and the rendering subsystem to cope with performance constraints.

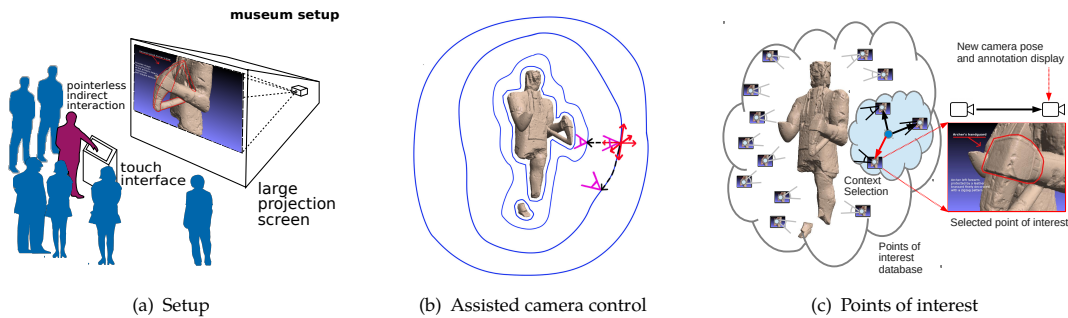


Figure 9.2: Method overview. Users focus on a large projection screen. Object exploration is achieved by moving the camera on isosurfaces of the object’s distance field, and point-of-interest selection is context sensitive and associated to display of auxiliary information.

9.2.2 Proposed setup

The widespread diffusion of touch devices, such as tablets or smartphones, has made people used to touch-based user interfaces based on direct manipulation of on-screen objects (R6). This kind of setting is increasingly being used at a larger scale, also for museum presentations, with users exploring models on touch-enabled monitors, tables, or walls [Hach 13]. Instead of using a large touch screen as input/output device for direct manipulation, we have preferred to decouple the devices for interaction and for rendering as to allow for large projection surfaces and enable multiple users to watch the whole screen without occlusion problems (R2, R7) and staying at a suitable distance from it when viewing large objects of imposing scale (R2). By contrast, using co-location of touch interaction and display would force users to stay within reach of the display, limiting its size. Many embodiments of this concept are possible. In this work, we present a low-cost solution based on a single PC connected to a small touch screen for input and display of help information and a projector illuminating a large screen for application display (see Fig. 9.2(a)). We emphasize that the touch screen does not display any visual feedback, encouraging the users to focus solely on the large display (R8) where the objects of interest are rendered (see Fig. 9.12(a)). Both single- and multi-touch controls have been implemented (see Sec. 9.5).

9.2.3 Camera control

3D variations of the well-known 2D multi-touch RST technique, that allows the simultaneous control of Rotations, Scaling, and Translations from two finger inputs are becoming increasingly used as viewpoint manipulation technique in this context [Kurt 97, Jank 13]. While no real standard for 3D touch-based interaction exists [Keef 13], touch surfaces are now so common that people are

encouraged to immediately start interacting with them, which is an important aspect of *walk-up-and-use* interfaces. Moreover, even if the mapping between 2D and 3D motions is non-trivial and varies for a user interface to the next, users are encouraged to learn by trial and error while interacting. Inspection of complex objects, especially for novice users, can be difficult, as, similarly to virtual trackball approaches, users are continuously shifting between pan, orbit, and zoom operations, and, in the case of topologically complex objects presenting many cavities and protrusions, the camera eye can cross the object surface, resulting in an undesirable effect. General 7DOF controllers also require special handling to avoid lost-in-space effects. These problems occur because the navigation interface is not context-aware, but simply operates on the model transformation, leaving the burden to adapt to the surface to the user. We have thus designed an object-aware technique explicitly targeted to constrained macro-scale and micro-scale object exploration (R6). Among the possible solutions discussed in Chapter 3, we focus on techniques that do not require on-screen widgets, in order to have a clean view of the work of art (R8). IsoCam (see Sec. 9.3) automatically avoids collisions with scene objects and local environments, while trying to maintain the camera at a fixed distance around the object while maintaining a good orientation (R5). When hovering around the surface, our method maintains the camera position on the current isosurface of the distance field generated by the model, and, once computed the new viewpoint, devises a good orientation based on temporal continuity and surface orientation at multiple scales. Zooming instead is obtained by changing the current isolevel. With this approach, collisions are avoided, and the camera accurately follows the model shape when close to the object, while smoothly evolving to move around a spherical shape when moving away from it (see Fig. 9.2(b)). This approach permits to browse in a smooth way the model always facing the camera in a proper direction, but also allows the user to move among disconnected parts, due to the topological structure of a distance field. User interaction does not require to provide a visual feedback or to implement picking, as needed, e.g., to select a pivot for rotation over a model. The user can update the camera position by interacting with the touch screen, while always looking at the large display where the object of interest is shown (R8). Among the many non-standardized mappings of 3D touch gestures to 3D manipulations [Keef 13], we choose the simple approach of mapping common 2D RST gestures, such as pinch-to-zoom, one- and two-fingers pan, and two-finger rotate to analogue 3D behaviors, as this encourages people to quickly learn the mapping by looking at the effect of their actions. We note that this is made possible because of the constrained nature of our interface, which reduces degrees of freedom in order to favor ease of use at

the expense of generality.

9.2.4 Points of interest

3D hot-spots, popularized by VRML/X3D browsers and geoviewers, are the most common methods for associating auxiliary information to the 3D model [Call 08, Jank 12]. These methods, however, require pointing methods and/or produce clutter. Instead of using hot-spots, we thus employ a context-based radial menu to select among a subset of context-selected views. The system thus supplies the user, on demand, a set of precomputed points-of-interest, corresponding to the best ones with respect to the observer position and view similarity metric (see Fig. 9.2(c) and Sec. 9.4). Each view is also enriched with an associated overlaid information, which is presented to the viewer (R10). As for constrained navigation, point-of-interest selection does not require any visual feedback on the interaction touch screen, nor the availability of a specific pointing method (see Sec. 9.5) (R8).

9.2.5 Scalability

In order to ensure real-time performance (R5) on large datasets (R2) presented at high resolution (R1) (e.g., billions of triangles and hundreds of points of interest per model), we employ specialized multiresolution structures and adaptive algorithms. In our approach, the renderer and the user-interface subsystem share structure and work on a view-adapted representation of the data-set (see Sec. 9.7). In particular, we have extended the Adaptive TetraPuzzles (ATP) approach [Cign 04] for massive model multiresolution visualization to implement all required geometric queries, and employ kd-trees to organize the points of interest. Since the complexity of all operations depend only on the (bounded) view complexity, rather than on the (potentially unbounded) model complexity, we can guarantee high display accuracy and real-time performance both for macro- and micro-structure inspection (R1, R2, R5).

9.3 Camera control

Previously proposed object-based orbiting approaches, well suited for indirect interaction, are based on the concept of sliding a target point on the object surface, and determining the camera orientation based on the local surface normal, using ad-hoc techniques to manage cavities and sharp turns. These requirements have two implications: first, the user can easily move only on connected parts of the

model, and second, the movement is highly sensible to the surface roughness and thus needs to be filtered properly to avoid abrupt accelerations.

We start from a concept similar to the one of the sliding surface, but the basic idea is to exploit the structure of the object's distance field. Thus, during hovering operations, the camera position is maintained at a constant isolevel, while zooming is achieved by changing the isolevel. Camera orientation is determined in this approach only after position is incrementally updated. One of the benefits is that isosurfaces of the distance field, being offset surfaces, are much smoother than the surface itself and change smoothly from the model shape to a spherical shape with increasing distance, combining increased smoothness with topological simplification. This permits to follow all the fine detail when the distance is small, while smoothly orbiting and moving across disconnected parts when the distance increases. For example, when inspecting a human figure, users can hover over a single leg when looking from a few centimeters, while orbiting around both legs from a further distance, see Fig. 9.3(a).

As we will see, orientation is computed by considering the gradient of a smoothed version of the distance field, an up-vector model, and temporal continuity considerations. Decoupling position and orientation computation allows us to have precise guarantees in terms of collision-free motion, which are not shared by hovering techniques which smooth the surface to reduce jittering [Moer 12]. In particular, distance field information is extracted with on-the-fly nearest-neighbor computation on multiresolution approximations of the current view-dependent approximation of the model (see Sec. 9.7). It should be noted here that position computation must be precise enough to avoid collisions, while orientation computation can trade precision with smoothness. Thus, a precise nearest neighbor search is used to keep the user at a fixed distance from the model surface, while a coarser nearest neighbor computation is preferred to compute the view direction, towards having smooth camera orientation variations. In the following, we define *fine target* as the nearest neighbor on the refined model with respect to the eye, while the *coarse target* corresponds to the nearest neighbor on a smoother coarse model (see Fig. 9.3(b)). Sec. 9.7 details how fine and coarse targets can be efficiently computed using the same multiresolution structure that is used for rendering.

9.3.1 Hovering

The hovering operation tries to maintain the camera on the current isolevel, looking straight at the model. Hovering is performed through an incremental update of the current position on the isosurface, followed by an update of the

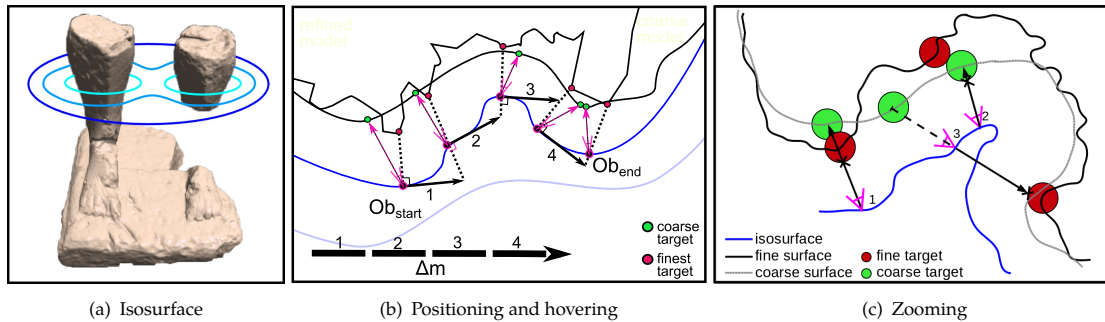


Figure 9.3: IsoCam. a) Some line of the isosurface. b) Hover movement orthogonal to gradient, user input Δm split into 4 steps to follow the isosurface. c) Zooming stopped in three situations: (case 1) approaching fine target, (case 2) approaching coarse target, (case 3) zooming backward (circle radius represents the minimum permitted distance from surface).

view direction. As user input, the camera controller takes a 2D vector indicating the desired direction and amount of motion in the camera plane. We assume in the following that the increment is small. Therefore, large user inputs (large velocities) imply multiple update steps. In order to update the position, we achieve motion by tracing an isoline starting from the current position. The tracing algorithm moves the observer position tangentially to the isosurface, re-projects it onto the isosurface, and finally updates the view direction, see Fig. 9.3(b). For that, each step is unprojected from screen to world space using the current projection and view transformation looking towards the fine target, thus producing a movement tangential to the isosurface. Projecting the observer onto the isosurface consists of moving the observer position towards the fine target to adjust the distance, and then recomputing the fine target. Working exclusively on fine targets for position computation ensures that we cannot have collisions with the model. Since the gradient direction changes from point to point, a single translation would not be enough to reach the original isosurface, and the projection algorithm is applied iteratively until the correct isolevel is reached within a prescribed relative tolerance (a small fraction of the isosurface value) or a maximum small number of iterations has been done. After this iterative adjustment, a new coarse target is recomputed for orientation computation. At this point the observer position and the view target are available, but we still need an up vector to define the current view transformation. Up-vector definition is a classic problem in camera motion control, and many solutions are available in the literature (see Khan et al. [Khan 05] for up-vector techniques suited for proximal navigation). In this work, we use a global up vector model, which is particularly suitable for exploring standing sculptures. In order to handle singularities, we limit the camera tilt with respect to the up vector, generating a camera motion similar to the one a human would perform around a statue.

9.3.2 Zooming

Zooming is achieved by switching between the different isosurfaces. The user input modifies the distance of the observer from the coarse target. The coarse target position, which defines the view direction, is not updated during the zooming, thus leaving unchanged the orientation and producing a smooth and intuitive movement. Moving between isosurfaces has the benefit that we always know the distance from the model surface, thus we can easily avoid collisions, simply limiting the distance from the model. We perform the movement only if the new distances from both the fine target and the coarse target are greater or equal than a minimum distance. We obviously need to check the fine target to avoid collisions. At the same time, we check the coarse target distance to avoid abrupt rotations when keeping facing the coarse target, see Fig. 9.3(c) (case 1, 2). Zooming in and zooming out use the same algorithm, since these checks are enough to prevent forward, but also backward collisions, see Fig. 9.3(c) (case 3).

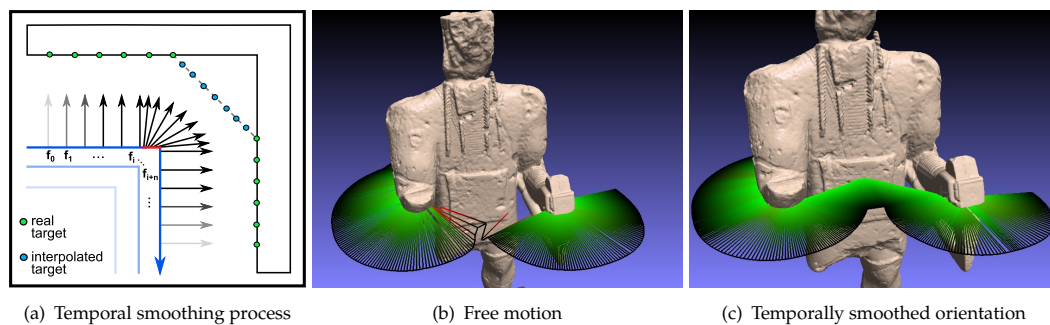


Figure 9.4: Temporal smoothing. Position and orientation interpolation when reaching coarse target discontinuity. In the 3D views, lines identify view direction in different frames. Large view direction discontinuities are highlighted in red.

9.3.3 Sudden orientation changes and temporal smoothing

A benefit of moving over an isosurface is the ability of early detecting collisions with invisible parts, as could happen in the HoverCam approach [Khan 05] when browsing a surface which presents a sudden change of orientation, as in Fig. 9.4(a). HoverCam solves this problem with prediction in the movement direction to avoid hitting the hidden surface. In our case, this is implicit when moving over an isosurface that covers the entire model. It is important to note that our view-dependent representation is complete, i.e., it covers the entire model without clipping it outside the view frustum (resolution of far/out-of-view nodes are only reduced) (see Sec. 9.7). Sudden surface orientation changes could cause, however, accelerations in user motion, due to fast changes of view direction (coarse target) even in a smoothed surface. This is a simpler situation to handle,

as it does not need to modify the path of the camera and only requires introducing acceleration control in the orientation. Working on coarse targets for orientation computations already introduces a degree of smoothness in orientation changes. Moreover, limiting the screen space movement of the coarse target produces a smooth change of orientation also in case of sudden coarse neighbor changes. To achieve this goal we introduce hysteresis in the process, achieving temporal smoothing. Positions and orientations are thus adaptively interpolated between previous and current view transformation. The interpolation value is given by the ratio between the user input movement and the coarse target screen movement, see Fig. 9.4(b), 9.4(c). Since orientation does not change linearly, we recursively apply this procedure until the coarse target screen movement is within a certain tolerance from the user input. This procedure converges in a few steps, generally 2 or 3. Finally, the current camera view transformation is interpolated at each step with previous camera transformation to produce a smoother movement.

9.4 Image-based navigation and points of interest

We provide the user also with a context-based guided navigation relying on nearest point-of-interest selection. This alternative control method allows the user to explore the object by traveling through a set of interesting views that have been previously defined and annotated. For that, the user is presented with a small set of images consisting of interesting views that can be reached from its position and can be selected with a radial selection interface shown in the projection screen (see Fig. 9.5). Selecting one of the presented images triggers a camera animation that moves the viewpoint to the location associated to the images and, then, presents in overlay the associated information (see Fig. 9.5). The information remains visible until the user decides to move to another position. This approach, based on static overlays, which can freely contain 3D sketches to be super-imposed to rendered 3D models, greatly simplifies the authoring of annotations by domain experts, which can use standard 2D graphics tools to generate vector graphics representations enhancing and commenting particularly interesting views.

No *hot-spots* or *hyperlinks* are visible during the navigation, but activated on demand by the user (see Sec. 9.5). Upon activation, a search is performed for determining the best view candidate set from the user viewpoint. The search of the best view candidates is performed on a separate kd-tree that contains all the views (see Sec. 9.7). For each view we store the associated image previously rendered, a description of the view contents, which is then rendered as overlay,

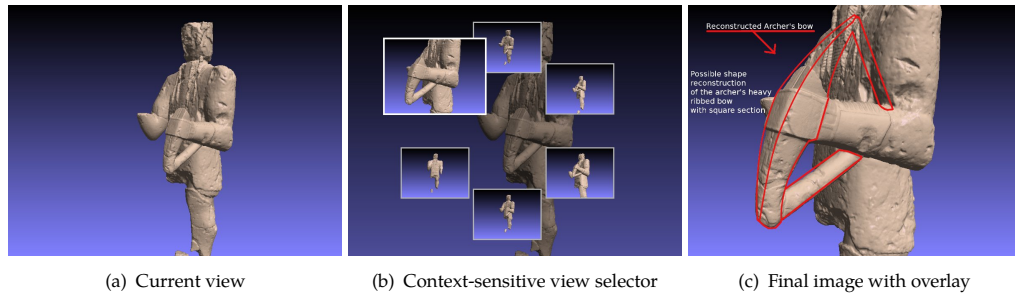


Figure 9.5: Radial view selector. A small set of images is presented to the user corresponding to the nearest views available from the current viewpoint. Selecting one of the presented images triggers a camera animation that moves the viewpoint to the location associated to the images and, then, presents in overlay the associated information.

and the viewing transformation from which we derive the point of view. Context-based selection selects from the database images that are similar to the current view and easily reachable from the current position (see Sec. 9.7).

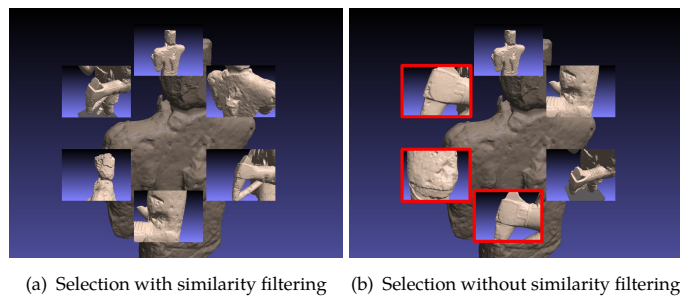


Figure 9.6: Context-based selection. Selected images with (a) and without (b) similarity filtering images from the point of view corresponding to the top view in the radial view selector. In the image without similarity filtering (b), we have highlighted in red the views less related to the current point of view, like the knee and the two closeup of the hand, which are close in terms of viewpoint but not in terms of similarity.

By using a radial selection interface, the user is no more required to look at the touch screen for switching between the different views, which are distributed onto a circumference around the center of the display. The upper position is held by the the nearest view with the following views in distance order being distributed in a clockwise fashion. Relative displacements from the initial touch position determine the selection in a natural way.

9.5 Device mapping

Our user interface for model manipulation and point-of-interest selection requires minimal user input, and can be mapped from input devices in a variety of ways. In this paper, we are interested in a situation where a touch screen is used for motion control and the rendering is displayed onto a big projective screen. For such a

setup, we avoid using the touch-screen to display content-related information, in order to encourage the user to focus on the visualization screen instead of concentrating on the user interface, see Fig. 9.12(a). It should be noted that, when using touch input, we not only have to deal with 2D-input-to-3D-output mapping, but also have to be aware of modality and precision issues [Keef 13]. The IsoCam interface helps in this respect, since it uses a constrained navigation approach that reduces degrees of freedom in order to simplify common operations, limits operations to zooming (1D), hovering (2D), and optional twist (1D), employs a fully incremental mode of operation that does not require pointing. We have implemented the interface both with a single touch as well as a multi-touch input. The single touch interface can be operated also with a one-button 2D mouse. Incremental camera motion control requires only to differentiate zoom vs. hovering and to continuously specify the associated 1D or 2D amounts. Point-of-interest selection requires the activation of the circular menu, and the specification of an angle to select the desired point-of-interest.

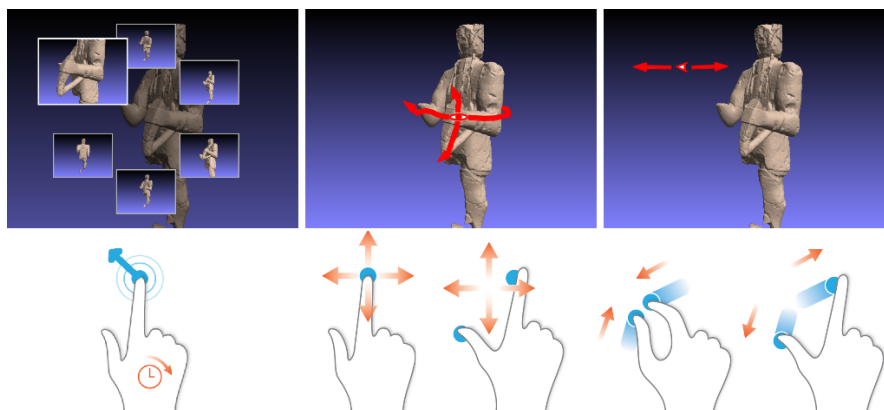


Figure 9.7: Multi-Touch gestures. Device mapping using a multi-touch device. Hand images adapted from Wikipedia (contributor GRPH3B18) under a Creative Commons Attribution-ShareAlike license.

9.5.1 Multi-touch mapping

Multi-touch is our preferred mode of operation. In order to reduce training times, we considered common single-touch and multi-touch gestures, as used for 2D actions in commodity products such as smart-phones or tablets, and mapped them to analogue 3D behaviors in our constrained controllers. To the same end, we also found useful to associate multiple gestures to the same action. While no standard exists for mapping 3D touch gestures to 3D manipulations [Keef 13], one-finger pan and two-finger rotate-scale-translate gestures are well established and accepted in the 2D realm. Since our controller is constrained and maps 3D motions to locally 2D actions, it is possible to map common 2D RST gestures to

analogue 3D motions. This has the advantage of encouraging walk-up-and-use behaviors, with people learning the 2D to 3D mapping while interacting.

One-finger long press is thus mapped to the activation of the point-of-interest menu, and a radial motion in the direction of the desired thumbnail followed by a release performs selection. Single finger panning gesture is used to control hovering. Direction and velocity are specified by the vector from the first touch position to the current finger position. Two-finger gestures can also be used to control hovering, as well as zooming and up-vector control. Two-finger panning has the same effect as single-finger panning (i.e., hovering control), while two-finger pinch controls zooming, and two-finger rotate controls twist (i.e., up-vector modification). See Fig. 9.7.

9.5.2 Single-touch mapping

The single-touch interface is used on desktop applications and in cases where a multi-touch screen is not available. Device mapping is similar to the multi-touch interfaces, with long press (or right button press for a mouse) for menu activation, and dragging gestures for hovering control and point-of-interest selection. Similarly to other works on unified camera control [Zeile 99], zooming in this case is activated by reserving a specific area of the touch device (a vertical bar placed on the left of the screen), and using dragging in this area for zoom-in (drag up) or zoom-out (drag down).

9.6 Extending support to light field displays

The evolution of 3D displays, with “light field displays” (see Section 3.3.6) as a good example, provide a well suited 3D visualization platform for exhibitions and museum installations. This kind of displays are now capable of providing multi-user immersive exploration of high quality 3D content. Besides, those displays provide variable resolution depending on the projection depth, where 3D objects lying in a depth range close to the screen plane are displayed at maximum resolution.

Aiming to extend our current system, we have studied how our method could be exploited for enhancing the visualization of complex 3D models on “light field displays”. For that purpose, we constrain the region of the 3D model which is in focus to lie in the correct depth range for optimal display resolution.

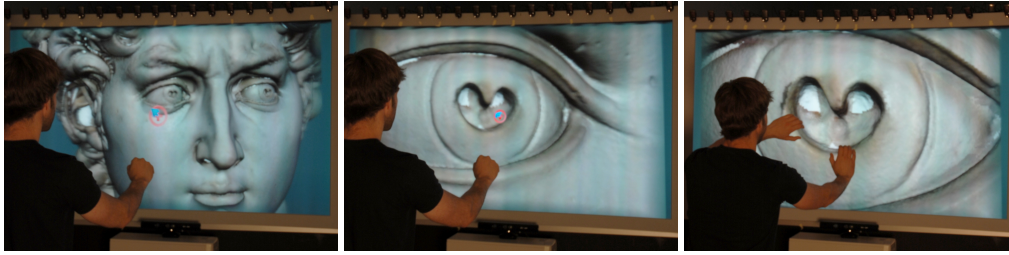


Figure 9.8: Natural immersive exploration of the David 0.25mm model (1GTriangle) on a 35MPixel light field display. Images taken with a hand-held camera. The 3D user interface allows casual users to inspect 3D objects at various scales, integrating panning, rotating, and zooming controls into a single low-degree-of-freedom operation, while taking into account the requirements for comfortable viewing on the light field display hardware. The model appears to be floating in the display workspace, providing correct parallax cues to multiple naked-eye observers.

9.6.1 Light field display: concepts and consequences

The light field display employed for this work uses a specially arranged projector array driven by a cluster of PCs and a holographic screen (see Fig. 9.9 left). The projectors are densely arranged at a fixed, constant distance from a curved (cylindrical section) screen. The projectors cast their respective images onto the holographic screen to create the light field. Mirrors positioned at the sides of the display reflect back onto the screen the light beams that would otherwise be lost, thus creating virtual projectors that increase the display field of view. The holographic screen has a holographically recorded, randomized surface relief structure able to provide controlled angular light divergence: horizontally, the surface is sharply transmissive, to maintain a sub-degree separation between views determined by the beam angular size. Vertically, the screen scatters widely, hence the projected image can be viewed from essentially any height. Thus, this approach creates a display with only horizontal parallax.

In order to cope with the parallax-only design, we employ a multiple-center-of-projection (MCOP) technique [Jones 07, Agus 08] to generate images with good stereo and motion parallax cues. The method is based on the approach of fixing the viewer's height and distance from the screen to those of a virtual observer in order to cope with the horizontal parallax. We assume that the screen is centered at the origin with the y axis in the vertical direction, the x axis pointing to the right, and the z axis pointing out of the screen. Given a virtual observer at V , the ray origin passing through a point P is then determined by $O = (E_x + \frac{P_x - E_x}{P_z - E_z}(V_z - E_z), V_y, V_z)$, where E is the position of the currently considered projector. The ray connecting O to P is then used as projection direction to transform the model in normalized projected coordinates. The parameters used for mapping screen pixels to screen 3D points can be determined by automated multi-projector calibration techniques [Agus 08].

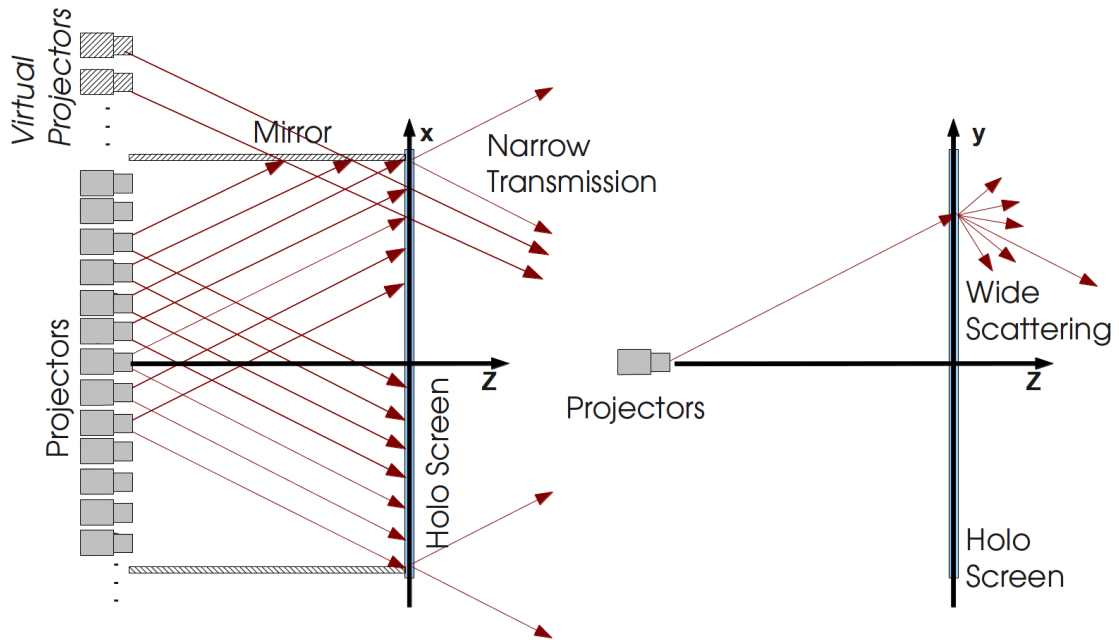


Figure 9.9: Light field display concept. The display uses a specially arranged projector array, a holographic screen, and side mirrors to increase the field of view. Left: horizontally, the screen is sharply transmissive and maintains separation between views. Right: vertically, the screen scatters widely so the projected image can be viewed from essentially any height.

By appropriately modeling the display geometry, the light beams leaving the various pixels can be made to propagate in specific directions, as if they were emitted from physical objects at fixed spatial locations. Freely moving, naked eye users can thus have the illusion of seeing virtual objects floating in the display workspace. It is important to note that the images of these objects are sharp only near the holographic screen, since the spatial resolution of the display is variable with respect to depth, approximately according to the equation $s(z) = s_0 + 2\|z\| \tan(\frac{\Phi}{2})$, where z is the distance to the holographic screen, and s_0 is the pixel size on the screen surface [Agus 08] (see Fig. 9.10 left). While blurred images are acceptable on the background, far from the viewer, excessive blurring near the viewer leads to discomfort.

Thus, the 3D display and related rendering methods have peculiar characteristics which impose constraints to the interaction and rendering system in order to generate compelling visualizations and reduce rendering artifacts. Specifically, the following characteristics have to be taken into account for the implementation of a natural interactive rendering system for massive models on a light field display:

- the spatial resolution of the display is variable with respect to depth, and objects far from the display's holographic screen appear blurred; thus, points of interest of the objects should be rendered near the screen surface;

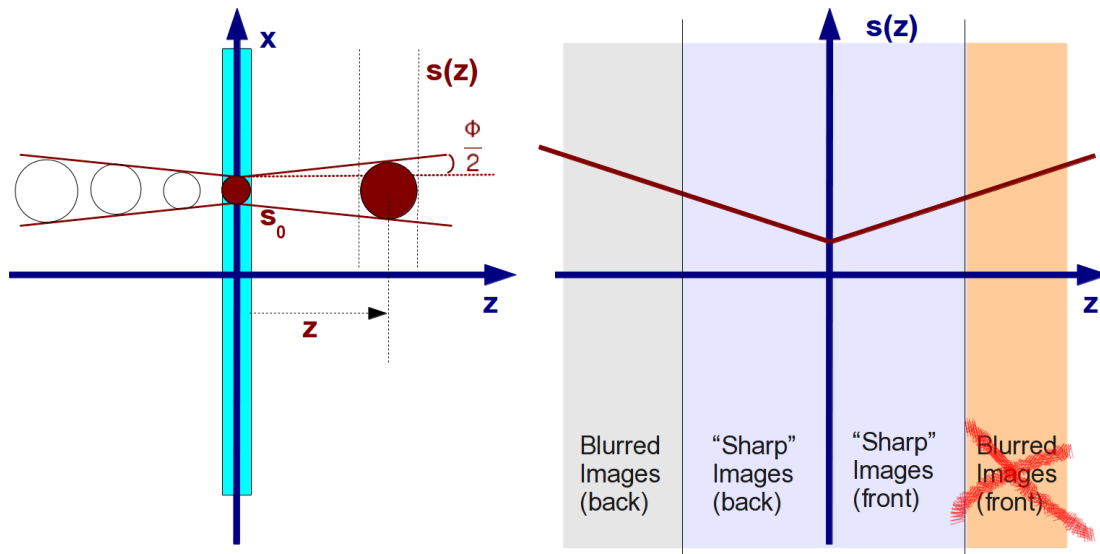


Figure 9.10: Light field display spatial resolution. The spatial resolution of the display varies with the depth. Only the region near the holographic screen is rendered sharply.

- the calibration technique minimizes errors only on the surface of the screen; thus, the effective depth of field of the display is reduced not only because of the diminishing spatial resolution, but also because of the spatially varying calibration accuracy;
- because of the display geometry, the angular field of view is limited and allows presentation of objects only within well defined angular bounds.

Thus, the best viewing experience is obtained when: (a) the scene is kept centered with respect to the screen; (b) the scene remains inside a limited depth range (at least in the front area of the display); and (c) the frequency details of the objects are adapted to the display's spatial accuracy. While (c) can be obtained by suitable rendering methods (see [Mart 12b]), (a) and (b) are best met by taking special care to position the scene within the display workspace.

9.6.1.1 Automatic model depth adjustment

We ensure that the model is always in contact with the display hot-spot, which should be at the center of the screen. Another requirement imposed by display characteristics is to keep the surface being manipulated at a good viewing depth. The display achieves its best resolution on its surface ($z = 0$). However, we found that users prefer to have the object slightly protruding from the screen in order to be able to virtually touch it (see Fig. 9.8 right). Thus, we would like the system to place the surface approximately at a depth H^* a few centimeters out of the screen. Simply placing the hot-spot at a fixed depth H^* is not sufficient, since the model can have complex asymmetric shapes around the hot-spot.

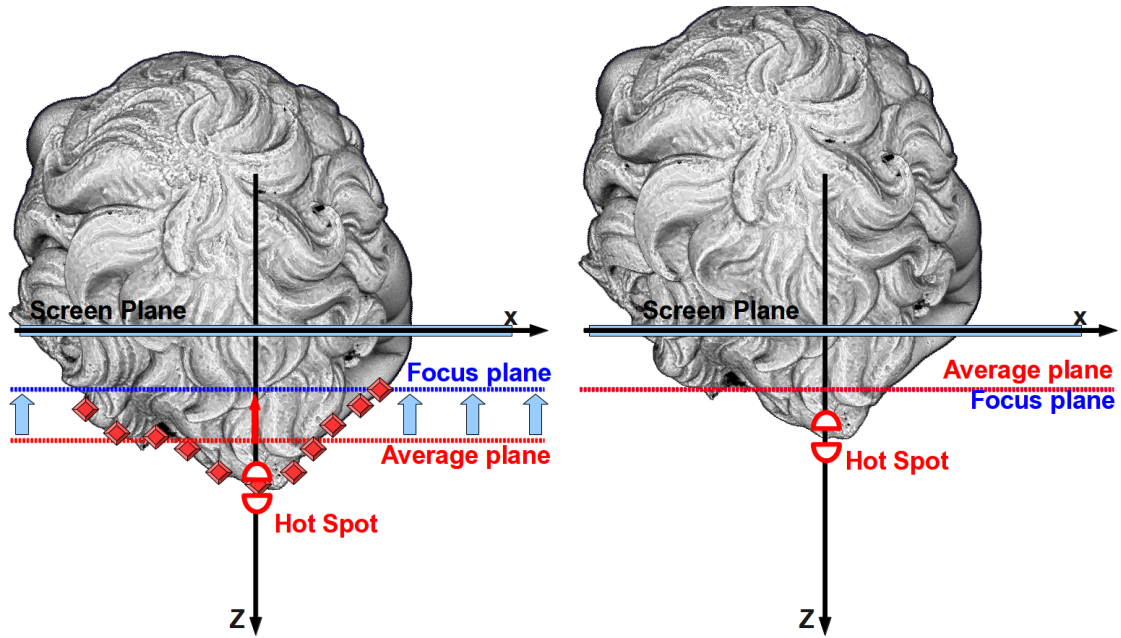


Figure 9.11: Automatic hot-spot placement. The depth of hot-spot is tuned automatically during interaction to keep the manipulated surface in a good viewing position. To do so, a least square plane of the points in the neighborhood of the hot-spot is computed.

To implement this depth adjustment feature, we developed a feedback correction scheme that automatically updates the model’s position (and thus the hot-spot depth) during interaction. For each user interaction step, our depth correction method extracts a coarse approximation of the surface in contact with the display hot-spot (see Fig. 9.11 left). This coarse point cloud, (P_0, P_i, \dots, P_N) , quickly extracted from our multi-resolution model representation (see Sec. 3.2.1), is then used to compute a weighted average depth s_z of the surface in the neighborhood of the hot-spot $H = (h_x, h_y, h_z)$:

$$s_z = \frac{\sum_i w^{(i)} P_z^{(i)}}{\sum_i w^{(i)}} \quad (9.1)$$

where the weight of each point $w^{(i)} = \Phi\left(\frac{\|(p_x^{(i)}, p_y^{(i)}) - (h_x, h_y)\|_2}{R}\right)$ is computed by a smooth, radially decreasing weight function for which we use the following compactly supported polynomial: $\Phi(x) = \max(0, (1 - x^2))^4$. Since the function has local support, only points within an xy-distance of R from the hot-spot contribute to determining the desired visible model surface depth s_z . For the purposes of this work, R was set to half the height of the display.

At this point, the amount of depth correction theoretically required is the difference between the average depth s_z and the comfortable depth H^* a few centimeters out of the screen (see Fig. 9.11 right).

In order to avoid abrupt changes in depth due to any surface discontinuities in the model and to reduce high-frequency vibrations, the depth correction is

temporally low-pass filtered by applying at each frame only a fraction λ of the full displacement (in our implementation $\lambda = 50\%$ adequately cut all undesired vibrations while still effectively correcting the scene depth). The overall model (and thus also the surface hot-spot) is thus translated at each frame by an amount $\lambda(s_z - H)$ in the z direction.

With this scheme, we are able to automatically keep the position of the approximated surface in a comfortable viewing position (close to the focal depth). Points near the hot-spot are therefore rendered at a good resolution and, since they are placed out of the screen, the surface can also be “touched” by users, increasing the quality of the experience. In our tests, a sampling rate twenty times coarser with respect to the original surface employed for rendering resulted to be computationally effective and sufficiently accurate for automatic model depth adjustment. A prototype system integrating this approach was presented in [Mart 12b].

9.7 Scalability

Both the rendering and user interface require specialized spatial indexing and multiresolution structures and adaptive algorithms to ensure real-time performance on large data-sets (billions of triangles and hundreds of points of interest per model).

We employ kd-trees to organize the points of interest, and have extended the Adaptive TetraPuzzles (ATP) approach [Cign 04], which already provides the features and performance required for massive model rendering, to implement the required geometric queries. The ATP coarse grained multiresolution structure is based on a hierarchical tetrahedral subdivision of the input volume. Each tetrahedron contains at leaf level a portion of the original model, while in inner levels, coarser representations are constructed by constrained simplification from children tetrahedra. Each node is made of a few thousands of triangles, leveraging the cost of the CPU traversal, amortizing node selection over many graphics primitives and properly exploiting current GPU bandwidth and processing capabilities. The algorithm is able to produce view-dependent adaptive seamless representations at high frame rates. The original algorithm has been improved, similar to what has been done by Balsa et al. [Bals 13c] by using a diamond structure which is refined with the use of a double heap refinement / coarsening strategy, to produce time critical interruptible view-dependent cut adaptations. At each frame, the view-adapted cut is used for all operations.

9.7.1 Nearest neighbor computation for camera control

The use of the multiresolution structure is mandatory to be able to perform, multiple times per frame, fast nearest neighbor search at multiple scales on massive models made up of hundreds of millions of samples, as required by our constrained camera controller (see Sec. 9.3). We perform all searches on the view-adapted tetrahedron graph. Each node of the tetrahedral hierarchy has been enriched with an internal bounding box hierarchy which permits to perform nearest neighbor computation at fast pace. This box tree is constructed by hierarchically bi-sectioning the node's vertex array and calculating the associated bounding box, exploiting the fact that the vertices in the triangle strip are sorted in a spatially-coherent order, similarly to ray strips methods [Bals 13c, Laut 07], which, however, construct a similar hierarchy on triangle strip. This is not required for our meshless approximation. Each node of the axis aligned box hierarchy contains, in addition the implicitly defined bounding box, the first and last indices of the vertex array portion it contains in order to be able to refine the search. The per-node bounding box hierarchy gives us the ability to perform nearest neighbor search at different levels of resolution, just by specifying the maximum level of the box hierarchy at which to perform the search. If the level is coarser than the box's maximum level, the search will return the point contained at half of the vertex array of that node. Returning a point belonging to the vertex array gives a better surface approximation with respect to returning for example the center of the bounding box, which could be placed away from the surface, depending on the point distribution inside the box. The fine search will look for the nearest neighbor among the leaf vertices (in our implementation there are $N < 16$ vertices at leaf level). Hence, coarse target search is performed near the root of the bounding box hierarchy, while fine target search is performed deeper in the hierarchy. In order to remove discrete sampling effects, each nearest neighbor search returns not only a single sample but a Gaussian weighted average $p^* = \frac{\sum_{i=1}^K w_i \cdot p_i}{\sum_{i=1}^K w_i}$ of the first K nearest neighbors identified by the knn-search, where $w_i = (1 - \frac{d_i^2 - d_{\min}^2}{d_{\max}^2 - d_{\min}^2})^4$ is the Gaussian-like weight for each of the K points, d_i is the distance of p_i to the eye, and d_{\max} and d_{\min} are the minimum and maximum distances among all d_i values. In our implementation we use $K = 8$ for fine target computation and $K = 16$ for coarse target computation to provide smoothing of sampling discontinuities at small CPU costs. Note, again, that this weighted averaging process is aimed at reducing discontinuities due to discrete sampling, not at smoothing the surface. Surface smoothing is not required for the fine target, used for position computation, while it is achieved for the coarse target through the sampling sparsification (subsampling) obtained by controlling search depth

in the box hierarchy.

9.7.2 Scalable selection of points of interest

Our context-based user interface requires scalable methods to efficiently select a good set of points-of-interest from a large database using a view similarity metric. We start from the fact that our situation is different from just image matching, since we have full information on the camera parameters as well as on the imaged 3D model. Thus, instead of computing image correlations or extracting and matching feature sets, we can compute similarity based on the fact that, for two similar views, a given point model would project to about the same image pixel. In order to efficiently perform the operation, we split the search into two phases. We organize all views in a kd-tree, indexed by the observer position. The selection of the M view candidates starts with a traversal of the kd-tree, looking for the $N \gg M$ points of view nearest to the user viewpoint that satisfy the following two constraints: (a) the angle ϕ between the user view direction and the candidate view direction must be below a certain threshold (in our case $\phi \leq 90^\circ$); (b) there should not be any obstacle between the user viewpoint and the candidate view (we perform a ray-casting between the two points of view to check for that constraint). Constraint (a) ensures that selected viewpoint has approximately the same orientation of the current view, while (b) ensures that the current viewpoint sees the target viewpoint, and a simple linear path can move the camera to the target point without collisions. Once the N nearest neighbors set satisfying both constraints has been determined, the set is further filtered to select the most similar views. Intuitively, two views are similar if surface samples in 3D model geometry project approximately to the same pixels in the two images. We perform the computation in Normalized Device Coordinates (NDC), see Fig. 9.6, using minimal information. For that, a small random set of 3D points is extracted from the view-adapted tetrahedron graph. This is done using a simple traversal of the graph leafs, selecting a few points per node. These points are then projected using the view parameters both from the candidate view and the user viewpoint, in order to calculate the average screen space distance between the two point sets, defined as $d = \frac{1}{S} \cdot \sum_{i=1}^S \|PV_j p_i - PV_{current} p_i\|$, where P is the projection matrix, V_j is the candidate's view matrix, $V_{current}$ is the current view matrix, and p_i corresponds to each of the S random selected points. Only $N \leq M$ views pass this last filter where $d \leq 2$ is required (as we work in NDC coordinates, a distance of two corresponds to screen width or height). A major advantage of this technique is that it is purely geometry-driven, and the only required information on the stored views are the viewing parameters.

9.8 Implementation and Results

We have implemented a prototype hardware and software system based on the design previously discussed in this chapter.

The techniques presented in this work have been implemented in a system for exploration of highly detailed models in museum settings. The system has been successfully used in a variety of settings, in particular for the exploration of a set of 3D models derived from the 3D scan acquisition of the statues of Mont'e Prama, ancient stone sculptures created by the Nuragic civilization of Sardinia, Italy, see Fig. 9.12(c),9.12(d). The 3D models of these statues are highly detailed and often made of a few disconnected parts, posing important problems to navigation techniques. See Bettio et al. [Bett 13, Bett 14b] for details on the Mont'e Prama dataset.

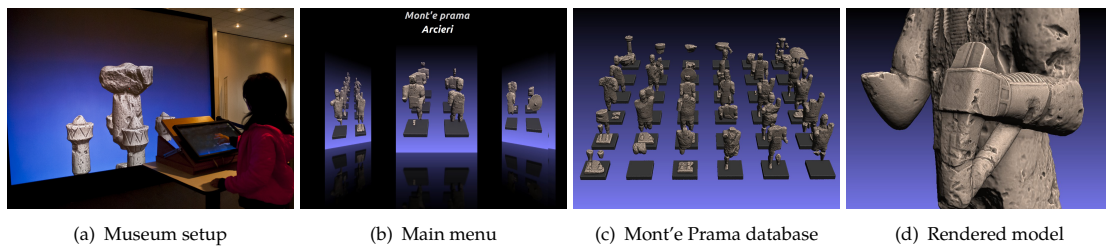


Figure 9.12: Application setup. a) Setup of touch-screen large projection device on a exhibition. b) Main menu interface. c) Mont'e Prama full database. d) Detail of a single Archer statue.

9.8.1 Reference implementation

A reference system integrating all techniques described in this paper has been implemented on Linux using OpenGL and GLSL for the rendering, and Qt 4.7 for the interface, while the database component relies on BerkeleyDB. The hardware setup for the interactive stations was composed of a 2.5m-diagonal back-projection screen, a 3000 ANSI Lumen *projectiondesign F20 SXGA+* projector, and a 23" ACER T230H optical multi-touch screen, both controlled by a PC with Ubuntu Linux 12.10, a Intel Core i7-3820 @ 3.6Ghz, with 8GB of RAM and a NVIDIA GTX 680 GPU. The multi-touch screen was designed to be placed one and a half meter away from the projection screen in order to allow visitors to see the whole display without being occluded by the user that was interacting.

The system has been tested on a variety of models including the Mont'e Prama statues. Models were rendered using a target resolution of 0.5 triangles/pixel, leading to graphs cuts containing on average 250 leaf nodes, with approximately 8K triangles/node. Interactive performance was achieved for all data-sets, with frame-rates constantly above 30fps and negligible interaction delays. Reusing

the multiresolution structure used for rendering also for object queries during interaction proved successful, as camera transformation computation in our camera controller always took just between 10% and 30% of the graph adaptation time required for rendering.

9.8.2 User study

In order to assess the IsoCam exploration controller we carried out a thorough user evaluation involving quantitative and subjective measurements based on interactive tasks carried out by a number of volunteers.

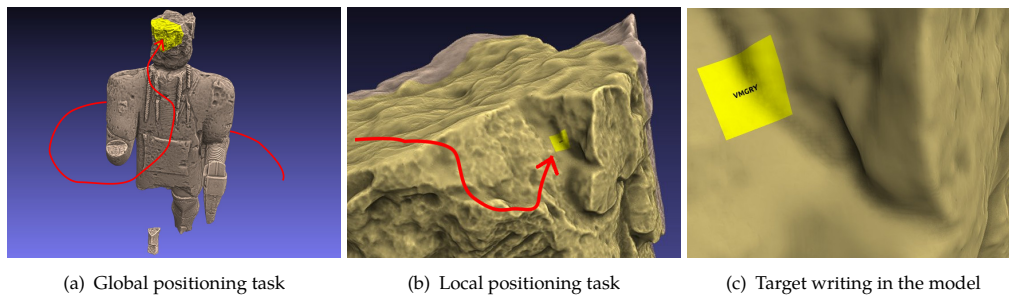


Figure 9.13: User interface evaluation. Tasks consisted of: (a) manipulating the model until reaching given specified global positions highlighted in yellow, (b) exploring a search area to find meaningful details (b), (c) identifying a 5-letter writing textured over the model.

9.8.3 Goal

The main goal of the evaluation was to assess whether the proposed camera controller is adequate for usage in the typical scenario of virtual museums, where many users with different skills and experiences try to interactively explore digital models in order to highlight details at various scales. Given the large number of 3D object exploration techniques, it is impossible to carry out a fully comprehensive comparison considering all the existent navigation systems. We thus limited our comparison to the virtual 3D trackball [Henr 04], and the HoverCam proximal navigation controller [Khan 05]. The virtual trackball was chosen because it is the interactive camera controller most commonly employed in virtual environments, and is considered easy and immediate to learn. HoverCam was considered because it is the most similar to IsoCam between the object-aware techniques available in literature.

9.8.4 Setup

The experimental setup considered the reference system implementation described in Sec. 9.8.1. All user interfaces were operated using the same precise

optical touch screen device using a multi-touch device mapping. IsoCam and HoverCam used the same device mapping presented in Sec. 9.5. The Virtual TrackBall was implemented by rigidly attaching the object to the user's hand and uses a typical RST approach: with one finger user generates 2-DOF rotations on the model's bounding sphere, while with two fingers (pinch) users perform zoom in or out by increasing or reducing the distance between the fingers, or pan the model by moving the fingers inside the workspace. As testing model, we selected a 80M triangles Archer statue from the Mont'e Prama collection, which has detailed carvings and is missing some parts: half bow, one hand and half leg (see Fig. 9.13(a)). The reconstructed model is thus composed by two disconnected parts (whole body and disconnected right foot), and presents many sharp features, surface saddles peaks and valleys, whose exploration represents a non-trivial task.

9.8.5 Tasks

The experiments consisted in letting users try the three different manipulation controllers (IsoCam, TrackBall [Henr 04], and HoverCam [Khan 05]) in the context of a target-oriented interaction task [Mart 09]. We designed our task to measure performance in the macro-structure and micro-structure inspections tasks typical of cultural heritage model explorations. Participants were asked to manipulate the model until reaching given specified global positions, and perform local small-scale explorations of the model surface with the goal of finding meaningful information. Targets were indicated by yellow patches lying on the Mont'e Prama bowman model, and the goal was to reach them as quickly as possible by starting from a fixed position and adequately manipulating the model in order to adequately position the viewpoint over the indicated areas (see Fig. 9.13(a)). When the target area was reached, the yellow patch area became an exploring area (see Fig. 9.13(b)) and users had to search and identify a 5-letter writing textured above the surface of the bowman (see Fig. 9.13(c)). Users had to read correctly the writing by moving locally the model up to reach an adequate orientation and scale, and identify it between 5 different choices indicated in buttons appearing under the touch area.

9.8.6 Participants

For the user analysis, 20 participants (17 males and 3 females, with ages ranging from 17 to 65, mean 36.5 ± 12.5 years) were recruited between the participants of the various exhibitions, CRS4 employees (including researchers, network administrators and administrative staff), and high school students. They were

subdivided in two groups according to their experience with 3D interfaces (12 novices and 8 experts).

9.8.7 Design

Each participant saw the three interfaces in randomized order. Users were first allowed to become familiar with the controller by watching a brief video showing how it works (part of the help system of the museum installation), trying it for two minutes, and performing one task just for testing. After the training session, the measured tests consisted of 5 trials, where targets were randomly selected from a list of 15 potential candidates, as to avoid any bias due to a-priori knowledge of target positions. For a complete testing session, users needed times ranging from 15 to 20 minutes. A trial was considered aborted if the target area was not reached within 40 seconds, or the inscription was not correctly identified within 40 seconds from reaching the target area. The times needed for reaching the target areas (global positioning) and for identifying the correct writings (local browsing) were measured and recorded. In summary, the complete test design consisted of 20 participants, each one testing 3 camera controllers with 2 tasks (positioning and identification) on 5 targets for a total of 600 time measurements. At the end of the experiments, participants were also asked to fill a questionnaire comparing the performance of the three controllers by indicating a score in a 5-point Likert scale with respect to the following characteristics: ease of learning, ease of reaching desired positions, and perceived 3D exploring quality. Finally, participants were asked to indicate their preferred controllers.

9.8.8 Performance evaluation

Before collecting the results, we expected that IsoCam would perform similarly to HoverCam and TrackBall for global positioning, while we expected a significant improvement in the local positioning task. We also expected that expert users might have similar performances for all the three camera controllers, and significant differences when the controllers are used by novice users. We performed an analysis of task completion times for global positioning and identification, and total times. Fig. 9.14 shows the boxplots of the task completion times, for novice and expert users, as rendered by the *R* package [R Co 13]. The bottom and top of each box are the first and third quartiles, the band inside the box is the second quartile (the median), and the ends of the whiskers extending vertically from the boxes represent the lowest datum still within 1.5 IQR (inter-quartile range) of the lower quartile, and the highest datum still within 1.5 IQR of the upper quartile. Outliers are indicated as small circles. An analysis of the boxplots in Fig. 9.14

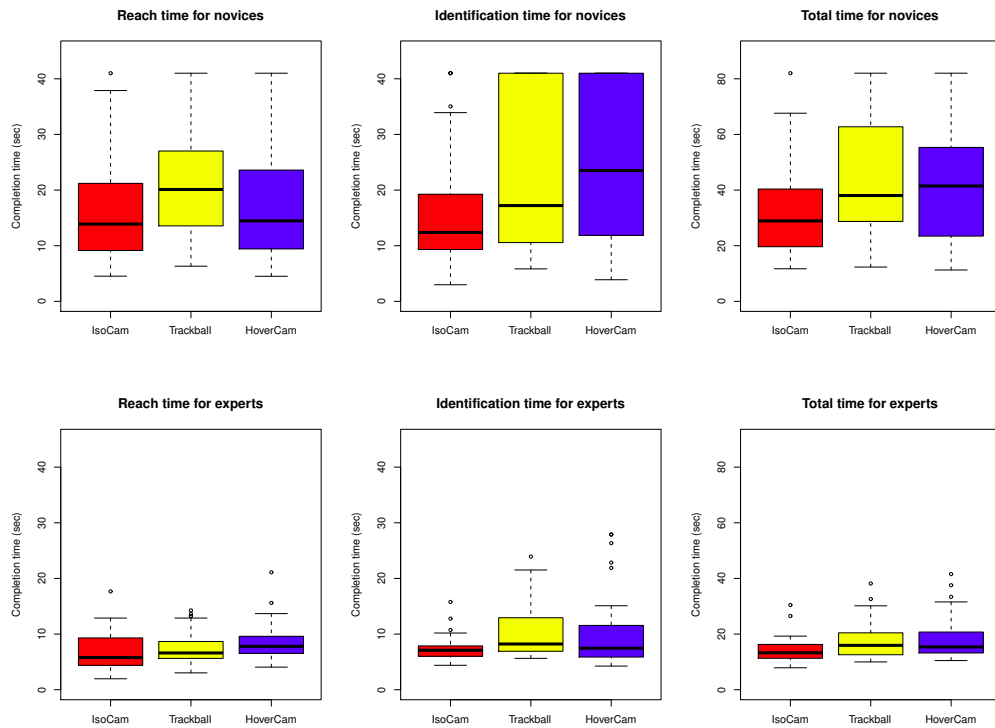


Figure 9.14: Performance evaluation. The proposed IsoCam camera controller was compared to the Virtual TrackBall [Henr 04] and HoverCam [Khan 05] with respect to performance and perceived navigation quality during the exploration of a complex Mont’e Prama statue model. In boxplots, red indicates IsoCam, yellow TrackBall, and blue HoverCam.

(bottom row) reveals that experts had similar performances with all interfaces, even if they were slightly faster with IsoCam. Through direct observation of expert users while performing the tasks, we noticed that the common strategy for completing the global positioning tasks consisted of moving the model at macro-scale before performing one zooming operation towards the target area. For the identification tasks, most expert users zoomed out the area in order to recognize the detail, and finally zoomed in to the target writing. In this way, for most of the targets, performances were more or less similar independently from the interface employed. However, for targets positioned in sharp areas of the model or close to disconnected parts, a number of users experienced problems with HoverCam and TrackBall. This explains the presence of some distant outliers in the boxplot of identification times for HoverCam and TrackBall. On the other hand, from top row in Fig. 9.14, it appears evident that novice users had better performances and felt more comfortable with IsoCam, especially during the identification task. Direct observation of interaction during the tasks revealed that naive participants did not follow any common strategy for completing the tasks, but the movement was mostly instinctive, with a gradual usage of zooming and the tendency of trying to slide along the surface when performing the exploration

and identification task. In most of the trials, independently from the position of the target, novice users performed continuous motion corrections, especially when using HoverCam and TrackBall, while with IsoCam they were able to keep smooth trajectories. To this end, particularly interesting is the size of IQR for the identification time (see second column of Fig. 9.14), from which it appears evident that both novices and experts had homogeneous performances during the local browsing task with IsoCam, but not with HoverCam and TrackBall. These results can be explained by considering that a direct surface orbiting technique like HoverCam can suffer when dealing with sharp features and disconnections, such as the ones contained the Mont'e Prama bowman model, while TrackBall needs many mode switches between rotations, translations and zoom to be able to inspect highly-detailed surface models while remaining at constant distance.

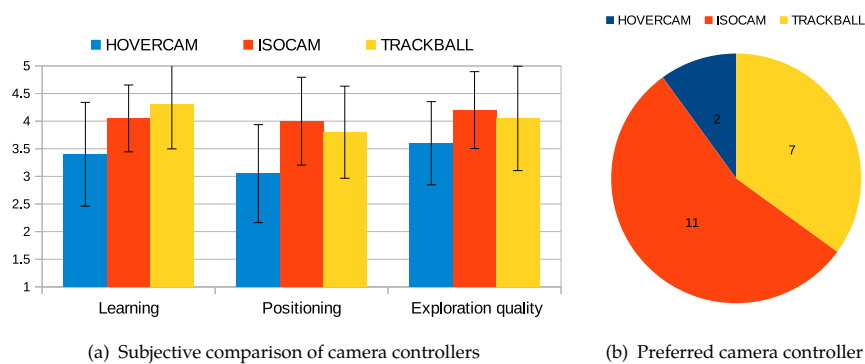


Figure 9.15: Qualitative evaluation. (a) Users were asked to evaluate the performance of the three camera controllers by indicating a score in a 5-point Likert scale with respect to ease of learning, ease of reaching desired positions, and perceived 3D exploring quality. (b) Users were asked to indicate the preferred controller.

9.8.9 Qualitative evaluation

Fig. 9.15(a), showing histograms with error bars for qualitative results, provides indication that users found the Virtual TrackBall easier to learn but not simple to operate, since they perceived that with IsoCam they could more easily reach targets, and they also felt more comfortable during exploration. The easy-to-learn property of the TrackBall is in large part due to the fact that the technique is (falsely) perceived as standardized, since most users have already been exposed to it, at least in standard 2D RST interfaces. It appears also evident that IsoCam revealed to be strongly appealing for users (see Fig. 9.15(b)). In addition, from think-a-loud comments, we derived that the large majority of novice users noted and appreciated the superior smoothness of the IsoCam camera controller with respect to the HoverCam.

9.8.10 Exhibition

The system has been successfully employed in a large exhibition held during the 65th International Trade Fair of Sardinia, held from April 25th to May 6th 2013 (see Fig. 9.1). The Mont'e Prama restoration project was presented to the visitors, together with virtual stands for interactively exploring the entire set of statues. In that setting, visitors were presented with a simple touch interface to select the statue to be explored. A gallery of models was shown on the multi-touch screen, and specific information about the currently selected object was shown in the projective screen. Sliding to left and right allowed the user to change the selected model, while a click on the central image invokes the associated action: showing the objects under the selected hierarchy group, or exploring the selected object. Once the model to explore has been selected, the visualization application came in, providing the user with an almost empty multi-touch screen (only the buttons for help and going back were shown), and the high resolution model being displayed in the projective screen, see Fig. 9.12(b). In order to interact with it, user simply had to perform touch gestures on the multi-touch screen without any visual feedback other than the visualization. Being these gestures relative to the initial touch position, user simply had to associate their movement on the screen surface with the behavior of the navigation, as would happen with a joystick, for instance. A total of approximately 3500 visitors attended the exhibition, and many of them interacted with the models using our interface (see accompanying video). Typically, groups of people gathered in front of the interactive stations and freely interacted with the system. The unobtrusive interface and setup was very effective in this setting. Informal feedback received during the exhibition was very positive. In particular, all users particularly appreciated the size of projection, the interactivity of the navigation, and the fine details of the statues provided by the multiresolution renderer.

A follow-up of this work is now the basis for permanent exhibitions on Mont'e Prama statues at the National Archaeological Museum in Cagliari, Italy and at the Civic Museum in Cabras, Italy. Each of these two installations employ a 7500 Lumen projector illuminating a 2.5m-high 16:10 screen positioned in portrait mode. This configuration permits rendering the statues at full scale. By matching projection parameters with the position of the viewer operating the console, we can achieve a natural viewing experience.

9.9 Discussion

We have presented an interactive system for natural exploration of extremely detailed surface models with indirect touch control in a room-sized workspace, suitable for museums and exhibitions. Furthermore, a museum installation has been set up and thousands of visitors had the possibility, during two weeks, of browsing high resolution 3D laser scan models of the 38 restored statues of the Mont'e Prama complex. We have presented a novel camera controller based on distance field surfing, enriched with a dynamic selection of precomputed views and point of interest, which does not require the visualization of hot-spots over the model, but is able to dynamically identify the best views near the current user position and to display connected information in overlay. The presented IsoCam camera controller has been compared with other two consolidated approaches, collecting quantitative and qualitative results from a series of tests involving 20 people. The camera controller appears to be intuitive enough even for casual users who quickly understand how to browse statue models in a short trial period. Our indirect control low-degree-of-freedom controller allows users to pan, rotate and scale, with simple and natural gestures, massive models, while always facing the model surface, without requiring precise pointing methods, and without the monolithic surface limitation of previous orbiting and hovering techniques. Moreover, we have shown how our camera controller can be exploited to enhance 3D rendering on light field displays. The resulting virtual environment, which combines ease of use with high fidelity representation and low-cost setup, appears to be well suited for installations at museums and exhibition centers. Our current work concentrates on improving interaction quality and on extending the system for supporting bidirectional connection between multiple multimedia types as well as narrative contents.

Advantages. The proposed setup is simple, low-cost and suitable for exhibitions and museums; it supports large scale display surfaces where multiple users can focus their attention and objects can be displayed at an imposing scale. The proposed navigation technique, which can also be employed in other more standard settings, lets inexperienced users inspect topologically complex 3D shapes at various scales and without collisions, integrating panning, rotating, and zooming controls into simple low-degree-of-freedom operations. Our approach, based on distance-field isosurfaces, significantly improves over previous constrained orbiting and proximal navigation methods [[Khan 05](#), [McCr 09](#), [Moer 12](#), [Mart 12b](#)] by better supporting disconnected components, more seamlessly transitioning between orbiting and proximal navigation, and providing smoother camera motion,

while being simpler to implement as it is based on a single unifying concept and does not require complex algorithms (e.g., motion prediction) to handle special cases. On-demand context-based selection of “nearby” points of interests is well integrated with the IsoCam interface, reduces clutter, does not require a pointing device for hot-spot selection, and supports overlaid information associated to target views. Our scalable implementation supports giga-triangle-sized models and hundreds of points-of-interest on commodity platforms.

Limitations. As for all current approaches, our approach has also limitations. The presented approach targets the traditional non-immersive setting with 2D screens and low-DOF input devices, and might be not suitable for more immersive setups (e.g., CAVEs). In addition, we only target the problem of 3D model exploration with image/text overlays. Using other associated multimedia information and/or supporting narratives are orthogonal problems not treated in this work. The proposed navigation technique achieves simplicity by constraining camera/model motion. As for many other constrained navigation techniques, not all view positions are thus achievable.

Scalability. The isosurface is computed on-the-fly by exploiting the underlying efficient data structures containing the geometry. This computation cost makes this method not suited for platforms with limited performance (i.e., mid-range mobile devices, or scripted environments such as Web browsers). In order to overcome this limitation, a discrete set of pre-computed isosurface levels could be used at the expense of considerable memory resources.

9.10 Bibliographical Notes

The major part of the content in this chapter is based on paper [Mart 14], where we present a user interface for navigating highly detailed 3D models using a object-aware camera controller and an interactive point-of-interest selector providing the user with the ability to explore in detail the 3D object without losing the focus and being able to query the system for information on close details. The content related to “light field displays” is based on paper [Mart 12b].

CHAPTER
10

.....
ExploreMaps: Ubiquitous Exploration of Panoramic View Graphs of Complex 3D Environments

In previous chapters, we have already explored constrained motion control for supporting guided navigation. The proposed methods are object-aware, and use some knowledge of the displayed scene to guide the user, by simply setting pivots for rotation, or constraining the camera to follow the displayed object's surface.

The approach presented in this chapter is focused, instead, on supporting ubiquitous interactive exploration of scenes with complex lighting. To achieve this, we rely on the approach presented in Chapter 7, which automatically constructs a graph of panoramic views and paths. Since the possible space of positions of the camera is extremely limited in this approach, it is reasonable to precompute images from all possible viewpoints (using maximum quality offline settings). The resulting camera-controller is extremely constrained, as it must only implement a graph visit.

10.1 Introduction

IN current 3D repositories, such as Blend Swap, 3D Café or Archive3D, 3D models available for download are mostly presented through a few user-selected static images. Online exploration is limited to simple orbiting and/or low-fidelity explorations of simplified models, since photo-realistic rendering quality of complex synthetic environments is still hardly achievable within the real-time constraints of interactive applications, especially on low-powered mobile devices or script-based Internet browsers.

Moreover, navigating inside 3D environments, especially on the now pervasive touch devices, is a non-trivial task, and usability is consistently improved by employing assisted navigation controls [Chri 08].

Spatially indexed photographic imagery, popularized by Internet systems such as Google StreetView and Bing Maps StreetSide, is, on the other hand, proving to be an effective mean for generating compelling interactive virtual tours of real locations. Designing such guided walk-throughs for synthetic 3D environments manually is, however, a hard and time-consuming task.

In this work, we introduce an approach aimed at automatically providing a richer experience in presenting 3D models on dedicated web sites. The method exploits the ExploreMaps visualization technique presented in Chapter 7, which provides a graph-based representation of the 3D scene, for designing a low *degrees-of-freedom* navigation interface. During exploration, the user is able to navigate between a series of predefined camera positions and inspect a panoramic view of the scene from them. Usability and sense of presence are increased by leaving orientation and field of view free when looking at the scene from a probe position, and gently converging to the closest target “good orientation” during transitions. Due to negligible CPU/GPU usage, real-time performance is achieved on emerging WebGL environments even on low-powered mobile devices

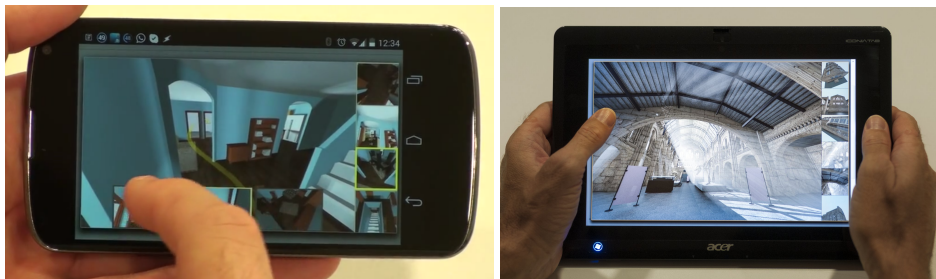


Figure 10.1: Mobile web-based exploration. The graph-based representation is exploited for providing visual indexes for the 3D scene and for supporting, even on low-powered mobile devices, interactive photo-realistic exploration based on precomputed imagery. On the left, browsing the German Cottage model on a Nexus 4 phone; on the right browsing the National Museum model on an Acer Iconia tablet.

10.2 Browsing Explore Maps

One of the main applications of ExploreMaps is supporting ubiquitous browsing of complex illuminated models using minimal CPU/GPU resources on web-based and mobile devices. We describe here the basic features of our reference JavaScript/WebGL implementation, which can run on any WebGL-enabled web browser, as shown in Fig. 10.1. The view graph is exploited both to provide a

visual index of the entire scene and to let users move within the environment. In automatic mode, the viewer traverses the graph by random walk. In interactive mode, the window is subdivided in three areas (see Fig. 10.3): the central area shows the high-res spherical panorama from the current probe position, while the right thumbnail bar (*probe bar*) shows all available probes, and the bottom context-sensitive thumbnail bar (*path bar*) shows selected probes reachable from the current position.

Probe Bar. Exploits a linearization of the views-graph in order to present a list of views where views which are close to each other are also close in the thumbnail list. For this purpose, we compute a weighted minimum linear arrangement (MLA) of the probes, i.e., a permutation Π of the probes such that the cost $\sum_{i,j \in E} w_{ij} |\Pi(i) - \Pi(j)|$ is minimal, where w_{ij} is the reciprocal of the length of the path connecting probe i to probe j . Intuitively, this ordering will attempt to cluster together the nodes that are close-by and connected by paths (see Fig. 10.2). We use this ordering at run-time to present nodes in a thumbnail bar using a logical exploration order. Since the MLA is known to be NP-hard, we heuristically approximate the solution using a multilevel solver [Safr 06].

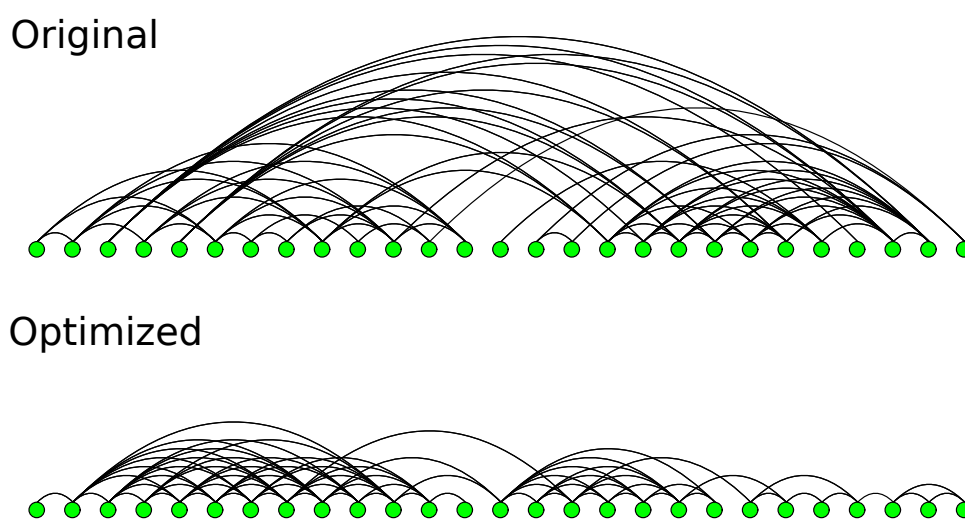


Figure 10.2: Graph optimization. Original and linearly arranged graphs. The optimized layout (bottom) reorders probes in a more coherent manner, moving nearby probes that are closely connected by short paths.

Path Bar. Panoramas in the path bar correspond to the target probes of the paths leaving the current probe. They are ordered based on the angle between the current view direction and the path direction, so as to always center the probe bar on the path most aligned with the current view, see Fig. 10.3. Panoramas are initially oriented towards their most preferential view direction. The user

is free to interactively change orientation with a dragging motion both in the central panorama and in the small panoramas appearing in the thumbnail bars. In addition, the central panorama is also zoomable. When a probe in one of the bars is selected, the path leading to it, if available, is shown in the main viewport. Clicking on the central viewport triggers a go-to action. When a non-directly connected probe is selected in the probe bar, the new probe is downloaded, and presented using a cross-dissolve transition. When going to a directly connected probe, the panoramic video corresponding to the selected path is started. The view direction is then interpolated over time, during video playback, between the one at the time of clicking, which depends on the user, to the arrival one, chosen among the best precomputed view directions. This improves the quality of the experience, since transitions are not repeated exactly (unless the starting position is exactly the same), and motion is consistent with the user-defined current orientation. Using precomputed video transitions with a single view direction would be too constraining, forcing the system to move the camera to the starting orientation of the video before transition, and forcing the arrival to a single fixed camera pose. Since we need a free viewpoint panoramic video, we render it by remapping frame areas on the 6 faces of the cube around the current point of view.

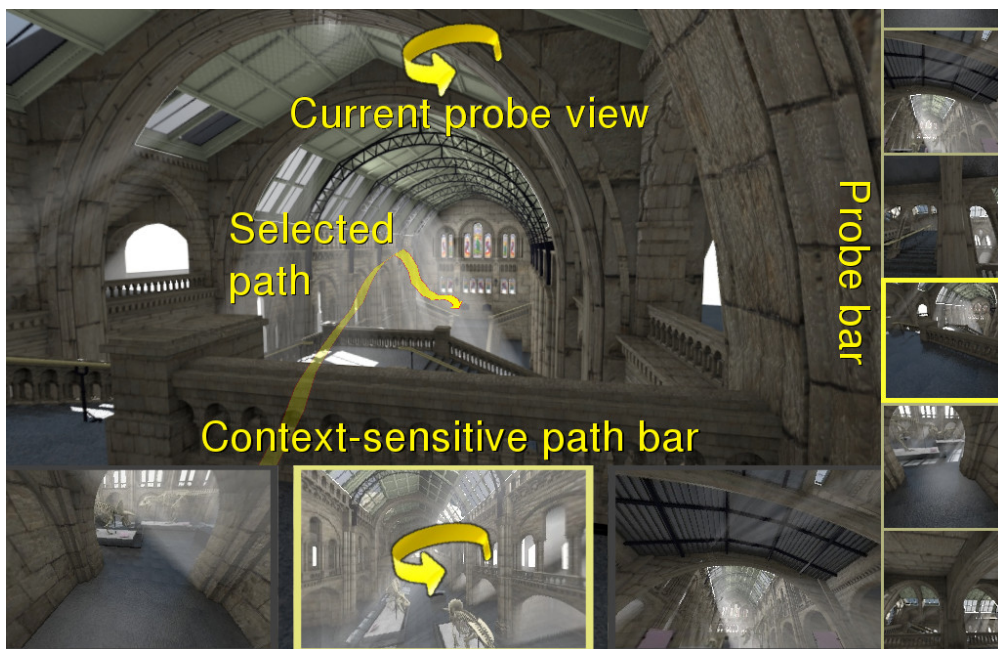


Figure 10.3: WebGL viewer. The central WebGL viewer area shows the currently selected probe, while the right thumbnail bar visually indexes the scene, and the bottom thumbnail bar shows a context-sensitive subset of reachable target position.

10.3 Implementation and Results

We have implemented a prototype hardware and software system based on the design previously discussed in this chapter. The browsing application has been written in JavaScript using WebGL and HTML5. It is able to deliver results in a HTML5 canvas running in WebGL-enabled browsers (Chrome 30 was used for tests in this work).

10.3.1 Test models

In order to evaluate our approach, we downloaded several models from public repositories (Trimble 3D Warehouse, Archive3D). These websites show a few views of each model so that users can judge if they are interested in downloading it. Therefore, these sites are a perfect example of how the ExploreMaps could be used for a higher quality browsing of 3D models. Samples from browsing various of the models used in our tests can be seen in Fig. 10.4, which have been selected to be representative of various model kinds, featuring complex illumination and/or geometry.



Figure 10.4: Browsing results. Browsing samples from different scene types. The whole scene can be explored by either following probe connections, or selecting target probe from the thumbnail bar.

10.3.2 Browsing

Our prototype client has been tested on a variety of devices, including a Nexus 4 phone (Qualcomm Snapdragon S4 Pro 4-core; 1280x768 screen) and an Acer

Iconia 500 tablet (AMD Fusion C-60 and Radeon HD6290; 1280x768 screen) connected to a wireless network. Our tests demonstrate our ability to sustain interactive performance on photo-realistic environments. During navigation, the user can look around within a single probe, and move to distant ones without losing the sense of location (see browsing samples from a variety of scenes in Fig. 10.4). The frame rate during probe exploration typically exceeds 50 fps, while the frame rate during video transitions drops down to about 20 fps due to video decompression and texture updates. Even though our WebGL application is not a full-fledged viewer, it shows the potential of this automated browsing approach. While some of the models could be explorable on such a mobile device in full 3D, this could definitely not be done while presenting the same quality images (see, for instance, volumetric illumination effects in the Museum example of Fig. 10.4). There is also a definite advantage on using thumbnails for quick scene browsing and panoramic videos for transitions with free selection of both the starting orientation and the arrival one.

10.4 Discussion

Our constrained navigation approach enables interactive exploration of scenes with complex lighting on a wide range of platforms where it would be impossible otherwise. Although our interface has the ability of exploring the whole scene thanks to the graph-based description, there are many possible improvements. In particular, there is no way to aim at a point of interest and move to the probe that has a best view of it, which may be frustrating for the user. Moreover, the view graphs should also be exploited to provide location awareness through the automatic generation of overhead views.

There are many potential applications for the ExploreMaps. In particular, we aim to open the way to a richer experience in presenting 3D models on dedicated web sites, no more limited to few still images or very constrained orbiting interfaces. Furthermore we can turn construction CAD into navigable previews for presentation to stakeholders/potential owners.

Advantages. The proposed interface provides the user with simple point and click navigation to neighboring points of view, as well as image-assisted navigation to travel to a given scene point of view through a thumbnail scroll bar.

Limitations. The proposed user interface is very limited and thus can be extended in many ways. Exploiting the transition videos connecting neighboring points of view, the user could be allowed to control the displacement in the direction of

movement of the video while being able to explore the panorama available at any frame. In this way, navigation could be extended to allow navigating through the connecting paths, in addition to inspecting point of view panoramas.

Scalability. With the only requirement being correct panorama rendering, typically implemented on graphics platforms supporting textured polygons, this technique is suitable for most platforms including web browsers running on desktop and mobile systems. Web browsers APIs for image and video decoding provide efficient data streaming, which together with WebGL result in a portable solution for a wide range of platforms.

10.5 Bibliographical Notes

Most of the contents of this chapter regarding is based on paper [Di B 14], where we presented an approach for rendering complex 3D scenes with complex illumination by computing a set of best views from the input model and letting the user navigate a graph-based representation of the scene through precomputed paths connecting an optimized set of camera positions. The approach presented in this chapter corresponds to the navigation method we designed for exploiting the visualization technique described in Chapter 7.

Part IV

Beyond Visual Replication

When exploring complex 3D models, information can be found at multiple scales, i.e., global shape or very fine details on the model surface. The exploration of this kind of 3D models requires guided navigation techniques in order to provide an easy-to-use interface that allows the user to concentrate on the 3D virtual object instead of the interaction itself. Moreover, 3D annotations are often used to enhance and integrate the understanding of the visual information at multiple scales by providing contextual information.

This last part focuses on navigation-oriented methods for the exploration of complex annotated 3D models with unobtrusive guidance and contextual information presentation without cluttering the visualization.

Type	Approach	Method	Published in	Features
Navigation-oriented	<ul style="list-style-type: none"> - Guided navigation - Stochastic recommendation system See Chapter 11		EuroVis'15	<ul style="list-style-type: none"> - Unobtrusive guided navigation - Non-cluttering information discovery - Non-linear information presentation - Simple authoring

Table 10.1: Annotated 3D model exploration methods.

CHAPTER

11

.....

Adaptive Recommendations for Enhanced non-linear Exploration of Annotated 3D Objects

While previous chapter have mostly focused on methods to compactly represent 3D scenes, rapidly render them, and efficiently explore them through camera controller, in this chapter we focus on the problem of letting casual viewers explore detailed 3D models integrated with structured spatially associated descriptive information. The basic idea is to use a graph-based representation of the information, encoding the preferred order of presentation of information associated to particular views. At run-time, we let users navigate inside the 3D scene, using one of the object-aware controllers presented in previous chapters, while adaptively receiving unobtrusive guidance towards interesting viewpoints and history- and location-dependent suggestions on important information, which is adaptively presented using 2D overlays displayed over the 3D scene. By combining information overlays with guided navigation we obtain an approach that meets many of the requirements set up in Chapter 2.

11.1 Introduction

DIGITAL multimedia content and presentations means are rapidly increasing their sophistication and are now capable in many application domains of describing detailed, complex, three dimensional representations of the physical world. Providing effective 3D content presentations is particularly relevant when the goal is to allow people to appreciate, understand and interact with intrinsically 3D virtual objects. Digital multimedia content and presentations means are rapidly increasing their sophistication and are now

capable in many application domains of describing detailed representations of the physical world. Providing effective 3D exploration experiences is particularly relevant when the goal is to allow people to appreciate, understand and interact with intrinsically 3D virtual objects. Cultural heritage (CH) valorization and Cultural Tourism are among the sectors that benefit most from this evolution, as multimedia technologies provide effective means to cover the pre-visit (documentation), visit (immersion) and post-visit (emotional possession) phases [Econ 11]. In order to effectively support a rich, informative, and engaging experience for the general public, 3D representations should, however, go beyond simple visual replication, supporting information integration/linking, allow shape-related analysis, and providing the necessary semantic information, be it textual or visual, abstract or tangible. Much of this information requires spatial association, as it describes, or can be related to, different spatial contexts. For instance, cultural artifacts are often very complex 3D objects, with subtle material and shape details, presenting information at multiple scales and levels of abstractions (e.g., global shape and carvings). Even the finest material micro-structure carries valuable information (e.g., on the carving process or on the conservation status).

Until recently, the most widespread ways to present information around 3D reconstructions have been through mostly passive visual presentation modalities, such as videos or computer-generated animations. Interest is, however, now shifting towards more flexible active modalities, which let users directly drive exploration of 3D digital artifacts. These active approaches are known to engage museum visitors and enhance the overall visit experience, which tends to be personal, self-motivated, self-paced, and exploratory [Falk 00]. In general, visitors do not want to be overloaded with instructional material, but to receive the relevant information, learn, and have an overall interesting experience. To serve this goal, user-friendly and flexible systems are needed, and many challenges need to be addressed in parallel [Kuf1 11]. Our work, motivated by a project for the museum presentation of cultural heritage objects (see Sec. 11.2) deals with the particular problem of letting casual viewers explore detailed 3D models integrated with structured spatially associated descriptive information in form of overlaid text and images.

Approach Our goal is to let users explore spatially annotated 3D models using a walk-up-and-use user-interface that emphasizes the focus on the work of art. Content preparation, done offline with an authoring tool, organizes descriptive information in an information graph. It should be noted that this graph does not provide a 3D scene description, as the usual scene graph, but is used to structure annotations and relate them to the 3D scene.

Each node associates a subset of the 3D surface seen from a particular viewpoint to the related descriptive annotation, together with its author-defined importance and its user-determined and evolving popularity.

Graph edges describe, instead, the strength of the dependency relation between information nodes, allowing content authors to describe the preferred order of presentation of information.

At run-time, users navigate inside the 3D scene, while adaptively receiving unobtrusive guidance towards interesting viewpoints and history- and location-dependent suggestions on important information, which is adaptively presented using 2D overlays displayed over the 3D scene. The approach is implemented within a scalable system, supporting exploration of information graphs of hundreds of viewpoints associated to massive 3D models, using a variety of GUI setups, from large projection displays to smartphones.

Contribution Our approach, motivated by a real-world visual presentation project in the CH domain, combines and extends state-of-the-art results in several areas. Our main contribution is the flexible integration of stochastic adaptive recommendation system based on a structured spatial information representation, centered around annotated viewpoints, with a walk-up-and-use user interface that provides guidance while being minimally intrusive.

11.2 Overview

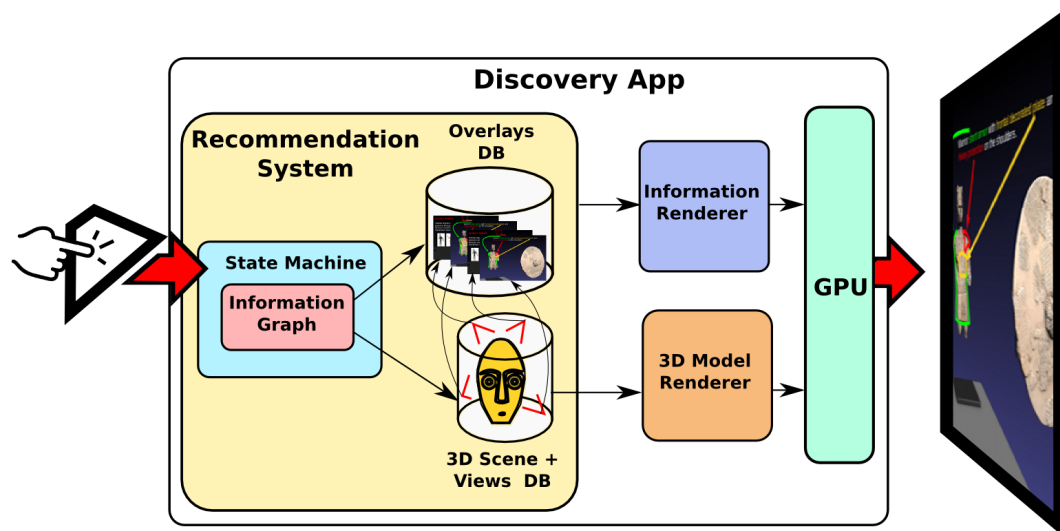


Figure 11.1: System overview. At run-time, users navigate inside the 3D scene, while adaptively receiving unobtrusive guidance towards interesting viewpoints and history and location-dependent suggestions on important information, which is adaptively presented using 2D overlays displayed over the 3D scene.

Requirements R1–R13, as well as our analysis of related work presented in Chapter 3, were used to drive our design process, which resulted in the definition of an approach based on the following concepts:

- **Information graph and authoring** We use a graph of 3D views to represent the various relations between annotations and their spatial position with respect to the 3D model. Each node associates a subset of the 3D surface (ROI) seen from a particular viewpoint to the related descriptive annotation (R11), together with its author-defined importance. Graph edges describe, instead, the strength of the dependency relation between information nodes, allowing content authors to describe the preferred order of presentation of information (R12). The information graph can be created off-line by selecting a reference view for each presented information, drawing an overlay image using standard 2D tools (R13), and indicating dependencies by selecting strong and weak predecessors for each view. This leads to a simple but effective procedure to create spatially relevant rich visual information in forms of linked overlays. Authoring details are orthogonal to the proposed method and are not detailed here. In practice, in this work, we used the exploration system to select the views that are to be annotated, and store snapshots as PNG images. Annotations are then created using a drawing system (libreoffice draw), and exporting the overlays as PNG images. The graph is then created with a simple image browser, that shows images+overlays and defines dependencies by referencing other images, saving the result as an XML file.
- **Exploration** In order to provide an engaging self-paced experience, we let users freely explore 3D models using an interactive camera controller (R6), with a user interface that presents in the main view only the 3D scene of interest (R8). An adaptive recommendation engine based on a state machine runs in parallel with user interaction, and identifies which are the current most interesting information nodes, using a scoring system based on the previous history of visited nodes, the dependency graph and the current user viewpoint (R11, R12, R13), see Fig. 11.1. A suggestion is then stochastically identified among these candidate nodes, with a probability proportional to the score (R12). The non-deterministic choice respects mandatory presentation orders, supporting classic authored storytelling, while introducing variations in the exploration experience providing non-linear information presentation. If the selected information node's view parameters are close enough to the current view, the user is unobtrusively guided towards it by smoothly interpolating camera parameters during interaction towards the

best view (see below). Otherwise, the proposal is visually presented for a limited time to the user in a small inset viewport (R8). If the user accepts it, a small animation is activated to bring the user to the selected target viewpoint. When the user is aligned with the target view, the corresponding textual and visual overlay is displayed on top of the 3D view (R8, R11). This approach avoids the use of a series of hot-spots over the model, which require pointing methods and/or produce clutter (R8). After a suggestion is taken or ignored, the information graph is updated, and a new suggestion is selected based on the new state. The so created story telling path is a non-linear dynamic exploration of the information graph, which is able to provide content in a consistent manner, but with different flavors depending on the user attitude to follow the proposed indications. This approach mimics the experience of a tour with an expert which describes and highlights the parts of the model on which the user is mostly interested.

- **User-interface and device mapping** The proposed approach poses little constraints on the GUI, as it requires only means for controlling the camera and accepting a suggestion (R9). In particular, we do not employ hot-spots (R8) and can rely on incremental controls for camera navigation, as, in particular, we do not require 2D or 3D picking. This makes it possible to implement the method in a variety of settings. In this work, we employ an approach that decouples the devices for interaction and for rendering as to allow for large projection surfaces and enable multiple users to watch the whole screen without occlusion problems and staying at a suitable distance from it when viewing large objects of imposing scale (R8,R9). The widespread diffusion of touch devices, such as tablets or smartphones, has made people used to touch-based user interfaces. While no real standard for 3D touch-based interaction exists [Keef 13], touch surfaces are now so common that people are encouraged to immediately start interacting with them, which is an important aspect of *walk-up-and-use* interfaces. Moreover, even if the mapping between 2D and 3D motions is non-trivial and varies for a user interface to the next, users are encouraged to learn by trial and error while interacting. In this work, we use a 3D variation of the well-known 2D multi-touch RST technique, that allows the simultaneous control of Rotations, Scaling, and Translations from two finger inputs to control a modification of a virtual trackball with auto centering capabilities [Bals 14a]), which provides automatic pivot without requiring precise picking. Accepting a suggestion is mapped to a long press, while rejection automatically occurs upon time-out. The motion of the trackball, in addition, is modified so as to attract the view towards the currently selected best view by applying a small nudge force in

the direction of the currently selected best view, but only use the component which is orthogonal to the current direction of motion (see Sec. 11.4). This helps gently guiding the user towards good viewpoints with associated information.

11.3 The recommendation engine

At the core of our approach is a recommendation engine, running in parallel with user navigation. It is based on a state machine (see Sec. 11.3.2) that evaluates the node contributions and stochastically selects one node with a probability proportional to a context-dependent score that depends on the current spatial position and the navigation history (see Sec. 11.3.3).

11.3.1 Data representation

The information exploited by the recommendation engine are the *3D model*, an *annotated view graph*, the *current viewpoint*, and the *interaction history*. The first two elements are static and provide the scene description, while the two latter ones are dynamic and evolve during navigation.

The 3D model can be any kind of surface model with a renderable representation. In this work, we use multiresolution triangulated surfaces (see Sec. 11.5 for scalability issues).

The view graph describes annotations in a structured form, as described in Sec. 11.2). We denote as $\gamma \in [0..1]$ the author-defined *importance* of each node and as $\omega \in [0..1]$ the dependency weight. Strict dependencies ($\omega = 1$) are useful to model cases where prior information is mandatory (e.g., global introduction is required before presenting some particular detail), while weak dependencies ($\omega < 1$) enable a more adaptive navigation, and if $\omega = 0$ no dependency exists. The descriptive information associated to each node is a 2D overlay image (a semitransparent bitmap or scalable vector graphics with the same aspect ratio of the rendered 3D view). The 2D overlay contains drawings, images or even text which is tightly attached to the object from the node's viewpoint (e.g., imagine a statue with a missing arm and a drawing proposing what could be the missing part, see Fig. 11.2). The 2D ROI consists of a bit-mask denoting the relevant part of the view that contains the information referenced in the textual information. We use this ROI in the ranking process, to identify the 3D region that participates in view similarity computation. All this information is connected to a viewpoint that is also stored in the node in the form of a view matrix. In order to speed-up

view-similarity computation (see Sec. 11.5), we also maintain with each node the bounding box of the 3D points contained in the ROI.

In the course of the navigation, we collect data on the user interaction with the system in order to extract aggregated information. This information is held in nodes attributes that provide aggregated information on user preferences like most visited nodes and the amount of time spent per node, providing new relevance weights for each node which can provide better suggestions to the user during navigation. This information is then exploited by the selection algorithm in order to improve future suggestions (see below).

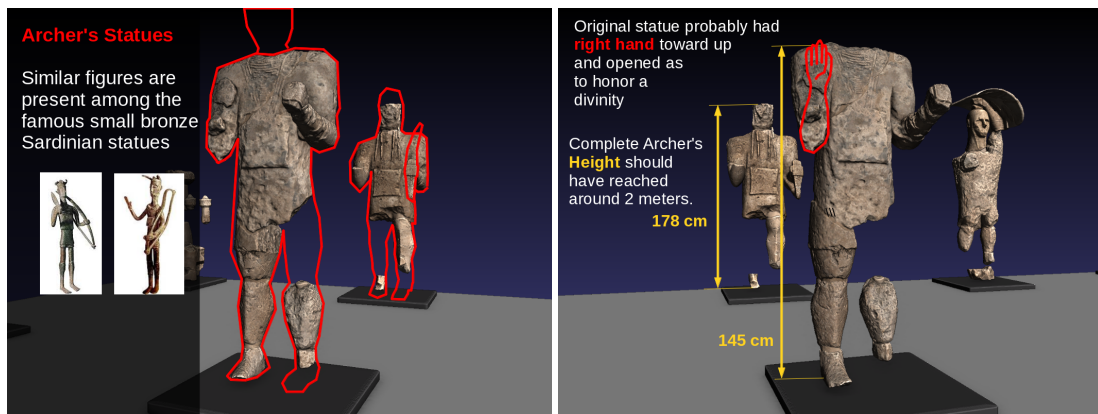


Figure 11.2: Overlaid information. Left: Drawing showing a possible reconstruction of the missing parts of the object; Right: Textual information is presented without cluttering the region of interest.

11.3.2 The recommendation state machine

The state machine (SM), see Fig. 11.3) runs in background while the user can freely move around the scene. The SM proposes specific views depending on the user behavior. On user acceptance the corresponding information is visualized. The SM, using the node graph, is able to produce a sequence of contents which tells a structured consistent story, according to user preferences. The SM states (*discover*, *attract*, *propose*, *go-to*, *show*) are here described in detail:

- **Discover:** it is the start state. Here the SM lets elapse a few seconds to avoid a continuous flow of suggestions, then it looks for a new node to propose (see Sec. 11.3.3). Once the node is selected the state passes to *attract*.
- **Attract:** a hidden attraction force is active while the user explores the scene, trying to drive her toward the active view (see Sec. 11.4). The machine can exit this state after a timeout, and in this case the state changes to *propose*, or if user gets close to the node, and in this case the state changes to *go-to*.

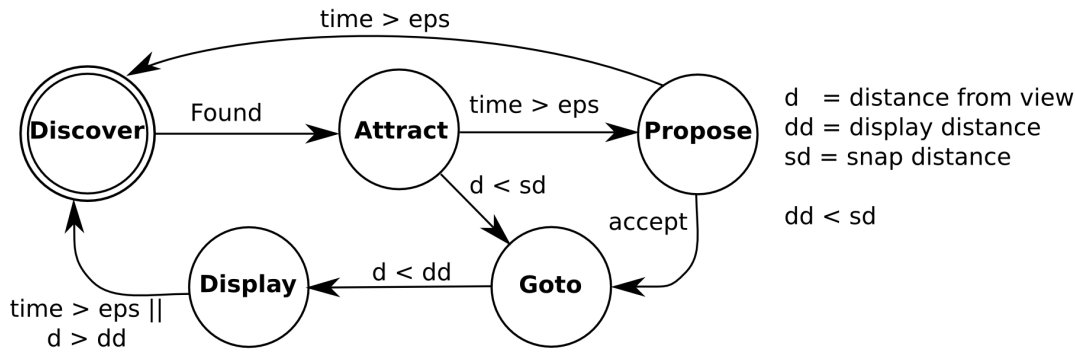


Figure 11.3: State Machine. State machine of the recommendation system.

- **Propose:** a thumbnail with a snapshot of the selected view is proposed to the user. If the user accepts the proposal, the state changes to *go-to*. Otherwise, if the proposed node is not accepted in a given time, the SM gets back to the *discover* state.
- **Go-to:** a small animation is computed and the user is moved to the point of interest associated to the node. After reaching the target point the state changes to *display*.
- **Display:** the information related to the node is displayed. After a time proportional to the length of the textual content is elapsed, or if the user moves away from this position, the state returns to *discover*.

11.3.3 Next best view selection

The adaptive recommendation system aims to guide the user to a structured exploration, taking into account either user movements and author preferences. At given times, the system selects the next best view to be proposed, possibly in the neighborhood of the area currently explored.

11.3.4 Selection algorithm

The selection is performed according to a ranking of all the visitable views in the graph. First of all, the views are partitioned into two sets (visible and invisible), based on view culling with respect to the current view position. At this point each visible view i is compared to the current view, according to the similarity measure described in next paragraph. If this measure is below a given threshold, the view is added to the set of invisible views, otherwise a score is computed according to the following equation:

$$S_i = \gamma_i \times D_i \times R_i \times \sigma_i \tag{11.1}$$

where γ_i is the author defined view relevance, D_i the dependency weight, R_i the recent navigation weight, and σ_i is the view similarity weight. All the weights appearing in Eq. 11.1 are inside the range [0..1]. The dependency weight D_i is a product among all the view dependencies, and it is computed as follows:

$$D_i = \prod_{j=1}^{N_i} (1 - \omega_j \times (1 - vis_j)) \quad (11.2)$$

where N_i the number of dependencies of view i , ω_j is the dependency weight of view j with respect to view i and vis_j is 1 if the view j has been already visited and 0 otherwise. The weight R_i takes into account the user recent navigation: giving lower priority to the views which have been recently displayed, or presented but not accepted. This value is 1 for all not visited and not proposed nodes, otherwise its value is computed by $R_i = \min((\frac{\Delta T}{T_{max}})^2, 1)$, where ΔT is the time elapsed from the last event (propose or visualization), and T_{max} is a time threshold. If at least a view in the visible set has a positive score, the next best view is selected randomly with a probability proportional to the score, otherwise the views inside the invisible set need to be considered. In this case, the scores are computed according to Eq. 11.1, but the view similarity weights employ a metric which is robust to distance, which will be detailed in next paragraphs. At this point, the next best view is selected among the ones with positive score, with a probability proportional to the latter.

11.3.5 View similarity metric for close views

For comparing the current view with respect to *visible* annotated views, we derived a metric based on the fact that two similar views would approximately project the same 3D points to the same image pixels. Therefore, the normalized squared sum of the distances of projected points provides an adequate distance metric for deriving the similarity measure used in Eq. 11.1:

$$\sigma_i = 1 - \frac{\xi^2(K - R) + \sum_{j=1}^R \min((PV_i \mathbf{s}_j - PV_{cur} \mathbf{s}_j)^2, \xi^2)}{K\xi^2} \quad (11.3)$$

where, in the current stochastic sampling composed by K samples, $\mathbf{s}_j = \{\mathbf{s}_1, \dots, \mathbf{s}_R\}$ is the set of R points inside the region of interest of view i , P is the current projection matrix, V_i is the view matrix of the view i , V_{cur} is the current observer view matrix, and ξ^2 is the maximum squared distance between two visible points in the normalized clipping cube.

11.3.6 View similarity metric for distant views

This above similarity measure is reliable when an adequate number of points are visible in the region of interest of view i , but it is not applicable to views outside the view frustum or with only few sample points in it. In these cases, similarity should not be computed in the image plane. Just computing the distance between view matrices, e.g., using L1 or Frobenius norms, is an applicable solution, but would not take into account the distance from the camera to the (average) look-at point. Thus, small variations in camera orientation, that could lead to large variations in image space, would not be captured. This is why we combine in our metric the motion of the viewpoint with motion of the look-at point, considering the eye-target-twist parametrization of viewing transformation, as a quick way to estimate the length of the path needed to reach the view V_i from the current view position V_{cur} . Specifically, the similarity metric is computed in this way:

$$\sigma_i = 1 - \frac{\|e_i - e_{cur}\| + \|t_i - t_{cur}\|}{2\delta} \quad (11.4)$$

where δ is the diagonal of the scene bounding box, e_i, e_{cur} are the eye positions associated to view i and the current view, t_{cur} is the center of visible points from current view, and t_i is computed as $t_i = e_i + \|t_{cur} - e_{cur}\|v_i$, with v_i the viewing direction of view i .

11.4 User interface

The recommendation engine can be integrated in a variety of settings, as, in terms of input, it requires only means for controlling the camera and accepting a suggestion, while, in terms of output, it requires only real-time 3D navigation, suggestion display, and overlay display. In this work, we focus on a museum setting that decouples input and output devices.

11.4.1 Setup and assisted camera control

3D models and associated information are presented on a large display (a back-projection screen in this work), controlled by a touch-enabled surface placed at a suitable distance in front of it. Note that we avoid using the touch-screen to display content-related information, in order to encourage the user to focus on the visualization screen instead of concentrating on the user interface, see Fig. 11.4. An alternative to this setup would be to use 3D devices, e.g., Kinect or Leap Motion, and gestures/posture recognition. Such an implementation, however, is less practical to deploy in crowded museum settings. Camera control is

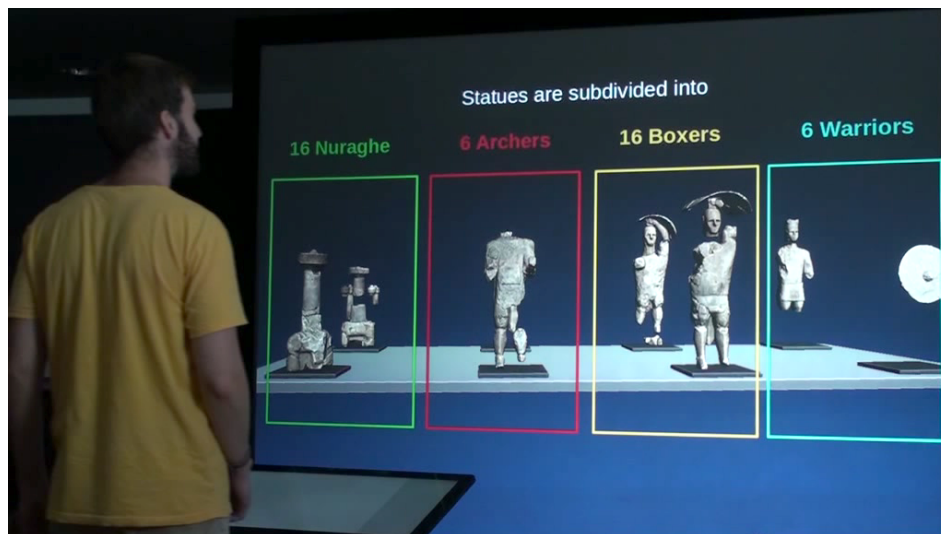


Figure 11.4: Large projection setup. 3D models and associated information are presented on a large display (a back-projection screen in this work), controlled by a touch-enabled surface placed at a suitable distance in front of it. Only incremental controls, without picking, are employed in the user-interface.

implemented through a multi-touch interface controlling the auto-centering virtual trackball. In order to reduce training times, we considered RST multi-touch gestures, as used for 2D actions in commodity products such as smartphones or tablets, and mapped them to analogue 3D behaviors in our constrained controllers. As we deal with statues, we use a fixed up-vector, and map two-finger pinch to dolly-in and dolly-out, two-finger pan to camera panning, and one-finger horizontal motion to orbiting. It should be noted that, similarly to Secord et al. [Seco 11], we deform the motion of the trackball in order to be attracted towards the currently selected view both during pan and rotate and during throwing (i.e., the small period of time after a release). We also add in a small friction force in the neighborhood of the selected view, so as to slow down near good viewpoints, and, when the view is sufficiently close, we snap it to the best view. A long press, instead, is used to accept the displayed suggestion. Suggestion is also accepted when the view similarity d is below a user-defined threshold. This means that when the view is almost similar, the user is automatically moved to the selected node's position, and the related overlay is accepted.

11.4.2 Displaying suggestions

Each time a new view is selected by the state machine, and the user has not moved close enough to trigger automatic acceptance within a small amount of time, a suggestion is displayed to the user. It should be noted that this situation does not occur very often, since the attraction force automatically drives the user

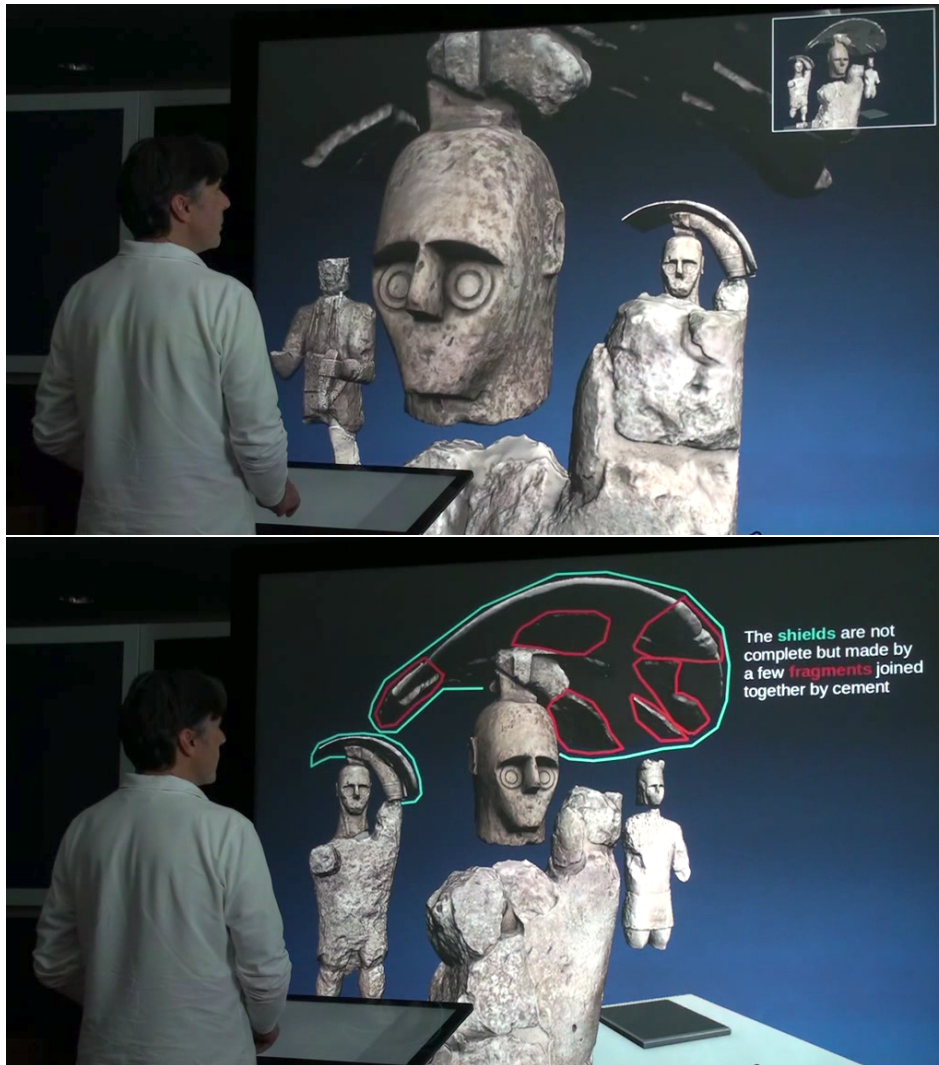


Figure 11.5: Suggestions and overlays. Top: Suggestions are presented in a small inset, using animations to relate them to the spatial context. These suggestions appear only when the attraction forces do not drive the user close enough to the current view. Bottom: when moving close to the currently selected view or accepting a suggestion, annotations are overlaid to 3D view.

towards the currently selected view during interaction. In our current implementation, suggestions are presented in small inset images using animations (see Fig. 11.5 top). First, the inset image fades in a corner of the main 3D view. The initial image presented is a clone of the target view. Then, an animation starts, showing a path from the current view to the target view. This animation is employed to inform the user on the location of the target without cluttering the main 3D view. The target image then remains fixed in the inset for a pre-determined amount of time. If, within this time, the user does not accept it (by moving close to the target or performing a long-press to trigger an automatic go-to), the suggestion is considered ignored, the inset image fades out, and the state machines starts looking for alternatives. In order to reduce distraction, the suggestions appears smoothly, combining fade-in/fade-out animation with

incremental zooming. We plan in the future to investigate less obtrusive methods, e.g., by using mechanisms for better exploiting change blindness events [Inti 02].

11.4.3 Visualizing overlays

When a target node is reached, the associated information is displayed in overlay (see Fig. 11.5 bottom). The information remains visible until the user decides to move to another position.

11.5 Scalability

Both the interactive inspection and the recommendation system require specialized spatial indexing and multiresolution structures and adaptive algorithms to ensure real-time performance on large data-sets (billions of triangles and hundreds of points of interest per scene). The most costly operations are 3D rendering and recommendation computation.

When computing recommendations, the graph is first partitioned in a set of feasible nodes, which are the ones for which predecessors are satisfied. Only these nodes, typically a small subset of the total graph, are checked for similarity. We associate to each node a bounding box, which contains all the points in its ROI, and keep the bounding boxes of potentially visible nodes in a bounding volume hierarchy (BVH). When ranking starts, the BVH is traversed, and nodes are compared with the current view frustum, classifying potentially visible and invisible ones. Potentially visible nodes are pushed in a priority queue, ordered by inverse difference in projected ROI area between target view and current view. Nodes are extracted from this queue one by one, starting with the nodes with most similar projected ROI area, view similarity is computed, and nodes are pushed in a queue sorted by recommendation score, until a small predefined number of nodes is found or there are no more nodes to check. Any node for which similarity is zero is pushed to the invisible set. If the set of nodes for which a score has been computed is non-empty, we stochastically select the suggestion by randomly picking from it with a selection probability proportional to the score. Otherwise, a score is computed for the invisible node, using a linear scan and a fast method that does not require view similarity computation. This approximate technique keeps the number of view similarity computations low in order to maintain interactivity.

View similarity computation is computed on top of the same adaptive multiresolution triangulation [Cign 04] used for rendering. A small random set of 3D points is extracted from the view-adapted tetrahedron graph. This is done

using a simple traversal of the graph leaves, selecting a few points per node. These points are then projected using the view parameters both from the candidate view and the user viewpoint, in order to calculate the average screen space distance between the two point sets and compare it according to Eq. 11.3. Note that only the points falling within the ROI of the target view participate in the score.

11.6 Implementation and User Study

A reference system integrating all techniques described in this paper has been implemented on Linux using OpenGL and Qt 4.7. The hardware setup for the interactive stations was composed of a 2.5m-diagonal back-projection screen, a 3000 ANSI Lumen *projectiondesign F20 SXGA+* projector, and a 27" Dell P2714T optical multi-touch screen, both controlled by a PC with with Ubuntu Linux 14.10, a Intel Core i7-3820 @ 3.6Ghz, with 8GB of RAM and a NVIDIA GTX 680 GPU.

The system, illustrated in the accompanying video, has been tested in a variety of settings. In this paper, we report on tests made using 8 representative models from the Mont'e Prama collection [Bett 14a], for a total of 390M triangles, which have been documented using a graph of 109 information nodes linked by 132 edges. Of these, 12 describe mandatory dependencies ($\omega = 1$), 36 strong dependencies ($\omega = 0.8$), and the remaining weak dependencies ($\omega = 0.2$). The graph is a hierarchical DAG with 5 levels, loosely ordered from general collection-level information to micro-structure description.

In all tests, the models were adaptively rendered using a target resolution of 0.5 triangles/pixel, leading to an average 2.5M triangles/frame and maintaining frame rates never going below 30Hz. The time required in the recommendation engine to generate a new recommendation has been generally less than 5 ms, leading to minimal interaction delays.

In order to assess the actual effectiveness of our approach for improving user satisfaction and increase emotional possession during museum visits, we designed and carried out a preliminary user study to try to quantify the performance, the cognitive load and the satisfaction in comparison with other strategies for associating information to visual exploration.

11.6.1 Goal

Our system is composed by a combination of narrative components [Sege 10] together with a free customized 3D user interface, which make it difficult to evaluate from a user perspective. In theory, for an adequate system evaluation, the

various parts should be considered separately in order to quantify their effects over users [Scho 06]. In our case, we opted to design our user study with the target to try to quantify the user satisfaction, in terms of fun and attractiveness, and the user performance, in terms of effort, learning curve, and information gathering [Diak 11]. To this end, we measured user performance during free explorations together with a simplified version of NASA task load index questionnaire [Liu 13]. Furthermore, we gathered information from think-a-loud comments.

11.6.2 Configurations

Various alternatives of the usage of the system were considered for the experiments: a free exploration interface with adaptive recommendations, and two versions in which the exploration is decorated with a thumbnail-based bar exploration interface [Mart 14]. In one of them the views are ordered according to the authoring importance (weights and dependencies), while in the other one the views are ordered according to the ranking of the recommendation system (views similarity and authoring criteria). In any moment, users could scroll the thumbnail-bar and decide to explore a specific view of the scene. The experimental setup considered the reference system implementation described in section 11.6. All exploration alternatives were operated using the same precise optical touch screen device using a multi-touch device mapping.

11.6.3 Tasks

The experiments consisted in letting users try and enjoy the system using the three different exploration strategies (with adaptive suggestions, with importance sorted thumbnails, and with rank sorted thumbnails) in the context of a free interaction task. We designed our task to measure learning and satisfaction performance in inspections tasks typical of cultural heritage model explorations. Participants were asked to freely explore the model and follow the narrative visualization with the goal of enjoying and acquiring as more useful information as possible.

11.6.4 Participants

For the user analysis, 15 participants (11 males and 4 females, with ages ranging from 28 to 53, mean 40.2 ± 6.9 years) were recruited between employees (including researchers, network administrators and administrative staff).

11.6.5 Design

Each participant tested the three exploration systems in randomized order. Users were first allowed to become familiar with the exploration systems by watching a brief video showing how it works (part of the help system of the museum installation). After the training session, the measured tests consisted of trying the 3 different narrative exploration interfaces for 5 minutes each one. For a complete testing session, users needed 15 minutes. In summary, the complete test design consisted of 15 participants, each one testing the 3 exploration interfaces for a total of 45 complete measurements. At the end of the experiments, participants were also asked to fill a questionnaire comparing the performance of the three systems by indicating a score in a 7-point Likert scale with respect to six factors: mental demand, learning time, physical demand, performance, effort, and frustration level. Since the objective of the tasks was to enjoy the models as to acquire interesting information as much as possible, we asked subjects to quantify as performance level their perception of satisfaction (how much they enjoyed the scene exploration).

11.6.6 Performance evaluation

The following measures were recorded during explorations using the adaptive recommendation system interface (ASI): number of nodes displayed, subdivided in nodes reached through attraction (overlays appearing during exploration), nodes reached through go-to animations, and nodes proposed and ignored during exploration. The subjects visited an average of 20.5 ± 3.1 nodes, of which 4.9 ± 1.6 where refused, 10.2 ± 2.5 where reached during the attract state, and 5.5 ± 2.3 were reached by explicit accept through go-to animation. This means that the adaptive recommendation system appeared to generally show appropriate contents with respect to subjects curiosity, and that in many cases this content appeared transparently during the navigation, without the need of additional inputs which could distract users from interaction. In order to compare the adaptive recommendation system with respect to the thumbnail-bar systems, we also measured for all the interfaces the total number of nodes displayed, and the time that subjects employed for observing overlay information (we assume it to be proportional to the interest to the content displayed), the time that they employed for 3D exploration of the scene, and the time that they employed for scrolling the thumbnail-bars. The number of nodes visited for thumbnail-based interfaces was 17.1 ± 2.9 for the interface with the authoring importance based thumbnail (ITI), while it was 19.4 ± 2.4 for the interface with the ranking based thumbnail (RTI). With respect of measured times, scrolling times were 96.7 ± 24.3 sec. for

ITI, and 86.3 ± 22.7 sec. for RTI, while overlay display times were 97.3 ± 11.5 sec. for ITI, 93.9 ± 15.8 sec. for RTI, and 87.5 ± 14.9 sec. for the adaptive suggestion interface (ASI), and finally 3D exploration times were 106 ± 24.5 sec. for ITI, 119.8 ± 31.5 sec. for RTI, and 212.5 ± 14.9 sec. for ASI. It appears evident that, even if with thumbnail bar interfaces the subjects were able to visit a slightly greater number of annotated views, the scrolling operation took approximately one third of total time, and users often lost the main focus of the 3D exploration.

11.6.7 Work-load evaluation

	ASI	RTI	ITI
Mental demand	1.93 ± 0.88	2.6 ± 0.82	3.0 ± 1.07
Physical demand	1.53 ± 0.64	2.13 ± 0.74	2.73 ± 1.28
Learning time	1.93 ± 1.16	2.87 ± 1.19	2.93 ± 1.1
Performance	6.33 ± 0.81	5.4 ± 1.05	5.4 ± 0.98
Effort	2.2 ± 1.14	2.73 ± 0.8	2.93 ± 1.16
Frustration	2.33 ± 1.35	2.47 ± 1.36	3.07 ± 1.62

Table 11.1: Results of NASA task load index questionnaire.

All factors of the NASA task load index questionnaire were individually analyzed in order to find differences between the three proposed interface. The average values of Likert-scores for the factors are presented in Table 11.1. We noticed a significant effect with respect to physical demand ($p = 0.004$ and $F(2, 42) = 6.23$), and a slight effect with respect to performance ($p = 0.01$ and $F(2, 42) = 4.74$) and mental demand ($p = 0.01$ and $F(2, 42) = 4.35$). We think that subjects considered distracting and physically demanding the scrolling operation on thumbnail bars, especially in the case of the importance based ordering. No significant effects were found with respect the other factors, namely effort, learning time, and frustration, meaning that subjects considered all three interfaces easy to learn and use.

11.6.8 Qualitative evaluation

We also gathered useful hints and suggestions also from think-a-loud comments made by subjects during the tests. In general, users perceived as appealing the overlays decorating the 3D models, and appreciated the transparent attraction force driving them to interesting views, while giving them the chance to freely explore the 3D scene. On the other side, few subjects considered intrusive the attractive force, while others considered the animation inset distracting with respect to 3D exploration. Finally, most users appreciated the adaptive suggestion system, and we noticed that the non-linear graph lead to a significant variability

in node exploration (all subjects carried out different paths and enjoyed different versions of the informative content). We plan to further explore this aspect in future.

11.7 Discussion

We have presented a new method and a scalable representation for letting casual users explore, at their own pace, spatially annotated 3D models. Our evaluation shows that the method appears to be well received and intuitive enough for casual users who quickly understand how to browse statue models in a short trial period. The resulting virtual environment, which combines structured information with a simple interface that does not require precise picking, appears to be well suited both for installations at museums and for interaction on mobile devices. We are currently focusing on improving the proof-of-concept prototype, and planning to perform large-scale tests in museum setting. So far, we mostly focused on the recommendation system, in order to provide meaningful navigation. Our future work will concentrate on improving the assisted navigation subsystem, in order to improve guidance towards interesting viewpoints during free navigation. Since the current evaluation focuses mostly on user satisfaction, more work is required to objectively assess the effectiveness of our user interface. Addressing this would require cognitive measures that are beyond the scope of the paper, and are an important avenue for future work. It will be also interesting to evaluate whether the proposed approach, currently tuned to museum applications, can be extended to more complex situation requiring specific tasks to be solved.

Advantages. Using a graph of views as a basis for information structuring has a number of practical advantages for authors and viewers. In particular, authors can use 2D tools for content preparation, leading to a simple but effective procedure to create spatially relevant rich visual information in forms of overlays, and can use weak dependencies to smoothly transition from constrained sequential presentations (stories) to more flexible independent annotations. The user interface, which automatically selects annotated views, suggesting them and smoothly guiding users towards them, is engaging as it gives users full control on navigation, while being unobtrusive and avoiding the requirements of precise picking, as opposed to the more common hot-spot techniques.

Limitations. This work only targets the problem of 3D model exploration with image/text overlays. Using other associated multimedia information (e.g., video) and/or supporting very complex narratives are orthogonal problems not treated

in this work. Moreover, while the proposed information presentation system is of general use, the proposed camera navigation technique is tuned for object inspection rather than environment walk-throughs. Finally, the current evaluation focuses mostly on user satisfaction. More work is required to objectively assess the effectiveness of the user interface in a variety of settings. Addressing this would require cognitive measures that are beyond the scope of the paper, and are an important avenue for future work.

Scalability. Many components used in the system (i.e., Compact Adaptive Tetra-Puzzles for rendering, HuMoRS camera controlling) have already been demonstrated on desktop and mobile platforms. The computation cost introduced by the recommendation engine together with the guiding component (i.e., attraction force computation) do not represent a barrier for its use on low- and mid-end systems such as mobile and web browser platforms. Nonetheless, recommendation computation scales linearly with the number of visible nodes and thus may impose some performance loss at very populated regions (i.e., hundreds of visible/active information nodes) on limited systems.

11.8 Bibliographical Notes

Most of the content in this chapter is based on paper on paper [Bals 15], where we presented an approach for free exploration of complex annotated 3D models with unobtrusive guidance towards interesting viewpoints.

Part V

Conclusions

In previous chapters we have described the application domain, studied current methods related to our problem domain, and proposed a number of solutions to overcome current limitations.

In this last part, we will summarize and discuss our results, and present our ideas for future work.

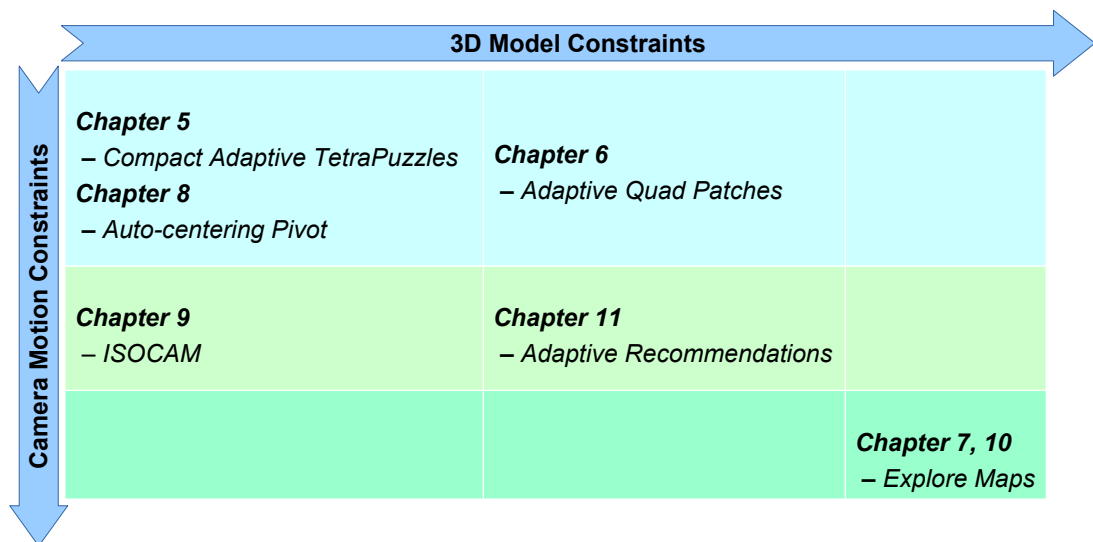


Table 11.2: Method classification. This chart classifies the methods presented in this thesis based on constraints on the input 3D model, and constraints on camera motion.

CHAPTER
12
Summary and Conclusions

This thesis has introduced scalable methods for distribution and rendering of complex 3D models on modern 3D platforms, as well as navigation techniques to aid the user during the exploration of those models at multiple scales. In addition, we have proposed methods for improving understanding of complex annotated 3D models exploiting guided navigation on top of a recommendation engine. This final chapter summarizes the results obtained and briefly discusses future work.

12.1 Conclusions

HIGHLY detailed 3D models are becoming increasingly common and represent a very useful tool for many application domains, both as a communication tool, enabling public access to vast amounts of information, and as a tool for experts, by providing means to classify, store, and analyze highly detailed 3D representations of real objects. However, the visualization of highly detailed 3D objects is a complex problem, requiring simulation of lighting conditions, non-trivial materials, and complex geometry, just to mention some relevant issues that would need to be addressed. There is no single scheme that can cope with all those requirements due to current 3D platform limitations, specially on mobile/Web platforms.

Several approaches in the literature propose solutions to tackle this limitations by introducing constraints, such as limiting camera freedom. These constraints allow for precomputation of a number of parameters that can be exploited during the visualization for providing real-time exploration. One typical constraint is considering objects to be static, which enables the utilization of efficient precomputed spatial subdivision and multiresolution approaches. The work presented in this thesis belongs to this group.

In this thesis, we impose constraints on the input geometry, in order to design compact data structures for improving scalability for distribution and rendering. We also introduce constraints on the camera to develop navigation methods for aiding the user during the exploration of complex 3D models.

With respect to constraints on the input 3D model, we have exploited scene characteristics to design specific algorithms and data representations aiming to address limitations on current 3D platforms. In Chapter 5, we presented a compact GPU-friendly representation for general dense 3D meshes, combining CPU and GPU compression technology for supporting efficient distribution and rendering of extremely detailed models on hardware-constrained mobile devices. In the pursue of better compression, we consider topologically simple 3D models, which can be parametrized into a quad-based domain. Then, exploiting this quad parametrization, in Chapter 6, we produce a fully regular compact image-based multiresolution representation, which relies on standard image algorithms for network distribution and CPU decoding, while most of the work for rendering is moved into the GPU, overcoming CPU performance limitations on scripted environments (i.e., Web browsers).

In Chapter 7, we impose tight constraints on the camera, limiting camera positions to a fixed set of optimized view positions, with associated panoramic views. This approach allows us to exploit image-based rendering by pre-computing all the possible panoramic views using off-line photo-realistic renderers. Thus, enabling interactive exploration of scenes with complex lighting on common 3D platforms.

Imposing constraints on the camera allows us to limit user freedom, in this case, aiming to help the user during the exploration. In Chapter 8, we introduced a free TrackBall camera with automatic pivot computation, simplifying the navigation by reducing the number of gestures required. This approach performed well, specially for expert users, but it is too free for novice users which get easily “lost-in-space”. We tackle this problem, in Chapter 9, by introducing an object-aware approach that provides both smooth transitioning between orbital inspection and proximal surface hovering, while always ensuring “good views”.

In order to provide unobtrusive information exploration, by avoiding cluttering the interface with indications, in Chapter 11, we paired the free Trackball with automatic pivot with an adaptive recommendation system. We further constrain the camera by applying an attraction force towards nearby information, selected by the recommendation engine, thus helping the user in finding contextual information which is presented when the camera gets close to its associated view position.

In Chapter 10, we presented a very constrained image-based navigation system relying on the graph-based representation from Chapter 7. This approach provides a very simple exploration interface, but also limited to the fixed set of view positions and the existing interconnections among them.

The results arisen from our tests proved good both in terms of performance and usability. Using our compact data representation from Chapter 5, we were able to explore $1GTriangle$ models (i.e., David Statue by Michelangelo) on iPhone devices and also on Android devices, with an average rendering throughput of $30Mtriangle/s$ on a 3rd generation iPad and $20Mtriangle/s$ on an Nexus 4. For data streaming, during navigation we reached peaks of $3.3Mbps$ on UMTS/HSPA connection, and $4.8Mbps$ under Wifi network. Using our approach from Chapter 6, we were able to explore 3D models between $8.4Mtriangles$ and $94.4Mtriangles$, on an Acer Iconia Tab W501, with an average frame rate of $37fps$ and a throughput of $34.2Mtriangles/s$, with peaks of $2.8Mbps$ for data fetching. Thus, both approaches provide interactive exploration of large 3D models on platforms with constrained hardware. The image-based rendering approach from Chapter 7 paired with graph-based navigation from Chapter 10 imposes no problems on current 3D platforms typically exceeding $50fps$ during navigation and $20fps$ during video transitions, due to video decoding, both on desktop and mobile platforms.

Our user tests on constrained camera motion controllers also showed interesting results. Typically, expert users prefer the freedom provided with our approach from Chapter 8, while inexperienced users tend to easily get “lost-in-space” thus preferring the more constrained approach from Chapter 9. Also our recommendation system proved well, with users appreciating the free exploration together with the attraction towards nearby information, in particular, thanks to our recommendation engine providing coherent contextual information.

Part of the work presented in this thesis has been integrated in various Museum and exhibition installations having thousands of visitors during the past year (see paper [Mart 14, Gobb 15]).

12.2 Future Work

The approaches presented in this thesis provide a suitable framework for distributing and exploring highly detailed 3D models on current 3D platforms. Our main objective in this thesis was to achieve ubiquitous exploration of complex 3D models, providing solutions to cover most of the wide range of system configurations which can be of use in the application domains.

From the distribution and rendering perspective, there is room for improvement on the compression side. Besides that, we consider that our approaches for compact data representation have proved efficient both in terms of data streaming and real-time rendering, running on desktop and mobile/Web platforms.

Our solution for supporting interactive exploration of scenes with photo-realistic quality is quite constrained. We would like to study wider interconnection between view nodes, and new path definitions to cover more parts of the scene, in order to provide a more flexible representation of the 3D model.

With respect to guided navigation, the free trackball with automatic pivot proved a good solution for experienced users, which appreciated the automatic pivot, while novice users had little problems with the multi-scale navigation. On the contrary, the isosurface navigation camera controller proved to help novice users having a good exploration experience, while experienced users felt too constrained. We plan to study an intermediate approach more similar to the one presented for information discovery, where we use a free trackball with automatic pivot while an attraction force is applied to the camera towards nearby information. Thus, the concept of having a “guidance field” in order to guide camera motion, both to provide good orientations and to attract towards interesting view points, seems a good avenue for future work. Exploiting the constrained camera motion controller for enhancing rendering on light field displays also proved good results, thus we do not plan further research in this direction.

Regarding our approach for graph-based exploration, our current implementation is too constrained. We plan to extend the approach to support more flexible exploration of the scene exploiting the panoramic transition videos and interpolations between neighboring panoramic views. For this, we would benefit from a richer input providing more redundant scene coverage.

Our approach for information discovery and presentation, on the other side, is quite limited and provides a starting point for further research. On this topic, we plan to study more flexible classifications of the information with respect to the spatial constraints, in order to widen the range of information that can be presented during exploration. We also plan to extend our semantic representation to include more flexible dependencies between nodes in order to provide better control on the course of the narration.

12.3 Bibliographical Notes

The scientific results obtained during this PhD work also appeared in related publications, listed below:

- E. Gobbetti, R. Pintus, F. Bettio, F. Marton, M. Agus, and M. Balsa Rodríguez. “Digital Mont’è Prama: dalla digitalizzazione accurata alla valorizzazione di uno straordinario complesso statuario”. *Archeomatica*, 2015. To appear
- M. Balsa Rodríguez, M. Agus, F. Marton, and E. Gobbetti. “Adaptive Recommendations for Enhanced non-linear Exploration of Annotated 3D Objects”. *Computer Graphics Forum*, May 2015. Conditionally accepted to EuroVis 2015
- M. Di Benedetto, F. Ganovelli, M. Balsa Rodríguez, A. Jaspe Villanueva, R. Scopigno, and E. Gobbetti. “ExploreMaps: Efficient Construction and Ubiquitous Exploration of Panoramic View Graphs of Complex 3D Environments”. *Computer Graphics Forum*, Vol. 33, No. 2, pp. 459–468, 2014. Proc. Eurographics 2014
- F. Marton, M. Balsa Rodríguez, F. Bettio, M. Agus, A. Jaspe Villanueva, and E. Gobbetti. “IsoCam: Interactive Visual Exploration of Massive Cultural Heritage Models on Large Projection Setups”. *ACM Journal on Computing and Cultural Heritage*, Vol. 7, No. 2, p. Article 12, June 2014
- F. Marton, M. Agus, E. Gobbetti, G. Pintore, and M. Balsa Rodríguez. “Natural exploration of 3D massive models on large-scale light field displays using the FOX proximal navigation technique”. *Computers & Graphics*, Vol. 36, No. 8, pp. 893–903, December 2012
- M. Balsa Rodríguez, M. Agus, F. Marton, and E. Gobbetti. “HuMoRS: Huge models Mobile Rendering System”. In: *Proc. ACM Web3D International Symposium*, ACM Press, New York, NY, USA, August 2014
- M. Balsa Rodríguez, E. Gobbetti, F. Marton, and A. Tinti. “Coarse-grained Multiresolution Structures for Mobile Exploration of Gigantic Surface Models”. In: *Proc. SIGGRAPH Asia Symposium on Mobile Graphics and Interactive Applications*, pp. 4:1–4:6, ACM, November 2013
- M. Balsa Rodríguez, E. Gobbetti, F. Marton, and A. Tinti. “Compression-domain Seamless Multiresolution Visualization of Gigantic Meshes on Mobile Devices”. In: *Proc. ACM Web3D International Symposium*, pp. 99–107, ACM Press, June 2013
- E. Gobbetti, F. Marton, M. Balsa Rodríguez, F. Ganovelli, and M. Di Benedetto. “Adaptive Quad Patches: an Adaptive Regular Structure for Web Distribution and Adaptive Rendering of 3D Models”. In: *Proc. ACM Web3D International Symposium*, pp. 9–16, ACM Press, New York, NY, USA, August 2012. (Best Long Paper Award)

Bibliography

- [Agus 08] M. Agus, E. Gobbetti, J. A. I. Guitián, F. Marton, and G. Pintore. “GPU Accelerated Direct Volume Rendering on an Interactive Light Field Display”. *Computer Graphics Forum*, Vol. 27, No. 2, pp. 231–240, 2008.
- [Andu 07] C. Andujar, J. Boo, P. Brunet, M. Fairén, I. Navazo, P. Vázquez, and A. Vinacua. “Omni-directional Relief Impostors”. *Computer Graphics Forum*, Vol. 26, No. 3, pp. 553–560, Sep. 2007.
- [Andu 12] C. Andujar, A. Chica, and P. Brunet. “Cultural Heritage: User-interface design for the Ripoll Monastery exhibition at the National Art Museum of Catalonia”. *Computers and Graphics*, Vol. 36, No. 1, pp. 28–37, 2012.
- [Babo 06] L. Baboud and X. Décoret. “Rendering geometry with relief textures”. In: C. Gutwin and S. Mann, Eds., *Graphics Interface*, pp. 195–201, 2006.
- [Bade 05] R. Bade, F. Ritter, and B. Preim. “Usability comparison of mouse-based interaction techniques for predictable 3d rotation”. In: *Proc. Smart Graphics*, pp. 138–150, Springer, 2005.
- [Bals 12a] M. Balsa Rodríguez, E. Gobbetti, F. Marton, R. Pintus, G. Pintore, and A. Tinti. “Interactive Exploration of Gigantic Point Clouds on Mobile Devices”. In: D. Arnold, J. Kaminski, F. Niccolucci, and A. Stork, Eds., *VAST: International Symposium on Virtual Reality, Archaeology and Intelligent Cultural Heritage*, The Eurographics Association, 2012.
- [Bals 12b] M. Balsa Rodríguez and P. Vazquez Alcocer. “Practical Volume Rendering in mobile devices”. In: *Proc. International Symposium on Visual Computing*, pp. 708–718, Springer Verlag, 2012.
- [Bals 13a] M. Balsa Rodríguez, E. Gobbetti, J. Iglesias Guitián, M. Makhinya, F. Marton, R. Pajarola, and S. Suter. “A Survey of Compressed GPU-based Direct Volume Rendering”. In: *Eurographics State-of-the-art Report*, pp. 117–136, May 2013.

- [Bals 13b] M. Balsa Rodríguez, E. Gobbetti, F. Marton, and A. Tinti. “Coarse-grained Multiresolution Structures for Mobile Exploration of Gigantic Surface Models”. In: *Proc. SIGGRAPH Asia Symposium on Mobile Graphics and Interactive Applications*, pp. 4:1–4:6, ACM, November 2013.
- [Bals 13c] M. Balsa Rodríguez, E. Gobbetti, F. Marton, and A. Tinti. “Compression-domain Seamless Multiresolution Visualization of Gigantic Meshes on Mobile Devices”. In: *Proc. ACM Web3D International Symposium*, pp. 99–107, ACM Press, June 2013.
- [Bals 14a] M. Balsa Rodríguez, M. Agus, F. Marton, and E. Gobbetti. “HuMoRS: Huge models Mobile Rendering System”. In: *Proc. ACM Web3D International Symposium*, ACM Press, New York, NY, USA, August 2014.
- [Bals 14b] M. Balsa Rodríguez, E. Gobbetti, J. Iglesias Guitián, M. Makhinya, F. Marton, R. Pajarola, and S. Suter. “State-of-the-art in Compressed GPU-Based Direct Volume Rendering”. *Computer Graphics Forum*, Vol. 33, No. 6, pp. 77–100, September 2014.
- [Bals 15] M. Balsa Rodríguez, M. Agus, F. Marton, and E. Gobbetti. “Adaptive Recommendations for Enhanced non-linear Exploration of Annotated 3D Objects”. *Computer Graphics Forum*, May 2015. Conditionally accepted to EuroVis 2015.
- [Beso 08] I. Besora, P. Brunet, M. Callieri, A. Chica, M. Corsini, M. Dellepiane, D. Morales, J. Moyés, G. Ranzuglia, and R. Scopigno. “Portalada: A Virtual Reconstruction of the Entrance of the Ripoll Monastery”. In: *Proc. 3DPVT*, pp. 89–96, June 2008.
- [Bett 13] F. Bettio, E. Gobbetti, E. Merella, and R. Pintus. “Improving the digitization of shape and color of 3D artworks in a cluttered environment”. In: *Proc. Digital Heritage*, October 2013. To appear.
- [Bett 14a] F. Bettio, A. Jaspe Villanueva, E. Merella, F. Marton, E. Gobbetti, and R. Pintus. “Mont’e Scan: Effective Shape and Color Digitization of Cluttered 3D Artworks”. *ACM JOCCH*, Vol. 8, No. 1, p. Article 4, 2014.
- [Bett 14b] F. Bettio, A. Jaspe Villanueva, E. Merella, R. Pintus, F. Marton, and E. Gobbetti. “Mont’e Scan: Effective shape and color digitization of cluttered 3D artworks”. *Submitted for publication*, 2014.

- [Blum 11] A. Blume, W. Chun, D. Kogan, V. Kokkevis, N. Weber, R. Petterson, and R. Zeiger. "Google Body: 3D human anatomy in the browser". In: *ACM SIGGRAPH 2011 Talks*, p. 19, ACM, 2011.
- [Bolc 07] C. Bolchini, C. A. Curino, E. Quintarelli, F. A. Schreiber, and L. Tanca. "A Data-oriented Survey of Context Models". *SIGMOD Rec.*, Vol. 36, No. 4, pp. 19–26, Dec. 2007.
- [Borg 05] L. Borgeat, G. Godin, F. Blais, P. Massicotte, and C. Lahanier. "GoLD: interactive display of huge colored and textured models". *ACM Trans. Graph.*, Vol. 24, No. 3, pp. 869–877, 2005.
- [Boub 05] T. Boubekur and C. Schlick. "Generic Mesh Refinement on GPU". In: *Graphics Hardware 2005*, pp. 99–104, July 2005.
- [Boub 08] T. Boubekur and C. Schlick. "A Flexible Kernel for Adaptive Mesh Refinement on GPU". *Computer Graphics Forum*, Vol. 27, No. 1, pp. 102–114, 2008.
- [Bowm 03] D. A. Bowman, C. North, J. Chen, N. F. Polys, P. S. Pyla, and U. Yilmaz. "Information-rich virtual environments: theory, tools, and research agenda". In: *Proc. ACM VRST*, pp. 81–90, ACM, 2003.
- [Burt 02] N. Burtnyk, A. Khan, G. Fitzmaurice, R. Balakrishnan, and G. Kurtenbach. "StyleCam: interactive stylized 3D navigation using integrated spatial and temporal controls". In: *Proc. ACM UIST*, pp. 101–110, ACM, 2002.
- [Burt 06] N. Burtnyk, A. Khan, G. Fitzmaurice, and G. Kurtenbach. "ShowMotion: camera motion based 3D design review". In: *Proc. ACM I3D*, pp. 167–174, ACM, 2006.
- [Butk 11] T. Butkiewicz and C. Ware. "Multi-touch 3D exploratory analysis of ocean flow models". In: *Proc. IEEE Oceans*, IEEE, 2011.
- [Call 08] M. Callieri, F. Ponchio, P. Cignoni, and R. Scopigno. "Virtual inspector: A flexible visualizer for dense 3D scanned models". *IEEE Computer Graphics and Applications*, Vol. 28, No. 1, pp. 44–54, 2008.
- [Call 13] M. Callieri, C. Leoni, M. Dellepiane, and R. Scopigno. "Artworks narrating a story: a modular framework for the integrated presentation of three-dimensional and textual contents". In: *Proc. ACM Web3D*, pp. 167–175, ACM Press, June 2013.
- [Calv 02] D. Calver. "Vertex decompression in a shader". *ShaderX: Vertex and Pixel Shader Tips and Tricks*, pp. 172–187, 2002.

- [Capi 08] T. Capin, K. Pulli, and T. Akenine-Moller. "The state of the art in mobile graphics research". *IEEE Computer Graphics and Applications*, Vol. 28, No. 4, pp. 74–84, 2008.
- [Chen 88] M. Chen, S. J. Mountford, and A. Sellen. "A study in interactive 3-D rotation using 2-D control devices". In: *Proc. SIGGRAPH*, pp. 121–129, ACM, 1988.
- [Chen 95] S. Chen. "Quicktime VR: An image-based approach to virtual environment navigation". In: *Proc. SIGGRAPH*, pp. 29–38, 1995.
- [Chhu 07] J. Chhugani and S. Kumar. "Geometry engine optimization: cache friendly compressed representation of geometry". In: *Proceedings of the 2007 symposium on Interactive 3D graphics and games*, pp. 9–16, ACM, New York, NY, USA, 2007.
- [Chri 08] M. Christie, P. Olivier, and J.-M. Normand. "Camera Control in Computer Graphics". *Computer Graphics Forum*, Vol. 27, No. 8, 2008.
- [Chri 09] M. Christie and P. Olivier. "Camera control in computer graphics: models, techniques and applications". In: *ACM SIGGRAPH ASIA Courses*, pp. 3:1–3:197, ACM, 2009.
- [Cign 04] P. Cignoni, F. Ganovelli, E. Gobbetti, F. Marton, F. Ponchio, and R. Scopigno. "Adaptive tetrapuzzles: efficient out-of-core construction and visualization of gigantic multiresolution polygonal models". *ACM Transactions on Graphics*, Vol. 23, No. 3, pp. 796–803, 2004.
- [Cign 05] P. Cignoni, F. Ganovelli, E. Gobbetti, F. Marton, F. Ponchio, and R. Scopigno. "Batched Multi Triangulation". In: *Proc. IEEE Visualization*, pp. 207–214, 2005.
- [Cign 07] P. Cignoni, M. Di Benedetto, F. Ganovelli, E. Gobbetti, F. Marton, and R. Scopigno. "Ray-Casted BlockMaps for Large Urban Visualization". *Computer Graphics Forum*, Vol. 26, No. 3, Sept. 2007.
- [Coff 12] D. Coffey, N. Malbraaten, T. B. Le, I. Borazjani, F. Sotiropoulos, A. G. Erdman, and D. F. Keefe. "Interactive Slice WIM: Navigating and Interrogating Volume Data Sets Using a Multisurface, Multitouch VR Interface". *IEEE Transactions on Visualization and Computer Graphics*, Vol. 18, No. 10, pp. 1614–1626, 2012.
- [Coma 02] D. Comaniciu and P. Meer. "Mean shift: A robust approach toward feature space analysis". *IEEE Trans. Pattern Anal. Mach. Intell.*, Vol. 24, No. 5, pp. 603–619, 2002.

- [Cram 09] M. Crampes, J. de Oliveira-Kumar, S. Ranwez, and J. Villerd. "Visualizing Social Photos on a Hasse Diagram for Eliciting Relations and Indexing New Photos". *IEEE Transactions on Visualization and Computer Graphics*, Vol. 15, No. 6, pp. 985–992, Nov. 2009.
- [Decl 09] F. Declé and M. Hachet. "A study of direct versus planned 3d camera manipulation on touch-based mobile phones". In: *Proc. MobileHCI*, pp. 32–35, ACM, 2009.
- [Di B 10] M. Di Benedetto, F. Ponchio, F. Ganovelli, and R. Scopigno. "SpiderGL: A JavaScript 3D Graphics Library for Next-Generation WWW". In: *Web3D 2010. 15th Conference on 3D Web technology*, 2010.
- [Di B 14] M. Di Benedetto, F. Ganovelli, M. Balsa Rodríguez, A. Jaspe Villanueva, R. Scopigno, and E. Gobbetti. "ExploreMaps: Efficient Construction and Ubiquitous Exploration of Panoramic View Graphs of Complex 3D Environments". *Computer Graphics Forum*, Vol. 33, No. 2, pp. 459–468, 2014. Proc. Eurographics 2014.
- [Diak 11] N. Diakopoulos. "Design Challenges in Playable Data". In: *CHI Workshop on Gamification*, 2011.
- [Dyke 09] C. Dyken, M. Reimers, and J. Seland. "Semi-Uniform Adaptive Patch Tessellation". *Computer Graphics Forum*, Vol. 28, No. 8, pp. 2255–2263, Dec. 2009.
- [Econ 11] M. Economou and E. Meintani. "Promising beginnings? Evaluating museum mobile phone apps". In: *Proc. Rethinking Technology in Museums Conference*, pp. 26–27, 2011.
- [Epsht 07] B. Epshtein, E. Ofek, Y. Wexler, and P. Zhang. "Hierarchical photo organization using geo-relevance". In: *Proc. ACM GIS*, pp. 18:1–18:7, ACM, 2007.
- [Falk 00] H. J. Falk and L. D. Dierking. *Learning from Museums: Visitor Experience and the Making of Meaning*. Rowman & Littlefield, 2000.
- [Fara 97] P. Faraday and A. Sutcliffe. "Designing effective multimedia presentations". In: *Proc. ACM SIGCHI*, pp. 272–278, ACM, 1997.
- [Fili 11] S. Filippini-Fantoni, S. McDaid, and M. Cock. "Mobile devices for orientation and way finding: the case of the British Museum multimedia guide". In: *Proc. Museums and the Web*, 2011.
- [Fitz 08] G. Fitzmaurice, J. Matejka, I. Mordatch, A. Khan, and G. Kurtenbach. "Safe 3D navigation". In: *Proc. ACM I3D*, pp. 7–15, ACM, 2008.

- [Flei 00] S. Fleishman, D. Cohen-Or, and D. Lischinski. "Automatic Camera Placement for Image-Based Modeling". *Computer Graphics Forum*, Vol. 19, No. 2, pp. 101–110, 2000.
- [Floa 05] M. S. Floater and K. Hormann. "Surface Parameterization: a Tutorial and Survey". In: *Adv. in Multires. for Geom. Model.*, pp. 157–186, Springer, 2005.
- [Fu 08] H. Fu, D. Cohen-Or, G. Dror, and A. Sheffer. "Upright orientation of man-made objects". In: *ACM Trans. Graph.*, p. 42, 2008.
- [Fu 10] C.-W. Fu, W.-B. Goh, and J. A. Ng. "Multi-touch techniques for exploring large-scale 3D astrophysical simulations". In: *Proc. ACM SIGCHI*, pp. 2213–2222, ACM Press, 2010.
- [Girg 09] A. Girgensohn, F. Shipman, L. Wilcox, T. Turner, and M. Cooper. "MediaGLOW: organizing photos in a graph-based workspace". In: *Proc. ACM IUI*, pp. 419–424, ACM, 2009.
- [Gobb 04a] E. Gobbetti and F. Marton. "Layered Point Clouds". In: *Proc. Eurographics Symposium on Point Based Graphics*, pp. 113–120, 2004.
- [Gobb 04b] E. Gobbetti and F. Marton. "Layered Point Clouds: A Simple and Efficient Multiresolution Structure for Distributing and Rendering Gigantic Point-Sampled Models". *Computers & Graphics*, Vol. 28, No. 1, pp. 815–826, 2004.
- [Gobb 12] E. Gobbetti, F. Marton, M. Balsa Rodríguez, F. Ganovelli, and M. Di Benedetto. "Adaptive Quad Patches: an Adaptive Regular Structure for Web Distribution and Adaptive Rendering of 3D Models". In: *Proc. ACM Web3D International Symposium*, pp. 9–16, ACM Press, New York, NY, USA, August 2012. (Best Long Paper Award).
- [Gobb 15] E. Gobbetti, R. Pintus, F. Bettio, F. Marton, M. Agus, and M. Balsa Rodríguez. "Digital Mont'e Prama: dalla digitalizzazione accurata alla valorizzazione di uno straordinario complesso statuario". *Archeomatica*, 2015. To appear.
- [Gosw 13] P. Goswami, F. Erol, R. Mukhi, R. Pajarola, and E. Gobbetti. "An Efficient Multi-resolution Framework for High Quality Interactive Rendering of Massive Point Clouds using Multi-way kd-Trees". *The Visual Computer*, Vol. 29, No. 1, pp. 69–83, 2013.

- [Gotz 07] T. Götzelmann, P.-P. Vázquez, K. Hartmann, A. Nürnberger, and T. Strothotte. "Correlating Text and Images: Concept and Evaluation". In: *Proc. Smart Graphics*, pp. 97–109, Springer-Verlag, Berlin, Heidelberg, 2007.
- [Gu 02] X. Gu, S. J. Gortler, and H. Hoppe. "Geometry images". *ACM Trans. Graph.*, Vol. 21, No. 3, pp. 355–361, 2002.
- [Guth 05] M. Guthe, A. Balázs, and R. Klein. "GPU-based trimming and tessellation of NURBS and T-Spline surfaces". *ACM Transactions on Graphics*, Vol. 24, No. 3, pp. 1016–1023, Aug. 2005.
- [Hach 13] M. Hachet, J.-B. de la Rivière, J. Laviolle, A. CohÃI, and S. Cursan. "Touch-Based Interfaces for Interacting with 3D Content in Public Exhibitions". *IEEE Computer Graphics and Applications*, Vol. 33, No. 2, pp. 80–85, March 2013.
- [Hanc 07] M. Hancock, S. Carpendale, and A. Cockburn. "Shallow-depth 3D interaction: design and evaluation of one-, two- and three-touch techniques". In: *Proc. ACM SIGCHI*, pp. 1147–1156, ACM, 2007.
- [Hanc 09] M. Hancock, O. Hilliges, C. Collins, D. Baur, and S. Carpendale. "Exploring tangible and direct touch interfaces for manipulating 2D and 3D information on a digital table". In: *Proc. ITS*, pp. 77–84, ACM, 2009.
- [Hans 97] A. J. Hanson and E. A. Wernert. "Constrained 3D navigation with 2D controllers". In: *Proc. IEEE Visualization*, pp. 175–182, IEEE Computer Society Press, 1997.
- [Hega 11] M. Hegarty. "The Cognitive Science of Visual-Spatial Displays: Implications for Design". *Topics in Cognitive Science*, Vol. 3, No. 3, pp. 446–474, 2011.
- [Henr 04] K. Henriksen, J. Sporring, and K. Hornbæk. "Virtual Trackballs Revisited". *IEEE Transactions on Visualization and Computer Graphics*, Vol. 10, No. 2, pp. 206–216, March 2004.
- [Hopp 97] H. Hoppe. "View-dependent refinement of progressive meshes". In: *Proc. ACM SIGGRAPH*, pp. 189–198, 1997.
- [Hu 10] L. Hu, P. Sander, and H. Hoppe. "Parallel view-dependent level-of-detail control". *IEEE Trans. on Visualization and Computer Graphic*, 2010.

- [Inti 02] S. S. Intille. "Change blind information display for ubiquitous computing environments". In: *Proc. UbiComp*, pp. 91–106, Springer-Verlag Berlin Heidelberg, 2002.
- [Isen 12] T. Isenberg and M. Hancock. "Gestures vs. Postures: Gestural Touch Interaction in 3D Environments". In: *Proc. 3DCHI*, pp. 53–61, 2012.
- [ISTI 12] ISTI-CNR Visual Computing Lab. "MeshLab for iOS: A powerful easy-to-use 3D mesh viewer for iPad and iPhone". www.meshpad.org, 2012.
- [Jang 09] C. Jang, T. Yoon, and H.-G. Cho. "A smart clustering algorithm for photo set obtained from multiple digital cameras". In: *Proc. ACM SAC*, pp. 1784–1791, ACM, 2009.
- [Jank 10] J. Jankowski, K. Samp, I. Irzynska, M. Jozwicz, and S. Decker. "Integrating Text with Video and 3D Graphics: The Effects of Text Drawing Styles on Text Readability". In: *Proc. ACM SIGCHI*, pp. 1321–1330, ACM, 2010.
- [Jank 12] J. Jankowski and S. Decker. "A dual-mode user interface for accessing 3D content on the world wide web". In: *Proc. WWW*, pp. 1047–1056, ACM, 2012.
- [Jank 13] J. Jankowski and M. Hachet. "A Survey of Interaction Techniques for Interactive 3D Environments". In: *Eurographics STAR*, 2013.
- [Jin 12] Y. Jin, Q. Wu, and L. Liu. "Unsupervised upright orientation of man-made models". *Graphical Models*, 2012.
- [Jone 07] A. Jones, I. McDowall, H. Yamada, M. T. Bolas, and P. E. Debevec. "Rendering for an interactive 360 degree light field display". *ACM Trans. Graph*, Vol. 26, No. 3, pp. 40–40, 2007.
- [Jova 08] B. Jovanova, M. Preda, and F. Preteux. "MPEG-4 Part 25: A Generic Model for 3D Graphics Compression". In: *Proc. 3DTV*, pp. 101–104, IEEE, 2008.
- [Jova 09] B. Jovanova, M. Preda, and F. Preteux. "MPEG-4 Part 25: A graphics compression framework for XML-based scene graph formats". *Image Commun.*, Vol. 24, No. 1-2, pp. 101–114, 2009.
- [Kazh 06] M. Kazhdan, M. Bolitho, and H. Hoppe. "Poisson surface reconstruction". In: *Proc. SGP*, pp. 61–70, 2006.

- [Keef 13] D. Keefe and T. Isenberg. "Reimagining the Scientific Visualization Interaction Paradigm". *Computer*, Vol. 46, No. 5, pp. 51–57, May 2013.
- [Khan 05] A. Khan, B. Komalo, J. Stam, G. Fitzmaurice, and G. Kurtenbach. "HoverCam: interactive 3D navigation for proximal object inspection". In: *Proc. I3D*, pp. 73–80, ACM, 2005.
- [Khod 03] A. Khodakovsky, N. Litke, and P. Schröder. "Globally smooth parameterizations with low distortion". *ACM Trans. Graph.*, Vol. 22, No. 3, pp. 350–357, 2003.
- [Kimb 01] D. Kimber, J. Foote, and S. Lertsithichai. "Flyabout: spatially indexed panoramic video". In: *Proc. ACM Multimedia*, pp. 339–347, 2001.
- [Klei 12] T. Klein, F. Guéniat, L. Pastur, F. Vernier, and T. Isenberg. "A Design Study of Direct-Touch Interaction for Exploratory 3D Scientific Visualization". *Computer Graphics Forum*, Vol. 31, pp. 1225–1234, 2012.
- [Knoe 11] S. Knoedel and M. Hachet. "Multi-touch RST in 2D and 3D spaces: Studying the impact of directness on user performance". In: *Proc. ACM 3DUI*, pp. 75–78, ACM, 2011.
- [Kopf 10] J. Kopf, B. Chen, R. Szeliski, and M. Cohen. "Street slide: browsing street level imagery". In: *Proc. SIGGRAPH*, pp. 96:1–96:8, 2010.
- [Krae 04] V. Kraevoy and A. Sheffer. "Cross-parameterization and compatible remeshing of 3D models". *ACM Trans. Graph.*, Vol. 23, No. 3, pp. 861–869, 2004.
- [Krat 10] S. Kratz and M. Rohs. "Extending the virtual trackball metaphor to rear touch input". In: *Proc. IEEE 3DUI*, pp. 111–114, IEEE, 2010.
- [Kufli 11] T. Kuflik, O. Stock, M. Zancanaro, A. Gorfinkel, S. Jbara, S. Kats, J. Sheidin, and N. Kashtan. "A visitor's guide in an active museum: Presentations, communications, and reflection". *Journal on Computing and Cultural Heritage (JOCCH)*, Vol. 3, No. 3, pp. 11:1–11:25, 2011.
- [Kurt 97] G. Kurtenbach, G. Fitzmaurice, T. Baudel, and B. Buxton. "The design of a GUI paradigm based on tablets, two-hands, and transparency". In: *Proc. ACM SIGCHI*, pp. 35–42, ACM, ACM, 1997.
- [Lab 09] L. Lab. "1 Billion Hours, 1 Billion Dollars Served: Second Life Celebrates Major Milestones for Virtual Worlds". <http://lindenlab>.

- [com/pressroom/releases/22_09_09](http://www.ibm.com/pressroom/releases/22_09_09), 2009. Retrieved on 15 Sep. 2010.
- [Laut 07] C. Lauterbach, S.-E. Yoon, and D. Manocha. “Ray-Strips: A Compact Mesh Representation for Interactive Ray Tracing”. In: *Proc. IEEE/EG Symposium on Interactive Ray Tracing*, pp. 19–26, IEEE Computer Society, 2007.
- [Lee 09] H. Lee, G. Lavoué, and F. Dupont. “Adaptive Coarse-to-Fine Quantization for Optimizing Rate-distortion of Progressive Mesh Compression”. In: *Proc. VMV*, pp. 73–82, 2009.
- [Lee 10] J. Lee, S. Choe, and S. Lee. “Compression of 3D Mesh Geometry and Vertex Attributes for Mobile Graphics”. *Journal of Computing Science and Engineering*, Vol. 4, No. 3, pp. 207–224, 2010.
- [Lee 98] A. W. F. Lee, W. Sweldens, P. Schröder, L. Cowsar, and D. Dobkin. “MAPS: Multiresolution Adaptive Parameterization of Surfaces”. *Comp. Graph. Proc.*, pp. 95–104, 1998.
- [Lipp 80] A. Lippman. “Movie-maps: An application of the optical videodisc to computer graphics”. *Proc. SIGGRAPH*, Vol. 14, pp. 32–42, July 1980.
- [Liu 13] Y. Liu, S. Barlowe, Y. Feng, J. Yang, and M. Jiang. “Evaluating exploratory visualization systems: A user study on how clustering-based visualization systems support information seeking from large document collections”. *Information Visualization*, Vol. 12, No. 1, pp. 25–43, 2013.
- [Liu 14] S. Liu, W. Cui, Y. Wu, and M. Liu. “A survey on information visualization: recent advances and challenges”. *The Visual Computer*, Jan. 2014.
- [Lloy 82] S. Lloyd. “Least squares quantization in PCM”. *IEEE Trans. Inf. Theory*, Vol. 28, No. 2, pp. 129 – 137, mar 1982.
- [Lueb 97] D. Luebke and C. Erikson. “View-dependent simplification of arbitrary polygonal environments”. In: *Proc. ACM SIGGRAPH*, pp. 199–208, 1997.
- [Lund 11] C. Lundstrom, T. Rydell, C. Forsell, A. Persson, and A. Ynnerman. “Multi-Touch Table System for Medical Visualization: Application to Orthopedic Surgery Planning”. *IEEE Transactions on Visualization and Computer Graphics*, Vol. 17, No. 12, pp. 1775–1784, Dec. 2011.

- [Magl 10] A. Maglo, H. Lee, G. Lavoué, C. Mouton, C. Hudelot, and F. Dupont. “Remote scientific visualization of progressive 3D meshes with X3D”. In: *Proc. Web3D*, pp. 109–116, 2010.
- [Malv 08] H. S. Malvar, G. J. Sullivan, and S. Srinivasan. “Lifting-Based Reversible Color Transformations for Image Compression”. In: *SPIE Applications of Digital Image Processing*, International Society for Optical Engineering, August 2008.
- [Manu 05] F. P. Manuel M.Oliveira. “An Efficient Representation for Surface Details”. Tech. Rep. RP 351, Universidade Federal do Rio Grande, January 2005.
- [Mari 12] P. Marion. “Point Cloud Streaming to Mobile Devices with Real-time Visualization”. www.pointclouds.org, 2012.
- [Mart 09] A. Martinet, G. Casiez, and L. Grisoni. “3D positioning techniques for multi-touch displays”. In: *Proc. ACM VRST*, pp. 227–228, ACM Press, 2009.
- [Mart 12a] A. Martinet, G. Casiez, and L. Grisoni. “Integrality and Separability of Multitouch Interaction Techniques in 3D Manipulation Tasks”. *IEEE Transactions on Visualization and Computer Graphics*, Vol. 18, No. 3, pp. 369–380, March 2012.
- [Mart 12b] F. Marton, M. Agus, E. Gobbetti, G. Pintore, and M. Balsa Rodríguez. “Natural exploration of 3D massive models on large-scale light field displays using the FOX proximal navigation technique”. *Computers & Graphics*, Vol. 36, No. 8, pp. 893–903, December 2012.
- [Mart 14] F. Marton, M. Balsa Rodríguez, F. Bettio, M. Agus, A. Jaspe Villanueva, and E. Gobbetti. “IsoCam: Interactive Visual Exploration of Massive Cultural Heritage Models on Large Projection Setups”. *ACM Journal on Computing and Cultural Heritage*, Vol. 7, No. 2, p. Article 12, June 2014.
- [Mart 79] G. N. N. Martin. “Range encoding: an algorithm for removing redundancy from a digitised message”. In: *Video and Data Recording Conference*, 1979.
- [McCr 09] J. McCrae, I. Mordatch, M. Glueck, and A. Khan. “Multiscale 3D navigation”. In: *Proc. I3D*, pp. 7–14, ACM, 2009.
- [Meye 10] Q. Meyer, J. Suessmuth, G. Sussner, M. Stamminger, and G. Greiner. “On Floating-Point Normal Vectors.”. *Computer Graphics Forum*, Vol. 29, No. 4, pp. 1405–1409, 2010.

- [Meye 12] Q. Meyer, B. Keinert, G. Sussner, and M. Stamminger. "Data-Parallel Decompression of Triangle Mesh Topology". *Computer Graphics Forum*, Vol. 31, No. 8, pp. 2541–2553, Dec. 2012.
- [Moer 12] C. Moerman, D. Marchal, and L. Grisoni. "Drag'n Go: Simple and fast navigation in virtual environment". In: *Proc 3DUI*, pp. 15–18, IEEE Computer Society, 2012.
- [Mosc 08] T. Moscovich and J. F. Hughes. "Indirect mappings of multi-touch input using one and two hands". In: *Proc. ACM SIGCHI*, pp. 1275–1284, ACM, 2008.
- [Mota 08] J. a. Mota, M. J. Fonseca, D. Gonçalves, and J. A. Jorge. "Agrafo: a visual interface for grouping and browsing digital photos". In: *Proc. ACM AVI*, pp. 494–495, ACM, 2008.
- [Nieb 10] F. Niebling, A. Kopeccki, and M. Becker. "Collaborative steering and post-processing of simulations on HPC resources: Everyone, anytime, anywhere". In: *Proceedings of the 15th International Conference on Web 3D Technology*, pp. 101–108, ACM, 2010.
- [Oliv 00] M. M. Oliveira, G. Bishop, and D. McAllister. "Relief Texture Mapping". In: S. Hoffmeyer, Ed., *Proceedings of the Computer Graphics Conference 2000 (SIGGRAPH-00)*, pp. 359–368, ACM Press, New York, July 23–28 2000.
- [Piet 10] N. Pietroni, M. Tarini, and P. Cignoni. "Almost Isometric Mesh Parameterization through Abstract Domains". *IEEE Transactions on Visualization and Computer Graphics*, Vol. 16, No. 4, pp. 621–635, 2010.
- [Pint 11] R. Pintus, E. Gobbetti, and M. Callieri. "Fast Low-Memory Seamless Photo Blending on Massive Point Clouds using a Streaming Framework". *ACM Journal on Computing and Cultural Heritage*, Vol. 4, No. 2, p. Article 6, 2011.
- [Poli 05] F. Policarpo, M. M. Oliveira, and J. L. D. Comba. "Real-time relief mapping on arbitrary polygonal surfaces". *ACM Trans. Graph*, Vol. 24, No. 3, p. 935, 2005.
- [Poly 11] N. F. Polys, D. A. Bowman, and C. North. "The role of Depth and Gestalt cues in information-rich virtual environments". *International Journal of Human-Computer Studies*, Vol. 69, No. 1-2, pp. 30–51, Jan. 2011.

- [Prau 03] E. Praun and H. Hoppe. "Spherical parametrization and remeshing". *ACM Trans. Graph.*, Vol. 22, No. 3, pp. 340–349, 2003.
- [Purn 05] B. Purnomo, J. Bilodeau, J. D. Cohen, and S. Kumar. "Hardware-compatible vertex compression using quantization and simplification". In: *Proc. ACM Graphics Hardware*, pp. 53–61, 2005.
- [R Co 13] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2013.
- [Rapp 98] D. Rappoport. "Image-Based Rendering for Non-Diffuse Synthetic Scenes". In: *Proc. Rendering techniques*, p. 301, 1998.
- [Reis 09] J. L. Reisman, P. L. Davidson, and J. Y. Han. "A screen-space formulation for 2D and 3D direct manipulation". In: *Proc. ACM UIST*, pp. 69–78, ACM, 2009.
- [Ried 06] M. O. Riedl and R. M. Young. "From linear story generation to branching story graphs". *IEEE Computer Graphics and Applications*, Vol. 26, No. 3, pp. 23–31, 2006.
- [Rivi 08] J.-B. de la Rivière, C. Kervégant, E. Orvain, and N. Dittlo. "CubTile: a multi-touch cubic interface". In: *Proc. ACM VRST*, pp. 69–72, ACM, 2008.
- [Rodr 12] K. Rodriguez-Echavarría, J. Kaminski, and D. B. Arnold. "3D Heritage on Mobile Devices: Scenarios and Opportunities". In: *EuroMed'12*, pp. 149–158, 2012.
- [Ross 01] J. Rossignac. "3D Compression Made Simple: Edgebreaker with Zip&Wrap on a Corner-Table". In: *Proceedings of the International Conference on Shape Modeling & Applications*, pp. 278–, IEEE Computer Society, Washington, DC, USA, 2001.
- [Rubi 13] I. Rubino, J. Xhembulla, A. Martina, A. Bottino, and G. Malnati. "MusA: Using Indoor Positioning and Navigation to Enhance Cultural Experiences in a Museum". *Sensors*, Vol. 13, No. 12, pp. 17445–17471, 2013.
- [Ryu 10] D.-S. Ryu, W.-K. Chung, and H.-G. Cho. "PHOTOLAND: a new image layout system using spatio-temporal information in digital photos". In: *Proc. ACM SAC*, pp. 1884–1891, ACM Press, 2010.
- [Safr 06] I. Safro, D. Ron, and A. Brandt. "Graph minimum linear arrangement by multilevel weighted edge contractions". *Journal of Algorithms*, Vol. 60, No. 1, pp. 24–41, 2006.

- [Sand 03] P. V. Sander, Z. J. Wood, S. J. Gortler, J. Snyder, and H. Hoppe. "Multi-Chart Geometry Images". In: *Eurographics Symposium on Geometry Processing*, pp. 146–155, June 2003.
- [Sank 12] A. Sankar and S. Seitz. "Capturing indoor scenes with smartphones". In: *Proc. UIST*, pp. 403–412, 2012.
- [Scho 06] J. Scholtz. "Beyond usability: Evaluation aspects of visual analytic environments". In: *Visual Analytics Science And Technology, 2006 IEEE Symposium On*, pp. 145–150, 2006.
- [Schr 04] J. Schreiner, A. Asirvatham, E. Praun, and H. Hoppe. "Inter-surface mapping". *ACM Trans. Graph.*, Vol. 23, No. 3, pp. 870–877, 2004.
- [Scot 03] W. R. Scott, G. Roth, and J.-F. Rivest. "View planning for automated three-dimensional object reconstruction and inspection". *ACM Comput. Surv.*, Vol. 35, No. 1, pp. 64–96, March 2003.
- [Seco 11] A. Secord, J. Lu, A. Finkelstein, M. Singh, and A. Nealen. "Perceptual Models of Viewpoint Preference". *ACM TOG*, Vol. 30, No. 5, pp. 109:1–109:12, Oct. 2011.
- [Sege 10] E. Segel and J. Heer. "Narrative visualization: Telling stories with data". *IEEE TVCG*, Vol. 16, No. 6, pp. 1139–1148, 2010.
- [Sene 04] J. G. Senecal, P. Lindstrom, M. A. Duchaineau, and K. I. Joy. "An improved N-bit to N-bit reversible Haar-like transform". In: *12th Pacific Conference on Computer Graphics and Applications*, pp. 371–380, Oct. 2004.
- [Shef 06] A. Sheffer, E. Praun, and K. Rose. "Mesh Parameterization Methods and Their Applications". *Foundations and Trends in Computer Graphics and Vision*, Vol. 2, No. 2, pp. 105–171, 2006.
- [Shiu 05] L.-J. Shiue, I. Jones, and J. Peters. "A realtime GPU subdivision kernel". *ACM Transactions on Graphics*, Vol. 24, No. 3, pp. 1010–1015, Aug. 2005.
- [Shoe 92] K. Shoemake. "ARCBALL: a user interface for specifying three-dimensional orientation using a mouse". In: *Proc. SIGGRAPH*, pp. 151–156, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1992.
- [Shum 07] H. Shum, S. Chan, and S. B. Kang. *Image-based rendering*. Springer, 2007.

- [Shur 11] D. Shuralyov and W. Stuerzlinger. "A 3D Desktop Puzzle Assembly System". In: *Proc. 3DUI*, pp. 139–140, IEEE Computer Society, 2011.
- [Sinh 12] S. Sinha, J. Kopf, M. Goesele, D. Scharstein, and R. Szeliski. "Image-based rendering for scenes with reflections". *ACM Trans. Graph.*, Vol. 31, No. 4, p. 100, 2012.
- [Snav 06] N. Snavely, S. M. Seitz, and R. Szeliski. "Photo tourism: exploring photo collections in 3D". In: *Proc. SIGGRAPH*, pp. 835–846, 2006.
- [Sonn 05] H. Sonnet, S. Carpendale, and T. Strothotte. "Integration of 3D Data and Text: The Effects of Text Positioning, Connectivity, and Visual Hints on Comprehension". In: M. Costabile and F. Patern s, Eds., *Human-Computer Interaction - INTERACT 2005*, pp. 615–628, Springer Berlin Heidelberg, 2005.
- [Stre 06] M. Strengert, M. Kraus, and T. E. Ertl. "Pyramid methods in gpu-based image processing". In: *Proc. VMV*, pp. 169–176, 2006.
- [Taub 98a] G. Taubin, A. Gu eziec, W. Horn, and F. Lazarus. "Progressive forest split compression". In: *Proc. ACM SIGGRAPH*, pp. 123–132, New York, NY, USA, 1998.
- [Taub 98b] G. Taubin and J. Rossignac. "Geometric compression through topological surgery". *ACM Trans. Graph.*, Vol. 17, No. 2, pp. 84–115, 1998.
- [Tomp 12] J. Tompkin, K. I. Kim, J. Kautz, and C. Theobalt. "Videoscapes: exploring sparse, unstructured video collections". *ACM Trans. Graph.*, Vol. 31, No. 4, p. 68, 2012.
- [Trin 11] D. R. Trindade and A. B. Raposo. "Improving 3d navigation in multiscale environments using cubemap-based techniques". In: *Proc. SAC*, pp. 1215–1221, ACM, 2011.
- [Van 08] S. Van Dongen. "Graph Clustering Via a Discrete Uncoupling Process". *SIAM Journal on Matrix Analysis and Applications*, Vol. 30, No. 1, pp. 121–141, 2008.
- [Vazq 02] P.-P. Vazquez, M. Feixas, M. Sbert, and W. Heidrich. "Image-Based Modeling Using Viewpoint Entropy". In: *Proc. CGI*, 2002.
- [Vinc 07] L. Vincent. "Taking Online Maps Down to Street Level". *Computer*, Vol. 40, pp. 118–120, December 2007.
- [Wang 03] L. Wang, X. Wang, X. Tong, S. Lin, S.-M. Hu, B. Guo, and H.-Y. Shum. "View-dependent displacement mapping.". *ACM Trans. Graph.*, Vol. 22, No. 3, pp. 334–339, 2003.

- [Wang 04] X. Wang, X. Tong, S. Lin, S. Hu, B. Guo, and H.-Y. Shum. "Generalized Displacement Maps". In: D. Fellner and S. Spencer, Eds., *Proceedings of the 2004 Eurographics Symposium on Rendering*, pp. 227–234, Eurographics Association, June 2004.
- [Weis 10a] K. Weiss and L. De Floriani. "Simplex and diamond hierarchies: Models and applications". In: *Eurographics 2010 - State of the Art Reports*, pp. 113–136, 2010.
- [Weis 10b] M. Weiss, S. Voelker, C. Sutter, and J. Borchers. "BendDesk: dragging across the curve". In: *Proc. ACM International Conference on Interactive Tabletops and Surfaces*, pp. 1–10, ACM, 2010.
- [Wils 03] A. Wilson and D. Manocha. "Simplifying complex environments using incremental textured depth meshes". *ACM Trans. Graph.*, Vol. 22, No. 3, pp. 678–688, 2003.
- [Wimm 10] R. Wimmer, F. Hennecke, F. Schulz, S. Boring, A. Butz, and H. Hussmann. "Curve: revisiting the digital desk". In: *Proc. NordiCHI*, pp. 561–570, ACM, 2010.
- [Wu 02] J. Wu and L. Kobbelt. "Fast mesh decimation by multiple-choice techniques". In: *Proceedings of 7th International Fall Workshop on Vision, Modeling, and Visualization*, pp. 241–248, 2002.
- [Xia 96] J. Xia and A. Varshney. "Dynamic View-dependent simplification for polygonal models". In: *Proc. IEEE Visualization*, pp. 327–334, 1996.
- [Yoon 04] S.-E. Yoon, B. Salomon, R. Gayle, and D. Manocha. "Quick-VDR: Interactive View-Dependent Rendering of Massive Models". In: *Proc. IEEE Visualization*, pp. 131–138, Washington, DC, USA, 2004.
- [Yoon 08] S. Yoon, E. Gobbetti, D. Kasik, and D. Manocha. *Real-time Massive Model Rendering*. Vol. 2 of *Synthesis Lectures on Computer Graphics and Animation*, Morgan and Claypool, August 2008.
- [Yu 10] L. Yu, P. Svetachov, P. Isenberg, M. H. Everts, and T. Isenberg. "FI3D: Direct-Touch Interaction for the Exploration of 3D Scientific Visualization Spaces". *IEEE Transactions on Visualization and Computer Graphics*, Vol. 16, No. 6, pp. 1613–1622, Nov. 2010.
- [Zelev 99] R. Zeleznik and A. Forsberg. "UniCam: 2D Gestural Camera Controls for 3D Environments". In: *Proc. Symposium on Interactive 3D Graphics*, pp. 169–173, ACM, 1999.

-
- [Zhao 11] Y. J. Zhao, D. Shuralyov, and W. Stuerzlinger. "Comparison of multiple 3d rotation methods". In: *Proc. IEEE VECIMS*, pp. 1–5, IEEE, 2011.

Curriculum Vitae

Marcos Balsa studied Computer Science at the Polytechnic University of Catalonia (UPC) in Spain, where he received his M.Sc. degree in September 2011. In 2002, he started working as research assistant at the IRI (Institut de Robòtica i Informàtica Industrial) of the UPC, working in projects related to Virtual Reality and medical image visualization. During 2007-2010 he worked in GEOvirtual, a company focused on 3D GIS, designing and developing a 3D Virtual Globe. In 2012, he was awarded with a Marie Curie Research Grant and hired as an Early Stage Researcher in the ViC group of CRS4 in Sardinia. His research is focused on massive model rendering. In particular, efficient and scalable methods for the interactive visualization of very large mesh models in constrained environments and mobile devices. His research interests cover real-time rendering, massive model rendering, parallel programming, volume rendering and mobile programming.

Contact Information

Name	Marcos Balsa Rodríguez
Address	CRS4, POLARIS Edificio 1, 09010 Pula (CA), Italy
Phone	+39 0709250283
E-mail	marbarod@gmail.com
Website	http://www.crs4.it/people

Personal Details

Date of Birth	April 18 th , 1981.
Place of Birth	Barcelona, Spain
Citizenship	Spanish
Languages	Spanish (native), Catalan (native), Italian (fluent), English (fluent), Galician (fluent)
Professional Affiliations	Eurographics (since 2012) - ACM (since 2012)

Education

- | | |
|------------------------|--|
| since Jan. 2012 | PhD Program in Computer Science , University of Cagliari, Italy. Working on the dissertation <i>Scalable Rendering of Highly Detailed 3D models</i> at the School of Mathematics and Computer Science. Tutor: Prof. Riccardo Scateni. Research Advisor: Dr. E. Gobbetti (CRS4). |
| 2006/07 - 2010/11 | Master Studies in Computing , Polytechnic University of Catalonia (UPC), Spain. Successfully completed the master's thesis titled <i>Volume visualization on mobile devices</i> , with 10/10. Advisor: Prof. Pere-Pau Vázquez Alcocer. |
| Sept. 1999 - Feb. 2006 | Engineering Degree in Computer Science , Polytechnic University of Catalonia (UPC), Spain. Successfully completed the thesis on <i>Falling leaves simulation on the GPU</i> , with 10/10. Advisor: Pere-Pau Vázquez Alcocer. |

Employment History

- | | |
|-----------------------------|--|
| since January 2015 | Consultant Researcher at the Visual Computing group of CRS4. |
| January 2012 - January 2015 | Marie Curie Early Stage Researcher at the Visual Computing group of CRS4 (Center for Advanced Studies, Research and Development in Sardinia). |
| May 2010 - Dec. 2010 | IT Consultant. Grup de Tecnologies Interactives (GTI) Universitat Pompeu Fabra, Barcelona, Spain. Development of Flash-based client-server technologies for tracking of the Barcelona World Race event. |
| Feb. 2007 - April 2010 | Computer Graphics developer. GEOVirtual. Barcelona, Spain. Research and development of a 3D Virtual Globe for geo-referenced information. |
| Dec. 2002 - Jan. 2007 | Research assistant. Institut de Robotica i Informatica Industrial. Polytechnic University of Catalonia, Barcelona, Spain. Research in Virtual Reality projects, Volume Rendering and GPGPU techniques. |

Software projects involved

2012-2014	Mont'e Prama project - 3D interactive visualization of highly detailed cultural heritage models (mobile and large display systems), and interaction methods adapted to the various setups.
2010	Flash-based 2D GIS tracking system for Barcelona World Race, including tracking of the various ships and dynamic weather forecast.
2007-2010	3D Virtual Globe with raster and vector (including 3D models) information. Also various web plugins enabling integration of the 3D globe in the web using Javascript.
2006	Visible Male - Out-of-core visualization and gradient precomputation of data from the Visible Human project.
2003-2005	ViHAP3D (http://www.vihap3d.org/news.html) - 3D museum builder, and museum viewer with 3D stereo capabilities and trigger-based multimedia events.
2002-2005	PVPC - Building Prototyping application. Qt GUI integrating modules for 2D sketch recognition, 3D model reconstruction from 2D (DXF), texturing, sunlight simulation, and 3D stereo visualization.

Scientific Talks, Tutorials and Lectures

Web3D 2014	HuMoRS: Huge models Mobile Rendering System. Vancouver, Canada. August, 2014
Master Game Day 2014	Huge surface model visualization on mobile devices. Universita' di Verona, Italy. May, 2014
Mobile Graphics & Interactive Applications 2013	Coarse-grained Multiresolution Structures for Mobile Exploration of Gigantic Surface Models. Hong Kong, China. November, 2013
Web3D 2013	Compression-domain Seamless Multiresolution Visualization of Gigantic Meshes on Mobile devices. San Sebastian, Spain. June, 2013
EuroGraphics 2013	A Survey of Compressed GPU-based Direct Volume Rendering. May 2013
ISVC 2012	Practical Volume Rendering in mobile devices. Rethymnos, Greece. July, 2012

Thesis Publications

Journal Articles

1. E. Gobbetti, R. Pintus, F. Bettio, F. Marton, M. Agus, and M. Balsa Rodríguez. “Digital Mont’è Prama: dalla digitalizzazione accurata alla valorizzazione di uno straordinario complesso statuario”. *Archeomatica*, 2015. To appear
2. M. Balsa Rodríguez, M. Agus, F. Marton, and E. Gobbetti. “Adaptive Recommendations for Enhanced non-linear Exploration of Annotated 3D Objects”. *Computer Graphics Forum*, May 2015. Conditionally accepted to EuroVis 2015
3. M. Di Benedetto, F. Ganovelli, M. Balsa Rodríguez, A. Jaspe Villanueva, R. Scopigno, and E. Gobbetti. “ExploreMaps: Efficient Construction and Ubiquitous Exploration of Panoramic View Graphs of Complex 3D Environments”. *Computer Graphics Forum*, Vol. 33, No. 2, pp. 459–468, 2014. Proc. Eurographics 2014
4. F. Marton, M. Balsa Rodríguez, F. Bettio, M. Agus, A. Jaspe Villanueva, and E. Gobbetti. “IsoCam: Interactive Visual Exploration of Massive Cultural Heritage Models on Large Projection Setups”. *ACM Journal on Computing and Cultural Heritage*, Vol. 7, No. 2, p. Article 12, June 2014
5. F. Marton, M. Agus, E. Gobbetti, G. Pintore, and M. Balsa Rodríguez. “Natural exploration of 3D massive models on large-scale light field displays using the FOX proximal navigation technique”. *Computers & Graphics*, Vol. 36, No. 8, pp. 893–903, December 2012

Conference Papers (based on full paper peer reviewing)

1. M. Balsa Rodríguez, M. Agus, F. Marton, and E. Gobbetti. “HuMoRS: Huge models Mobile Rendering System”. In: *Proc. ACM Web3D International Symposium*, ACM Press, New York, NY, USA, August 2014
2. M. Balsa Rodríguez, E. Gobbetti, F. Marton, and A. Tinti. “Coarse-grained Multiresolution Structures for Mobile Exploration of Gigantic Surface Models”. In: *Proc. SIGGRAPH Asia Symposium on Mobile Graphics and Interactive Applications*, pp. 4:1–4:6, ACM, November 2013
3. M. Balsa Rodríguez, E. Gobbetti, F. Marton, and A. Tinti. “Compression-domain Seamless Multiresolution Visualization of Gigantic Meshes on Mobile Devices”. In: *Proc. ACM Web3D International Symposium*, pp. 99–107,

ACM Press, June 2013

4. E. Gobbetti, F. Marton, M. Balsa Rodríguez, F. Ganovelli, and M. Di Benedetto. "Adaptive Quad Patches: an Adaptive Regular Structure for Web Distribution and Adaptive Rendering of 3D Models". In: *Proc. ACM Web3D International Symposium*, pp. 9–16, ACM Press, New York, NY, USA, August 2012. (Best Long Paper Award)

Other Publications

Journal Articles

1. M. Balsa Rodríguez, E. Gobbetti, J. Iglesias Guitián, M. Makhinya, F. Marton, R. Pajarola, and S. Suter. "State-of-the-art in Compressed GPU-Based Direct Volume Rendering". *Computer Graphics Forum*, Vol. 33, No. 6, pp. 77–100, September 2014

Conference Papers (based on full paper peer reviewing)

1. M. Balsa Rodríguez, E. Gobbetti, J. Iglesias Guitián, M. Makhinya, F. Marton, R. Pajarola, and S. Suter. "A Survey of Compressed GPU-based Direct Volume Rendering". In: *Eurographics State-of-the-art Report*, pp. 117–136, May 2013
2. M. Balsa Rodríguez, E. Gobbetti, F. Marton, R. Pintus, G. Pintore, and A. Tinti. "Interactive Exploration of Gigantic Point Clouds on Mobile Devices". In: D. Arnold, J. Kaminski, F. Niccolucci, and A. Stork, Eds., *VAST: International Symposium on Virtual Reality, Archaeology and Intelligent Cultural Heritage*, The Eurographics Association, 2012

Book contributions

1. M. Balsa Rodríguez and P. Vazquez Alcocer. "Practical Volume Rendering in mobile devices". In: *Proc. International Symposium on Visual Computing*, pp. 708–718, Springer Verlag, 2012

