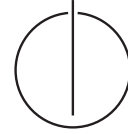




Technische Universität München
Fakultät für Informatik



Dissertation

**Real-Time Multi-View 3D Reconstruction
for Interventional Environments**

Stavros Ladikos

TECHNISCHE UNIVERSITÄT MÜNCHEN

Lehrstuhl für Informatikanwendungen in der Medizin & Augmented Reality

Real-Time Multi-View 3D Reconstruction for Interventional Environments

Stavros Ladikos

Vollständiger Abdruck der von der Fakultät für Informatik der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktors der Naturwissenschaften (Dr. rer. nat.)

genehmigten Dissertation.

Vorsitzender: Univ.-Prof. M. Beetz, Ph.D.

Prüfer der Dissertation: 1. Univ.-Prof. Dr. N. Navab

2. Prof. P. Sturm, Ph.D., INRIA, Grenoble, Frankreich

Die Dissertation wurde am 30.08.2010 bei der Technischen Universität München eingereicht und durch die Fakultät für Informatik am 11.03.2011 angenommen.

Für Doris

Abstract

This thesis addresses the topic of real-time multi-view 3D reconstruction in interventional environments. Due to the special working conditions in interventional rooms this is a very challenging topic. However, the ability to recover the 3D structure of a dynamic scene in real-time is essential for many novel and innovative interventional applications. In this thesis we take a step towards this goal by presenting the design of a multi-view reconstruction system taking into consideration the conditions typically encountered in interventional environments. Two installations of the system were created: One in a laboratory environment and one in a real interventional room. While the laboratory system was used for development and evaluation purposes, the hospital system was used to learn about the conditions which have to be taken into account in a clinical setting. These conditions which typically include a cluttered environment with complex lighting and occlusions are discussed in detail and recommendations are given for dealing with them.

Based on the proposed system several applications were developed to demonstrate the benefits of real-time 3D reconstruction in interventional environments. In a first application the real-time reconstruction is used to ensure the safe operation of automated medical devices such as C-arms by detecting potential collisions between the device and its environment (staff, patient, utilities) in advance. A second application is aimed at estimating the physician's radiation exposure by tracking his reconstruction and accumulating the radiation received by each body part over time. The promising experimental results obtained using these applications show the possible impact of our work for the medical community and illustrate the clinical value of multi-camera systems in the development of intelligent, integrated interventional suites. This is underlined by the fact that the proposed system has already been used by other researches for recovering the workflow of surgical procedures [PMW⁺09].

During the course of this thesis the topic of reconstruction and organization of large image collections was also investigated. This research area is gaining importance since it has become easy and affordable to acquire large image collections due to the proliferation of digital cameras. However, the huge amount of images available poses new challenges for existing algorithms. The focus of this part of the thesis therefore lies on the efficient use of large image collections. Three related problems are investigated, namely obtaining high quality reconstructions using graph-cut methods, increasing the reconstruction efficiency by clustering image collections and building a unified framework for various applications related to image organization, such as image-based navigation, image set reduction and scene summarization.

Zusammenfassung

Diese Arbeit befasst sich mit dem Thema der Echtzeit Multi-View 3D-Rekonstruktion in Interventionsräumen. Aufgrund der speziellen Arbeitsbedingungen in diesem Bereich ist dies ein sehr anspruchsvolles Thema. Nichtsdestotrotz ist die Fähigkeit die 3D-Struktur einer Szene in Echtzeit zu erfassen essentiell für viele neuartige und innovative klinische Anwendungen. In dieser Arbeit leisten wir einen Beitrag zum Erreichen dieses Ziels, indem wir das Design eines Multi-View Rekonstruktionssystems präsentieren, welches die Bedingungen, die in typischen Interventionsräumen herrschen, berücksichtigt. Wir haben zwei Installationen des Systems erstellt: Eine in einer Laborumgebung und eine in einem echten Interventionsraum. Das Laborsystem wurde für Entwicklungs- und Versuchszwecke benutzt, wogegen das klinische System dazu benutzt wurde, um mehr über die Gegebenheiten zu erfahren, die in einem klinischen Umfeld zu berücksichtigen sind. Diese Gegebenheiten, welche typischerweise eine ungeordnete Arbeitsumgebung und komplexe Beleuchtungsverhältnisse umfassen, werden im Detail diskutiert und es werden Lösungsvorschläge zum Umgang mit selbigen vorgestellt.

Basierend auf dem vorgestellten System wurden einige Anwendungen entwickelt, um die Vorteile einer Echtzeit 3D-Rekonstruktion in Interventionsräumen aufzuzeigen. In einer ersten Anwendung wird die Echtzeitrekonstruktion dazu benutzt, um den sicheren Betrieb von automatisierten medizinischen Geräten sicherzustellen, indem potentielle Kollisionen zwischen dem Gerät und der Umgebung frühzeitig erkannt werden. Eine zweite Anwendung zielt darauf ab die Strahlenbelastung des Arztes zu approximieren, indem seine Rekonstruktion getrackt und die Strahlung, die von jedem Körperteil empfangen wird, akkumuliert wird. Die vielversprechenden experimentellen Ergebnisse, die in diesen Anwendungen erlangt wurden, zeigen die Bedeutung unserer Arbeit für die medizinische Gemeinschaft und illustrieren den klinischen Wert von Multikamerasystemen für die Entwicklung von intelligenten, integrierten Interventionslösungen. Dies wird durch die Tatsache unterstrichen, dass das hier vorgestellte System schon von anderen Forschern zum Bestimmen des Arbeitsablaufes eines operativen Eingriffs benutzt wurde [PMW⁺09].

Im Laufe dieser Arbeit wurde außerdem die Rekonstruktion und Organisation von großen Bildersammlungen untersucht. Dieser Forschungsbereich hat in den letzten Jahren an Bedeutung gewonnen, da es aufgrund der weiten Verbreitung von Digitalkameras einfach und günstig geworden ist große Bildersammlungen zu erstellen. Allerdings bereitet die große Anzahl an Bildern den existierenden Algorithmen Schwierigkeiten. Der Fokus in diesem Teil der Arbeit liegt daher auf der effizienten Nutzung großer Bildersammlungen. Dabei werden drei miteinander verwandte Problemstellungen untersucht, nämlich das Erstellen hochqualitativer Rekonstruktionen mittels Graph-Cut Methoden, die Steigerung der Rekonstruktionseffizienz durch das Clustern von Bildersammlungen und das Erstellen eines einheitlichen Frameworks für verschiedene Bildorganisationsanwendungen.

Acknowledgments

First of all I would like to thank Professor Nassir Navab for giving me the opportunity to work on this exiting topic and for supervising me. I'd also like to thank Selim Benhimane who was very involved in the first part of this thesis and who helped me in finding into the subject. Furthermore, I'm also thankful to all the people at the chair with whom I had helpful discussions and collaborations. In particular I'd like to thank Cedric Cagniart, Ali Bigdelou, Edmond Boyer and Slobodan Ilic. Finally I'd like to thank Martin Horn for getting all the equipment necessary for performing this work and for helping me with the installation of the real-time reconstruction system both in our lab and in the hospital.



Contents

I	Real-Time Multi-View 3D Reconstruction for Interventional Environments	1
1	Introduction	3
1.1	Application Areas of Real-Time 3D Reconstruction	3
1.1.1	Mixed Reality	4
1.1.2	Free Viewpoint TV	5
1.1.3	Motion Capture	5
1.2	Real-Time 3D Reconstruction Systems in Interventional Environments	7
1.2.1	Benefits	7
1.2.2	Challenges	8
1.3	Contributions	8
1.4	Outline	9
2	3D Reconstruction Techniques and Systems	11
2.1	3D Reconstruction Techniques	11
2.1.1	Active Methods	11
2.1.2	Passive Methods	15
2.2	Multi-Camera Reconstruction Systems	17
2.3	Conclusion	20
3	The Visual Hull	21
3.1	Definition and Properties	21
3.2	Visual Hull Computation	23
3.2.1	Volumetric Approach	23
3.2.2	Polyhedral Approach	25
3.2.3	Other Approaches	26
3.3	Conclusion	29

CONTENTS

4	Proposed 3D Reconstruction System	31
4.1	Overview	31
4.2	Hardware	31
4.2.1	PC Cluster	31
4.2.2	Cameras	32
4.2.3	Synchronization	33
4.3	System Architecture	33
4.4	Calibration	34
4.4.1	Robust Point Extraction	35
4.4.2	Factorization	35
4.4.3	Euclidean Stratification	36
4.4.4	Radial Distortion Correction	36
4.4.5	Registration	37
4.4.6	Results	38
4.5	Background Subtraction	38
4.5.1	Histogram Approach	39
4.5.2	Gaussian Mixture Models	40
4.5.3	Codebook Approach	41
4.5.4	Discussion and Results	43
4.5.5	Handling Static Occluders	44
4.6	Visual Hull Computation	45
4.7	Visualization	45
4.8	Conclusion	46
5	GPU-based Visual Hull Computation	49
5.1	Introduction	49
5.2	Related Work	49
5.3	Precomputation-based Algorithm	50
5.4	Direct Algorithm	51
5.5	Results	52
5.6	Conclusion	55

CONTENTS

6	Incremental Visual Hull Computation	57
6.1	Related Work	57
6.2	Incremental Reconstruction Approach	58
6.3	Ray Casting	59
6.4	Ray Buffers	61
6.5	Analysis	61
6.6	Results	63
6.7	Conclusion	67
7	Applications	69
7.1	Collision Avoidance in Interventional Environments	69
7.1.1	System Description	70
7.1.2	Results	71
7.1.3	Conclusion and Future Work	72
7.2	Controlling Radiation Exposure in Interventional Environments	72
7.2.1	Related Work	73
7.2.2	Reconstruction and Tracking System	74
7.2.3	Results	75
7.2.4	Discussion	76
7.2.5	Conclusion and Future Work	76
7.3	Workflow Analysis	77
7.4	Mixed Reality Interactions	77
7.4.1	Related Work	78
7.4.2	Pong	79
7.4.3	Video Compositing	81
7.4.4	Conclusion	82
7.5	Conclusion	82
8	Conclusion	85
8.1	Discussion and Future Work	85
II	Multi-View 3D Reconstruction and Organisation of Image Collections	89
9	Introduction	91
9.1	Reconstructing and Organizing Large Image Collections	91
9.2	Contributions	94

CONTENTS

10 Graph-Cut-based Reconstruction	97
10.1 Introduction	97
10.2 Related Work	98
10.3 Volumetric Graph-Cuts	99
10.3.1 Consistency Measure	99
10.3.2 Surface Optimization with Graph-Cuts	100
10.4 Proposed Reconstruction Method	101
10.4.1 Surface Normal Optimization	101
10.4.2 Robust Visibility Test	102
10.4.3 Narrow Band Graph-Cut	103
10.5 Results	104
10.6 Conclusion	106
11 Spectral Camera Clustering	107
11.1 Introduction	107
11.2 Related Work	108
11.3 Camera Clustering	110
11.4 Results	114
11.5 Conclusion	116
12 Region Graphs	117
12.1 Introduction	117
12.2 Related Work	119
12.3 Building Region Graphs	120
12.3.1 Identifying Overlap Between Images	120
12.3.2 Identifying Overlapping Regions	121
12.3.3 Constructing the Region Graph	121
12.4 Using Region Graphs	122
12.4.1 Image Set Reduction	122
12.4.2 Canonical Views	123
12.4.3 Image-based Navigation	124
12.5 Experimental Results	126
12.5.1 Image Set Reduction	126
12.5.2 Canonical Views	126
12.5.3 Image-based Navigation	129
12.6 Conclusion	130

CONTENTS

13 Conclusion	131
13.1 Discussion and Future Work	131
Own Publications	133
Bibliography	135

Part I

**Real-Time Multi-View 3D
Reconstruction for Interventional
Environments**

Chapter 1

Introduction

Recovering the 3D structure of a scene from images has long been one of the core interests of Computer Vision. While a considerable amount of work in this area has been devoted to achieving high quality reconstructions regardless of run time, other work has focused on the aspect of real-time reconstruction and its applications. Especially in the last decade the interest in real-time 3D reconstruction systems has increased considerably due to the many applications, industrial as well as scientific, which are enabled by this technology. This thesis fits into this trend by presenting a real-time 3D reconstruction system making use of novel and efficient reconstruction algorithms. Using this system we showcase several innovative applications, focusing in particular on interventional settings in order to show the advantages and benefits of including multi-camera 3D reconstruction systems in future interventional rooms.

1.1 Application Areas of Real-Time 3D Reconstruction

Before considering the topic of real-time 3D reconstruction in interventional environments more closely in the next section, we want to first give a brief overview of some popular uses of 3D reconstruction systems in order to show the wide range of possible applications. In particular we will have a look at mixed reality, 3D television and motion capture.



Figure 1.1: Small scale system for hand-based interaction with virtual objects [AMR⁺07].



Figure 1.2: Demonstration of the *VGate* system [PLBR09]. The user can interact with virtual objects in the scene, including seeing his own reflection in a virtual mirror.

1.1.1 Mixed Reality

Mixed Reality can be considered one of the first and primary applications of real-time 3D reconstruction. While the topic of using 3D reconstructions of events for introducing them into virtual environments has been around for at least as long as the creation of the CMU dome [KRN97], it has only been in the last decade that systems achieving this in real-time have been introduced.

One application which immediately comes to mind is the interaction with virtual objects. This is made possible by the fact that due to the real-time 3D reconstruction all the information about the position and the intersection between virtual and real objects can be deduced. This allows a human to touch and move virtual objects among other things without requiring a complex (skeletal) tracking of the user.

In the following we will take the *GrImage* platform developed at INRIA Grenoble as a showcase to highlight different mixed reality applications. The *GrImage* platform is a mature multi-view 3D reconstruction system and runs at real-time frame rates. Several interactive mixed reality applications have been built on it.

One interesting application was demonstrated at SIGGRAPH 2007[AMR⁺07]. In it a user is interacting with a virtual jack in the box. The user's hands are reconstructed in 3D and the forces he exerts on the virtual jack in the box are computed using a physics-based interaction framework based on the intersection of the hands and the virtual object. The result of the reconstruction and the interaction can be observed on a screen as shown in figure 1.1.

A more immersive interaction can be achieved when using an HMD instead of a screen. This was done in another application [PLBR09] which uses the system for full-body immersive interactions with virtual objects. The user sees the virtual environment including himself through an HMD (see figure 1.2). As in the previous application the user can interact with virtual objects. In addition, new interaction metaphors such as mirrors and the cloning of one's own reconstruction are inserted into this application. It is clear that all these interactions are only possible due to the availability of a real-time 3D reconstruction.

Another possible application is remote collaboration and telepresence [PLM⁺10]. In this sce-



Figure 1.3: 3D TV for a soccer game. The first image shows one of the input images, while the other two images show two renderings of the recovered scene from novel viewpoints. [GTH⁺07]

nario two multi-view reconstruction systems at different locations are linked and the users are placed in the same virtual space. This way both users can see each other and interact with the same virtual objects.

1.1.2 Free Viewpoint TV

The idea behind free viewpoint TV as opposed to traditional television is to let the user freely choose his vantage point. For instance during a soccer game a user might want to place himself on the playing field in order to view the action more directly. In order to achieve this goal the content (in this case the soccer game) has to be recorded in 3D. As the broadcast is usually live the reconstruction also has to be performed at real-time. In contrast to studio-based applications which can assume favorable conditions the scenario in an outdoor sporting event is much less restricted. For instance background subtraction is made difficult by changing lighting conditions, changing background and the movement of the crowd. For soccer and other ball games played on a green field the situation is slightly more advantageous since it can be assumed that the playing field is green, which significantly improves the task of foreground object extraction [GTH⁺07]. In the last years some solutions have been presented which achieve a quality sufficient for broadcast (see figure 1.3). Especially the BBC has been quite active in this area in the form of the iView project [iP] and already produced some sporting events in 3D. While free viewpoint TV at this point in time is still not widely available, the entertainment industry is actively researching this area, which not only includes the issue of 3D reconstruction but also the issues of transmission and display at the end user location.

1.1.3 Motion Capture

The goal of motion capture is to transfer an actor's motions to an animated character in order to obtain realistic animations in movies or video games. To achieve this goal the actor's joint configuration needs to be recovered. This is traditionally done by attaching markers to the actor which are tracked and used to infer the joint configuration. While common in practice the disadvantage of these systems is that they are cumbersome to use and usually require the actor to wear tight clothing, since a skeleton has to be fit to the observations. Ideally one would like to remove the markers and to have the actor wear arbitrary clothing. Research in this direction has been pushed in particular by the group around Adrian Hilton at the

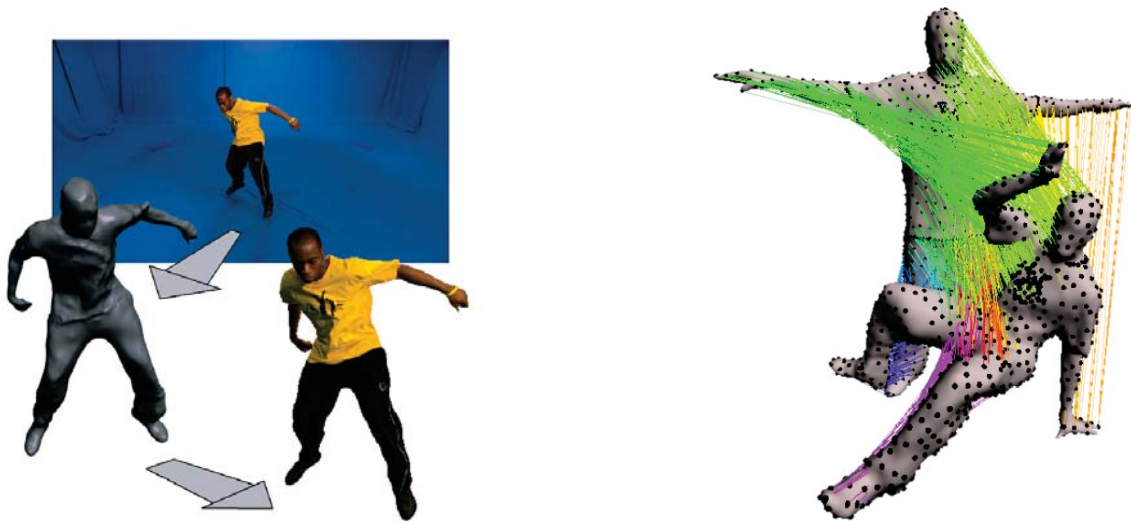


Figure 1.4: Surface capture [SH07b] and subsequent correspondence labeling [SH07a] between a reference surface and the current surface.

University of Surrey. They reconstruct actors using a 3D reconstruction system and use mesh tracking algorithms to infer the movement of the actor over time by establishing dense correspondences between 3D reconstructions at different time steps [SH07a] which in turn allows to recover the joint configuration (see figure 1.4). Another approach to motion capture is to fit a skeleton to the observations (i.e. the independent 3D reconstructions) and thereby deduces the actor's pose [GSdA⁺09]. Other work yet decomposes the initial reconstruction into patches and matches these patches to the current reconstruction [CBI10b] as shown in figure 1.5. Based on this matching the initial reconstruction is deformed in order to match the current one.

Motion Capture is an example of an application whose ultimate goal goes beyond the mere reconstruction of objects, but which uses the reconstruction results for further more sophisticated tasks. At this point the existing motion capture applications do not run in real-time.



Figure 1.5: Example of patch-based correspondence labeling over time for mesh-based deformable tracking [CBI10b].

1.2 Real-Time 3D Reconstruction Systems in Interventional Environments

In contrast to the previously discussed application areas using a real-time 3D reconstruction system in an interventional environment is an up to now little considered research area. Although there are quite a few interesting challenges associated to bringing a reconstruction system to an interventional environment, the benefits and possibilities are huge and easily outweigh the challenges. In the following we will take a closer look at both sides of the coin, discussing both the benefits and the challenges of bringing real-time 3D reconstruction into an interventional setting.

1.2.1 Benefits

Bringing a real-time 3D reconstruction system into an interventional room has several advantages. By performing a 3D reconstruction of the scene many novel applications arise. For instance it becomes possible to create a 3D video of the intervention which can be used to train novel physicians by having them look at the intervention from several viewpoints. A 3D video could also be used to evaluate and compare the performance of several physicians and to analyze the movement of the staff in the interventional room in order to optimize the room layout. Another application which is based on a 3D reconstruction of the scene is collision avoidance between automated devices and other objects. This issue is becoming more important as more automated devices are introduced into the interventional environment. Another possible application is to model the radiation exposure of the physician by considering the X-ray scatter radiation created during the intervention. Even more interesting is the use of the 3D reconstruction for workflow analysis in order to automatically determine



Figure 1.6: Complexities in interventional rooms: The background and foreground have similar colors, patient table and monitors as well as medical devices create occlusions and are movable and due to the limited space the working environment becomes cluttered once an intervention is underway.

which phase an intervention is in. This information can then be used in order to only display information relevant to the current working step to the physician, to help in documenting the intervention and to analyze and compare the performance of several surgeons.

1.2.2 Challenges

While the range of new applications enabled by real-time 3D reconstruction systems is big, interventional environments also poses several challenges to existing systems. Most existing systems work in a lab environment under favorable conditions such as blue screens, controlled illumination and with little occlusion and clutter. In a real interventional room, however, these conditions cannot be controlled. The background often has a similar color as the foreground objects we wish to reconstruct and is cluttered with tool tables, trays and cupboards, which might be opened and closed or even moved during the intervention. Occlusions are created by ceiling-suspended monitors and lamps and most importantly by the patient table and automated devices such as C-arms. The fact that these objects move during an intervention adds to the complexity of the problem. In addition, there may be several people in the scene occluding each other. The lighting conditions can also change abruptly, for instance by turning on spot lights. There may also be mirrors and transparent surfaces (for instance a visitor window). Figure 1.6 shows some images taken in a real interventional room highlighting the complexity of the environment. As we will see in the rest of this thesis taking these conditions into account is a challenging task.

1.3 Contributions

The main contribution of this part of the thesis is the design of a real-time 3D reconstruction system for interventional environments and the applications built on it. On the technical side two visual hull reconstruction algorithms are presented, namely an efficient GPU-based method and an incremental method for dynamic scenes. On the application side a particular emphasis is put on the use of the proposed system in interventional environments in order to demonstrate the benefits of bringing a real-time 3D reconstruction system to an interventional setting. The proposed applications include collision avoidance with automated medical devices, radiation modeling and workflow analysis. In addition, the use of the system for mixed reality applications is also investigated. Last but not least, a list of challenges associated with bringing a 3D reconstruction system to an interventional environment is given and possible solutions to these challenges are proposed and discussed.

The following list once more breaks down the contributions made in this part of the thesis.

- Design and implementation of a real-time 3D reconstruction system targeted at interventional environments
- Development of a fast GPU-based algorithm for visual hull computation
- Development of an incremental algorithm for visual hull computation
- Development of a collision avoidance application for interventional environments

- Development of a radiation estimation application for interventional environments
- Development of an occlusion-aware mixed reality interaction application
- Discussion of challenges in bringing a 3D reconstruction system into an interventional environment and proposal of possible solutions

1.4 Outline

The remainder of this part of the thesis is structured as follows. In chapter 2 we discuss different 3D reconstruction techniques and multi-camera systems. In particular we focus on techniques which enable real-time performance. We then take a look at the visual hull - our 3D reconstruction technique of choice - and its properties in chapter 3. Our proposed 3D reconstruction system is presented in chapter 4. Chapter 5 presents the GPU-based method we developed to achieve real-time performance on commodity hardware while chapter 6 introduces a technique for incrementally updating the 3D reconstruction. In chapter 7 we present several applications which are only made possible by the availability of a real-time 3D reconstruction, focusing mainly on interventional applications. The first application deals with collision avoidance for automated medical devices, while the second one deals with tracking the physician's X-ray scatter radiation exposure. An additional application shows the use of our system for mixed reality interactions. We conclude with chapter 8 by summarizing the challenges encountered in interventional environments and discussing possible solutions.

Chapter 2

3D Reconstruction Techniques and Systems

There exists a large number of 3D reconstruction techniques. Unfortunately many of these are not real-time capable and therefore not applicable in our system. Nevertheless, we will give a brief overview of all major 3D reconstruction techniques in this chapter in order to give a perspective on our choice of the visual hull (see chapter 3). In addition, we will also give an overview of past and current multi-view reconstruction studios.

2.1 3D Reconstruction Techniques

There is a wide variety of 3D reconstruction techniques. They can be divided into active and passive methods. Active methods use active sensors (e.g. a laser) to recover the 3D structure of the scene. Passive methods on the other hand do not interact with the scene, but instead only use the light emitted by the scene (captured in the form of images) to perform a reconstruction. As is to be expected active methods perform better in general, providing a higher accuracy at the cost of more complex and sometimes impractical setups.

2.1.1 Active Methods

Active 3D reconstruction methods include laser range scanning, structured light, photometric stereo and time of flight (TOF) cameras. In the following sections each method will be described briefly.

Laser Range Scanners

The principle behind laser range scanners is to use a laser beam to sample the scene. In its simplest form the laser beam is projected onto the scene and creates a single dot which is recorded by a camera calibrated to the laser source. Using the stereo geometry of the laser-camera system the 3D position of the point can be computed by triangulation. Projecting a

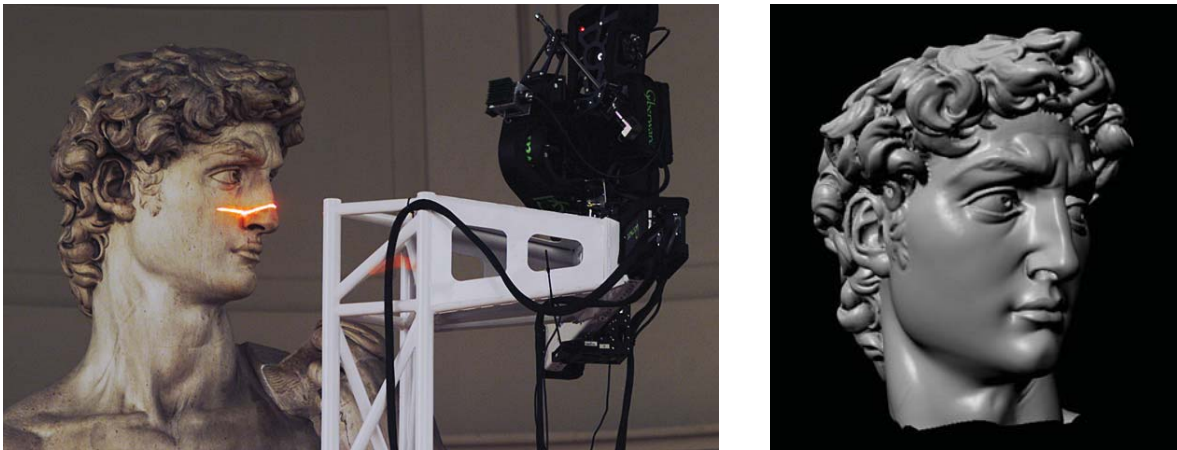


Figure 2.1: Left: Laser scanner used in the Digital Michelangelo Project [LPC⁺00]. Right: Result of a scan of the David statue. (Images taken from <http://graphics.stanford.edu/projects/mich/>)

single dot has the disadvantage that it requires the scanner to swipe the whole scene point-by-point. A more efficient technique is to project a line as shown in figure 2.1. This allows the scanner to recover a bigger part of the scene in one scan, speeding up the acquisition. In general multiple scans, which then have to be stitched together, are required to capture an object completely. The output of a laser scan is a dense 3D point cloud of the scene. Using meshing algorithms the point cloud can be converted into a mesh representation which can then for instance be rendered. Figure 2.1 shows a rendering of a laser scan. Laser scanners are highly accurate and often used for creating 3D models of real objects for use in cultural heritage preservation [LPC⁺00] or the movie industry. However, since the laser has to swipe the surface, this method is not real-time capable. In addition, reflective and (semi-)transparent surfaces pose a problem for this method due to their optical properties.

Structured Light

The principle behind structured light [BMS98] is similar to that of laser scanning. However, instead of projecting a single dot or a line a two-dimensional light pattern is projected onto the object. This pattern is often chosen to be a set of parallel stripes. To help identify single stripes in the pattern, alternating patterns are projected forming a gray code for each stripe [SI85] (see the middle two images in figure 2.2). By observing the deformation of the light pattern the shape of the object can be inferred. If desired the accuracy can be improved by performing multiple acquisitions with slightly shifted light patterns.

Structured light was used for instance to obtain the ground truth for the Middlebury stereo evaluation [SS03] which is up to this date the standard evaluation set for multi-view reconstruction algorithms and therefore requires very accurate ground-truth. Since the projection and the acquisition of the light pattern can be performed quite fast this method is used in real-time industrial inspection and production processes. As with laser scanners reflective and (semi-)transparent surfaces can be problematic.

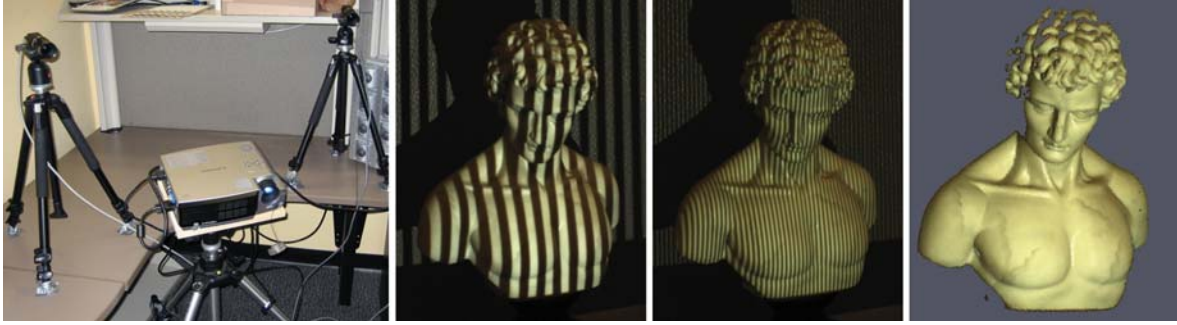


Figure 2.2: From left to right: Projector camera system for structured light, two images used to encode the stripe position with a gray code, reconstruction result. (Taken from <http://mesh.brown.edu/dlanman/courses/cs220/Lanman-structured-light.pdf>)

Photometric Stereo

Photometric stereo [Woo80] is different from the previously discussed methods in that it only uses photometric information for determining the shape of an object, while the previously described active methods use geometric information (i.e. point or line correspondences) in order to determine the object shape. The basic idea behind photometric stereo is to vary the direction of the light source while keeping the camera viewing direction constant. This creates different intensities at each surface element due to the reflection properties of the surface. Since the camera viewing direction is constant pixel correspondences are given automatically and no feature matching is required. Using the recovered intensities for different light source directions together with a reflectance model of the surface, the surface normal can be determined at each pixel. By integrating the normals the shape of the object is recovered.

In its standard form photometric stereo cannot be used to recover the shape of moving objects since it is necessary to acquire multiple images of the object under constant viewing direction but with changing lighting direction. A solution to this problem was proposed by Hernandez [HVC07]. They use three differently colored light sources and separate the color channels in the image in order to simultaneously acquire three images of the object under different illumination directions. This allows to also capture moving objects. Figure 2.3 shows some reconstruction results. The reconstruction times they report lie in the range of 20 seconds per frame making this method not real-time capable. In addition, the high complexity of the setup and the need for controlled illumination conditions with little ambient light make this method hard to use in a practical setting.

TOF Cameras

In contrast to laser scanners and structured light, TOF cameras do not require a complex setup and do not illuminate the scene in the visible spectrum. Instead they come in a handy camera-size format and can be used almost out of the box. These advantages have led to a quick adoption of their use in the computer vision community. TOF cameras produce a depth map of the scene by sending out a modulated infrared flash. By measuring the phase

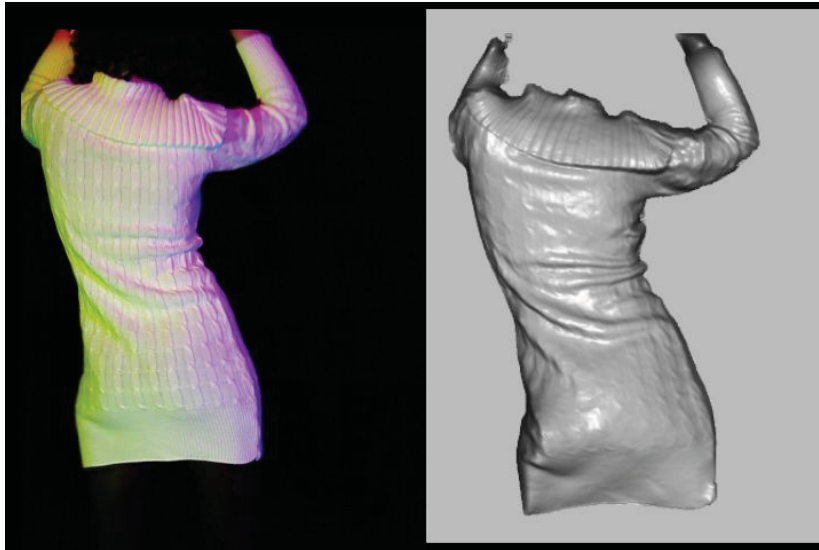


Figure 2.3: Reconstruction results for a sweater using photometric stereo (taken from [HVC07])

shift of the reflected flash the scene depth with respect to the camera can be computed. Currently the resolution of TOF-cameras is still rather limited (e.g. 176×144 for the Swissranger SR4000, see figure 2.4) and their accuracy cannot be compared to that of laser scanners and structured light. TOF cameras operate at real-time frame rates and are therefore also used in real-time applications. As most active methods they have problems with objects with unfavorable reflective properties which leads to partial outliers in the depth estimation. These have to be corrected by appropriate post-processing. Figure 2.4 shows the result of a post-processed depth-map obtained with a TOF camera.

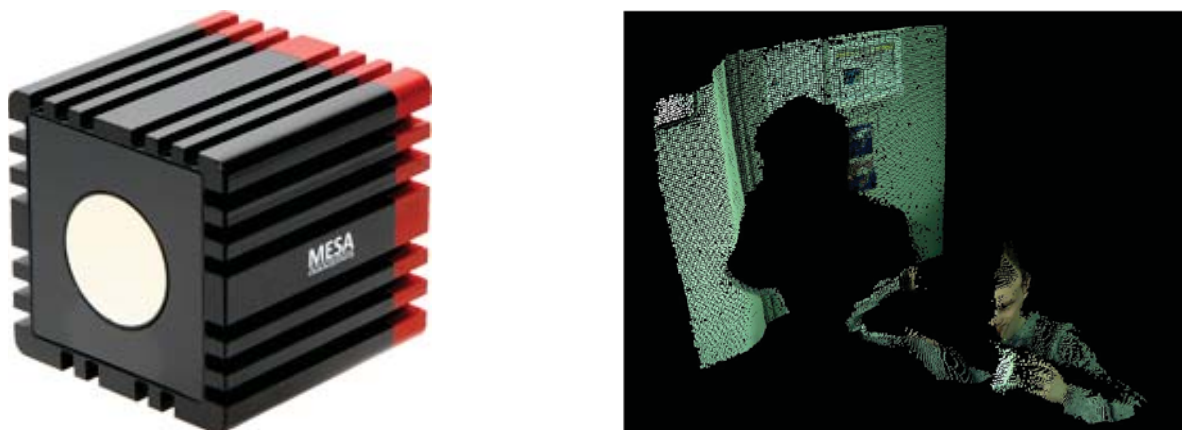


Figure 2.4: Left: TOF camera SR4000 from Mesa Imaging (www.mesa-imaging.com) Right: Depth Map produced by a TOF camera (taken from [HSJS08])



Figure 2.5: Two images from the Middlebury stereo evaluation dataset and the corresponding disparity map.

2.1.2 Passive Methods

Passive 3D reconstruction methods do not actively scan the scene, but only use photometric information. This makes these methods more general and easy to use. In contrast to active methods there are, however, additional problem cases such as untextured objects.

Stereo

Stereo is one of the oldest methods in computer vision for recovering the 3D-structure of a scene. In general two cameras are assembled into a stereo setup with a given baseline. To recover the scene structure, point correspondences between the left and the right image have to be obtained. To simplify this task the images are rectified, warping them in such a way that corresponding points appear on the same scan line in both images. The x-displacement along a scan line between the left and the right image is called the disparity and can be used together with the stereo geometry to compute the point depth. Figure 2.5 gives an example taken from the Middlebury stereo evaluation [SS02] showing two input images and the corresponding disparity map.

Since the goal is to obtain a dense disparity map, every pixel in the first image should be matched to a pixel in the second image assuming that there are no occlusions present. The simplest class of matching methods uses a local window around the point of interest in the left image and searches along the corresponding scan line in the right image to find the best matching window. Typical similarity measures used for matching include the SSD and the NCC. An overview of different image similarity measures used in stereo matching is given in [HS09]. Due to their simplicity these methods are quite fast and even real-time capable. However, their accuracy is limited, since they do not enforce any kind of regularization making them problematic in regions of low texture. Other methods perform regularization along the scan lines using for instance dynamic programming [Bel96]. There also exist graph-cut and variational methods which also consider regularization between scan lines and therefore in the whole image [KZ01]. Their results are more accurate than the results of local methods. However, they are not real-time capable. A good overview of dense stereo algorithms is given in [SS02]. Using a single stereo pair one can only obtain a depth map of the scene. In order to obtain a complete 3D reconstruction multiple stereo pairs have to be acquired and their corresponding depth maps have to be fused. This is time consuming and the merging



Figure 2.6: Input images and reconstruction results from the Middlebury multi-view evaluation. Left: *temple* [GSC⁺07]. Right: *dino* [FP07]

of the depth maps can lead to artifacts.

Multi-View Stereo

In contrast to traditional stereo methods, multi-view stereo uses a set of images for recovering the 3D structure of the scene. A recent overview of this area is given by Seitz *et al.* [SCD⁺06]. Often an initial estimate of the scene structure is given (e.g. by the visual hull or a bounding box). This initial estimate is then refined using different techniques.

In space carving [SD99, KS00] the scene is discretized into voxels and each voxel is checked for photo-consistency. Inconsistent voxels are removed from the reconstruction until only the photoconsistent part of the scene (the photo hull) remains. The drawback of this approach is that it does not contain any regularization and that it is greedy, so that a voxel which has been removed once cannot be restored later on. This can lead to the carving of the whole scene, when the photoconsistency model (which is lambertian in general) fails in regions of specular reflection.

Variational methods start out with an initial mesh or a level set representation of the scene and deform it using a gradient descent scheme based on surface photoconsistency [FK98, PKF07]. As all gradient-descent methods variational methods are local and can fall into local minima.

Graph-cut based methods [KZ02, VETC07] like variational approaches try to find the maximum photoconsistent reconstruction. However, they are global in nature and are therefore less likely to fall into local minima. As with all photometric methods, an accurate photo measure is needed and the assumptions underlying its design (such as lambertian surface properties) have to be fulfilled.

Next to these methods which deform a given reconstruction there also exist reconstruction methods which start by finding features on the object using feature detectors. Then patches are grown around these features and new patches are added at the borders of the initial ones [FP07]. If available some multi-view methods also include silhouette constraints. Yet other methods start out using a dense stereo algorithm on image pairs and fuse the resulting depth maps [BBH08]. Multi-view stereo methods are not real-time capable but deliver the

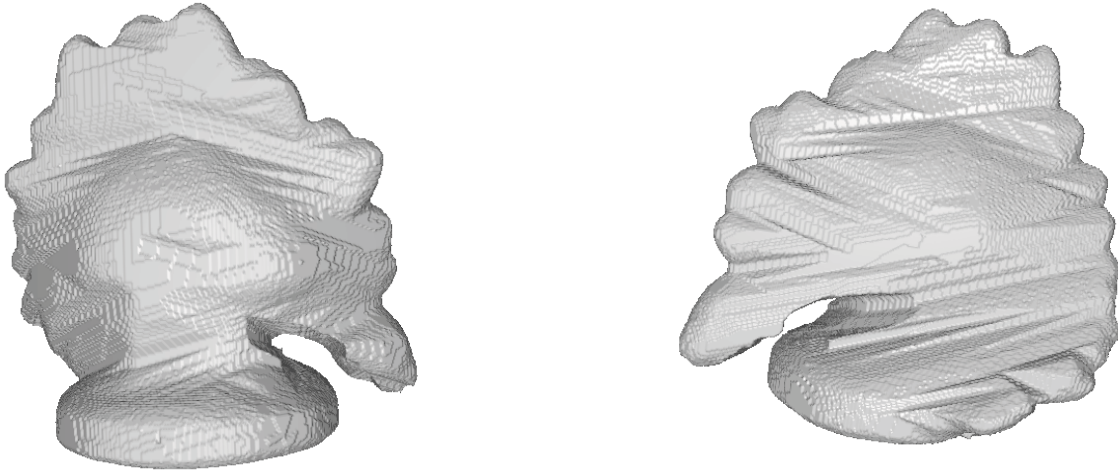


Figure 2.7: Visual hull of the *dino* dataset from the Middlebury multi-view stereo evaluation.

best possible results when using only photometric information. Some reconstruction results on the Middlebury multi-view evaluation [SCD⁺06] are shown in figure 2.6.

Shape from Silhouette

In contrast to stereo and multi-view stereo, Shape-from-Silhouette methods [Lau94, Sze93] do not directly use any photometric information about the scene and in particular are not concerned with the photometric consistency of the reconstruction. Instead they assume the availability of silhouette images showing the foreground objects in the scene and pose the problem purely geometrically as that of finding the shape which is maximally consistent with the observed silhouette images. The shape they recover is called the visual hull. Figure 2.7 shows the visual hull of the *dino* dataset from the Middlebury multi-view stereo evaluation [SCD⁺06].

Due to the simplicity of the reconstruction process Shape-from-Silhouette methods are real-time capable. However, they have the drawback that they require silhouette images, which are not always available. In addition, the visual hull is in general only a superset of the true reconstruction and does not have the precision reached by the previously described methods. On the other hand uniformly textured objects do not pose a problem for this method as long as silhouette images can be obtained (e.g. by background subtraction). This makes Shape-from-Silhouette approaches interesting for real-time systems which require speed and robustness. More details on this method will be given in the next chapter.

2.2 Multi-Camera Reconstruction Systems

In the last decade many systems for 3D reconstruction have been proposed, most of them using the visual hull. While early systems did not achieve real-time performance due to the limited technology of their time, present systems can run at real-time frame rates.

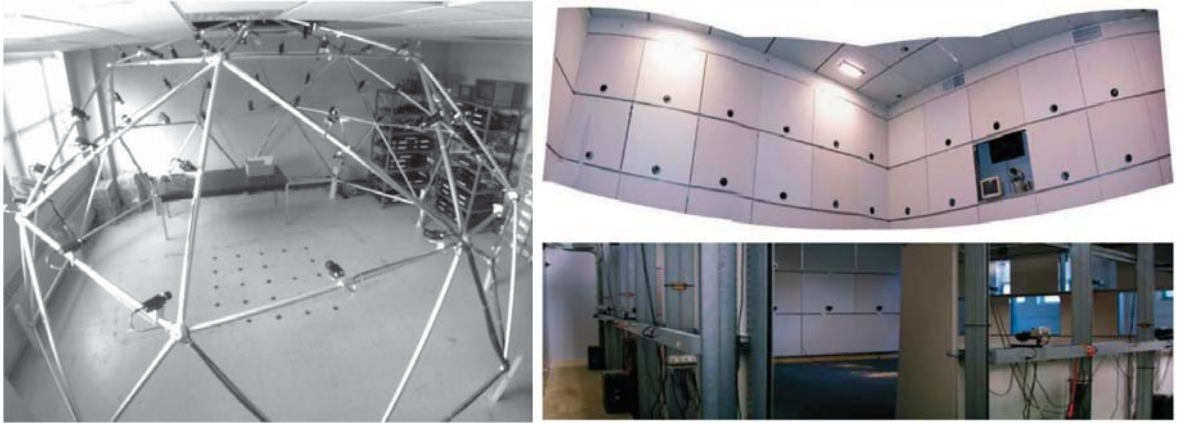


Figure 2.8: Left: CMU dome. Right: CMU Virtualization Studio. Images taken from [KN07]

One early system constructed around 1995 which was using stereo instead of the visual hull was the CMU dome [KRN97] consisting of 51 cameras mounted on a dome 5 meters in diameter (see figure 2.8 left). The cameras were analog and used 3.6 mm lenses in order to achieve a near 90 degree viewing field. This resulted in an overlapping area about 3m x 3m x 2m in size. The cameras were synchronized and a synchronization signal was outputted on video. The video was recorded on S-VHS tapes for later digitization and processing. Hence, this system was not real-time.

In 1998 the 3D room was constructed at CMU [KN07]. It used digital image acquisition using 49 color cameras. However, due to the limited memory size of the time, the recording time was limited to about 800 frames which is about 18 seconds at 15 fps (see figure 2.8 right).

Later in 2002 the CMU Virtualization Studio was built using 48 cameras attached to 24 PCs enabling fully digital image acquisition. The size of the acquisition space was 6.1m x 6.7m x 4.3m. The cameras were using 3CCD chips (i.e. one chip per color channel) which made the system quite expensive.

Another multi-camera studio not directly aimed at real-time performance was proposed by Starck *et al.* [SMN⁺09] using 8 HD cameras. They cover a working volume of 4m x 4m x



Figure 2.9: Multi-view systems at Surrey, Kyoto and INRIA respectively. Note the controlled lighting and the uniform background to improve the background subtraction results. (Images taken from [SMN⁺09] and GrImage website)



Figure 2.10: Blue-c cave and blue-c power wall. Images taken from [GWN⁺03]

2m. For reconstruction they use a visual hull based approach. The initial visual hull is then further refined using graph-cut techniques [SH07b]. In order to simplify the background subtraction task controlled lighting and a blue background are used (see figure 2.9).

At Kyoto university a system consisting of 15 cameras was developed [MWTN04]. For reconstruction a visual hull based approach is used. The visual hull is subsequently improved by deforming the recovered mesh based on photoconsistency and silhouette forces [MWTN04]. Wu *et al.* [WM03, WTM06] use the system with a plane intersection test for visual hull reconstruction. They report values of 12 fps with 9 cameras. This system also uses controlled lighting and a uniform background in order to simplify the background subtraction (see figure 2.9).

INRIA Grenoble proposed the GrImage platform [AFM⁺06]. The GrImage platform uses a polyhedral visual hull approach for reconstruction and achieves real-time performance. The system is used among other things for interaction between virtual and real objects [PLBR09]. It has been commercialized and is being sold to industry and university clients. The number of cameras is varying depending on the current setup. Franco *et al.* [FMBR04] parallelize the polyhedral approach over a cluster of 8 dual Xeon 2.66 GHz PCs to achieve frame rates of 30 fps using four cameras. Hasenfratz *et al.* [HLGB03, HLS04] also report frame rates of 30 fps with four cameras using a voxel-based method (resolution 64^3) implemented on the GPU of a SGI Onyx IR-3000 with 8 R12000 processors. Allard *et al.* [AFM⁺06, AMR⁺07] use the system with a polyhedral visual hull algorithm using 11 dual-Xeon 2.6 GHz PCs. They report frame rates of 30 fps using 6 cameras. A controlled background and professional lighting are used to simplify background subtraction (see figure 2.9).

The ETH Zurich developed the blue-c system [GWN⁺03]. It has a reconstruction area of $3\text{m} \times 3\text{m} \times 2.5\text{m}$ and uses 16 cameras. For the real-time reconstruction the visual hull is used. On the technical side this system proposes a very interesting way to combine a cave environment with a reconstruction system. The user is standing in a cube enclosed by large projection screens. These screens can be switched between being transparent and displaying the cave visualization. The cameras are synchronized with the transparent phase of the screens. Due to the rapid switching speed the user does not notice the switching (see figure 2.10).

The Max-Planck Institute also proposed a system running in real-time. Theobalt *et al.* use this system with 8 cameras [TLMpS03]. Li *et al.* [LMS03a] use the system with a polyhedral visual hull reconstruction approach and report frame rates of 15 fps using 6 cameras.

2.3 Conclusion

There exist many techniques for 3D reconstruction. However, only structured light, time of flight cameras and Shape from Silhouette are real-time capable. The disadvantage of structured light methods is that they usually use a light pattern in the visible spectrum which can be seen by the user apart from requiring a complex setup. Time of Flight cameras yield promising results but are still somewhat immature in terms of resolution and accuracy. Shape from Silhouette methods on the other hand are unintrusive and quite robust, requiring only silhouette images which can be obtained by background subtraction even for non-textured objects. The downside is that the visual hull is only an approximation to the true shape. However for most applications this is not a major constraint which is underlined by the fact that all current real-time reconstruction systems use this approach. If a more accurate reconstruction is desired the initial visual hull can be further processed using multi-view stereo methods in an offline step. In addition, most current systems use professional lighting and a uniform background in order to simplify the background subtraction stage and to obtain more accurate results.

Chapter 3

The Visual Hull

As seen in the previous chapter the visual hull lies at the heart of almost all current 3D reconstruction systems. In this chapter we will give a formal definition of the visual hull, discuss its properties and present different methods for computing it in practice.

The decision to use the visual hull in our system is based on the possibility to compute it in real-time using calibrated silhouette images, its robustness to segmentation errors and the fact that it also works for untextured objects. Other 3D reconstruction methods targeted at producing high quality results are too computationally expensive, since they require costly optimizations over the shape of the object, as discussed in the previous chapter. The visual hull on the other hand is straightforward to compute from a set of calibrated silhouette images. Although being only an approximation to the true shape, it is sufficient for many application areas since it captures the essence of most shapes encountered in a multi-view studio well.

3.1 Definition and Properties

The visual hull [Lau94] is defined as the shape of maximum volume whose projection is consistent with the silhouette images. Geometrically the visual hull is obtained by intersect-

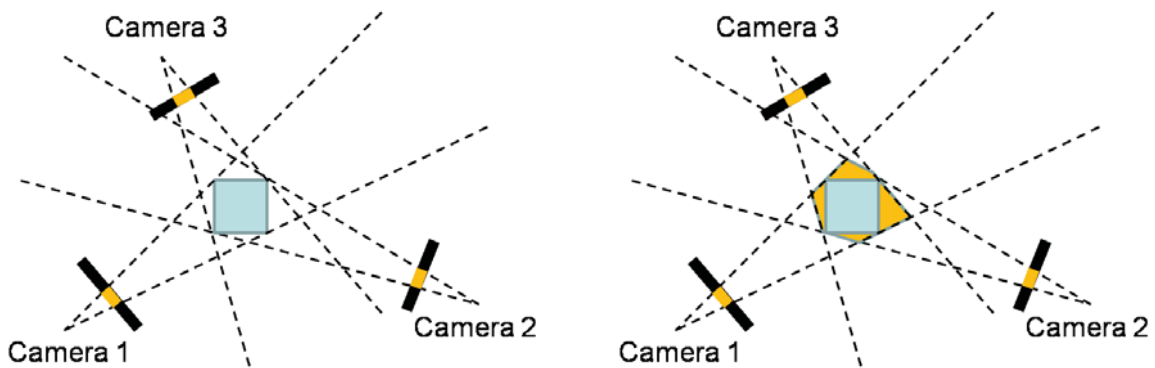


Figure 3.1: Visual Hull construction for a 2D example.

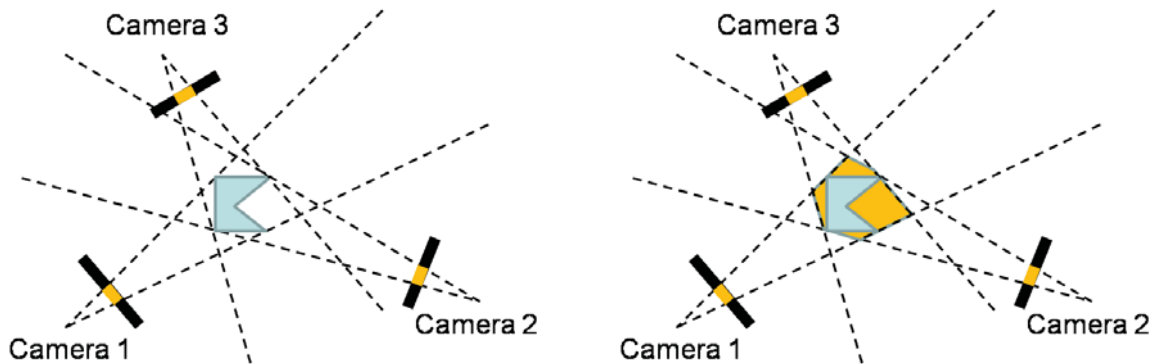


Figure 3.2: Visual Hull construction for a 2D example with a concave object. Concavities not visible in the silhouette images cannot be recovered.

ing the generalized viewing cones created by the viewing rays emanating from the camera center and passing through the silhouette contours.

The visual hull has several important properties. We will illustrate these properties using the simple 2D example in figure 3.1 which shows a setup with a square which is imaged by three cameras. In 2D the generalized cones simplify to (infinite) triangles. For each camera an infinite triangle is created by extending the rays passing through the camera center and the silhouette borders to infinity (the dashed lines in the figure). Then all triangles are intersected with each other and only the overlapping part is kept. This is the visual hull. As can be seen from the illustration, the visual hull only approximates the shape of the object. However, it is the best estimate of the shape which is possible given the silhouette images.

As more cameras are added the approximation becomes increasingly more accurate. If the object is convex the visual hull will become identical to the true shape as the number of cameras approaches infinity. This is however not the case for some non-convex objects, because concavities not seen in the silhouette images can by construction not be recovered. This is for instance the case for the example in figure 3.2. Since the concavity does not appear in any silhouette images - regardless of where the cameras are placed - it is not recovered and the reconstruction is identical to the one obtained in the previous example.

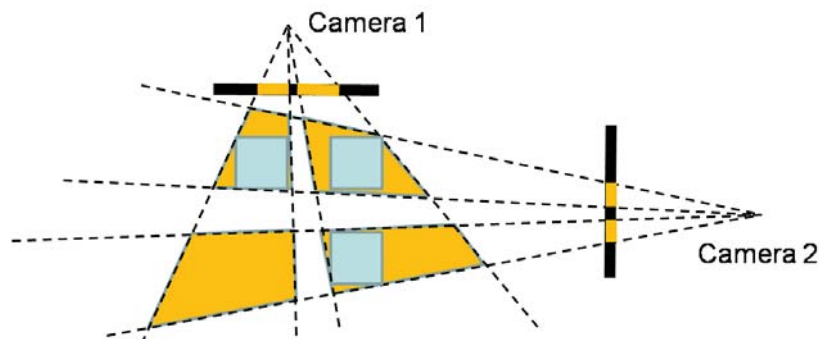


Figure 3.3: Visual Hull construction with multiple objects for a 2D example. Due to the limited number of cameras a fourth object which does not actually exist is reconstructed.

Using a too small number of cameras often leads to artifacts in the reconstruction, especially when multiple objects are part of the scene. Figure 3.3 gives an example of this problem. Due to the limited number of cameras a fourth object is reconstructed which does not exist in reality. This problem can usually be solved by adding more cameras.

3.2 Visual Hull Computation

Many methods have been proposed for computing the visual hull in practice. The most popular ones are the volumetric and the polyhedral approach. Therefore, we will focus in particular on these two methods and discuss other methods more briefly.

3.2.1 Volumetric Approach

In the volumetric approach the visual hull is computed using a voxel representation of space [Pot87, Sze93]. The reconstruction volume is discretized into voxels. This requires the extent and the placement of the reconstruction volume in the world coordinate system to be known. Each voxel in the reconstruction volume is projected into all of the silhouette images using the known camera projection matrices. If the projection of a voxel lies inside the silhouette in all images it is marked as occupied, otherwise it is marked as empty. This is justified by the fact, that if the projection of a voxel is outside the silhouette in even one of the silhouette images, no object lies on the ray passing through this voxel, i.e. the voxel has to be empty. While the fact that a voxel projects outside the silhouette in at least one of the silhouette images is a positive proof that the voxel is indeed empty (assuming the silhouette images are correct), the fact that a voxel projection is inside all silhouettes does not necessarily mean that it is also occupied (think of a hollow half-sphere which will be reconstructed as a solid half-sphere). This is the reason why the visual hull is only an approximation of the true shape. Algorithm 1 outlines the volumetric computation approach.

The volumetric approach is very robust since it does not rely on the silhouette contours but on the foreground regions in the silhouette images. One shortcoming of the volumetric approach is the discretization of space. Depending on the size of the voxels discretization artifacts can become visible in the final result. This can be seen in the example in figure 3.4.

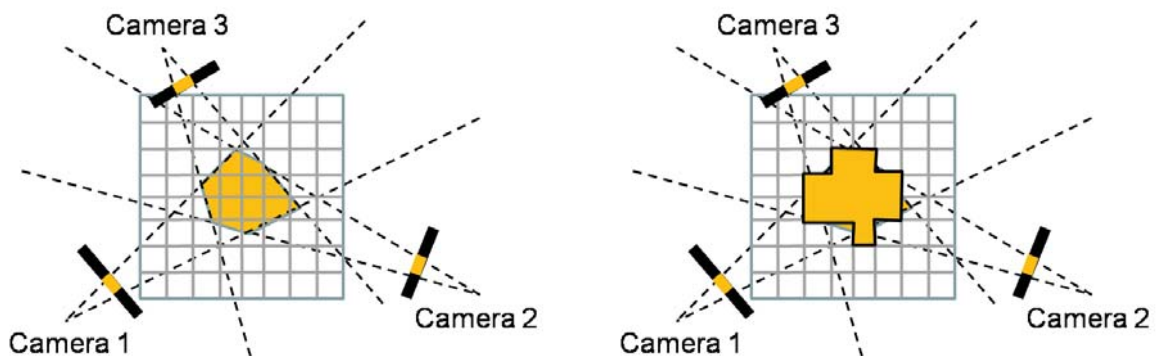


Figure 3.4: Visual Hull discretization in the volumetric approach.

Algorithm 1 Volumetric Visual Hull Computation

```

1: for each voxel  $v$  do
2:   occupied = true
3:   for each silhouette image  $I$  do
4:     Project  $v$  into  $I$ 
5:     if projection outside silhouette then
6:       occupied = false
7:     end if
8:   end for
9:   if occupied then
10:    Mark voxel as occupied
11:  else
12:    Mark voxel as empty
13:  end if
14: end for

```

While it is in principle possible to make each voxel small enough that discretization does not play a significant role anymore, this is made difficult in practice due to memory and runtime constraints. On the other hand the volumetric approach is ideal when the target application only requires a certain accuracy (for instance for collision avoidance). In that case the voxel size and therefore the computational load can be limited to the required accuracy.

The volumetric method as described up to now is very easy to implement. However, it has a major problem. Every single voxel has to be tested for occupancy. Although a single test is very fast the number of voxels required for a reasonable reconstruction (e.g. $128^3 = 2,097,152$) makes it almost impossible to achieve real-time performance. A significant speed-up can be achieved by using an octree representation of space [Sze93]. In this case the reconstruction volume is partitioned into an octree structure. In the beginning the whole reconstruction volume consists of only one octree cell. Each octree cell is treated like a voxel in the flat approach, i.e. it is projected into all silhouette images. The difference to the flat approach is that the octree cell can be subdivided if necessary. In practice this can be achieved as follows: First it is checked whether the projection of the cell is fully outside the silhouette in any of the silhouette images. If so the cell is considered empty and not subdivided further.

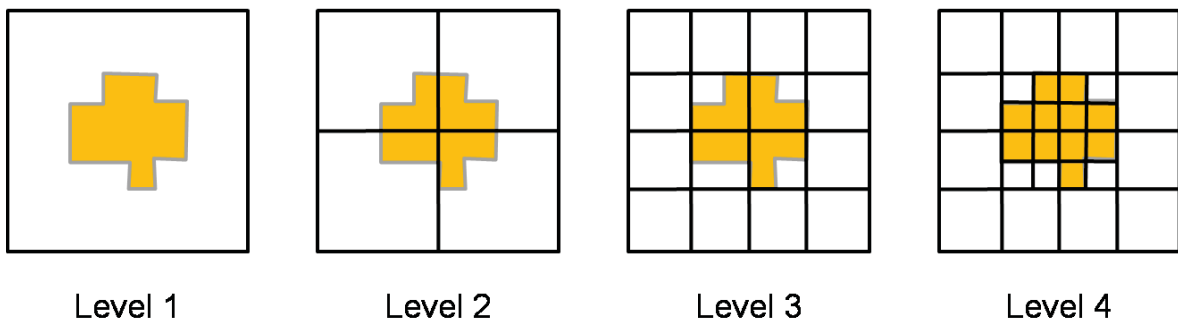


Figure 3.5: Visual Hull computation using an octree representation of space. If an octree cell spans both foreground and background regions it is subdivided further.

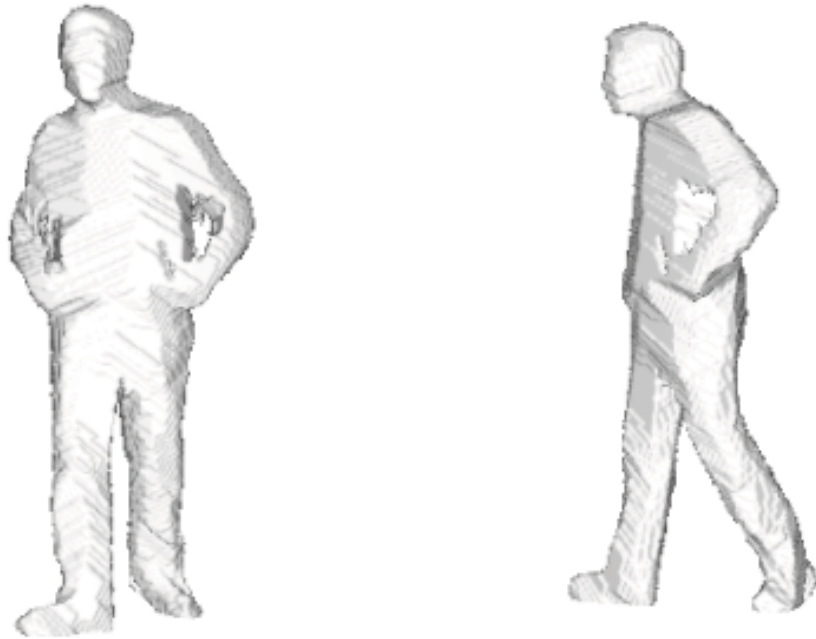


Figure 3.6: Result of the visual hull computation using the volumetric approach on a data set of a walking person consisting of 16 images.

The same is true if it projects inside the silhouettes in all silhouette images. In this case it is marked as occupied and not subdivided any further. In all other cases it is subdivided into 8 subcells (4 subcells in 2D) of equal size. The computation is finished when no cell has to be subdivided any further or when the maximum octree depth is reached. By using this approach the computations are significantly sped up since large empty or filled regions of space can be quickly skipped. Figure 3.5 shows a 2D example with 4 levels. Even in this simple example a significant improvement in the number of occupancy checks which have to be performed can be observed. In the octree approach only 38 occupancy checks have to be performed while in the flat approach 64 occupancy checks have to be performed. To achieve even further speed-ups the computations can be distributed over multiple processors [SMRR07] or over a distributed system.

The output of the volumetric method is a voxel volume. Depending on the application it might be more appropriate to have a mesh-based representation of the scene, for instance for rendering. This can be achieved by applying the Marching Cubes [LC87] algorithm to the voxel volume. Figure 5.4 gives an example of the visual hull obtained on a data set of a walking person consisting of 16 images.

3.2.2 Polyhedral Approach

In the polyhedral approach geometric properties are used to compute a mesh representation of the visual hull [MBM01, FB03, LFP07]. In [FB03], for instance, the silhouette contours are first discretized into polygonal image contours. The vertices of the polygonal image

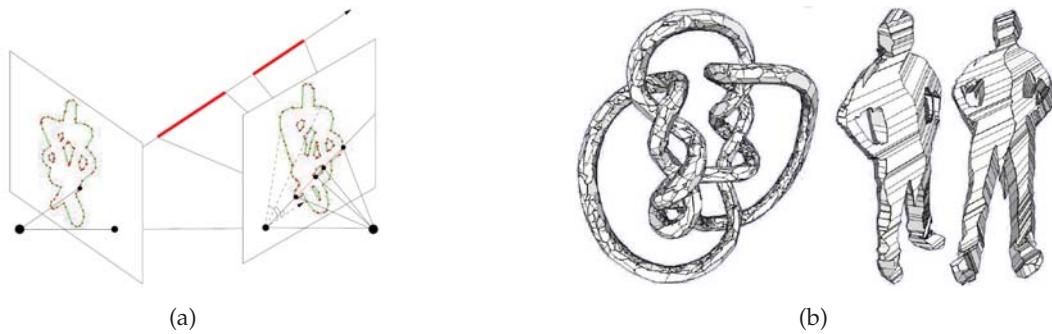


Figure 3.7: (a) Illustration of viewing edge computation. Only the red segments (i.e. the viewing edges) of the viewing ray are retained. (b) Visual Hulls obtained using the polyhedral approach. Note that the resulting meshes consists of quite irregular facets. Images taken from [FB03].

contours are backprojected to space creating lines in 3D. These 3D lines are then projected into the other silhouette images and only the segments projecting inside the silhouettes in all other images are retained (see figure 3.7 (a)). These segments are called viewing edges and by construction lie on the visual hull. However, they only form an unconnected subset of the desired visual hull mesh. In a further processing step the connections between the viewing edges as well as missing edges are found and added to the mesh. This results in a watertight visual hull mesh which projects exactly into the polygonal image contours. In a final step planar contours are identified for each face of the polyhedron. Typical results are shown in figure 3.7 (b). As can be observed the facets of the recovered meshes are quite irregular. This poses problems for certain mesh-based algorithms such as dense surface tracking [CBI10b] or variational approaches requiring a regular sampling of the mesh.

The advantage of the polyhedral approach lies in the fact that the resolution of the reconstruction is defined in image space. Hence, no accuracy is lost due to voxel discretization of space as is the case for the volumetric approach. However, the computational cost increases quickly with finer silhouette discretization and an increasing number of images. Another shortcoming of this approach is that it is susceptible to errors in the silhouette images and calibration errors, since the intersections of viewing cone boundaries are then not well defined and sensitive to numerical instabilities. To achieve real-time performance a method for distributing the computations over multiple processors has been proposed [FMBR04].

3.2.3 Other Approaches

There also exist other approaches for computing the visual hull. They are often targeted at specific problem settings, which are different from the standard problem where the silhouette images are considered to be binary and given in advance. The most important group of these approaches are the implicit view-based methods, although they cannot be considered full visual hull computation methods since they only produce a view-based representation of the visual hull instead of recovering the full 3D shape.

Implicit view-based approaches

Implicit approaches [MBR⁺00, LMS03b, LMS03a, LMS04, MH06] do not compute the visual hull explicitly, but only render images of the visual hull from a given viewpoint. These methods are interesting for the case where an explicit representation of the scene is not required. This is typically the case when the goal is to render the scene from a new viewpoint. Due to the fact that only a view-based reconstruction is performed, these methods achieve a better performance than explicit methods. In terms of computations the implicit methods are similar to the polyhedral approach. However, they only focus on computing line segments and visual hull patches visible in the target view, instead of building the full mesh. This is the reason why they are more efficient than for instance the polyhedral approach. At the same time they are not suited for applications which require the full information about the shape of the reconstructed objects or which need to perform post-processing on the reconstruction. Figure 3.8 shows some reconstruction results obtained using the method proposed by [MH06].



Figure 3.8: Results of the exact view-dependent visual hull algorithm presented in [MH06]. The results visually resemble the output of the polyhedral method. However, no full reconstruction is performed. Only the parts of the visual hull necessary for rendering the object from the given viewpoint are computed.

Probabilistic approach

In [FB05] a probabilistic approach for computing the visual hull is presented. The segmentation of the input images is not considered a binary process, but instead the probability of every pixel being inside the silhouette is modeled explicitly. By considering the probabilities of all pixels into which a voxel projects, a probability of occupancy is computed for every voxel. The resulting probabilistic voxel volume is then thresholded in order to extract the highest probability surface.

Graph-cut approach

In [SVZ00] a graph-cut based method incorporating smoothness constraints is presented. Similar to the probabilistic approach it does not make hard decisions regarding the silhouette images, but instead uses the difference between the current image and a prerecorded

background image to estimate the voxel occupancy. For each voxel the intensity differences in the voxel projections are considered. Additionally, a smoothing term is used, which penalizes the assignment of different occupancy states to neighboring voxels. The cost function optimized in the graph cut is of the form

$$E(f) = \sum_{v \in V} D_v(f_v) + \lambda \sum_{v, v' \in N(v)} (1 - \delta(f_v - f_{v'})) \quad (3.1)$$

where f is the labeling of the voxels (i.e. inside or outside the visual hull), V is the voxel set, $D_v(f_v)$ is the cost of assigning label f_v to voxel v , $N(v)$ is the voxel neighborhood of v (e.g. the 6 or the 27 neighborhood) and δ is the unit impulse function being 1 at zero and 0 everywhere else. λ weights the contribution of the smoothing term. The choice of D_v is critical for the reconstruction result. It should be chosen to be large when the intensity difference is large in all images and small otherwise.

Hybrid approach

In [BF03] a method is proposed which partially combines the polyhedral and the volumetric approach. It aims at developing a space carving method which avoids the complexity due to the regular space discretization associated with the volumetric approach by using some techniques from the polyhedral approach. The idea behind this method is to first identify points lying on the visual hull surface. For this purpose the end points of the viewing edges as computed in the polyhedral approach are used. Based on these points a Delaunay tetrahedrization of space is built. In a final step each tetrahedron is checked and marked as belonging or not belonging to the visual hull. In this approach the tetrahedrons can be likened to the voxels in the volumetric approach. However, their shape corresponds more closely to the actual visual hull surface and their density is only high around the actual visual hull surface and not in the whole volume as in the volumetric approach. Figure 3.9 shows some results obtained using this method.

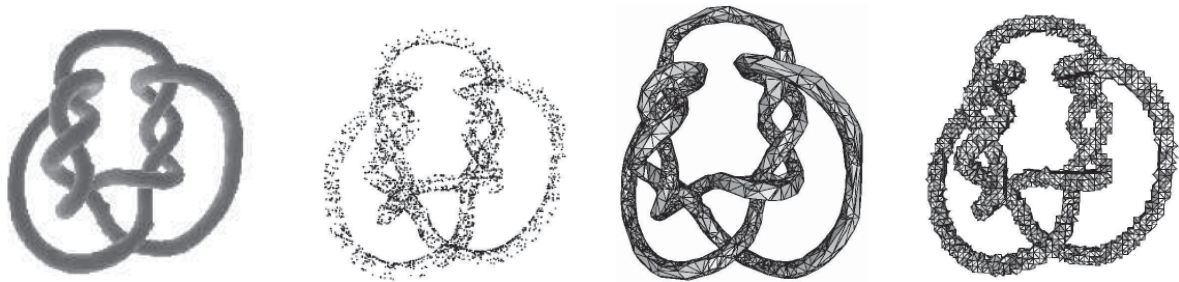


Figure 3.9: Results of the hybrid method. From left to right: One of the input images, point cloud obtained from viewing edges, hybrid reconstruction, reconstruction using standard voxel-based approach (resolution 60). Images taken from [BF03].

Visual Shapes of Silhouette Sets

In [FLB06] a method is introduced which instead of only computing the visual hull computes a whole class of visual shapes consistent with the silhouette images. In this class the visual

hull is the member with the maximal volume. Like the hybrid approach this method is also based on the viewing edges used in the polyhedral approach. Instead of taking the whole viewing edge only a thinned version or a single contact point is retained. In the case of retaining only a single contact point a local second order assumption is made on the shape. This yields smoother results and is in particular useful for objects of which it is known that they are locally smooth. Figure 3.10 shows some visual shapes of a sphere obtained using this method.

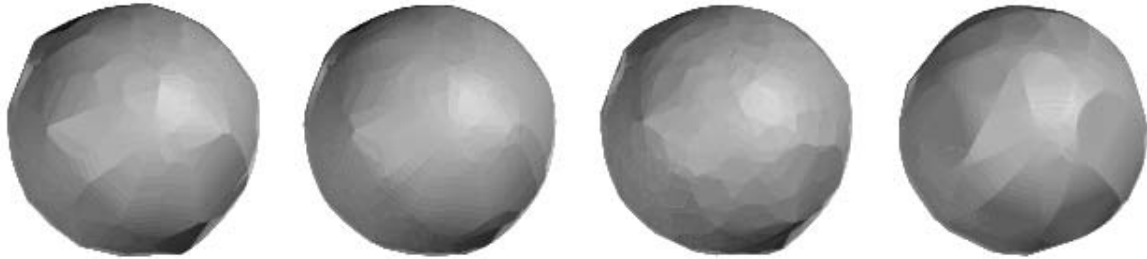


Figure 3.10: Examples of visual shapes of a sphere using 10 images. The images show: visual hull, visual shape with slightly thinned viewing edges, visual shape with one randomly chosen contact point, visual shape with one contact point and second order constraints. Image taken from [FLB06]

3.3 Conclusion

In this chapter we discussed the visual hull which is the concept lying at the core of almost all current real-time 3D reconstruction systems. The reason for its popularity are that it is robust, does not rely on texture information and can be computed in real-time. However, it is generally only an approximation to the real object shape and cannot be used to recover concavities not seen in any of the silhouette images. For computing it, there exist several algorithms, the most popular ones being the volumetric approach which uses a discretization of space and the polyhedral approach which uses geometric principles. There also exist other approaches for more specialized scenarios, most notably the implicit algorithms which do not recover an explicit 3D representation of the visual hull, but instead only produce a rendering showing the visual hull from a given viewpoint.

Chapter 4

Proposed 3D Reconstruction System

In this chapter we present our real-time 3D reconstruction system. We discuss the design of the system, including the calibration procedure, the background subtraction stage, the reconstruction strategy and the visualization, as well as the distribution of the computational load over a PC cluster and the software and hardware measures taken to ensure a stable and robust operation. We also focus on the requirements for use of the system in an interventional environment.

4.1 Overview

Our system is designed so that the computations are distributed over a PC cluster. This makes the system scalable since it is always possible to add more computers in order to accommodate more cameras. The whole system is controlled from a single computer which performs the visualization of the reconstruction results and provides an interface for accessing the reconstructed scene by other applications. The remaining computers are dedicated to capturing the images from the cameras and performing the steps required for reconstruction. Since we are targeting interventional environments we also take the particular requirements of such a setting such as constraints on camera placement and the complex nature of the scene into account.

4.2 Hardware

We use a PC cluster consisting of five computers and a total of 16 cameras. The cameras are synchronized using a hardware trigger signal in order to assure synchronous image acquisition. The following sections explain all the hardware aspects of the system in detail.

4.2.1 PC Cluster

There are a total of five PCs in our system. Four of these (the slaves) are dedicated to capturing images from the cameras and to performing a partial reconstruction. They are equipped

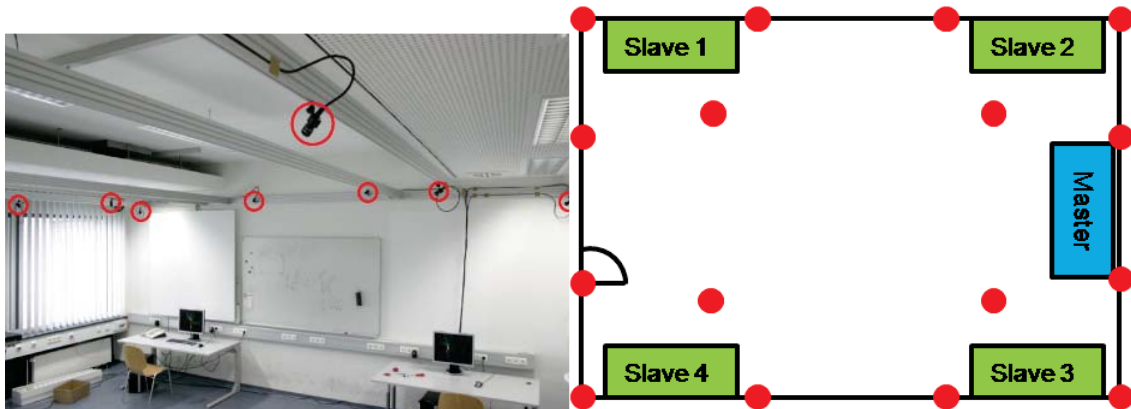


Figure 4.1: Lab setup for our real-time 3D reconstruction system. The cameras are marked with red circles (not all cameras are seen in this image).

with an Intel 2.6 GHz Quad-Core CPU (Q6700), 2 GB of main memory and a NVIDIA 8800 GTX graphics board with 768 MB of memory. In addition each PC has two IEEE 1394b adapter cards for connecting up to four cameras. The fifth PC (the master) controls the other PCs and visualizes the result. It is equipped with an Intel 3.0 GHz Dual-Core CPU (E6850), 2 GB of main memory and a NVIDIA 8800 GTS graphics board with 640 MB of memory. The slave PCs have faster CPUs and GPUs because they are doing most of the processing while the processing load of the master is application specific and usually much lower.

The PCs are connected through a switched Gigabit-Ethernet network. Data transmission such as the sending of partial reconstructions from the slaves to the master is performed using UDP due to the real-time requirements and for avoiding the overhead associated with handling TCP connections. The control commands between the master and the slaves are sent using Multicast. This lowers the burden on the network and simplifies the configuration of the system, since a new slave machine only needs to know the address of the multicast group.

4.2.2 Cameras

We use a total of 16 PointGrey Flea2 IEEE-1394b cameras. The cameras are mounted on a movable aluminum frame suspended from the ceiling. This allows us to easily reconfigure the camera positions while at the same time avoiding unintentionally moving the cameras and thereby destroying the calibration. This is particularly important in an interventional environment, not only to avoid a costly and time-consuming recalibration of the system in case a camera is accidentally moved, but also to keep the cameras out of the working space of the medical staff. Figure 4.1 shows the aluminum frame and the cameras as well as a schematic of the camera layout. We do not use a top-down camera in the center, which is often done in other multi-view systems, since this is typically not possible in an interventional environment due to ceiling suspended monitors and guide rails. On the cameras we use 5 mm wide angle lenses in order to cover a big working volume using a comparatively small number of cameras. The actual working volume varies with the exact placement and orientation of the cameras. It is however in general around the dimensions of $4m \times 4m \times 2.5m$.



Figure 4.2: The trigger box used for synchronizing the cameras.

Images are acquired at a resolution of 1024×768 using the Bayer encoding and subsequently converted to RGB. The frame rate of the cameras depends on the trigger rate of the external synchronization system and can reach up to 30 Hz.

4.2.3 Synchronization

In order to obtain a consistent 3D reconstruction it is important to acquire images simultaneously on all cameras. This is achieved by using an external trigger signal which can be set to the desired frame rate. The trigger signal is generated by a standalone PC and is propagated to all cameras through a trigger box (see figure 4.2) which adjusts the trigger signal to the electrical requirements of the cameras.

4.3 System Architecture

As previously mentioned all computations are distributed over a PC cluster in order to achieve real-time performance. In addition to the global workload distribution each computer in the cluster distributes its local computations over all available CPU cores and the GPU.

The slave PCs are responsible for computing a partial reconstruction using their locally attached cameras. This computation can be split into several substeps, namely image acquisition, background subtraction, volume reconstruction and volume compression and transmission. The substeps are organized into a pipeline. Each step is running in a separate thread and therefore ideally on a separate CPU core. The reconstruction runs on the GPU. Due to the pipelining of the computations the segmentation of the current frame can for instance run in parallel with the visual hull computation of the previous frame. This increases the throughput of the system.

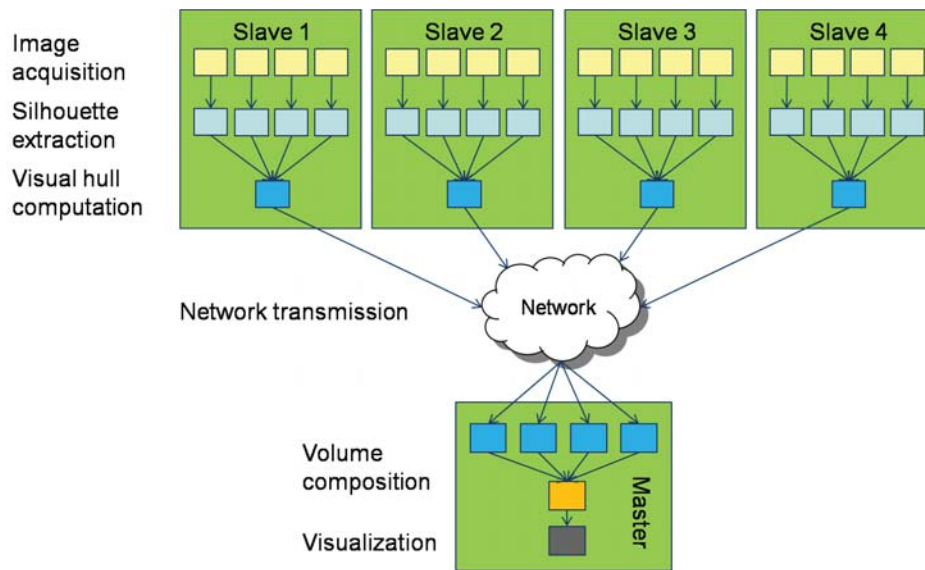


Figure 4.3: System architecture: Each slave acquires images from its locally attached cameras and computes the visual hull on the GPU. The resulting partial reconstructions are sent to the master which combines them and visualizes the result.

The master PC is responsible for composing the partial reconstructions and visualizing them. Here, as well, the processing is distributed into several steps. There is a separate thread for handling network communication, compositing the partial reconstructions and visualizing the result. In addition, an application specific stage using the reconstruction result can be introduced. An overview of the global architecture of our system is shown in figure 4.3.

4.4 Calibration

A good calibration of the system is essential for obtaining accurate 3D reconstructions. However, standard calibration methods [Zha00] were not designed for calibrating large multi-camera systems. They assume that a calibration object (usually a checkerboard) is visible in all cameras simultaneously. This is hard to achieve in a multi-camera setup since the overlapping area of all cameras is either small leading to an unstable calibration outside this area or there may not even be an area observed by all cameras simultaneously. To solve this problem and to allow easy and fast calibration of multi-camera systems Svoboda *et al.* proposed a multi-camera calibration system [SMP05]. Their method does not require a traditional calibration target but instead relies on the availability of point correspondences between the cameras. These can be created by moving a point light source through the room and detecting it in the images. In addition, their method does not assume, that points are visible in all cameras simultaneously. Therefore, even systems having no common overlapping region can be calibrated. Moreover, the calibration procedure is easy to learn and can therefore be performed by a non-expert. Typically a calibration run takes around 20 minutes. Both of these facts, the ease of use and the short calibration time, are important in an interventional environment.



Figure 4.4: Laser pointer used for creating the point correspondences in the images. Note the little piece of plastic at the top which creates a diffuse point.

In the following sections the steps involved in the calibration procedure will be described in more detail.

4.4.1 Robust Point Extraction

The first step in performing the calibration is to acquire a synchronized image set with point correspondences. To create the point correspondences a laser pointer is modified to act as a bright point light source by attaching a piece of plastic to it as shown in figure 4.4. During acquisition the light in the room is dimmed in order to simplify the task of segmenting the laser pointer in the images. Typically a sequence of about 1000 images is acquired in which the whole reconstruction area is covered with the laser pointer. The points are extracted with subpixel accuracy from the images by first subtracting the mean image and then fitting a Gaussian point spread function to points considered to be created by the laser pointer. Outliers due to misdetections are removed from the point correspondences using epipolar constraints and RANSAC.

4.4.2 Factorization

The next step is to compute the camera extrinsic and intrinsic parameters from the 2D point correspondences. This is done by a technique called factorization [ST96].

In the following we will assume that we have n calibration points and m cameras. Let $\mathbf{X}_j = [X_j \ Y_j \ Z_j \ 1]^\top$, $j = 1, \dots, n$ be the 3D coordinates of the calibration points and let P^i , $i = 1, \dots, m$ be the projection matrices of the cameras. The projection of a point \mathbf{X}_j into

camera i is given by

$$\lambda_j^i \begin{bmatrix} u_j^i \\ v_j^i \\ 1 \end{bmatrix} = \lambda_j^i \mathbf{u}_j^i = P^i \mathbf{X}_j, \quad \lambda_j^i \in \mathbb{R}^+ \quad (4.1)$$

where $\mathbf{u}_j^i = [u_j^i \ v_j^i \ 1]^\top$ are the image coordinates of the point and λ_j^i is a projective scale factor. We need to recover all P^i and λ_j^i in order to calibrate the system. To accomplish this, all measurements are assembled into the so-called scaled measurement matrix W_s :

$$W_s = \begin{bmatrix} \lambda_1^1 \begin{bmatrix} u_1^1 \\ v_1^1 \\ 1 \end{bmatrix} & \dots & \lambda_n^1 \begin{bmatrix} u_n^1 \\ v_n^1 \\ 1 \end{bmatrix} \\ \vdots & \ddots & \vdots \\ \lambda_1^m \begin{bmatrix} u_1^m \\ v_1^m \\ 1 \end{bmatrix} & \dots & \lambda_n^m \begin{bmatrix} u_n^m \\ v_n^m \\ 1 \end{bmatrix} \end{bmatrix} = \begin{bmatrix} P^1 \\ \vdots \\ P^m \end{bmatrix}_{3m \times 4} [\mathbf{X}_1 \ \dots \ \mathbf{X}_n]_{4 \times n} = PX \quad (4.2)$$

Assuming that enough measurements are given and the λ_j^i are known W_s has rank 4 and can be factorized into P and X . The unknown projective scales are determined using the method proposed in [ST96]. For the common case that not all points are visible in all cameras (e.g. due to occlusions) the measurement matrix contains some unknown entries which have to be filled before factorization. This is done by applying a rank 4 constraint as proposed in [MP02]. However, even if P and X are obtained, this still leaves an ambiguity as an arbitrary 4×4 matrix H can be inserted into the decomposition.

$$W_s = PX = PHH^{-1}X = \hat{P}\hat{X} \quad (4.3)$$

4.4.3 Euclidean Stratification

The calibration computed in the previous step is still projective. In order to obtain a Euclidean calibration the transformation H fixing the projective and affine properties of the calibration has to be recovered. This is done through self-calibration [HZ04]. Based on the assumption that the principal points are known and that each camera has zero skew and square pixels an equation system can be derived which is then solved yielding the stratifying transformation H . At least 3 cameras are needed in order to perform the stratification.

4.4.4 Radial Distortion Correction

Up to now only a linear camera model was used. However, this is problematic when using wide angle lenses, such as fish-eye lenses, which is common practice in multi-camera systems. Based on the initial linear calibration, the radial distortion parameters are estimated using the MATLAB Camera Calibration Toolbox [Bou]. Using the recovered radial distortion parameters the factorization is rerun using the corrected point coordinates. This can be repeated iteratively until the calibration error converges.



Figure 4.5: Calibration target for registering the camera coordinate system to the room coordinate system.

4.4.5 Registration

The camera coordinate system computed by the calibration procedure has to be registered to the room coordinate system in order to be able to recover metric properties from the reconstruction. As the camera coordinate system is already Euclidean we only need to determine the similarity transformation (i.e. scaling, rotation and translation) which registers the two coordinate systems. This is done by means of a rectangular registration target placed on the floor of which we know the exact corner coordinates in the room coordinate system (see figure 4.5). Images showing the target are acquired on at least two and optimally all cameras which see the target completely. The corner points of the target are manually selected in every image and their 3D coordinates are computed in the camera coordinate system by triangulation. In the following let the projection matrices be defined as above. Further let the 2D corner points of the registration target in camera i be \mathbf{x}_i^1 to \mathbf{x}_i^4 with $\mathbf{x}_i^k = [x_i^k \ y_i^k \ 1]^\top$. Let the corresponding 3D coordinates in the room coordinate system be \mathbf{X}_1^R to \mathbf{X}_4^R . The point coordinates in the camera coordinate system \mathbf{X}_1^C to \mathbf{X}_4^C can be obtained by linear triangulation. For every camera pair (i, j) with $j > i$ and one of the corner points (e.g. \mathbf{X}_k^C) a submatrix A_{ij}^k is constructed as

$$A_{ij}^k = \begin{bmatrix} x_i^k P_3^i - P_1^i \\ y_i^k P_3^i - P_2^i \\ x_j^k P_3^j - P_1^j \\ y_j^k P_3^j - P_2^j \end{bmatrix} \quad (4.4)$$

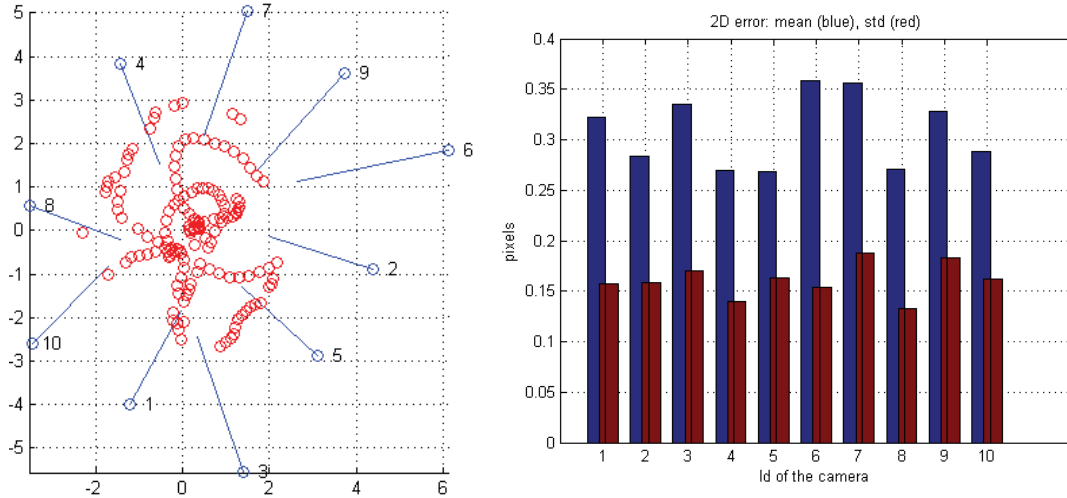


Figure 4.6: Calibration results for a setup with 10 cameras. The left figure shows the recovered camera positions and orientations (blue circles and lines) and the positions of the detected laser points (red circles) in the camera coordinate system. The right figure shows the mean (blue) and the standard deviation (red) of the reprojection error for every camera. The mean reprojection error over all cameras is 0.31 pixels and the standard deviation is 0.17 pixels.

where P_k^i is the k -th row of P^i . After stacking all A_{ij}^k in a matrix A^k the solution to the linear equation system $A^k \mathbf{X}_k^C = \mathbf{0}$ gives the solution for the 3D point \mathbf{X}_k^C . The next step is to find the similarity transform $\mathbf{T} = \begin{bmatrix} s\mathbf{R} & \mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix}$ relating \mathbf{X}^C to \mathbf{X}^R . It is computed using the method proposed in [WSV91]. Once \mathbf{T} has been obtained the projection matrices are updated as $P^i = P^i \mathbf{T}$.

4.4.6 Results

The reprojection error after calibration is in general around 0.3 pixels when also correcting for radial distortion. Figure 4.6 shows the reprojection error for each camera and the recovered camera positions for a setup with 10 cameras. The calibration can be done within less than 20 minutes by a person familiar with the system. This keeps the downtime of the system due to recalibration small. As the cameras are mounted on the ceiling recalibration has to be performed only very rarely when one of the cameras is actually moved, since accidentally moving the cameras is very unlikely.

4.5 Background Subtraction

Background subtraction is concerned with detecting foreground objects in the scene. This is necessary in order to obtain the silhouette images needed for the computation of the visual hull. To detect foreground objects a reference image of the background is taken when the

scene is empty. At runtime the current image is compared to the reference and pixels which changed with respect to the reference are marked as foreground.

A popular approach for background subtraction in multi-view studios is chroma keying. In this approach the background has a uniform color (e.g. blue). Therefore, segmentation can be achieved by mapping out the background color in the input images. The disadvantage of this approach is, that scene objects with the same color as the background will also be removed. In addition, chroma keying requires a very controlled environment including controlled lighting in order to ensure, that the background always has the same color. While it is possible to achieve this in a studio setting, this cannot be done in an interventional environment. We therefore cannot use chroma keying in our system and instead have to use background subtraction methods which can deal with complex backgrounds and changing lighting conditions.

There exist many algorithms for background subtraction in the literature. For the use in our system we compared three approaches, namely the Gaussian mixture model (GMM) approach presented in [ZvdH06], the codebook approach presented in [KCHD05] and a histogram-based approach presented in [FII⁺06]. The criteria according to which we evaluated the approaches are speed, accuracy and behavior in the face of illumination changes. All of these aspects, especially changing lighting conditions for instance due to turning on and off the surgical lamp, are important in an interventional setting. Some background subtraction methods, for instance the GMM approach, update the background model over time. This means that objects which enter the scene but stay static for a long time will become part of the background. This is undesirable in an interventional setting. For instance a tool table might be brought into the room at the beginning of the intervention and then stays at the same position for the duration of the procedure. However, we still want to keep it as part of the foreground, since for instance in a collision avoidance application it would be quite dangerous to consider the space where the table is standing as empty. We therefore disable background updating in the GMM approach.

In the following sections we will first describe each algorithm in more detail and then present a comparison with respect to our evaluation criteria.

4.5.1 Histogram Approach

The key idea behind this method is to adapt the reference background image to the current illumination conditions for every incoming frame. This allows to deal with quick illumination changes, since they are corrected for by the background adaption stage. The actual background subtraction is based on a blockwise decomposition of the image. For each block an adaptive threshold is used to mark it as either foreground or background. Blocks on object boundaries are further processed at a pixel level in order to extract detailed boundaries. In the following each processing step will be described in more detail.

Adaption of the background image

Let the background image be denoted by BG and the current image at time t by I_t . In order to adapt BG to the current illumination conditions the change in the color distribution has to be computed. Let $BG(x, y)$ be the RGB color value of the background image at position

(x, y) and similarly $I_t(x, y)$ the color value at the same position in the current image. Further let A'_t be the estimated background region at time t . A'_t is computed by first computing the absolute difference between I_t and I_{t-1} . The difference image is then thresholded to obtain the region with the smallest differences. A'_t is obtained as the logical AND between the thresholded region and the previous background image A_{t-1} . For the case when A'_t is the whole image or empty (for instance due to strong illumination changes), A'_t is taken to be a dilated version of A_{t-1} . A_0 is the entire image.

Next, a conversion table between the colors in the original background image BG and the colors in the current image I_t inside the region A'_t is generated by computing a histogram containing a bin for each possible color pairing. For every pixel (x, y) inside the region the bin $(I_t(x, y), BG(x, y))$ is incremented. This is done for all three color channels independently. The histogram is then used to find for each color in BG the corresponding color in I_t with the highest bin count. Intensities which do not have corresponding entries in the histogram are estimated by linear interpolation between the neighboring intensities. By substituting the corrected color for the original one in BG the adapted background image BG_t is obtained.

Background subtraction

In order to obtain the new segmentation mask A_t , the images BG_t and I_t have to be subtracted. This is done in a block-wise fashion by computing the sum of the absolute differences for each block (per color channel). If the sum of the differences over all color channels is below the sum of the color channel thresholds the block is considered as foreground. The threshold is changed dynamically per block and color channel and computed based on the intensities in the histograms obtained in the previous step, in order to accommodate different lighting conditions. For blocks on the boundary of the object each pixel is further classified as inlier or outlier by the same procedure in order to obtain an accurate outline of the object. For more details refer to [FII⁺06].

4.5.2 Gaussian Mixture Models

In this approach the appearance of each pixel is modeled as a mixture of Gaussians, each representing the distribution of a color. When a new color is observed for a pixel its probability of belonging to one of the background distributions is evaluated. If it does not match any of the background distributions it is considered as foreground. The initial background distributions are learned from a training set. In order to deal with lighting changes and changes in the background the background color distributions are not fixed but updated over time. This allows to incorporate objects which become part of the background during runtime (e.g. a chair placed and left in the room). However, as previously mentioned, this is undesirable in an interventional environment. We therefore disable background updating as described later on.

The color distribution of a pixel with a history of colors $X_T = \{X^t, \dots, X^{t-T}\}$ over the last T time steps is modeled using M Gaussians. The probability of observing the pixel value x is then given by

$$p(x|X_T) = \sum_{m=1}^M \omega_m N(x; \mu_m, \sigma_m^2 \mathbf{I}) \quad (4.5)$$

where M is the number of components in the mixture model, ω_m is the weight of each component and μ_m and σ_m are the mean and standard deviation of the m -th Gaussian respectively. \mathbf{I} is an appropriately sized identity matrix, meaning that the dependency between color channels is not modeled. The weights ω_m are non-negative and sum up to one.

When a new observation x_t is made the parameters are updated as follows:

$$\omega_m = \omega_m + \alpha(o_m^t - \omega_m) \quad (4.6)$$

$$\mu_m = \mu_m + o_m^t (\alpha / \omega_m) \delta_m \quad (4.7)$$

$$\sigma_m^2 = \sigma_m^2 + o_m^t (\alpha / \omega_m) (\delta_m^T \delta_m - \sigma_m^2) \quad (4.8)$$

where $\delta_m = x^t - \mu_m$ and $\alpha = 1/T$ is the update rate based on a time interval T . The term o_m^t is one if the new sample is close (in terms of the Mahalanobis distance) to component m and zero otherwise. If the new sample does not match any of the existing distributions a new distribution is created with $\omega_{M+1} = \alpha$, $\mu_{M+1} = x^t$ and $\sigma_{M+1} = \sigma_0$ where σ_0 is an initial variance. If the maximum number of components is reached the one with the smallest weight is discarded.

Since every new observation enters the distribution, we need to define a number B of components representing the background model. The remaining components are not considered part of the background model. In practice the components are sorted by descending weights and the first B components are taken to represent the background model. If a new object stays stationary for a long enough time its weight will increase relative to the other weights and it will thus become part of the background model.

The method as described up to now is the classical GMM approach similar to the one presented in [SG98] and uses a fixed number of components M . We use the approach by Zivkovic *et al.* [ZvdH06] who also estimate the number of components in the GMM automatically. This leads to an improved performance. Since we want to disable background updates we set α to zero after the initial training phase. For more details on automatically estimating the number of components refer to [ZvdH06].

4.5.3 Codebook Approach

In the codebook approach every pixel is modeled by a codebook consisting of a set of codewords. Each codeword models a certain set of allowed color and brightness variations for a pixel. During the training phase codewords are added to the codebook based on the observed pixel colors. If a new training sample does not match any existing codeword a new codeword is added to the codebook. Each pixel can have a different number of codewords, depending on the complexity of the background model. Each codeword c_i consists of a RGB color tuple $\mathbf{v}_i = (R_i, G_i, B_i)$ and a 6-tuple $\mathbf{aux}_i = (\check{I}_i, \hat{I}_i, f_i, \lambda_i, p_i, q_i)$. \check{I} and \hat{I} are the minimum and maximum brightness, respectively, of all pixels assigned to the codeword, f is the

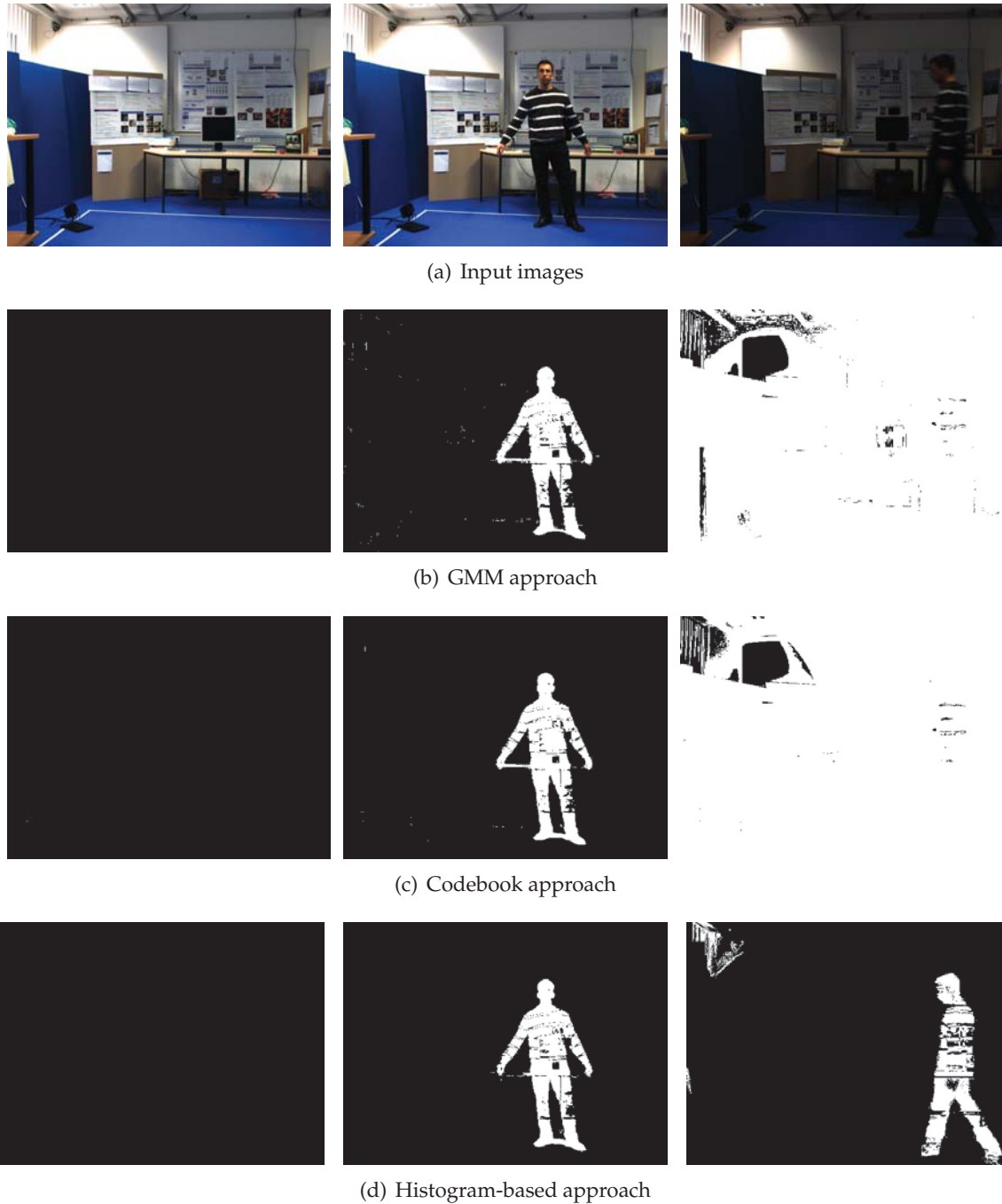


Figure 4.7: Comparison of the background subtraction results under strong illumination changes. Only the histogram based method gives reasonable results.

frequency at which the codeword has occurred, λ is the longest interval during the training phase in which the codeword has not reoccurred, and p and q are the first and last time step that the codeword has occurred. During runtime the observed pixel colors are compared to each codeword in terms of color distortion and brightness variation, i.e. the new sample has to fall within the min and max bounds of the codeword intensities. If a match is found the

pixel is marked as background, otherwise it is marked as foreground. More details on this approach can be found in [KCHD05].

4.5.4 Discussion and Results

In order to select the best algorithm for use in our real-time system, we compared the three approaches described above in terms of performance and accuracy on a test sequence recorded in our multi-camera studio. The sequence shows a person walking inside the room.

The runtime performance is 9 ms for the codebook and the histogram method and 32 ms for the GMM approach (using the author's reference implementation [ZvdH06]) on images of resolution 512×384 . The tests were performed on one of the slave PCs. This already precludes the GMM method from use in our system, since we need a performance of at least 15 ms in order to also perform the reconstruction and still achieve real-time performance for the whole pipeline.

In terms of segmentation quality there is not a big difference between the three approaches. In general the GMM gives a slightly better result. However, it is too slow for our purposes. The histogram-based method and the codebook method perform similarly with the codebook method exhibiting more noise in general.

We also tested how the algorithms cope with (quick) illumination changes. The disadvantage of GMMs is that they cannot cope with quick illumination changes since the background color distributions cannot adapt to the change in intensities quickly enough. This would require a very high update rate which would let any object fade into the background almost immediately. In addition, we do not want to update the background model in order to avoid losing objects to the background. The codebook approach has the same problem, since the codebook built during the training phase is not valid anymore when the illumination changes drastically, even though color normalization is applied to the input colors. Therefore both methods are not suited for scenes with large illumination changes. The histogram-based method on the other hand deals quite well with illumination changes since the back-

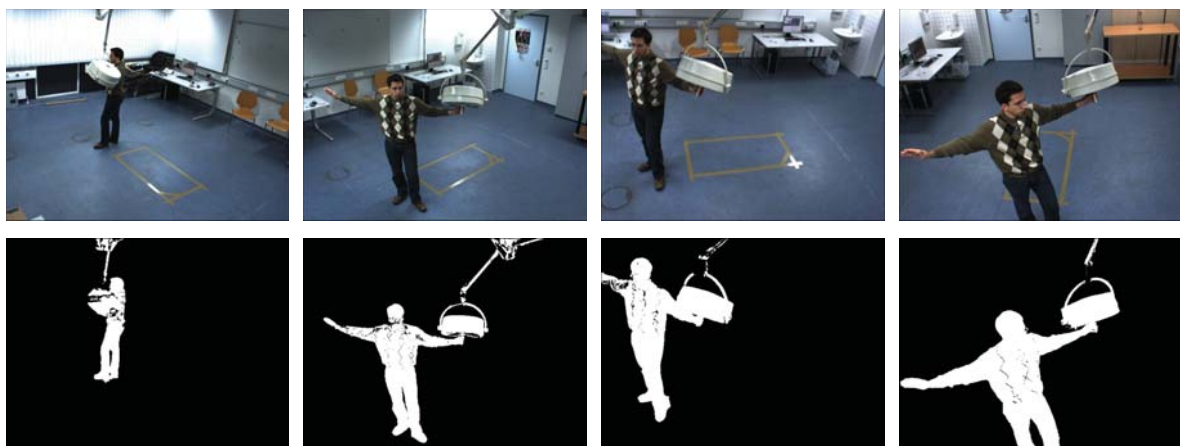


Figure 4.8: Background subtraction results on some images from our multi-view studio using the method by Fukui *et al.* [FII⁺06].

ground is always adapted to the current lighting conditions. In addition, the codebook and the GMM method require a training sequence, while the histogram-based method only requires a single frame. This combined with the fact that this method is the fastest and gives good segmentation results, led us to adopt it in our system. Figure 4.7 shows the results of the three methods on a sequence with strong illumination changes. Only the histogram-based method also works under strong illumination changes. The reason for the stripes in the segmentation results is that the user is wearing a striped pullover and the color of the white stripes is very similar to the background color. In a real interventional setting such problems would be avoided by having the surgeons wear distinctly colored clothes, such as green or blue surgical robes. Figure 4.8 shows more segmentation results obtained using the histogram-based method. Notice that even such fine details as the ceiling support of the operating light are correctly segmented. As with any segmentation method there are small regions which are incorrectly marked as background due to the fact that they have the same color as the foreground. In our reconstruction approach such small segmentation errors do not play a significant role, since the resolution of the voxels is larger than the extent of the typical holes in the segmentation. In addition, the holes can be partially removed using a morphological closing operator.

4.5.5 Handling Static Occluders

One problem which has to be addressed during background subtraction regardless of the method used is the presence of static occluders in the scene. Static occluders are objects inside the working volume which cannot be removed, such as tables mounted to the floor. Hence static occluders are also present in the background images. This is particularly important in interventional environments, since here the operating table is usually fixed to the floor and cannot be removed. The assumption during background subtraction, however, is that all foreground objects are located in front of the background. This is not the case in the presence of an occluder because a foreground object could move behind the occluder and effectively disappear from the silhouette image as illustrated in figure 4.9. This will result in the partial or complete removal of the object from the reconstruction. To overcome this problem, the areas in the silhouette images corresponding to the static occluder have to be disregarded during the visual hull computation. We achieve this goal by building a 3D representation of the occluder and projecting it into the cameras or by manually segmenting the occluder in the reference images. This gives us a mask for every camera in which the static occluder is marked as foreground. This mask is then added (logical OR) to the silhouette images computed during runtime. A similar approach was suggested in [GSFP06].

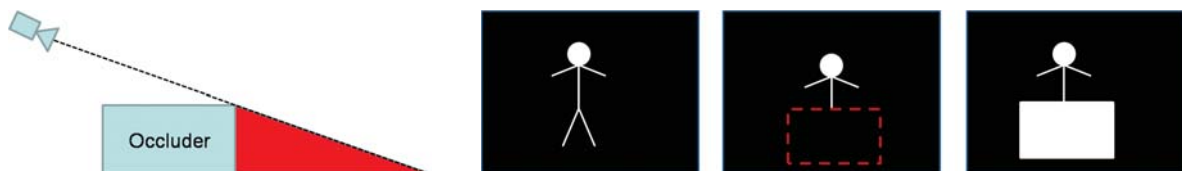


Figure 4.9: Static occluders. Left: Any object in the red area will not be visible in the silhouette image. Hence it will not appear in the reconstruction. Right: Effect of the occlusions mask. From left to right: Without occluder, with occluder, with occlusion mask.

4.6 Visual Hull Computation

In our system the visual hull is computed using the octree-based volumetric approach described in section 3.2.1 at a voxel resolution of 128^3 . In a first attempt we implemented the reconstruction on the CPU exploiting the multi-core architecture. However, although heavily optimized the implementation did not achieve real-time performance at the target resolution of 128^3 voxels. We therefore developed a GPU-based reconstruction method which achieves real-time performance and which will be described in detail in chapter 5. In this section we will explain the distribution of the computation of the visual hull over multiple PCs.

In the classical visual hull computation only the space in the intersection of the viewing cones of all cameras is reconstructed. Since this overlapping volume in our case is smaller than the desired working volume we have to apply measures to make certain that we can have a bigger reconstruction volume without needing to use extreme wide angle lenses which would allow every camera to observe the whole reconstruction volume. Using such lenses would not only decrease the spatial resolution in the images but also lead to bigger radial distortion. The way we achieve the goal of having a larger reconstruction volume than the overlapping volume is tightly linked to the distribution of the cameras to different PCs in the system. As mentioned previously each slave PC computes a partial reconstruction of the scene using its locally attached cameras. Instead of applying the standard approach of only reconstructing the overlapping volume of the local cameras, we reconstruct all objects in the union of the viewing cones of the local cameras. In practice, this means that we do not consider voxels which project outside of the silhouette images to be empty. Instead, we only consider the contributions from the cameras in which the voxels project into the silhouette image. This naturally introduces a certain amount of ghost volumes into the partial local reconstruction. However, these partial reconstructions, which every slave PC computes independently, are then sent to the master PC, which combines them. This is done by performing a logical AND operation on the volumes. The effect is that only voxels which are occupied in all partial volumes are kept in the final reconstruction. In essence this means that an object has to be seen in at least one camera of each of the four slave PCs in order to appear in the final reconstruction. This removes the ghosting artifacts and increases the size of the working volume by removing the strict constraint that an object has to be seen by all cameras in order to be part of the reconstruction.

In order to obtain nicer results during visualization as described in the next section we do not use binary voxel volumes. Instead we compute a partial voxel occupancy, by considering for each occupied voxel the number of pixels inside the silhouette over the total number of pixels in the voxel projection. This way voxels on object boundaries receive only a partial occupancy which leads to smoother results during the visualization.

4.7 Visualization

The output of the reconstruction process is a volumetric representation of the scene. Although it could be rendered using volume rendering [Lev88] it is easier to first obtain a mesh representation of the scene which can then also be used for other editing tasks. The conversion from the volumetric to the mesh representation is performed using the Marching Cubes algorithm [LC87]. Marching Cubes extracts the isosurface of the volume by labeling

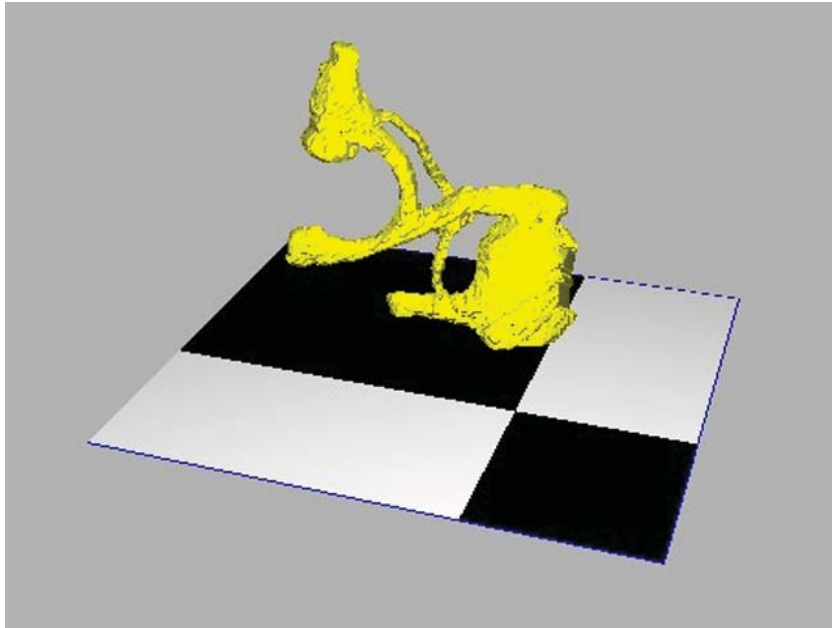


Figure 4.10: Reconstruction and visualization of a C-arm obtained using our system.

each voxel as being either inside or outside the isosurface based on a user-defined threshold (128 in our implementation as voxels have an occupancy value between 0 and 255). Subsequently a surface consisting of several triangles is placed in each imaginary cube spanned by 8 neighboring voxels which crosses the inside-outside boundary (i.e. some of the voxels are labeled as inside and some as outside). In our system we use the CUDA implementation provided with the NVIDIA CUDA SDK. A CPU implementation did not work in real-time on volumes of size 128^3 . Figure 4.10 shows a reconstruction of a C-arm obtained and visualized using our system.

4.8 Conclusion

We presented the design of our system and highlighted the design choices taken due to the intended use of our system in an interventional environment. Our system is based on a distributed computer system and performs the reconstruction in real-time on commodity hardware by offloading the computationally expensive visual hull computation onto the GPU. The cameras are placed on the ceiling in order to avoid accidentally moving them and to keep them out of the working space of the medical staff. To ensure a synchronous image acquisition the cameras are externally triggered. The calibration procedure is easy to learn and only requires minimal system downtime. Since we cannot use chroma keying in an interventional environment, we use a robust background subtraction algorithm for segmenting the foreground objects in the scene, which can deal with complex backgrounds and quick illumination changes. In addition, it does not incorporate static foreground objects into the background model which is also of importance in an interventional setting. Moreover, we include a scheme for handling static occluders inside the scene, such as the surgical table. Finally, we propose a method for increasing the size of the reconstruction volume extending

it beyond the overlapping volume of all cameras.

Chapter 5

GPU-based Visual Hull Computation

As mentioned in the previous chapter we perform the visual hull computation on the GPU. This enables us to keep the system cost low compared to other real-time 3D reconstruction systems which are using very powerful and expensive hardware to achieve real-time frame rates [FMBR04, AFM⁺06, AMR⁺07].

5.1 Introduction

Volumetric visual hull computation is a perfect example of a highly parallel algorithm. Therefore, it is well suited to be implemented on the GPU. However, there are still different design choices for performing the computations. We evaluate two different approaches. In the first approach we precompute the bounding boxes of the voxel projections in each image and store them in a lookup table which is used during the visual hull computation. In the second approach we downsample the images so that a voxel approximately projects into one pixel. This allows us to only project the voxel center point to obtain the corresponding pixel in the image.

The first approach is most useful when the camera configuration is static which is the case for a 3D reconstruction system. If, on the other hand, one just wants to compute the visual hull once for a given camera configuration, the overhead in terms of memory and time that the precomputation requires, makes this approach less attractive. The second approach does not perform any precomputations and is therefore suited for both cases. We implemented both an octree version and a non-octree version of each approach. Using an octree representation of space speeds up the computations, but constrains the size of the reconstruction volume to cubes with power of two side lengths. The non-octree version also allows to compute the visual hull on non-cubic volumes but requires a longer computation time.

5.2 Related Work

In contrast to our proposed method, previous GPU-based methods for visual hull computation make use of hardware texturing. Hasenfratz *et al.* [HLGB03, HLS04] use the GPU to

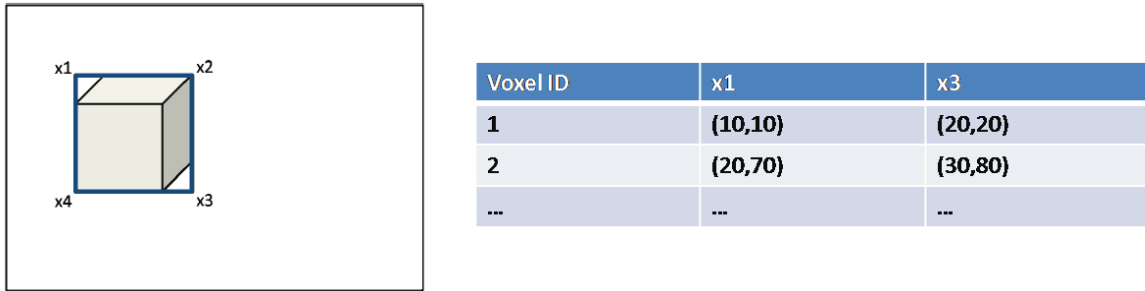


Figure 5.1: Bounding boxes and lookup table layout used in the precomputation-based algorithm.

map the silhouette images as textures onto slices through the reconstruction volume and intersect them using image blending. Hornung *et al.* [HK06c] suggest to project every voxel into the images using a fragment shader and texture mip-mapping. We propose a method making use of CUDA [Nvi] to perform the reconstruction by using kernels which compute the projection of every voxel and accumulate the occupancy information. We present two implementations of this approach and compare them in terms of performance.

5.3 Precomputation-based Algorithm

In this algorithm, which we shall call GPU1, we first compute the bounding boxes of the voxel projections in each image and store them in a lookup table by specifying the upper left and lower right corner (see figure 5.1). This is done in an offline step. The memory required by the lookup table is proportional to the number of images and the voxel resolution used. For the octree version we compute a lookup table for every octree level. The lookup table is copied onto the GPU during an initialization step. For every new set of silhouette images, we first compute the corresponding integral images. The integral image S of an image I is computed as

$$S(x, y) = \sum_{x' \leq x} \sum_{y' \leq y} I(x', y') \quad (5.1)$$

The integral image contains at every pixel the sum over all pixel values in the rectangle spanned by the image origin and the considered pixel. This allows us to efficiently evaluate the number of occupied pixels given a bounding box in the image by using only four memory accesses instead of looking at every pixel in the bounding box. The number of occupied pixels A in the rectangular region spanned by the corner points $(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4)$, with \mathbf{x}_1 the upper left corner and the other vertices listed in clockwise order, is computed as

$$A = S(\mathbf{x}_3) - S(\mathbf{x}_2) - S(\mathbf{x}_4) + S(\mathbf{x}_1) \quad (5.2)$$

This is illustrated in figure 5.2. The time required to compute the integral images is significantly shorter than the overhead of looking at every pixel in the bounding box. All the before mentioned steps are performed on the CPU. To speed them up, the computations are parallelized.

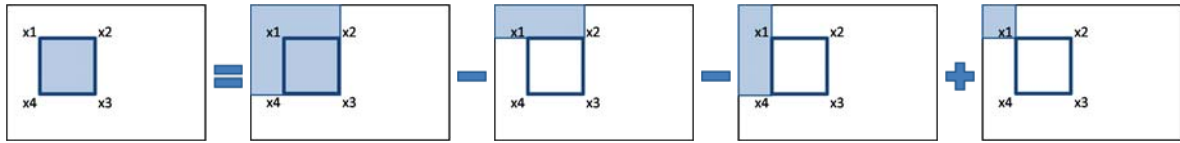


Figure 5.2: Illustration of the computation of voxel occupancy using integral images.

The integral images are then copied onto the GPU using page-locked memory to increase the transfer rates. In the non-octree version, a kernel is executed for each voxel. The kernel accesses the lookup table to find the corner points of the voxel bounding box. The corner coordinates are used as an offset into the integral images to compute the number of occupied pixels. The sum of occupied pixels and the sum of total pixels is accumulated over all images and their quotient is assigned as the occupancy to the voxel. This gives smoother results than only assigning one and zero for the voxel occupancy. If a bounding box is found to be empty in one image the remaining images are not checked and the voxel is assigned a value of zero.

In the octree version of the algorithm (GPU1_OT) a similar procedure is executed for each octree level. We maintain a list of active octree cells (we talk of cells instead of voxels since an octree cell consists of multiple voxels except at the highest octree level) containing the IDs of the cells which have to be checked at the current octree level. We start at level 4 because the overhead of using the octree approach is higher than computing the result directly when starting at a lower level. We therefore initialize the active cell list with the ID of every cell at level 4 (4096 cells). We only start as many kernels as there are active octree cells and use the ID of the kernel as an index into the active octree cell list to get the cell ID. The computation of the occupancy is performed the same way as in the non-octree version, except that a different lookup table is used for every level. Depending on the result of the occupancy test, we set an entry in a cell list. If the cell is either totally occupied or totally empty, we set the entry to zero, indicating that we are not considering the cell at the next level and then fill in the corresponding voxels in the volume accordingly. In case of partial occupancy we mark the eight sub-cells corresponding to this cell on the next octree level in the cell list. After the kernel has finished executing we use the CUDPP library [SHZO07] and a small additional kernel to compact the cell list to only contain the IDs of the active cells, which are then used as the input to the kernel on the next level. This process is illustrated in figure 5.3. The resulting voxel volume is copied off the GPU using page-locked memory.

5.4 Direct Algorithm

The direct algorithm, which we shall call GPU2, does not perform any precomputations. Instead the silhouette images are downsampled so that every voxel approximately projects into a single pixel. This allows us to only compute the projection of the voxel center point and to perform a single memory access in the silhouette images. In the non-octree version we downsample the image to match the voxel resolution using a Gaussian smoothing followed by a downscaling. The kernel is then executed for every voxel. It derives the 3D position of its corresponding voxel from its ID and projects the voxel center point into the image. The occupancy value of the voxel is computed as the mean over the values at all voxel projec-

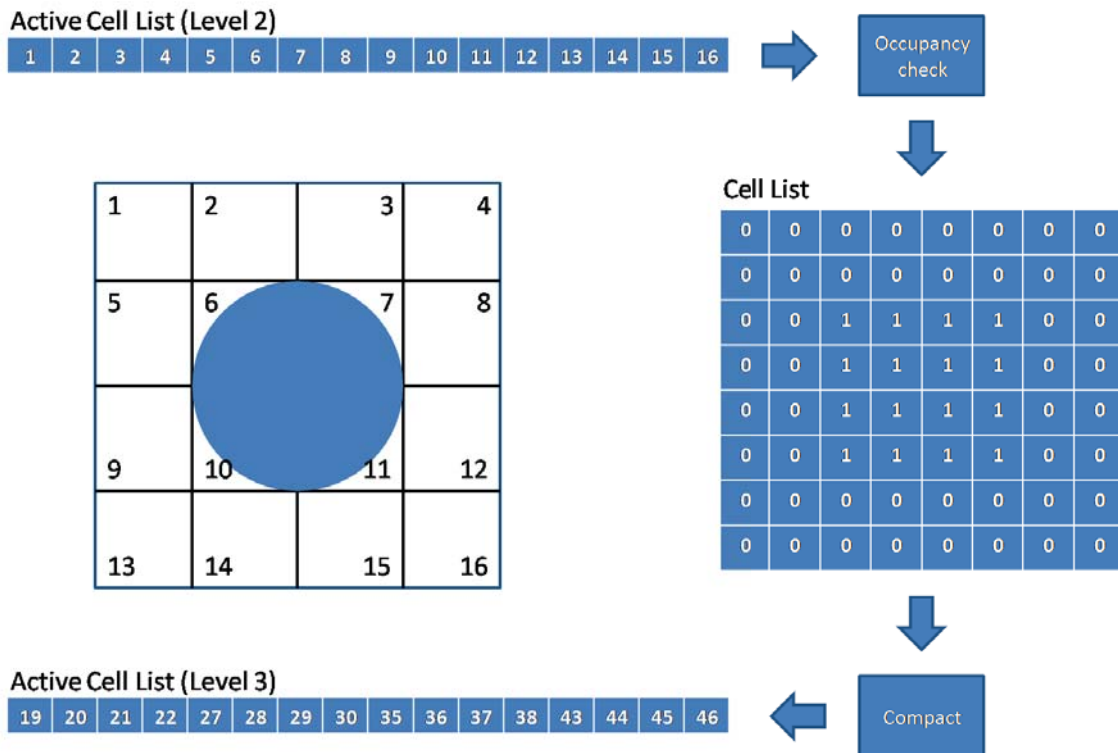


Figure 5.3: Illustration of the computation scheme for the octree versions of the proposed algorithms. The 2D scene shown on the left is represented in the Active Cell List (ACL). For every cell in the ACL a kernel checks the occupancy and sets the entries of the corresponding subcells on the next level to 0 if the cell is either occupied or empty and to 1 otherwise. The list is then compacted by an additional kernel and the corresponding cell IDs are added to the ACL for the next level.

tions. If a value is zero in one image the remaining images are not checked anymore and the voxel occupancy is set to zero.

In the octree version (GPU2_OT) the silhouette images are downsampled in a Gaussian pyramid, so that one image is obtained for every octree level. The kernel is then executed for every level using only the active octree cells which are determined in the same way as in the precomputation-based algorithm. The resulting voxel volume is copied off the GPU using page-locked memory.

5.5 Results

To evaluate the performance of the visual hull computation algorithms, we tested them on three data sets at different voxel resolutions.

The first data set is used to assess the performance for a scene typically encountered in a real-time 3D reconstruction system. This data set was captured with our system and shows a person walking through the room. We first computed the visual hull using all 16 images

Method	64^3	128^3	256^3
GPU1	46.40 ms	64.44 ms	x
GPU2	18.60 ms	113.88 ms	870.06 ms
GPU1_OT	45.46 ms	51.94 ms	x
GPU2_OT	15.24 ms	25.91 ms	73.53 ms

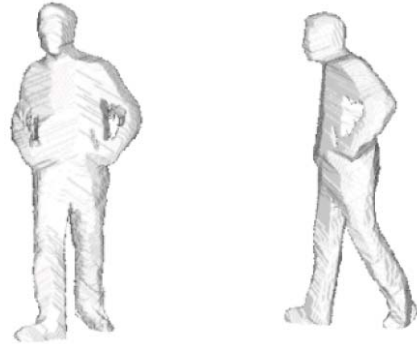


Figure 5.4: Runtimes on the *person* dataset consisting of 16 images (1024×768). The precomputation-based algorithms cannot run at level 256^3 due to the size of the lookup table.

(resolution 1024×768). The runtimes are shown in figure 5.4. It can be seen that even at a resolution of 128^3 we still achieve real-time performance using the direct algorithm (GPU2_OT) which consistently provides the best reconstruction times at all resolutions. Also note that the precomputation-based algorithms (GPU1, GPU1_OT) fail to run at a resolution of 256^3 because the lookup table size exceeds the available storage on the GPU (768MB).

To test the scalability of the algorithms, we ran them on the same data set using different numbers of images. Figure 5.5 shows the resulting runtimes, plotted over the number of input images. The first plot shows the results of the non-octree algorithms. It can be seen that the precomputation-based algorithm (GPU1) has a complexity linear in the number of images, while the direct algorithm (GPU2) has a complexity which first increases steeply with the number of images and then only grows very slowly. The highly linear behavior of the precomputation-based method can be explained by the computation of the integral images which is linear in the number of images. In the direct algorithm this step is not necessary, so that it only grows much slower (the down-sampling of the images is very fast).

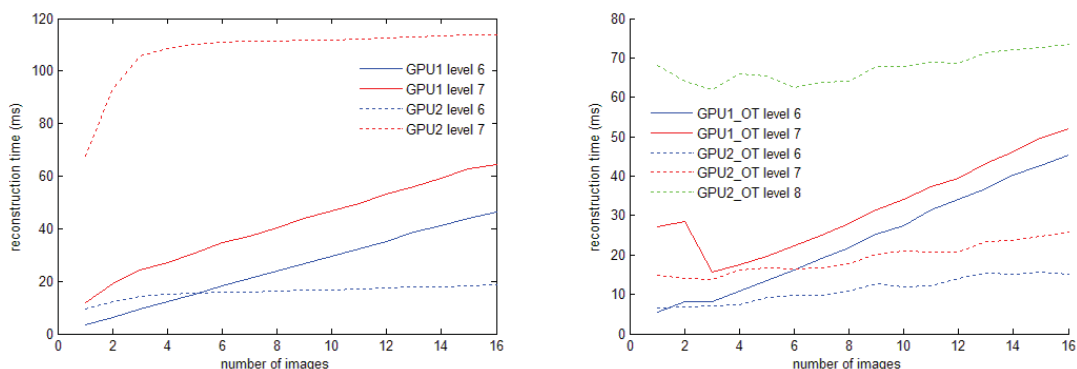


Figure 5.5: Runtimes on the *person* data set over number of images used. The image resolution used was 1024×768 .

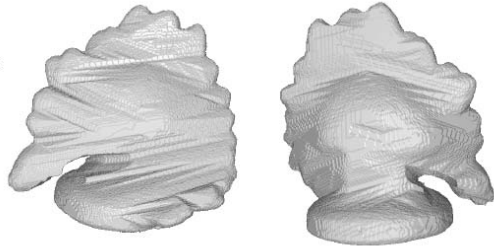
Method	64^3	128^3	256^3		
GPU1	95.26 ms	x	x		
GPU2	52.76 ms	417.79 ms	3033.89 ms		
GPU1_OT	63.95 ms	x	x		
GPU2_OT	39.94 ms	99.89 ms	296.71 ms		

Figure 5.6: Runtimes on the *dinoRing* dataset consisting of 48 images (640×480). The precomputation-based algorithms can only run at 64^3 due to the size of the lookup table.

It should also be noted that the probability of finding a totally empty voxel projection early is increasing with a higher number of images, so that it is not unexpected to see only minor changes in the runtime as the number of images increases. An interesting observation is that the non-octree algorithms can provide real-time results. In particular the GPU1 method can be used with four images and a resolution of 128^3 to obtain reconstruction times of about 30 ms. Since no octrees are used, this means that it is also possible to reconstruct non-cubic volumes with the same performance as long as they have a similar number of voxels. In this case it is advisable to use the precomputation-based algorithm (GPU1) since it is faster at higher resolutions than the direct algorithm when using a reasonable number of images.

The second plot shows the performance of the octree-based algorithms. We can observe the same behavior as for the non-octree algorithms, albeit at a higher performance level. The direct algorithm (GPU2_OT) exhibits a slightly higher growth rate using the octree than without using it. This is due to the computation of the image pyramid in the octree version. The fact that for a low number of images the runtimes of the algorithms are sometimes decreasing when using more images can be explained by the fact, that by using more images the visual hull will be more constrained and hence smaller. This allows the octree methods to stop checking some octree cells at an early octree level, which increases the performance. After a certain number of images this effect disappears, because adding new images only changes the visual hull slightly. When using the octree version of the algorithms the direct method should be used (GPU2_OT), because it shows a consistently better performance than the precomputation-based method.

The second and third data set are taken from the Middlebury multi-view evaluation [SCD⁺06]. We use them to test how the algorithms perform with large numbers of images and complex scenes. The results obtained on the *dinoRing* data set consisting of 48 images (resolution 640×480) are shown in figure 5.6. The memory requirements of the precomputation-based algorithms were so high that they could only run at the lowest resolution. The direct octree algorithm on the other hand computes the visual hull quite fast, at a resolution of 64^3 even at 25 fps. The results obtained on the *templeRing* data set consisting of 47 images (resolution 640×480) are shown in figure 5.7. This data set was used to test how the performance changes with increasing scene complexity. While the non-octree methods show a similar performance as on the *dinoRing* data set, the octree methods have an increased runtime. This is expected, because the probability of being able to terminate the computations on a low octree level is lower, when the scene is more complex. It is clear from these experiments that

Method	64^3	128^3	256^3
GPU1	97.42 ms	x	x
GPU2	51.61 ms	372.95 ms	3022.10 ms
GPU1_OT	62.83 ms	x	x
GPU2_OT	56.48 ms	170.91 ms	516.80 ms

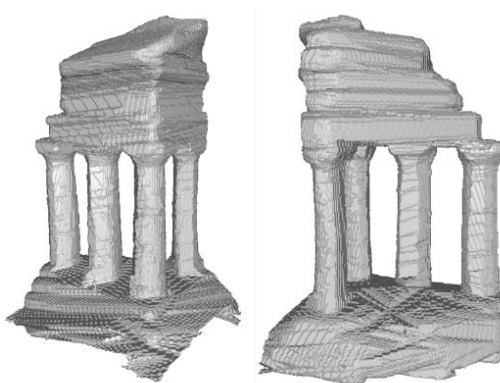


Figure 5.7: Runtimes on the *templeRing* dataset consisting of 47 images (640×480). The pre-computation based algorithms can only run at 64^3 due to the size of the lookup table.

for large data sets and high resolutions the direct octree method (GPU2_OT) provides the best results.

One also has to consider the precomputation times when using the precomputation-based algorithms. They usually lie in the range of a few seconds for moderately sized data sets. This means that if one wants to only compute the visual hull once with a given camera configuration, it is better to use the direct method. In a real-time system where the camera configuration does not change this is not an issue.

5.6 Conclusion

We presented two GPU-based approaches for efficient visual hull computation. One approach uses a precomputed lookup table, while the other directly computes the voxel projections. Both methods were implemented in an octree and a non-octree version on the GPU using CUDA. We compared the performance of both methods on different data sets. Our results indicate that for small non-cubic resolutions the precomputation-based non-octree algorithm (GPU1) should be used, while for high resolution volumes and large numbers of images the direct octree algorithm (GPU2_OT) performs better. We therefore use the direct octree algorithm in our real-time system.

Chapter 6

Incremental Visual Hull Computation

While the GPU-based method described in the previous chapter is very fast, it can be argued that it performs more work than necessary. This is because it does not take temporal consistency in the scene into account. Instead, it highly parallelizes the computation of the visual hull in order to obtain real-time performance.

In this chapter we propose an incremental visual hull reconstruction approach taking temporal consistency into account. By exploiting the fact that there usually occur only small changes between consecutive frames, we can reduce the runtime and calculation complexity. This allows us to perform real-time reconstruction on a standard processor without having to parallelize the computations.

6.1 Related Work

Although there are methods for performing time consistent reconstructions in the context of multi-view reconstruction [SH07b, VZBH08], there is little work dealing with temporal visual hull reconstruction. Cheung *et al.* [CBK05a, CBK05b] consider the problem of aligning multiple silhouette images of a non-rigidly moving object over time in order to improve the quality of the visual hull. However, this is quite different from our approach. More recently Aganj *et al.* [APSK07] proposed a method for spatio-temporally consistent visual hull reconstruction by performing a 4D reconstruction over the whole sequence using Delaunay meshing. The drawback of their method is that it only works offline, requiring the whole sequence to be available for performing the reconstruction. This precludes a real-time on-line use of their method. In addition the runtime of their method is far from real-time due to its high computational complexity. In [Keh05] another method taking temporal consistency into account is presented. At every pixel a list of voxels which projects into this pixel is stored. At runtime the voxels which have to be checked for changes are therefore immediately available given the changed pixels. However, the memory requirements for storing the voxel list for every pixel grow rapidly with the voxel resolution. In addition for large image resolutions the memory requirements grow even further. We avoid this overhead by sampling the voxels along the ray going through a changed pixel at run time.

Our approach does not make any assumptions about the number or the shape of the objects in the scene and does not use any additional information other than that provided by the

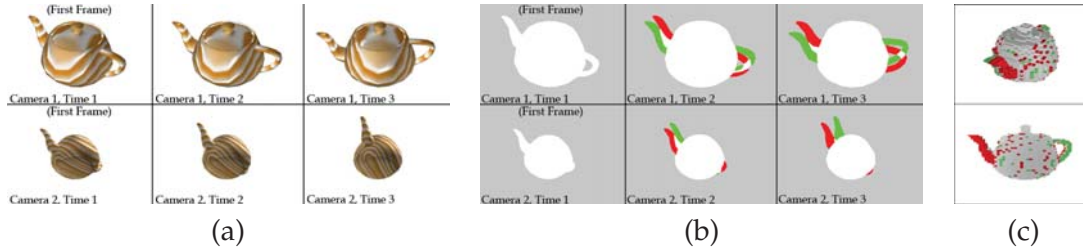


Figure 6.1: The rotating teapot model: (a) Rendered views from two different cameras in three successive time steps; (b) Corresponding silhouette and difference images with highlighted changed areas. Red pixels have been removed from the foreground since the previous time step and green pixels have been newly added in the current time step; (c) Colored recently removed and added voxels from two novel views.

silhouette images. Our results are identical to the results one would obtain when using a normal reconstruction method for each frame but the computation time is considerably reduced leading to real-time performance on off-the-shelf hardware.

6.2 Incremental Reconstruction Approach

In dynamic scenes such as those captured by multi-camera systems the changes between frames are limited by the speed at which the objects in the scene move or deform. Hence, the number of voxels that change in the reconstruction does not alter dramatically. Therefore, it is inefficient to reconstruct each frame independently. Theoretically the only voxels that have to be updated to transform the previous reconstruction into the current one, are the ones with changed occupancy. In the following we will call this subset of voxels the *Changed Set*. In practice we normally cannot compute the *Changed Set* directly. Therefore, we approximate it by a superset, which we call the *Search Space*.

We propose a *Search Space* whose size is close to the *Changed Set*. The main idea is to consider the changes in the silhouette images between the previous and the current frame, since they are caused by the change in voxel occupancy. Using ray casting [AW87] we efficiently find these changed voxels. Since most of the search process is done in two dimensional image space, the required amount of operations is reduced noticeably compared to performing a full reconstruction. Figure 6.1 illustrates our method for three frames of a rotating teapot. Figure 6.1(b) shows the change in the silhouettes (green being newly added foreground pixels and red being the removed foreground pixels), while figure 6.1(c) shows the added (green) and removed (red) voxels between the frames. It is clear that the number of changed voxels is small compared to the total number of voxels.

The first frame of a sequence does not have a corresponding previous reconstruction which can be updated. However, by assuming that the previous reconstruction was empty and consequently labeling every foreground pixel as added pixel, the proposed method can also be used. However, this is less efficient than using a standard visual hull computation method such as the GPU-based method proposed in the previous chapter. The method proposed here shows its true strength when only subparts of the scene are changed. This is commonly the case between two frames of an image sequence.

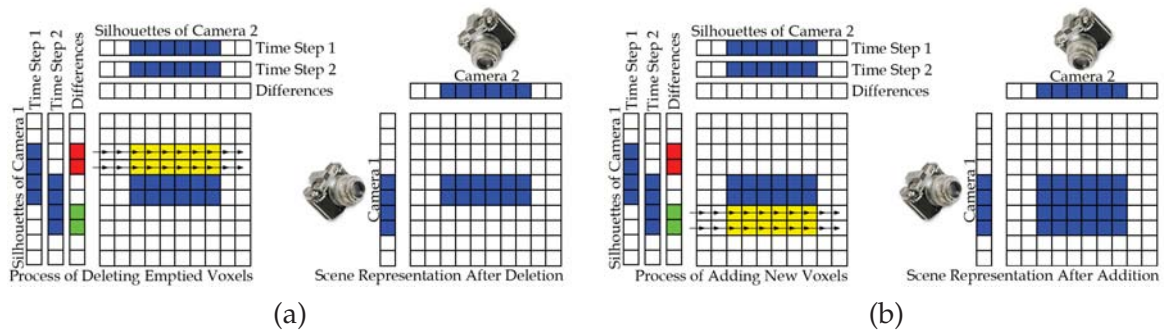


Figure 6.2: The voxel traversing process: (a) Rays from removed pixels. Yellow voxels are removed; (b) Rays from new pixels. Yellow voxels are added.

In order to update the reconstruction, the difference in the silhouette images between the previous and the current time instant have to be computed. This determines which pixels were added or removed from the foreground with respect to the previous time step. We then find the corresponding voxels by traversing the rays starting from the camera center and passing through the changed pixels. However, there usually is not a one-to-one relation between voxels and pixels due to different resolutions and the effects of perspective projection. Normally the voxel resolution is smaller than the pixel resolution; hence the projection of most voxels extends to more than one pixel. In this situation the occupancy of each voxel may be checked multiple times, when using ray casting. Despite that, the redundancy can be reduced considerably by using an appropriate sub-sampling depending on the voxel and image resolution. In other words only a subset of the pixels is used to update the changed voxels. This is achieved by sub-sampling the silhouette images. This is similar in principle to the direct GPU method discussed in the last chapter, where the images were also downsampled in order to match the voxel and the pixel resolution.

The information about which pixels were added and removed is used to find the set of removed and added voxels. While the set of voxels corresponding to the removed pixels only has to be set to empty, the set of voxels corresponding to the added pixels has to be checked for occupancy. This is done using the original silhouette images. Since we only change the traversing process in voxel space, the reconstructed visual hull produced by our approach is equivalent to the one produced by other visual hull reconstruction methods. The implementation of the deletion and addition phase depends on the algorithm used. We have developed two different algorithms based on this approach, which we will discuss in the following two sections.

6.3 Ray Casting

Here, we explain our first algorithm for incremental visual hull reconstruction in dynamic scenes, which uses ray casting to update the visual hull. For each changed pixel in the difference images, we create a ray from the optical center of the camera passing through the pixel and traverse all voxels, which the ray passes, using ray casting [AW87]. The size of the search space in this implementation is equal to the number of voxels projecting into the changed regions of the difference images.

All changes after the first frame can be reconstructed by an update phase using this approach. Even in the first frame, the whole visual hull can be computed using blank silhouettes as the images from the previous time step. However, using this approach for reconstructing the whole scene is inefficient, since the search space would include every voxel. We therefore use our GPU-based method for the first frame.

To perform the update of the reconstruction two phases are required. In a first phase newly empty voxels are deleted from the reconstruction, while in a second phase newly occupied voxels are added.

In the deletion phase, the occupancy of all voxels lying on rays passing through removed pixels (red in the images such as figure 6.1) is set to empty. This is because these areas are the projection of removed voxels since the previous time step. This is illustrated in figure 6.2(a). Algorithm 2 shows the pseudo code for the deletion process.

Algorithm 2 Pseudo code for the deletion phase in the ray casting method

```

1: for each image  $i$  in the sub-sampled difference image list do
2:   for each removed pixel  $p$  in image  $i$  do
3:     Create ray  $r$  from optical center passing through  $p$ 
4:     Set  $v$  to the first voxel along  $r$  in the reconstruction volume
5:     repeat
6:       Set occupancy state of  $v$  to empty
7:     until set  $v$  to the next occupied voxel along  $r$  is unsuccessful
8:   end for
9: end for

```

To complete the updating process, after removing deleted voxels, newly occupied voxels are added in the addition phase. Figure 6.2(b) and the pseudo code given in algorithm 3 show the process of creating newly occupied voxels in the reconstruction results of the previous phase. The addition phase of this method is similar to the deletion phase. However, before creating new voxels, their occupancy has to be checked. A voxel is marked as occupied if its projection on all silhouette images maps to a foreground pixel.

Since ray casting is completely implementable on a GPU [SKB⁺06, MSE06], this algorithm also can be implemented on modern graphical processors.

Algorithm 3 Pseudo code for the addition phase in the ray casting method

```

1: for each image  $i$  in the sub-sampled difference image list do
2:   for each added pixel  $p$  in image  $i$  do
3:     Create ray  $r$  from the optical center passing through  $p$ 
4:     Set  $v$  to the first voxel along  $r$  in the reconstruction volume
5:     repeat
6:       if  $v$  is not occupied then
7:         Check voxel occupancy at  $v$  and mark  $v$  accordingly
8:       end if
9:     until set  $v$  to the next empty voxel along  $r$  is unsuccessful
10:   end for
11: end for

```

6.4 Ray Buffers

Although the ray casting approach described in the previous section is already very efficient, we can further improve the performance by storing additional data from the previous time step. To this end we store per pixel ray information as described in [BMG05, BM06] for each time step. The information saved includes the starting point and the direction of the ray passing through a pixel as well as the first surface voxel hit by the ray. We call this per pixel information ray buffers. This information allows us to directly jump to the first voxel on the surface for a given ray without having to sample the ray first. This helps us to define an even smaller search space. However, the information in the ray buffers has to be updated after every update step in the voxel volume which creates additional processing overhead.

The deletion phase is similar to the ray casting method. Using the ray buffers we can directly find the first surface voxel which should be removed. After removing a surface voxel all ray buffers in which this voxel was visible have to be updated to the next visible voxel. In large scenes this reduces the search space size and consequently the amount of required computations for finding the first visible surface voxel noticeably.

On the other hand, in the addition phase, we cannot directly find new voxels using the available data in the ray buffers, because we only store information about the current surface voxels. New voxels can however appear at any point on the ray. For improving performance, after finding the first new voxel - using the same approach as in the addition phase of the ray casting method - we check the neighboring voxels using a 3D region growing method in voxel space, starting on the newly added voxel. While new voxels are found which should be occupied we continue the region growing process and then, by projecting the newly added voxels into the input images, mask out all corresponding changed pixels from the difference images. The ray buffers are also updated accordingly. This helps us to avoid further redundant processing.

6.5 Analysis

Our proposed approaches update the scene based on the changed areas in the images, so each change in the visual hull of a model should be reflected in at least one of the silhouette images used for reconstruction. In this section we show the correctness of this assumption.

As described before, the scene volume is divided into voxels. We define the set V such that it contains all voxels in the scene. We will refer to each voxel with v . For constructing the visual hull, we use n calibrated cameras viewing the scene. The set C contains all cameras and we will refer to each individual camera in this set using c_1, c_2, \dots, c_n . Each of these cameras has an associated projection matrix P_{c_i} . $P_{c_i}(U)$ is the 2D projection of a 3D point U in camera c_i . Silhouette images for each of these cameras are referred to using I_{c_i} . The value in the silhouette image at location u is given by $I_{c_i}(u)$. In the silhouette images pixels can have two distinct values, foreground and background. We refer to foreground with value one and to background with value zero. Extending these notations for dynamic scenes requires adding a time step index. For example for referring to the camera image c_i in time step t we use $I_{c_i}^t$ and so on.

Using this notation, we will represent the visual hull reconstruction process. The reconstructed visual hull of a scene is defined as

$$D = \left\{ v \in V \mid \forall_{c \in C} (I_c(P_c(v)) = 1) \right\}$$

which is a subset of V including only occupied scene voxels. D contains all voxels, whose projections are mapped to foreground pixels in all images. The state of a voxel can change between two time steps t and $t + 1$ (*Changed Set*). Unchanged voxels are either empty or occupied at both times, while changed voxels are occupied at t and empty at $t + 1$ or vice versa.

For creating the voxel representation of the scene at time $t + 1$ it is sufficient to only invert the state of the voxels which are in the *Changed Set*. Formally the *Changed Set* is given by:

Lemma 1: The set of voxels S , defined as

$$S = \left\{ v \mid (v \in D^t) \wedge \exists_{c \in C} (I_c^{t+1}(P_c(v)) = 0) \right\} \cup \left\{ v \mid (v \notin D^t) \wedge \forall_{c \in C} (I_c^{t+1}(P_c(v)) = 1) \right\}$$

contains all voxels in the changed set.

Proof: The equation defining the set S is the union of two sets. The left is the set of voxels occupied at time t ($v \in D^t$) and empty at time $t + 1$. The second term on the left ($\exists_{c \in C} (I_c^t + 1(P_c(v)) = 0)$) is based on the observation that there should be at least one camera in which the projection of v at time $t + 1$ is zero if this voxel is actually empty. The right part adds the set of voxels, which are empty at time t ($v \notin D^t$) and occupied at time $t + 1$. The second term on the right ($\forall_{c \in C} (I_c^{t+1}(P_c(v)) = 1)$) is the set of occupied voxels, since by definition their projection on the images of all cameras at time $t + 1$ is one, i.e. it falls into the foreground region. \square

Lemma 1 gives us a clear definition for the optimum set of voxels, which only contains the changed voxels at time $t + 1$ with respect to time t . However, computing the subset S itself is not possible directly. Therefore, we propose a superset, which approximates S closely.

Definition: We define two new binary images I_c^{RV} , I_c^{AV} for any given camera at times t and $t + 1$ and compute their corresponding values at the location u as described by the following formulas:

$$I_c^{RV}(u) = I_c^t(u) \wedge \neg I_c^{t+1}(u), I_c^{AV}(u) = \neg I_c^t(u) \wedge I_c^{t+1}(u)$$

I_c^{RV} represents the projection of the removed voxels, whose disappearance causes differences between I_c^t and I_c^{t+1} and I_c^{AV} represents the projection of the added voxels, whose appearance causes differences between I_c^t and I_c^{t+1} . In figure 6.1(b) the red and green colored areas are showing I_c^{RV} and I_c^{AV} respectively.

Next we will show that if any change happens in the scene, which can be captured by the visual hull, we will see it in the difference images. The next theorem proves that our proposed search space is a superset for the *Changed Set*.

Theorem 2: The subset E , which is defined as described below, contains all changed voxels between times t and $t + 1$:

$$E = \left\{ \bigcup_{c \in C} \{v | I_c^{RV}(P_c(v)) = 1\} \right\} \cup \left\{ \bigcup_{c \in C} \{v | I_c^{AV}(P_c(v)) = 1\} \right\}$$

Proof: Again, the above equation contains the union of two sets. The left part is denoting all voxels for which there exist at least one camera with value one at the voxel projection in the corresponding image I^{RV} of that camera. Similarly on the right side of the union operator, we have a subset of voxels, denoting all voxels for which there exist at least one camera with value one at the voxel projection in the corresponding image I^{AV} of that camera. By rewriting the above equation we obtain:

$$E = \left\{ v | \exists_{c \in C} (I_c^{RV}(P_c(v)) = 1) \right\} \cup \left\{ v | \exists_{c \in C} (I_c^{AV}(P_c(v)) = 1) \right\}$$

By replacing I_c^{RV} , I_c^{AV} from the above definitions we get:

$$\begin{aligned} E &= \left\{ v | \exists_{c \in C} (I_c^t(P_c(v)) = 1 \wedge I_c^{t+1}(P_c(v)) = 0) \right\} \cup \left\{ v | \exists_{c \in C} (I_c^t(P_c(v)) = 0 \wedge I_c^{t+1}(P_c(v)) = 1) \right\} \supseteq \\ &\left\{ v | \forall_{c \in C} (I_c^t(P_c(v)) = 1) \wedge \exists_{c \in C} (I_c^{t+1}(P_c(v)) = 0) \right\} \cup \left\{ v | \exists_{c \in C} (I_c^t(P_c(v)) = 0) \wedge \forall_{c \in C} (I_c^{t+1}(P_c(v)) = 1) \right\} = \\ &\left\{ v | (v \in D^t) \wedge \exists_{c \in C} (I_c^{t+1}(P_c(v)) = 0) \right\} \cup \left\{ v | (v \notin D^t) \wedge \forall_{c \in C} (I_c^{t+1}(P_c(v)) = 1) \right\} = S \end{aligned}$$

For reaching the third line, we have used the fact that if a voxel can be seen from all cameras at time t , ($\forall_{c \in C} (I_c^t(P_c(v)) = 1)$), it is part of the reconstructed volume at time t , which gives $v \in D^t$. Similarly, if there exists at least one camera from which the voxel can not be seen ($\exists_{c \in C} (I_c^t(P_c(v)) = 0)$) at time t , that voxel is empty at time t ($v \notin D^t$). Using the result of lemma 1 and the fact that $S \subseteq E$ the correctness of this theorem can be stated. \square

Conclusion 1: For any newly removed voxel from the scene at time $t + 1$, which alters the visual hull, there should be at least one I^{RV} image, that reflects this change.

Conclusion 2: For any newly added voxel into the scene at time $t + 1$, which alters the visual hull, there should be at least one I^{AV} image, that reflects the change.

This theorem helps us to define a new search space E , whose size is close to the real changed set S as described in lemma 1. Moreover, because I^{AV} and I^{RV} are made up from the projection of changed portions of the scene we get a small 2D area to consider in each image. In other words, the size of the search space is directly related to the size of the changed portion of the scene; for instance if the scene is static, the search space will be empty because I^{AV} and I^{RV} are equal to zero.

6.6 Results

We have tested our proposed incremental reconstruction approaches on a publicly available data set provided by Daniel Vlasic [VBMP08] (see figure 6.3). We compared the proposed methods to existing visual hull computation approaches, such as the GPU method

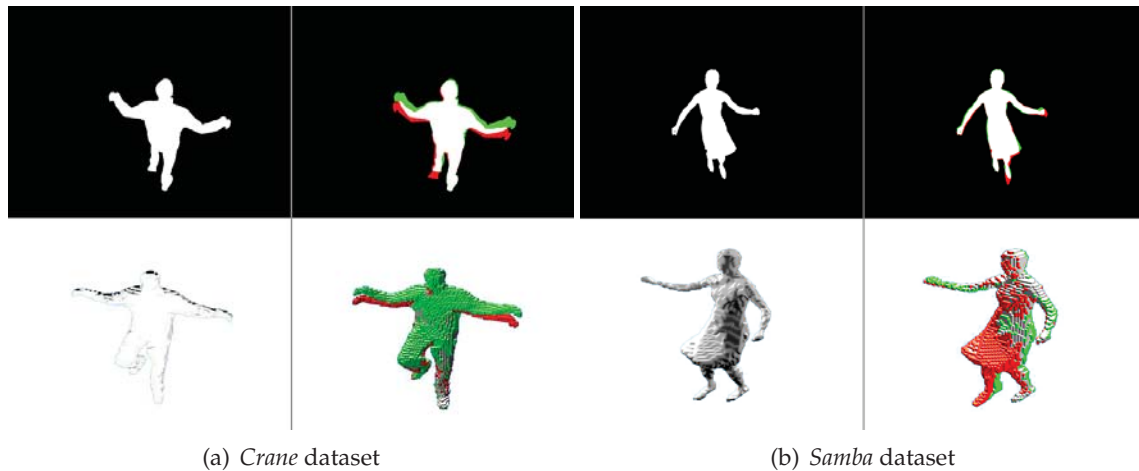
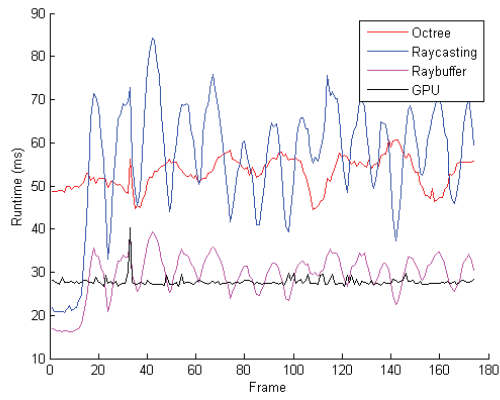


Figure 6.3: Data sets used for testing (provided by [VBMP08]). The first row shows silhouette and difference images while the second row shows the reconstruction from a novel viewpoint.

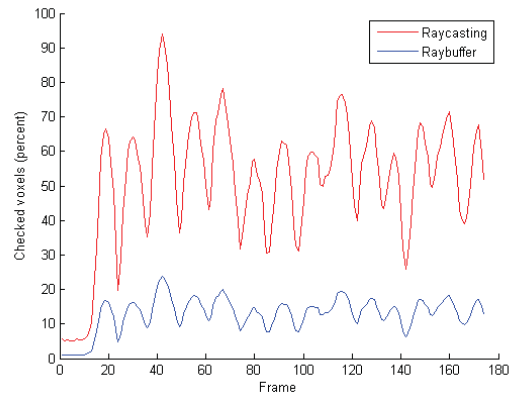
presented in the previous chapter, on different aspects such as run time and the number of occupancy checks with various voxel and image resolutions. The results show the significantly improved performance of our incremental approaches. Sometime the performance of the proposed methods even surpasses the efficient GPU-based implementation presented in the previous chapter while only using a single CPU core.

Figure 6.4 and 6.5 shows the results we obtained on the *Crane* and *Samba* data sets with respect to runtime and number of checked voxels at different voxel resolutions. Since the size of the search space of our approach is directly related to the size of the changed portion of the model, as expected, both the number of occupancy checks and the runtime are related to these changes. Therefore frames with many changes take more time to compute. For showing this, we plotted the percentage of checked voxels with respect to the total number of voxels for each model. The ray casting and ray buffer method are consistently faster than the CPU based octree reconstruction approach even though they only use one CPU core while the octree implementation uses four. This is expected, since we consider only the changed parts of the scene in our computations. We even partially outperform the highly optimized GPU-based reconstruction method presented in the previous chapter. The number of checked voxels is smaller in the ray buffer method than in the ray casting method since we can avoid checking many voxels by directly jumping to the first surface voxel using the information in the ray buffers.

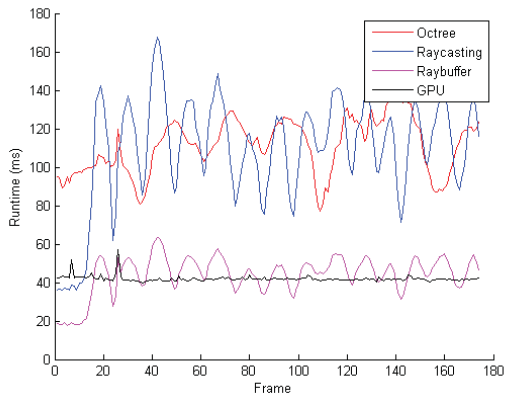
On the other hand, as shown in figure 6.6, image resolution and the number of cameras have little impact on the performance of the previous reconstruction methods, but they can affect our approaches noticeably due to the 2D image based search space. Generally, without subsampling the performance of our algorithms is directly related to the number of cameras and the image resolution. To limit this effect we subsample the input images to match the voxel resolution.



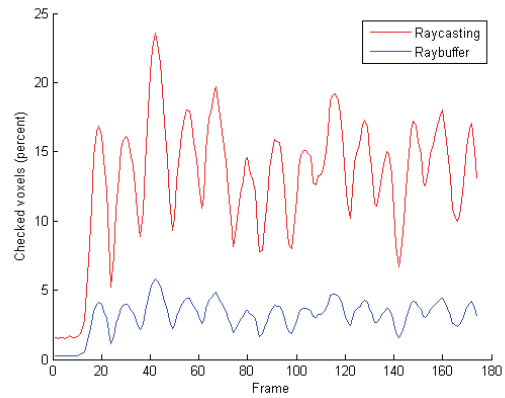
(a) Runtime (ms) 64^3



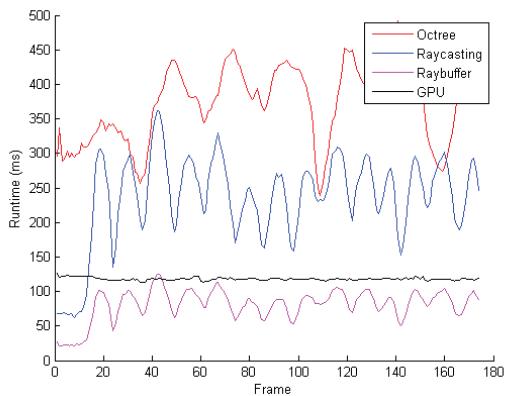
(b) Occupancy checks (percent) 64^3



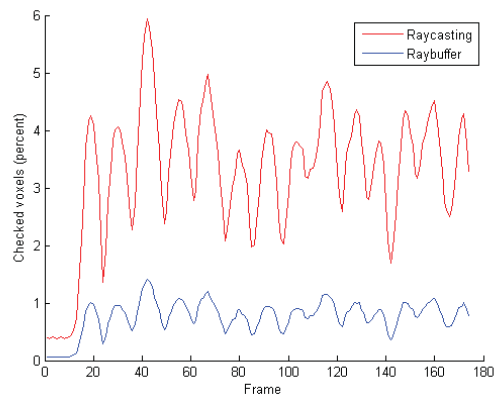
(c) Runtime (ms) 128^3



(d) Occupancy checks (percent) 128^3

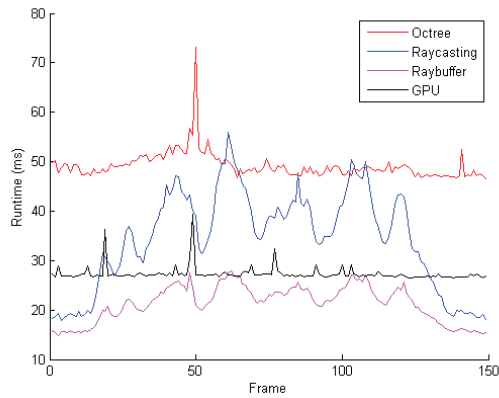


(e) Runtime (ms) 256^3

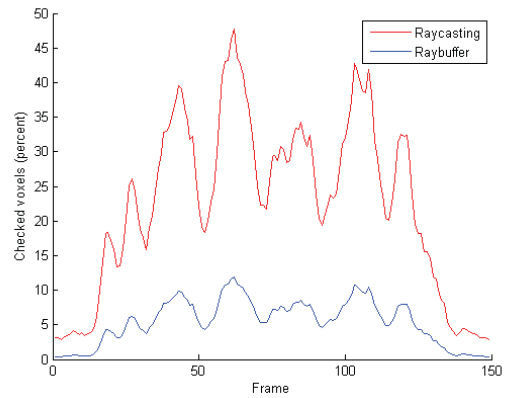


(f) Occupancy checks (percent) 256^3

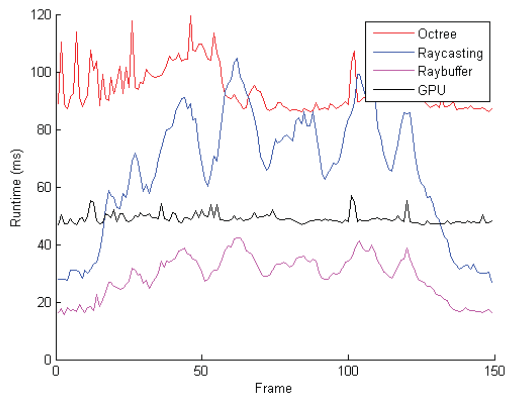
Figure 6.4: Runtime and occupancy check count for the *Crane* dataset at three different voxel resolutions. The runtime of the incremental reconstruction is consistently below the runtime of the GPU based method.



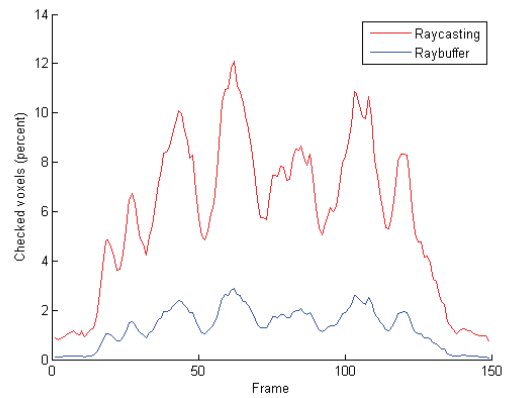
(a) Runtime (ms) 64^3



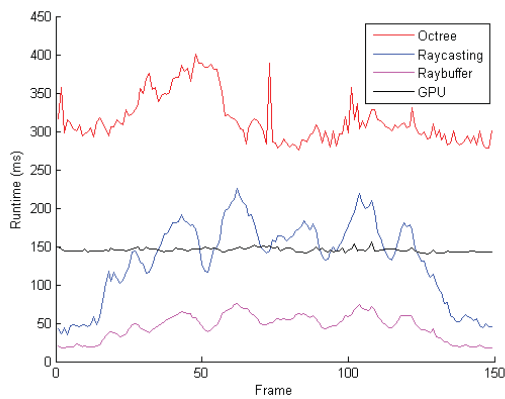
(b) Occupancy checks (percent) 64^3



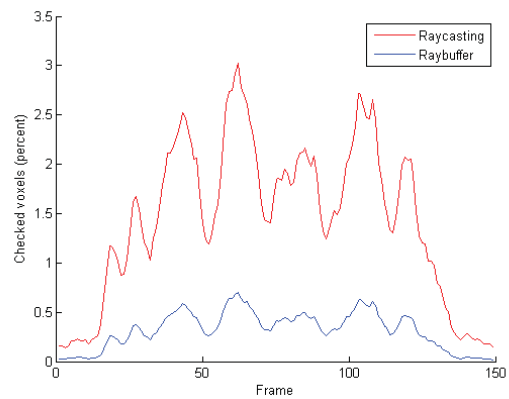
(c) Runtime (ms) 128^3



(d) Occupancy checks (percent) 128^3



(e) Runtime (ms) 256^3



(f) Occupancy checks (percent) 256^3

Figure 6.5: Runtime and occupancy check count for the *Samba* dataset at three different voxel resolutions. The incremental approach performs at a similar level as the GPU-based method, the exact runtime depending on the number of checked voxels.

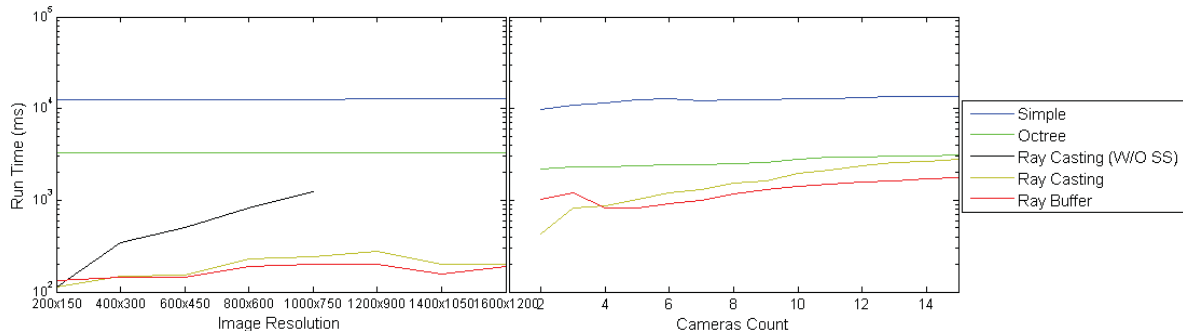


Figure 6.6: The left diagram shows the effect of using different image resolutions (frame 11 of the *Bouncing* dataset). The right diagram shows the performance with different numbers of cameras (frame 5 of the *Teapot* dataset). The volume resolution used was 256^3 .

6.7 Conclusion

We proposed a new incremental visual hull reconstruction approach using the difference in the silhouette images between two frames to efficiently find potentially changed voxels in the scene. Two ray-casting-based algorithms were proposed to this end, one of which stores additional per-frame information for speeding up the computations. Furthermore, the validity of our method was shown using a mathematical analysis. We demonstrated that our incremental method significantly improves the performance of the reconstruction process, outperforming existing reconstruction methods and making it suitable for real-time applications.

Despite these encouraging results, we do not use this method in our system, because the GPU-based approach performs in real-time at our target resolution of 128^3 and most importantly has a constant worst case runtime compared to the variable runtime of the incremental approach which depends on the amount of interframe changes in the scene. The worst case runtime is a very important aspect for our system, especially when considering the intended use of the system in an interventional environment, since we have to fulfill strict real-time constraints.

Chapter 7

Applications

Based on our 3D reconstruction system we investigated several applications which take advantage of the real-time 3D reconstruction of the scene. In particular we discuss two applications aimed at use in interventional rooms, namely a collision avoidance application for automated medical devices and a radiation modeling application. In addition, we briefly discuss the use of our system for clinical workflow recovery. We use these applications to show the benefits of introducing a real-time 3D reconstruction system into the interventional environment and to motivate further research in this area. Next to these interventional applications, we also present a more general mixed-reality application for occlusion-aware interactions.

For developing and testing the applications presented in this chapter we built two installations of our system: One in our lab and one in a real interventional room. Due to the difficulties of running a reconstruction system in an actual interventional environment, we performed all experiments using our lab system. The clinical setup was used to gather experience in running a multi-view 3D reconstruction system in an interventional environment. The lessons learned out of this installation will be discussed in the next chapter. In the remainder of this chapter we will discuss the different applications we propose and the experimental results we obtained.

7.1 Collision Avoidance in Interventional Environments

Today, it is not uncommon to have one or more automated devices, such as C-arms and medical robots, inside an interventional room. However, there is a danger of collision between these devices and other equipment such as ceiling-suspended monitor arms, radiation protection shields or the medical staff. A collision can damage both the device and the colliding object and sometimes requires to take the device out of service until a technician has evaluated the damage. The intervention may be suspended and the patient may be moved to a different room. Such collisions could be both dangerous and costly.

Nevertheless there currently is no fully automated solution to the problem of collision avoidance. Indeed, many systems rely on the discretion of the operating physician to avoid collisions. Technical measures include contact sensors and reduced movement speed. Some

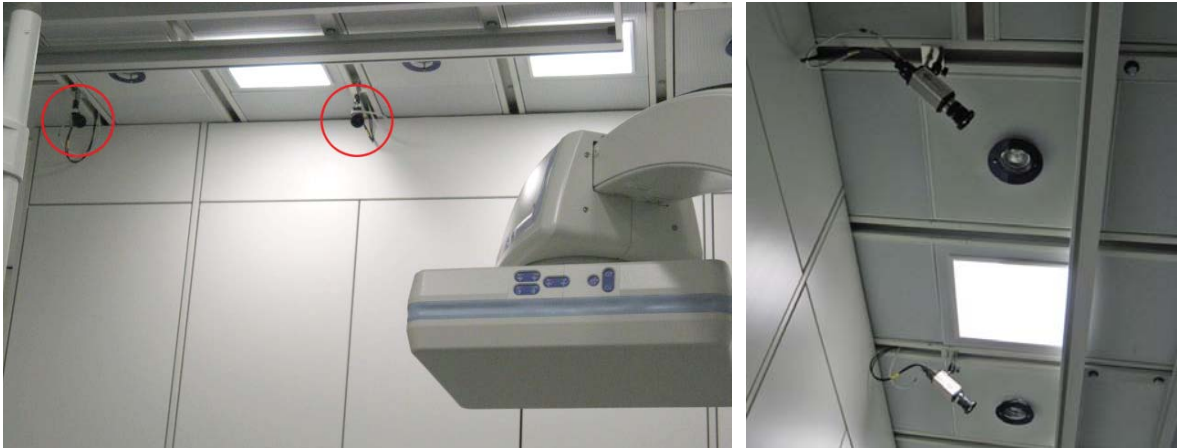


Figure 7.1: Left: C-arm and two cameras (red circles) in our clinical installation. Right: Closeup of the cameras.

devices also perform a slow test run to check the trajectory for obstacles. The disadvantage of these mechanisms is that they only detect a collision when it has already occurred. Indeed, they are more focused on minimizing the damage than on preventing the collision in the first place. In the case of C-arms, these safety considerations directly affect the speed of the device for certain movements. For instance, it has been shown in [GLV⁺05] that 3D rotational angiography can be performed in less time and with less contrast agent using angular C-arm motions instead of orbital motions. However, the C-arm speed required for this would cause serious injury in case of collision. With an automated collision avoidance, this method could be used more often and with increased safety for the medical staff.

In recent years, medical robots [FFK⁺07, AKA⁺04, SWM⁺04] have become an active research topic and some systems like the daVinci are regularly used in many hospitals. Sometimes such robots are also used in conjunction with other devices such as C-arms, navigation systems and electric tools. In such environments, staff unfamiliar with the movement range of all the devices can unknowingly enter the working volume of the device and cause an accident. This issue will become even more important in the future, since there will be many interventional rooms where multiple imaging modalities are used.

To overcome these problems, we propose a real-time collision avoidance system to increase the safety in the interventional room and allow faster device operation.

7.1.1 System Description

The collision avoidance system is based on the real-time 3D reconstruction system presented in chapter 4. As we want to avoid collisions between automated devices and other objects, the first step before running the system is to determine the working volume of the automated device. This can be done either manually by specifying the working volume in the room coordinate system or by reconstructing the automated device using the system itself. In this case, since the automated device typically cannot be removed from the interventional room, it has to be segmented in the silhouette images in order to obtain its reconstruction. The information about the working area of the device is later used in order to decide which

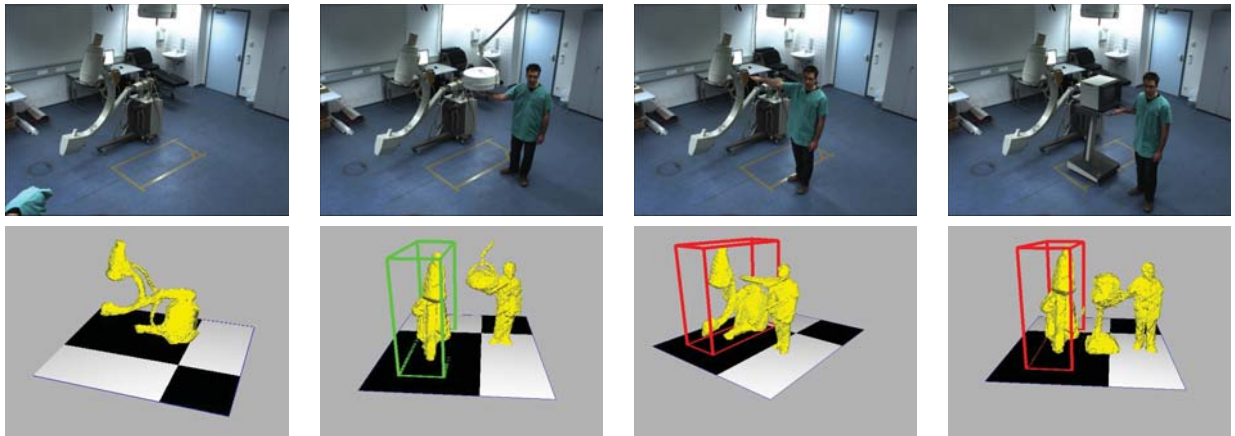


Figure 7.2: Results of the collision avoidance with a C-arm. Each column shows one of the input images in the upper row and the reconstruction in the bottom row. The first column shows the reconstructed C-arm. The second column shows the C-arm in a safe state (green bounding box), while the third and fourth column contain an object in the safety zone of the C-arm (red bounding box).

areas of the room are safe for people to enter during device operation. To this end safety zones are computed around the reconstructed objects and checked for intersections with the working volume of the device. Currently we use the bounding box of an object as its safety zone. If another object's bounding box enters the operating range of the device, a warning is given to the physician. To aid the physician in quickly finding the responsible object, we also visualize the reconstruction. The system is only active when the device is being operated, so that the physician is not disturbed by alarms, when it is safe to enter the working volume of the device.

The advantage of this method compared to existing approaches is that it is fully automated and detects possible collisions before they occur. It is also non-intrusive since the cameras are mounted on the ceiling and are therefore not in the way of the clinical staff. This solution works for any kind of automated device. In addition, once the system has been set up, very little maintenance work is required. Our system also meets the requirements for the presence of cameras in an interventional room since the images do not need to be saved. To the best of our knowledge, no previous work has considered a multi-camera-based collision avoidance system for an interventional room.

7.1.2 Results

For the collision avoidance application we duplicated our lab system setup in an interventional room in one of our partner hospitals (see figure 7.1). This allowed us to learn about the characteristic difficulties and challenges of working in an actual interventional room. These include a highly complex background, changing lighting conditions and crowded scenes among others. Due to these complexities we did not use the interventional system for our experiments but instead used the lab system.

The working environment we used has dimensions of size $3.7 \text{ m} \times 3.2 \text{ m} \times 2.2 \text{ m}$. This corre-

sponds to voxels of approximately 2.2 cm side length at a voxel resolution of 128^3 . This is sufficient in practice since we would like to maintain a security distance of at least 20 cm to the device. The experiments were performed with a C-arm. During runtime, we compute the bounding box for each object in the scene and test it for intersection with the working area of the C-arm, which is modeled as a rectangular working volume that extends 20 cm beyond the C-arm bounding box. If there is an intersection, a warning is displayed (bounding box turns red) and an alarm is sounded. Figure 7.3 shows some reconstruction results and some configurations with objects inside and outside the safety zone. All intersections with the safety zone of the C-arm were successfully detected.

7.1.3 Conclusion and Future Work

Although we only performed experiments using the lab system, our clinical partners are very interested in the system, which is why we were able to install it in a real interventional room. As already mentioned, the reasons we could not use the interventional system at this point are related to the challenging working conditions in interventional environments. The main difficulties are obtaining a good foreground segmentation and choosing the camera placement and the number of cameras in such a way, that ghosting artifacts in the reconstruction are kept to a minimum. This is hard due to the presence of many people and objects in the scene creating dynamically changing occlusions. These and further issues related to the challenges in actual interventional environments will be discussed in more detail in chapter 8.

In the future it would be interesting to track the automated device in order to obtain more accurate silhouette images. This could be achieved by placing optical markers on the device which are observed by the cameras. Together with a 3D model of the device the projection of the automated device in all silhouette images could be obtained and added as an occluder mask. This would be more efficient than only considering the maximum working area of the device which could be rather large. Another interesting aspect for future work would be to directly interface with the automated device and to use its internal configuration data in order to not only obtain the current pose without using markers, but to also know its future trajectory. This would be of great benefit for the collision avoidance application since it would allow to detect collisions even earlier. Finally, it would be of interest to either stop the device automatically or to compute an alternate trajectory on the fly which avoids the detected obstacles.

7.2 Controlling Radiation Exposure in Interventional Environments

In this application we are concerned with another important interventional topic, namely the radiation exposure of the physician. In contrast to even 10 years ago the use of interventional X-ray imaging and therefore the dosage received by the physician has increased dramatically, especially for fluoroscopy guided procedures. In fluoroscopy the X-ray source may be turned on for more than 30-60 minutes [ME96] during which the physician is standing very close to the patient and therefore the X-ray source. Due to the increase in the use of fluoroscopy guided procedures, interventional radiologists have nowadays a much greater risk of radiation damage. Although the dose received in a single intervention is not large, it

is well known, that long-term radiation exposure even at low doses can lead to negative effects on the body, which in the extreme can lead to cell mutations and cancer. The physician is usually protected by a lead vest and sometimes a lead collar around his neck. However, this leaves some body parts still exposed to the radiation, most notably the head, the arms and the hands. Especially the hands are at danger since they are used to work on the patient and therefore closest to the radiation source [ME96, TTK⁺08]. To control the radiation received a dosimeter is typically worn under the lead-vest. However, since the dosimeter is worn under the protective vest, its readings are not representative of the radiation received by the unprotected body parts. Moreover since it is placed at the level of the chest it is further away from the X-ray source than for instance the hands and therefore only of limited use. We want to present a solution to physicians, especially the less experienced ones, which allows them to quickly gauge their radiation exposure during an intervention and over the course of a longer time frame, in order to sensitize them for the dangers associated with X-ray radiation and the options of limiting their exposure. To this end we propose using our real-time 3D reconstruction system together with a geometric tracking algorithm to reconstruct and track the physician in the interventional room during the procedure and - with the knowledge about the position and movement of the X-ray source - to approximate his radiation exposure.

The contributions of this work are twofold: On the technical side we adapt a powerful methodology for tracking the physician from his real-time 3D reconstruction [CBI10a]. The tracking is a non-trivial problem since the physician has to be tracked based only on his geometry and correspondences have to be established between corresponding body parts in different reconstructions. In addition, the scatter radiation is simulated and the results of this simulation are used to determine the radiation deposited in an object at a given position relative to the C-arm, such as the physician.

On the medical side we show a concept for a system which sensitizes the physician to his radiation exposure and which helps him to take more informed decisions when using X-ray devices to minimize his risk for radiation related dangers. The system displays a color-coded map of the physician after each intervention which shows his radiation exposure during the procedure. This allows the physician to keep track of his long-term radiation exposure. The system can also be used for making inexperienced physicians more aware of the dangers of radiation exposure.

7.2.1 Related Work

There have been many medical studies concerning the radiation exposure of patients and surgeons during different interventions [SAHML07, BBT08, TTK⁺08, GBMR09]. In [TTK⁺08] for instance the radiation exposure of the patient and the surgeon is modeled mathematically. However, only estimates about the average distance of the surgeon to the X-ray source are used. Many studies conclude that the radiation exposure of the physician during fluoroscopy guided interventions is very high in the area of the hands [ME96] and in the lower extremities [SAHML07]. This has sparked some discussion about the possibility of protecting the surgeon and making him more aware of the dose he has already received. Our system attacks at this point since it allows the physician to check his exposure easily in an everyday setting.

There has also been work on simulating the radiation exposure for training novel physicians [WDDDB09, BWD⁺09]. In this work the C-arm was modeled in a simulation and the radiation dose was measured at several spherical detectors around the C-arm. However, the position of the physician was not taken into account and the system was not meant for use in interventional rooms but for training. We are not aware of any work which tracks the physician during the intervention and which accumulates his radiation exposure over the course of an intervention.

The tracking stage of our system is responsible for establishing dense correspondences between the independent 3D reconstructions of the scene computed at each time frame. As we are interested in the movement of the physician, our work relates directly to the vast body of literature addressing markerless human motion-capture in multi-camera environments. Most of these vision-based tracking algorithms use kinematic models in the form of a skeleton as discussed in the survey by Moeslund et al. [MHK06]. Such skeletal models allow to effectively constrain the problem and reduce its dimensionality but they usually require manual initialization of the joint positions, as well as lengthy parametrization of the tracked person's body characteristics from generic mesh models. In contrast to these approaches, we chose to use the framework recently presented by Cagniart *et al.* [CBI10a]. This algorithm doesn't rely on strong priors on the nature of the tracked object and instead deforms the first reconstructed mesh to fit the geometry in the rest of the sequence.

7.2.2 Reconstruction and Tracking System

We use our 3D reconstruction system described in chapter 4 to reconstruct the shape of the objects inside the interventional room in real-time. The reconstruction of the physician is then tracked using a mesh deformation framework. In a final step, the scatter radiation created by the C-arm is modeled and accumulated using the reconstruction of the physician. In the following sections we will describe the tracking and radiation modeling components of the system in more detail.

Tracking

The result of the 3D reconstruction stage is a sequence of meshes independently reconstructed at each time frame. We use the non-rigid mesh registration framework presented by [CBI10a] to iteratively deform a reference mesh to fit the geometry in the rest of the sequence. The reference mesh is taken from the first frame of the sequence. We prefer this tracking method to skeletal-based alternatives as it does not need an initialization of the pose and of the body characteristics of the tracked person. The only requirement is that the tracked physician must be clearly distinct from the rest of the reconstructed geometry in the reference frame. The tracking algorithm first divides the reference mesh into elementary surface patches. These patches are considered rigid. During runtime each patch is tracked using an ICP-based method. However, if each patch was tracked independently the resulting deformed mesh would not be consistent. Therefore, the predicted patch positions are determined using a weighting scheme taking the predictions of the neighboring patches into account. This is expressed in a global cost function based on the data term (i.e. the ICP-based tracking score) and the rigidity constraints over neighboring patches. By optimizing

this cost function a consistent deformed mesh is obtained. More details on this method are given in [CBI10a]. The output of the tracking algorithm is a dense trajectory for each vertex of the reference mesh across the sequence, which is necessary to accumulate the radiation exposure over time.

Radiation Modeling

Scatter radiation is created by the interaction of X-rays with matter due to physical processes such as Compton Scattering, Rayleigh Scattering and the Photoelectric Effect. These processes create secondary particles such as electrons and weaker X-rays which are deflected into certain directions according to the scatter spectrum of the particular interaction in question. This interaction typically occurs inside the patient's body. We therefore assume the source of the scatter radiation to be at the center of the patient. As the energy of an X-ray beam decreases quadratically with the distance from the source due to the dispersion of the particles over a larger volume we use this quadratic falloff of the energy as the primary modeling principle. Based on an initial energy at the scatter source, we compute a radiation volume which contains the energy of the particles in every voxel of the volume. This radiation volume is precomputed using some basic settings of the C-arm (e.g. the initial ray energy) and composed with the tracked mesh obtained in the previous step. This allows us to accumulate the radiation received by each vertex and by interpolation the radiation received by the whole mesh.

In order to obtain a more accurate simulation of the radiation the exact characteristics of the X-ray source as well as the patient geometry and composition (for instance obtained by a registered CT scan) as well as the location and composition of other objects in the room would have to be considered. This is, however, beyond the scope of our study.

7.2.3 Results

To validate our system we recorded an interventional scenario in our lab consisting of a C-arm, a patient and a physician. The physician is moving around the C-arm while the C-arm is constantly radiating. This is a typical scenario for a fluoroscopy guided intervention, since in fluoroscopy the C-arm is also almost constantly radiating. The frame rate at which the reconstruction system is running is 20 fps. Once we obtain the reconstruction of the physician, we start the tracking algorithm. The physician is then tracked during the entire sequence. With the knowledge of the position of the X-ray source and the patient we can compute the source of the scatter radiation which is subsequently modeled using the radiation simulation framework. Since we have the tracked mesh of the surgeon with corresponding vertices over time, we can simply add up the radiation at each vertex position for each frame. By performing this addition over the whole sequence we obtain the final radiation dose collected by the physician for each vertex of his reconstruction. These values are then interpolated to obtain the radiation on the whole mesh. This is reasonable since the mesh consists of only small triangles. Finally, we visualize the accumulated radiation using a heat map (see figure 7.3). The scaling on the heat map is set so that the maximum amount of radiation received by the physician is marked as bright red. This makes it easier to visually gauge the exposure. It can be seen, as also observed in [TTK⁺08], that the hands receive most of the radiation.

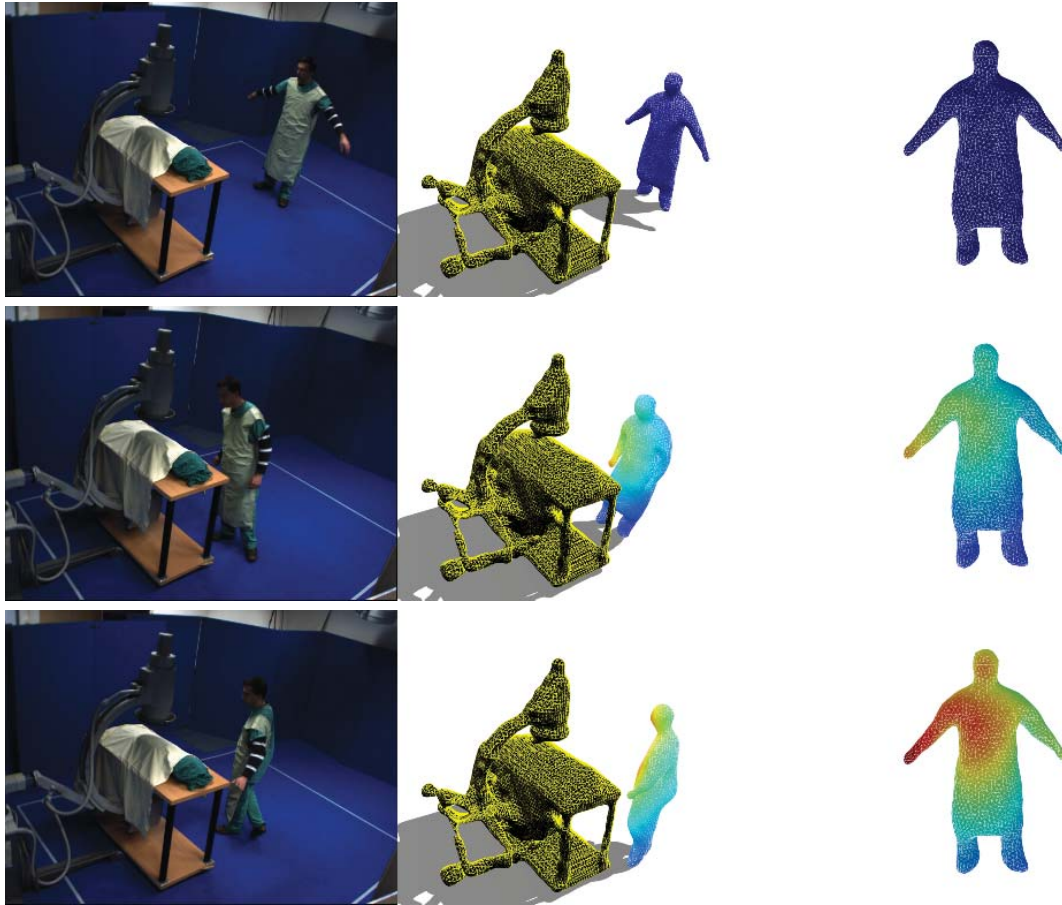


Figure 7.3: Results on a sequence recorded in our lab. The three rows show the radiation exposure at the beginning, the middle and the end of the intervention. The first column shows one of the input images, the second column shows the tracked 3D scene and the final column shows the physician in his reference pose with the color-coded radiation exposure.

7.2.4 Discussion

We understand our system as a proof-of-concept. Our goal was not to develop a system which can be directly used in an operating room, but to show what problems have to be addressed, which methods are available to solve them and how they can be combined in a sensible way. In particular the exposure computed by our system needs to be validated by experimental measurements and the dynamic and static environment of the interventional room needs to be taken into account in more detail.

7.2.5 Conclusion and Future Work

We presented a system for modeling the radiation received by a physician during an intervention. Our system builds on a real-time 3D reconstruction of the interventional room which is used for tracking the 3D-mesh of the physician using a mesh deformation framework. Our contribution is the combination of real-time 3D reconstruction and mesh-based

tracking to help the physician estimate his radiation exposure and to allow him to collect statistics about his long-term exposure. Our system can also be used for making novice physicians more aware of their radiation exposure. Future work includes validating the radiation estimation by using sensors attached to the physician, creating profiles for each physician and bringing the system to a real interventional room.

7.3 Workflow Analysis

Workflow analysis deals with recovering and analyzing the workflow of an intervention [LSM⁺05, LNA⁺06, PBE⁺07]. This analysis is performed using signals obtained during the intervention (e.g. the presence of certain instruments) and a model of the surgery to determine the current stage of the surgical procedure. This allows for instance to give an estimate of the remaining time of the procedure or to adapt the user interface of the surgeon's workstation to only show the information relevant for the current step of the procedure. In addition, workflow analysis could be used to help document the surgery and to analyze and compare the proficiency of different surgeons in performing the same procedure. However, all these goals can only be achieved when it is possible to automatically obtain signals which allow to deduce the current state of the intervention. In practice these signals although plentiful are hard to obtain in an automated fashion without user intervention and without interfering with the surgeon's work. The real-time 3D reconstruction provided by our system on the other hand is non-intrusive and contains enough information to be used as an input signal to a workflow recovery algorithm.

Consequently Padoy *et al.* [PMW⁺09] explored the use of our system for monitoring the workflow of an interventional procedure. They use the 3D reconstruction to extract 3D motion features. These motion features are used as input to a Hidden Markov Model (HMM) describing the surgical procedure. Using this model and the computed 3D motion features, the current state of the surgery is determined. The 3D motion features used are based on an extension of the Lukas-Kanade method (KLT) [BM04] to 3D. For computing them the reconstruction volume is subdivided into a set of evenly spaced cells. For each cell a histogram of the motion orientations - determined using KLT - are computed. In a final step the dimensionality of the histograms is reduced using PCA. These reduced histograms are given as input vectors to the HMM model. They are used both to train the HMM and to determine the current phase of the intervention at runtime.

The fact that our system was already successfully adopted by other researchers for use in interventional environments in our opinion underlines the significance of our work for the medical community.

7.4 Mixed Reality Interactions

Next to the interventional applications, we also developed a mixed reality application, in order to demonstrate the wide range of uses of our 3D reconstruction system. One of the key aspects in mixed reality is the integration of virtual objects into a real scene and their interaction with real objects. However, this is a non-trivial problem. To create the illusion of

actually belonging to the scene, a virtual object has to behave properly in the face of occlusion. Many existing systems are not capable of handling this case, leading to unconvincing augmentations, where the virtual object appears in front of the occluder. Another important aspect for a convincing presentation is the ability to interact with virtual objects. For instance, this would allow the user to pick up a virtual object and place it at another position.

We present a system which is capable of addressing both the occlusion and the interaction issue to create a convincing mixed reality environment. The user can interact with virtual objects without requiring any additional tools or external tracking while at the same time occlusions are seamlessly handled. Our system is based on the reconstruction of the 3D shape of objects inside an interaction space. Our proposed real-time 3D reconstruction system provides us with a 3D representation of every object in the scene, which in turn allows us to convincingly add virtual objects and to handle occlusions automatically. One of the key advantages of such a system over more traditional tracking-based systems is that we do not require any a priori information about the objects in the scene. We also do not need any prior setup or calibration for someone to use our system. There can even be multiple people in the interaction space at the same time. This makes our system a good candidate for use in real environments where people can just enter the interaction space, start to interact naturally with the virtual scene and then leave, without having to put on any special equipment. This significantly lowers the barrier to try the system and makes it attractive for presenting it to a wider audience, for instance in museums.

We implemented two exemplary applications which highlight the aspects of interaction and occlusion handling respectively. The first application is loosely based on the game Pong. The goal of the game is to prevent a virtual ball which is bouncing between the user and a wall from leaving the interaction space, by placing oneself in its path (see figure 7.4). This application shows the interaction between real and virtual objects. The second application is more focused on providing correct occlusion handling. This is done in the context of video compositing. We record several sequences in the interaction space at different points in time and use the depth map computed from the 3D reconstruction to join the sequences while correctly handling occlusions.

7.4.1 Related Work

In recent years, several real-time 3D reconstruction systems which explicitly recover the visual hull have been proposed [CKBH00, BSD03, WTM06, AMR⁺07, HLS04, LBN08]. However, only [HLS04, AMR⁺07, LBN08] actually run at frame-rates which allow interactivity. Other researchers have focused on implicitly computing the visual hull [MBR⁺00, PCF⁺02, DMB07, LNWB03]. The main difference to the explicit systems is that they only generate an image of the visual hull from a novel viewpoint without recovering an explicit 3D reconstruction. This is acceptable in some cases, but does not allow any form of interaction which requires the full 3D shape (e.g. taking the volume of the object into account). However, it is still possible to use it for collision detection [LNWB03, DMB07]. As is to be expected these systems run faster than comparable systems performing an explicit 3D reconstruction. However, today explicit reconstruction systems reach real-time performance, so that there is no drawback to making use of the additional information.

Some early work on real-time 3D content capture for mixed reality, was presented in [PCF⁺02].

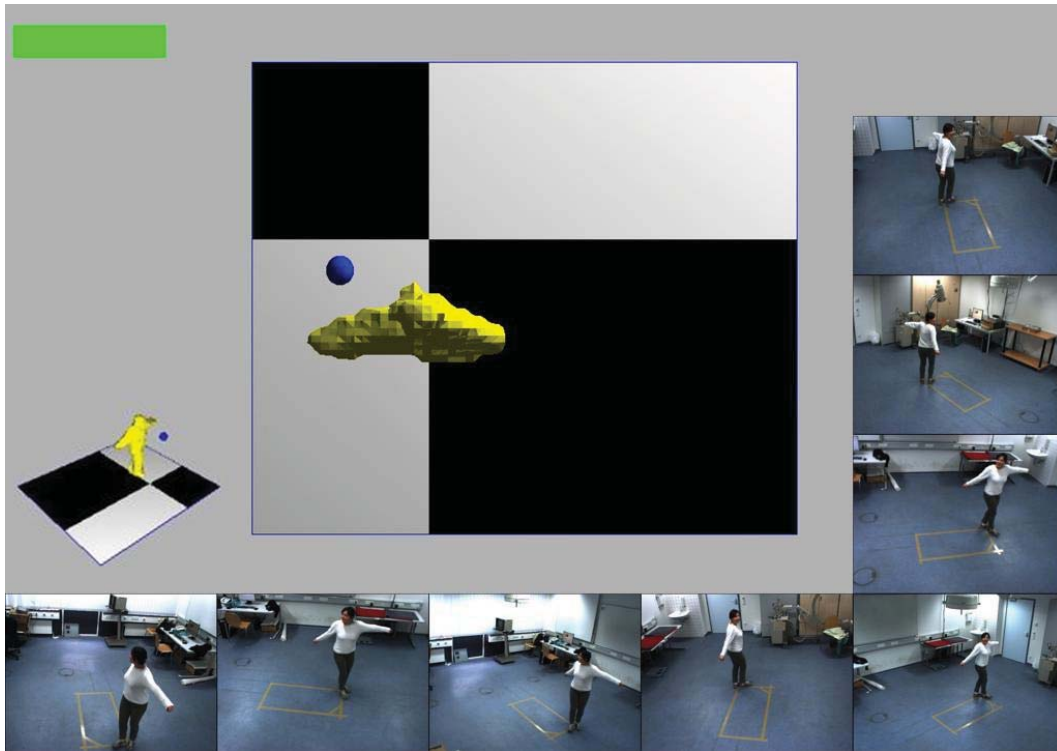


Figure 7.4: Our system allows the user to interact with virtual objects using natural movements due to a real-time 3D reconstruction. The images placed around the center show some of the input views while the center and the left side show an orthographic and a perspective view of the scene respectively.

In this paper a novel view generation system was used to insert 3D avatars of real objects into a virtual environment. The system runs at approximately 25 fps using 15 cameras. However, the aspect of interaction between real and virtual objects was not considered. In [DMB07] the authors present a collision detection scheme which extends the work in [LNWB03] allowing the interaction between real and virtual objects. However, they are also not using an explicit 3D reconstruction. In addition their system is running at only 10 fps using 7 cameras which is rather low for real interactivity. Petit *et al.* [PLBR09] as well as Hasenfratz *et al.* [HLS04] also present mixed-reality applications allowing the interaction between real and virtual objects based on the GrImage platform developed at INRIA.

Using our system we recover the explicit 3D reconstruction of all objects in the scene at a real-time frame rate of 30 Hz using 16 cameras. This allows us to also perform interactions with objects which are occluded by other objects and would therefore not be visible in a system based on an implicit reconstruction.

7.4.2 Pong

The idea of using mixed reality to create a game in which users interact with virtual objects has already been introduced with the ARHockey system [OSYT98]. The ARHockey system used a lot of tracking devices and HMDs to enable the illusion of having a virtual puck

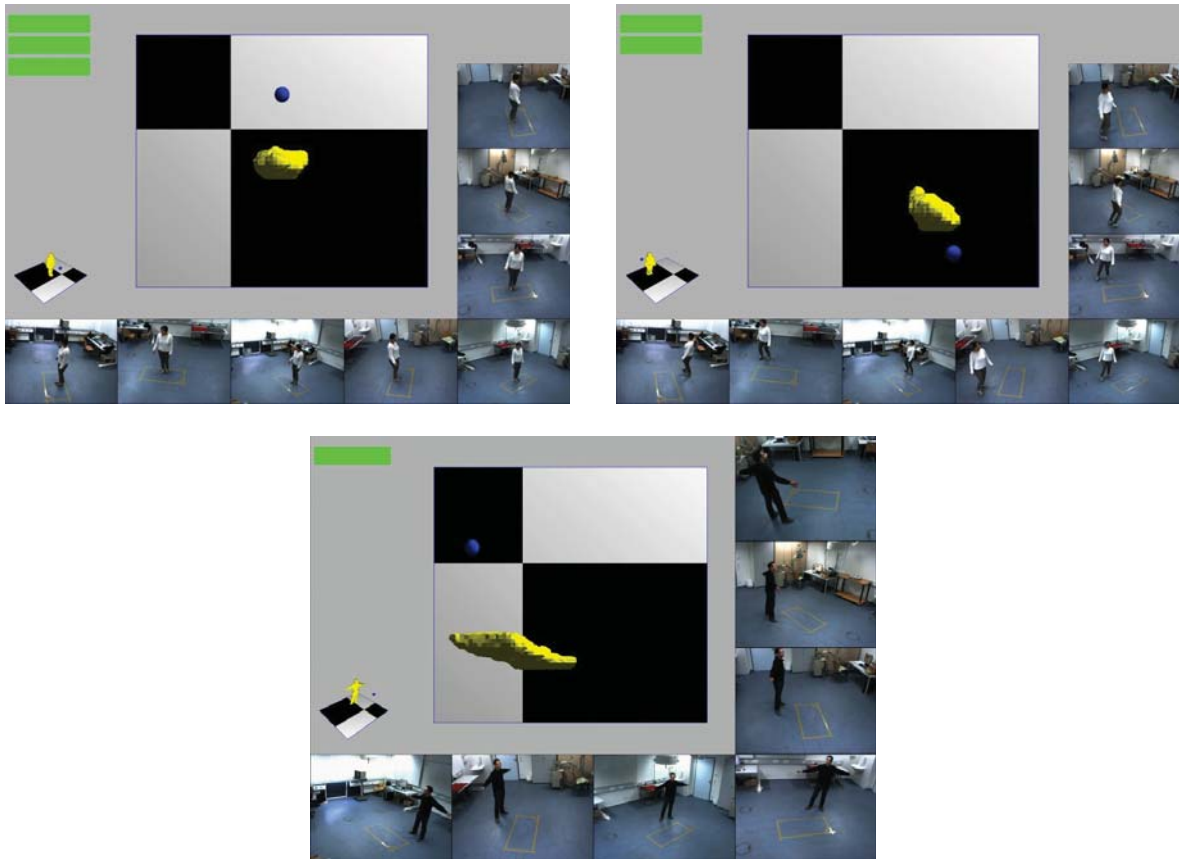


Figure 7.5: Our system allows the user to play Pong by interacting with a virtual ball. In the upper left corner the player's remaining life points are shown.

which is controlled by the hands of the users. Picking up on this idea we used our system to implement a game which is loosely based on the game Pong. In the original game two players each control a pad and pass a ball between each other. If a player fails to catch the ball his opponent gains a point. We modified the game so that one player is playing against a wall. The goal is to keep the ball from exiting the scene. The player has a certain amount of life points and has to try to keep the ball in the game for as long as possible.

We use a video projector to display the reconstruction of the interaction space on a wall. The user can see himself moving in 3D and he has to position himself, such that the ball is reflected off of him (see figure 7.5). The collision test is performed between the virtual object and the visual hull. There are two modes. In the first mode we only use the bounding box of the visual hull to perform the collision test. This has the advantage that it is easier for the user to hit the ball, because there is a bigger interaction area. Using the bounding box also allows children to easily capture the ball, because they can extend their arms to compensate for their lesser body size. The second mode performs the collision test directly between the visual hull and the virtual object. This leads to a more natural interaction, because it is very intuitive. However, the problem here is that it is hard for the user to estimate the height of the virtual ball, so that it might happen that he extends his arm, but the ball passes below it. This problem can be reduced by showing several views of the reconstruction. Optimally a

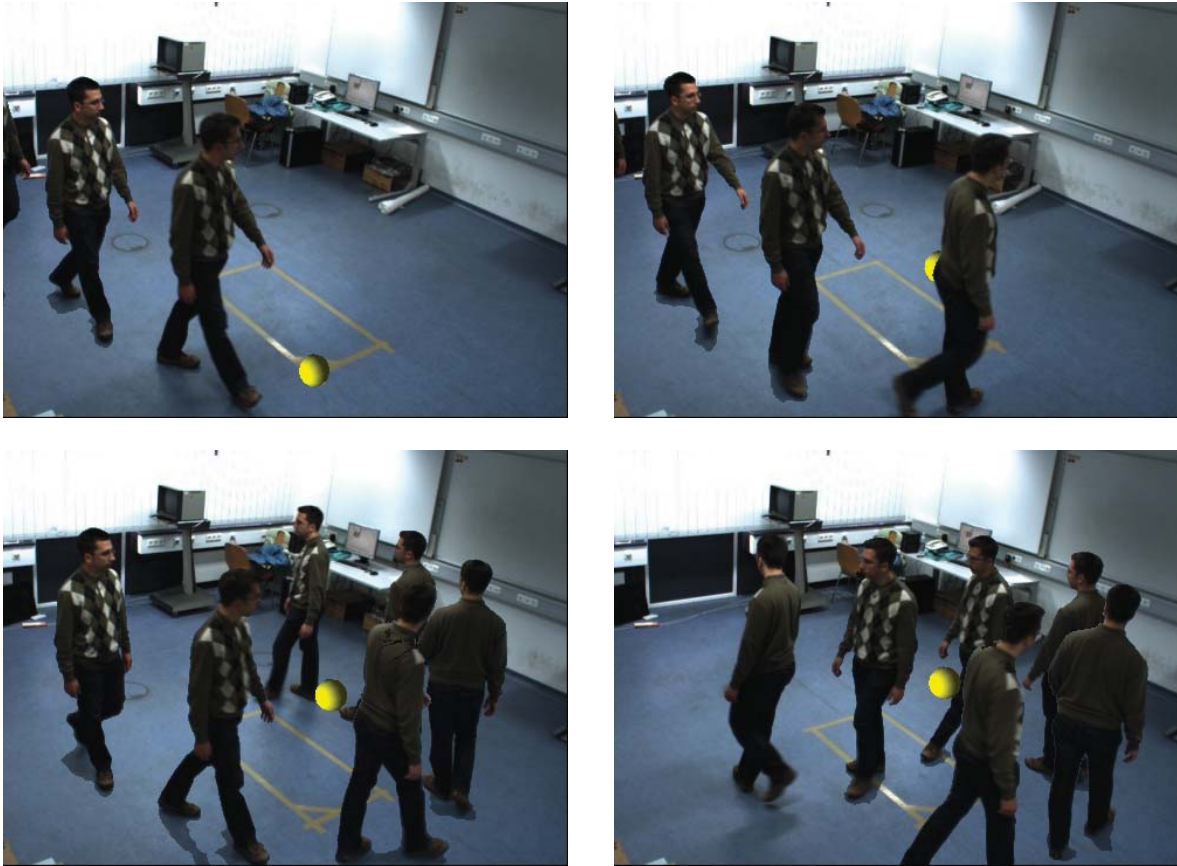


Figure 7.6: Video compositing. We created a new sequence by composing the same video six times in 1 second intervals. Note the correct occlusion handling with respect to the virtual ball and the different time steps of the original sequence.

stereoscopic HMD would allow the most natural interaction.

It is also possible for multiple people to play the game at the same time. Due to the use of our reconstruction system the player can use his whole body to catch the ball. The interaction is very natural and people intuitively know how to move to catch the ball. Their movement is not hindered by any additional equipment as would be the case in a tracking-based solution. Even when using a tracking-based solution it would be quite complex to correctly compute the extents of the body. Due to its easy usability and the fact that no setup or training phase is necessary for the user, our system is well suited for use in a real environment, for instance in a museum.

7.4.3 Video Compositing

As a second application we implemented a video compositing system which properly handles occlusions. This is an important topic in mixed and augmented reality [Ber97, KYS03]. Using our system we recorded a sequence of a person walking inside the interaction volume. The goal was to compose the same scene multiple times in one of the input cameras.

We therefore first used the reconstruction to compute the depth map for the target camera for every frame in the sequence by tracing a ray for every pixel into the scene and finding the distance to the first intersection point of the reconstruction. In order to compose two different parts of the scene into one frame we first need to select one frame as the reference frame. The foreground objects from the second frame are then transferred into the reference frame using the segmentation as mask. For every pixel in the second frame which is marked as foreground the depth at this pixel and the corresponding pixel in the reference image are compared. If the depth of the pixel in the reference image is larger, the pixel from the other image is used. Otherwise the pixel in the reference image is kept.

By using both the information from the depth map and the segmentation we created a sequence which shows the same scene at six time steps with an interval of one second in between at the same time (see figure 7.6). The effect is that instead of one person you can see a queue of 6 copies of the same person walk inside the room. Due to the use of the depth map we correctly handle the occlusion effects. In addition, we added a virtual bouncing ball to the scene which also correctly obeys the occlusion constraints. For creating the composited scene we currently do not apply any image-based refinement on the silhouette borders, but this could be easily added into the system.

The compositing results we obtained would be very hard to achieve using purely image-based techniques which do not consider any information about the 3D structure of the scene. It would require a (manual) segmentation of the objects of interest in the entire sequence which is extremely time consuming especially for long sequences. With our solution the segmentation and the depth information is automatically recovered without any additional intervention from the user.

7.4.4 Conclusion

We presented a real-time system for occlusion-aware interactions in mixed reality environments. The 3D scene reconstruction is used to allow users to interact naturally with virtual objects inside the scene while correctly handling the problem of occlusions in the augmentation. This is an important aspect in mixed and augmented reality. We demonstrated the results of our system in two application scenarios. The first application is an interactive game which focuses on the interaction aspect, while the second application is a video compositing task which focuses on occlusion handling.

7.5 Conclusion

We presented several applications making use of our real-time 3D reconstruction system. In particular we focused on interventional applications, showing how our system can be used for collision avoidance, radiation modeling and workflow analysis. These interventional applications demonstrate the value our system adds to an interventional environment by enabling novel interventional applications.

We created two installations of the system: One in our lab and one in a real interventional room. All experiments were performed using our lab system, while the clinical installation

was used to collect information for improving the design of future real-time 3D reconstruction systems used in interventional environments. This will be discussed in more detail in the next chapter. In addition to the interventional applications we also presented a mixed reality application which enables occlusion-aware interactions. We are certain that the encouraging results obtained in the presented applications as well as the fact that our system was already used successfully by other researchers for clinical applications motivate further research in bringing multi-camera 3D reconstruction systems into the interventional room.

Chapter 8

Conclusion

We presented a real-time multi-view 3D reconstruction system for interventional environments paying particular attention to the special conditions encountered in interventional rooms. On the algorithmic side we proposed both a high performance visual hull computation algorithm running on the GPU and an incremental visual hull reconstruction approach. On the application side we focused on two interventional applications, namely collision avoidance with automated medical devices and modeling the radiation exposure of the physician due to scatter radiation. In addition, we investigated a mixed reality application allowing interactions between virtual and real objects and shortly looked at workflow analysis using 3D data. The main goal of this work lies in showing the many possible uses of real-time 3D reconstruction systems in interventional environments and in paving the way for actually bringing such systems into clinical application.

8.1 Discussion and Future Work

Bringing a multi camera system to an interventional room poses several challenges related to the working conditions in such an environment. We therefore created two installations of our system: one in our lab and one in a real interventional room. The lab system was used for development and performing experiments, while the hospital system was used for learning about the conditions in interventional environments (see figure 8.1). In the following discussion we will list the challenges we encountered when bringing the system to the hospital and talk about our proposed solutions to these problems.

The experience we collected with the installation of the system in an interventional room, shows that there are several issues which are of importance in interventional environments, but which do not typically occur in multi-camera studios. First and foremost the **changing and complex background** has to be mentioned. Tooltables, cupboards and other medical instruments and appliances are located around the patient table. These are moved, replaced or taken away during the intervention. Often the background and foreground also have similar colors and there may be transparent and mirroring objects inside the scene. On top of that the **lighting conditions** are quite challenging since spot lights are popular and some physicians prefer to work almost in the dark with only a few spot lights illuminating the working space. These problems have to be addressed during the background subtraction

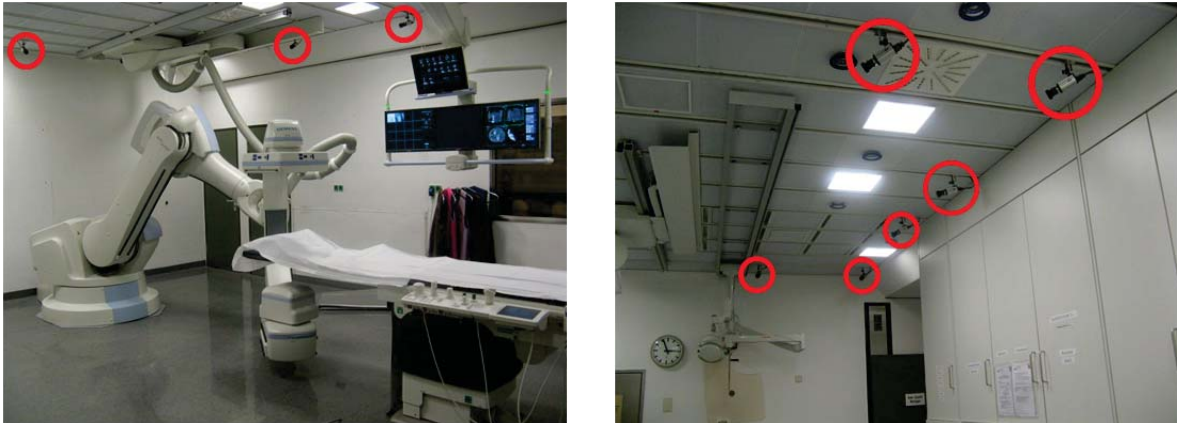


Figure 8.1: Images of our camera setup in the interventional environment. The cameras are highlighted with red circles. Note that the C-arm and the patient table have a similar color as the background. The C-arm and the ceiling suspended monitors also create dynamically changing occlusions.

stage of our system. We use a robust background subtraction algorithm which can deal with quick illumination changes. This makes background subtraction more robust, but still does not solve all problems. A problem which is in principle very hard to address are the **similar colors of the background and the foreground objects**. One way of dealing with this is to include TOF cameras in the interventional room, which do not depend on the color of the background in order to perform a segmentation. They could be used to perform a segmentation based on object depth. Additionally, the use of a multi-camera system should be taken into account during interventional room design by making certain that the background (walls, cupboards, etc.) are of a different color than the typical grayish tones used for medical devices.

Occlusions, such as those created by ceiling suspended devices, monitors and the patient table, also pose problems. In addition, the medical devices (in our case the C-arm) also dynamically occlude the scene and have to be modeled explicitly. These problems can be addressed by adding static occlusion masks to the segmentation results in the case of static occluders. For dynamic occluders tracking them explicitly using an articulated model is one option. This would require to either add markers to the objects or to use a markerless

Challenge	Proposed Solution
Complex and changing background	Robust background subtraction, TOF cameras
Same color background	TOF cameras, Design of interventional room
Transparent objects and mirrors	Explicit modeling
Static Occluders	Occlusions masks
Dynamic occluders	Tracking and dynamic occlusion mask, TOF cameras
Changing lighting conditions	Robust adaptive background subtraction
Crowded environment	Careful selection of number and location of cameras

Table 8.1: List of challenges encountered in interventional environments and our proposed solutions.

tracking algorithm which focuses on some easily detectable structures on the object. Another option would be to use TOF cameras and to segment objects inside the reconstruction area based on their depth.

Last but not least and also related to the issue of occlusion the interventional environment is usually quite **crowded** making it hard to properly reconstruct all objects without too many ghosting artifacts. This has to be addressed by carefully choosing the location and the number of cameras based on the typical occupancy pattern of the room.

In table 8.1 the challenges and our proposed solution are summarized. It is our belief that in order to make multi-camera reconstruction practical, the requirements of multi-view reconstruction systems have to be taken into account during the design of the interventional room, just as the presence of automated devices had to be taken into account when they were first introduced. The addition of TOF cameras could significantly improve the quality of the reconstruction, since they simplify one of the most challenging aspects of a purely passive vision-based approach, namely the difficulty of obtaining a good foreground-background segmentation.

Despite all these challenges, we believe that it is worth to tackle this problem since the addition of a multi-camera system adds a lot of value to the interventional environment by providing such innovative applications as collision avoidance, radiation modeling and workflow analysis, just to name a few.



Part II

Multi-View 3D Reconstruction and Organisation of Image Collections

Chapter 9

Introduction

In the second part of this thesis we will be concerned with the reconstruction and organization of (large) image collections. This topic has received increased attention in the last years due to the advent of Internet photo sharing sites like Flickr. By searching for a certain location on a photo sharing site a tremendous amount of images can be obtained almost instantaneously. However, there is also a downside to the ready availability of large amounts of visual data. Traditional reconstruction algorithms cannot properly deal with such huge numbers of images, as their runtime and memory consumption quickly goes beyond acceptable boundaries. At the same time not all images returned by a keyword search are related to the object of interest. Searching for 'Notre Dame' for instance not only returns images of the cathedral in Paris, but also of the US university of the same name among many other even totally unrelated search results. In addition, images are taken at different times of day and in different seasons and therefore include strong illumination changes and a variable background. Therefore, one important issue in dealing with such image collections is not only to perform efficient reconstructions but to also deal with organizing them and removing irrelevant or duplicate images.

In the following chapters we will be dealing with these issues. First, we investigate a method for obtaining high quality reconstructions of a scene since only using the visual hull is insufficient. Secondly, we are interested in clustering large image collections into smaller subsets, which can in turn be reconstructed more efficiently than the whole image set. Finally, we are interested in organizing image collections, i.e. browsing through them, finding representative images (so called canonical views) and removing redundant images for the purpose of reconstruction. However, before discussing each of these topics in more detail in the next chapters, we will first give a brief overview of the area of reconstructing and organizing large image collections.

9.1 Reconstructing and Organizing Large Image Collections

When considering reconstruction from image collections two areas have to be distinguished. The first aims at producing high-quality reconstructions using a moderate number of well-selected images, while the second considers using thousands if not ten thousands of images taken from Internet photo sharing sites such as Flickr for reconstruction. While the ultimate



Figure 9.1: Reconstruction results taken from [VKLP09]. The left and right show untextured and textured reconstruction results respectively. From top to bottom: Mountain (51 images), Sculpture (27 images), Castle (10 images), Herz-Jesu (8 images)



Figure 9.2: Examples of organizing and navigating through Internet Photo Collections taken from [SSS08b]. The left view shows the recovered camera positions of some images, while the right view shows the user interface for navigating through the image set.

goal of both areas is to obtain a 3D reconstruction, the challenges associated with using photos from Internet photo sharing sites are larger, since, as already mentioned, the amount of images and their variability, do not allow to simply use a traditional reconstruction approach without any prior preprocessing.

In the area of obtaining high-quality reconstructions there has been considerable work in the last decade [SCD⁺06]. The techniques used in this category typically belong to the area of multi view stereo and were already briefly discussed in chapter 2. Most current approaches start from an initial approximation of the object shape as obtained for instance through the visual hull or by triangulation and deform it until the reconstruction is maximally photo-consistent with the input images.

Based on the representation of the reconstruction and on how the optimization is performed several categories of methods can be distinguished. In graph-cut based approaches [BK03, VETC07, HVC07] the reconstruction is typically represented on a voxel grid and the optimal shape is found using a graph-cut on this grid. The advantage of this approach is that it gives the optimal solution in theory. In practice this is not always the case. We will have a closer look at this in the next chapter. Another shortcoming of this approach is the memory overhead due to the spatial discretization of the scene into voxels which can quickly grow beyond reasonable boundaries if a high-resolution reconstruction is desired.

Another class of methods are the variational approaches [FK98, PKF07, VKLP09] which iteratively deform a surface using gradient descent methods. The scene may be represented as a levelset function or as a mesh. The disadvantage of this method is that the gradient descent algorithm can fall into local minima. On top of that when using the levelset representation the same memory concerns apply as for the voxel grids used in the graph-cut approach. Therefore one of the currently most popular approaches is to use meshes since the memory overhead is smaller and the resolution can be adapted to the level of detail of the actual object being reconstructed.

One of the up to date most impressive reconstruction results was presented by Vu *et al.* [VKLP09]. They reconstruct large-scale outdoor scenes using a variational approach on

meshes which is efficiently implemented on a PC cluster to obtain reasonable runtimes. Some reconstruction results are shown in figure 9.1.

Regarding reconstruction from large Internet photo collections the Photo Tourism project [SSS08a] can be considered as one of the first major works. In that paper a large set of images taken from Internet photo collections is used for performing a point-based 3D reconstruction of the scene. An exhaustive pairwise matching followed by an incremental bundle adjustment phase are used both for the reduction of the image set and for 3D reconstruction. Follow-up work focused more on the aspect of organizing image collections for different purposes. In [SGSS08] the authors focused on navigating through large image collections by selecting appropriate neighboring views given an initial starting image and allowing the user to explore the scene visually by moving through the images in an intuitive manner (see figure 9.2). In other work Simon *et al.* [SSS07] considered summarizing the scene by selecting canonical views, i.e. images which contain much of the visual variance of the scene. Later Snavely *et al.* [SSS08b] worked on speeding up the initial reconstruction process by building a skeletal graph over the input image set which removes irrelevant images in order to perform an efficient and fast initial reconstruction which can then optionally be refined by adding back some of the removed images. Li *et al.* [LWZ⁺08] presented an application for performing reconstruction and recognition on large image sets. They construct a so called iconic scene graph which relates canonical views of the scene and use it for 3D reconstruction. This allows to already remove many irrelevant images before even performing an exhaustive pairwise matching which was a significant bottleneck in the Photo Tourism project. There have also been earlier attempts at reconstruction from unordered image collections [SZ02], however the number of images used was still small compared to the Photo Tourism project. Recently the computation time for obtaining high-quality reconstructions has been driven down quite dramatically using smarter bundle adjustment techniques (e.g. skeletal graphs [SSS08b]), GPUs and computer clusters [ASS⁺09]. Some recent quite impressive results from [ASS⁺09] are shown in figure 9.3. Around 150,000 images of Rome, 250,000 images of Venice and 57,000 images of Dubrovnik were collected from the Internet. After initial processing and clustering the largest connected components still contained thousands of images, which were then used to reconstruct some of the sites shown in figure 9.3.

9.2 Contributions

The contributions made in this part of the thesis are threefold. We first present an algorithm for obtaining high-quality reconstructions from moderately sized image sets. The proposed method is based on the graph-cut framework for reconstruction and adds a mechanism for preserving protrusions which are removed in the standard graph-cut formulation due to the minimum surface bias inherent to this method. The second work deals more directly with efficiently reconstructing large image collections. We propose a method to cluster large image sets into smaller subsets which can then be reconstructed independently and merged afterwards. This allows us to use existing reconstruction algorithms on large image collections. The third work proposes a general scheme for organizing large image collections using a spatial data structure relating scene regions across different images. Based on this structure we propose several applications related to organizing image collections, such as canonical view selection, image-based navigation and image set reduction, i.e. removing



Figure 9.3: Reconstruction results taken from [ASS⁺09]. Left (top and bottom): Two reconstructions of Dubrovnik's old city center (4,619 images). Right: Colosseum (2,106 images, top) and Trevi fountain (1,936 images, bottom).

images not contributing to a 3D reconstruction. Each of these contributions directly relates to the processing steps required for reconstructing large image collections, namely organizing the image collection, clustering it and performing the reconstruction.

Chapter 10

Graph-Cut-based Reconstruction

As seen in the first part of this thesis, the visual hull is only an approximation to the true shape of an object. We would like to improve on this approximation in order to obtain a more accurate 3D reconstruction. There exist many methods for this purpose as already discussed in chapter 2. In this chapter we use the volumetric graph-cut framework [BK03] for 3D reconstruction from multiple calibrated images and present a new algorithm for reducing the minimal surface bias associated with this approach. Our algorithm is based on an iterative graph-cut over narrow bands combined with an accurate surface normal estimation. At each iteration, we first optimize the normal to each surface patch in order to obtain a precise value for the photometric consistency measure. This helps in preserving narrow protrusions with high curvature which are very sensitive to the choice of normal. We then apply a volumetric graph-cut on a narrow band around the current surface estimate to determine the optimal surface inside this band. Using graph cuts on a narrow band allows us to avoid local minima inside the band while at the same time reducing the danger of taking "shortcuts" and converging to a wrong "global" minimum when using a wide band. Reconstruction results obtained on standard data sets clearly show the merits of the proposed algorithm.

10.1 Introduction

Reconstructing the shape of an object given a set of calibrated input images is a topic which has been extensively studied in the computer vision community as recently surveyed in [SCD⁺06]. A first class of methods reconstruct the visual hull of the object [Sze93, Lau94, FB03, LFP07]. These methods exclusively use silhouette information; consequently, they do not recover the concavities of the shape. A better shape reconstruction can be obtained using voxel-coloring or space carving which take the photometric consistency of the surface across the input images into account and allow the recovery of the photo-hull that contains all possible photo-consistent reconstructions [SD99, KS00, Kut00, SCM⁺04]. This approach works well in general; unfortunately, if a voxel is wrongly carved, it cannot be restored in later iterations. This can lead to the carving of the whole neighboring region (if not the whole volume). In addition, since space-carving is a greedy approach, it is hard to enforce smoothness constraints on the reconstruction. Another class of methods optimizes the surface integral of a consistency function over the surface shape. One way of minimizing this cost function

is the variational/level set formulation [FK98, HS04, PKF07]. In this formulation the surface is iteratively deformed using a gradient descent method. It is also possible to add regularizers to the cost function, such as smoothness constraints. However, being a local method it can fall into a local minimum. A second way of minimizing the surface integral is to use graph cuts [BK03, SP05, VTC05, BL06, FP06, HK06b, LBI06, TD06, YAC06, HVC07, SMP07]. One problem is that the cost function which is minimized is a minimum surface functional and hence the global solution is biased towards smaller shapes. Therefore, the global minimum might not correspond to the actual surface. This is also true for level set methods, but due to the local convergence properties of level sets, the effect is not as strong as in the case of graph-cuts. In practice this leads to the carving of narrow protrusions, since the graph-cut solution prefers shorter cuts over long cuts. This problem has been addressed by incorporating silhouette constraints [SP05, TD06, FP06, SMP07] or adding a ballooning term [VTC05, HVC07]. However, using silhouette constraints is only viable when exact silhouettes are available and a global ballooning term has the side effect of also pushing out concave regions of the object. In addition to the optimization method, the photometric consistency measure and its accurate computation also play an important role for the quality of the reconstruction results. In fact, an incorrect estimation may lead to overcarving the volume and eliminating protrusions.

We propose a method to reduce the bias of graph-cuts for smaller cuts without requiring exact silhouette images or using a ballooning term. To this end, we use iterative volumetric graph-cuts over narrow bands to minimize the influence of shortcuts on the reconstruction coupled with an exact normal optimization for each surface patch which is used to compute an accurate photoconsistency score. It is through this combination that we are able to avoid the overcarving of narrow protrusions. Using each technique by itself does not avoid the overcarving. Using only narrow bands still allows overcarving when the photoconsistency score is incorrectly estimated, while using an exact photoconsistency score in a wide band also has a high probability of overcarving, because of the high variation in possible path lengths (see figure 10.1(a)). At each iteration, we first compute the visibility of the current surface estimate and optimize the normals to the surface in order to obtain a precise value for the photometric consistency measure. Then, we apply a volumetric graph-cut in order to determine the optimal surface inside a narrow band around the current surface estimate. The band size can be adjusted to achieve a tradeoff between the ability to overcome local minima and the ability to preserve protrusions by discouraging shortcuts. We show that iteratively searching for the correct orientation of the surface and considering narrow bands is able to deal with the graph-cut's inherent bias for smaller shapes without requiring a ballooning term or exact silhouettes. Thereby our algorithm reduces problems with overcarving and preserves protrusions of the surface.

10.2 Related Work

The use of graph-cuts on narrow bands was proposed by Xu *et al.*[XBA03] in the context of image segmentation. Hornung *et al.*[HK06b] suggested the use of hierarchical iterated graph-cuts for 3D reconstruction. However, using graph-cuts on narrow bands alone does not necessarily preserve protrusions. Another important aspect is the exact computation of the surface consistency score. This is typically performed by computing the NCC of a small

patch in the tangent plane of the surface over several images. A critical aspect in this computation is the choice of the normal. The current surface estimate given by the graph-cut is not well-suited to determining accurate normals to the surface due to the discretization. In addition, the current surface estimate can be quite different from the actual surface. Therefore it is not sufficient to only evaluate the consistency measure using the normal given by the current surface. Instead it has to be optimized in order to find the most consistent normal over all input images in which the surface point is visible. This helps significantly in determining correct consistency values even for high curvature regions which cannot be well represented on the voxel grid. The concept of optimizing the surface normal appeared in other work related to 3D reconstruction [GM04, FP07, HK06a, HK07, ZPQS07]. Habbecke and Kobbelt [HK06a, HK07] use a normal optimization in order to find the orientation of 3D patches which are combined to approximate the surface. A similar idea is used by Furukawa *et al.* [FP07]. Both of these assume a reference patch with respect to which the normal is optimized. This is in contrast to our work which does not assume a reference view. None of these consider the use of normal optimization in conjunction with narrow band graph-cuts.

Prior work related to reducing the minimum surface bias in graph-cut reconstructions uses either a ballooning term [VTC05, BL06, VETC07, HVC07] or includes silhouette constraints [TD06, FP06, SMP07]. The use of a global ballooning term as suggested in [VTC05] has the drawback of also affecting concavities. It is often not possible to find one single value which achieves the desired effect of preserving protrusions and not affecting concavities. Hernandez *et al.* [HVC07] use an intelligent ballooning term which is based on the evidence of regions being inside or outside the volume. This yields improved results over using a global ballooning term. Boykov *et al.* [BL06] use the photoflux as an intelligent ballooning term to drive the reconstruction towards the object boundaries, thereby allowing the recovery of thin structures. The results they achieve on the *Gargoyle* data set are comparable to ours. However, we achieve this by using the normal optimization which does not require to add extra links to the graph for incorporating the intelligent ballooning term. The use of silhouette constraints is only viable when exact silhouettes are available. This is not always the case. In addition, the silhouettes only constrain the shape of the surface on the rims. The method proposed by Yu *et al.* [YAC06] uses graph-cuts on surface distance grids. The drawback of their method is that they assume, that the initial estimate is already quite close to the final result.

10.3 Volumetric Graph-Cuts

In this section we describe the volumetric graph-cut approach on which our reconstruction algorithm is based. We focus in particular on the photoconsistency measure used and the design of the graph.

10.3.1 Consistency Measure

One of the most common measures for evaluating the photo-consistency of a surface patch in the input images is the NCC due to its invariance to linear illumination changes. Similar to [FP06, SMP07], we compute the NCC for a surface patch over the image projection of its

tangent plane. The tangent plane is sampled with a uniform grid in 3D and each point is projected into the images in which the surface patch is visible. Since the NCC is only defined for image pairs, it has to be extended to multiple images in order to get one consistency score. One way of doing this is to select a reference image and to compute the NCC between this image and all other images. Using this approach, the reliability of the score depends very much on the reference image: If the reference image is of bad quality or if it contains occlusions which have not been captured by the model, the correlation will be low. Therefore, we choose to compute the mean of the pairwise score of all image pairs. If the NCC score between images i and j of the voxel \mathbf{x} is $\eta_{ij}(\mathbf{x})$, the photo-consistency score is given by

$$\rho(\mathbf{x}) = \Psi \left(\frac{2}{n(n-1)} \sum_i \sum_{j>i} a_{ij} \eta_{ij}(\mathbf{x}) \right) \quad (10.1)$$

where $\Psi(x) = \min(1-x, 1)$ normalizes the score to the interval $[0, 1]$. n is the number of images and a_{ij} is a weight depending on the angle between the two cameras. Since this consistency measure is based on the tangent plane it is important to obtain a good normal to the surface. A wrong normal can result in a low consistency score which in turn will lead to overcarving. Therefore, in our algorithm, we use the normal which results in the highest photo-consistency score as described in section 10.4.1.

In our implementation, we sample the tangent plane with a 5×5 grid. The spacing between the grid points is chosen so that no pixels in the image are missed. We only use views forming an angle of less than 60 degrees with the surface normal. The weights a_{ij} are chosen as the cosine of the angle between the viewing directions of camera i and j thereby penalizing big angles.

10.3.2 Surface Optimization with Graph-Cuts

3D reconstruction can be cast as an optimization problem to find the minimum cost surface where the cost of the surface S is modeled as the surface integral over a consistency score $\rho(\mathbf{x})$ for each surface patch \mathbf{x}

$$E = \iint_S \rho(\mathbf{x}) dA \quad (10.2)$$

Volumetric graph-cuts provide a way to find an approximately optimal solution to this minimization problem. This is done by converting the continuous problem into a discrete formulation over a voxel grid [BK03].

Given an initial estimate of the surface, a band is constructed around it and the best surface inside this band is found using a graph-cut [BVZ01, KZ04]. For every voxel \mathbf{x} in the band the consistency score $\rho(\mathbf{x})$ is computed, where a lower cost signifies a better consistency. In order to be able to compute the consistency for a voxel, it is necessary to know the visibility and depending on the consistency measure also the normal of the voxel. This information is propagated from the original surface by assigning the visibility and the normal from the closest voxel in the original surface. Alternatively, it can also be estimated for each layer in the band using the surface given by this layer.

Every voxel in the band is represented as a node in the graph. The source of the graph is placed inside the object while the sink is placed outside. All voxels on the inner boundary of

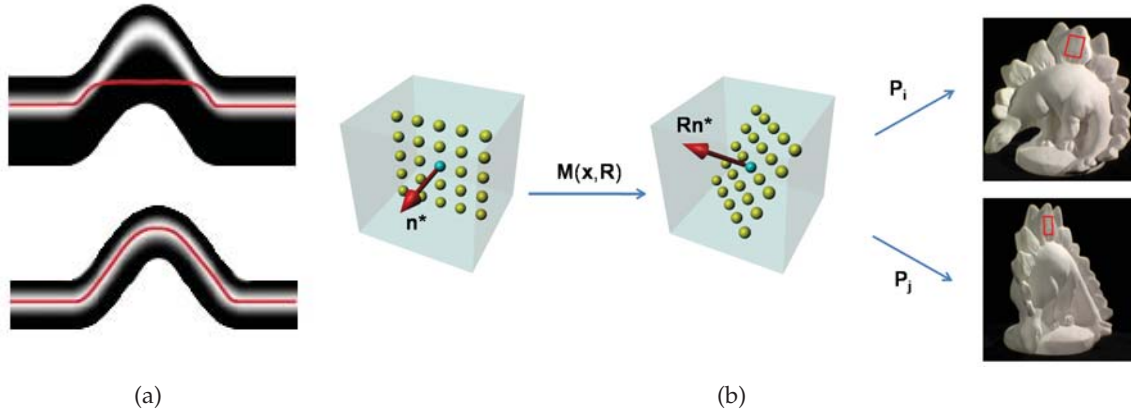


Figure 10.1: Figure (a) shows the advantages of using a narrow band. When using wide bands protrusions can be totally filled out. The graph-cut will then prefer the shortcut through the high-cost region, because its accumulated cost is less than that of the long path through the low-cost region. When using a narrow band, the graph-cut will take the low-cost path because there is no shortcut and any deviation from the low-cost path will incur a high cost. Figure (b) shows the photo-consistency computation. We find the plane which best approximates the surface passing through a voxel.

the band are connected to the source and all voxels on the outer boundary are connected to the sink by edges of infinite weight. Neighboring voxels \mathbf{x}_k and \mathbf{x}_l are connected by edges of weight

$$w_{kl} = c_{kl} \frac{\rho(\mathbf{x}_k) + \rho(\mathbf{x}_l)}{2} \quad (10.3)$$

where the term c_{kl} is a weight proportional to the distance between the two voxels [BK03]. Typical neighborhood systems are the 6- and the 26-neighborhood. All voxels still connected to the source after applying the graph-cut are part of the optimal volume.

In our implementation, we use the 26-neighborhood of the voxel instead of the 6-neighborhood when constructing the graph. This helps to avoid some of the discretization artifacts associated with graph cuts.

10.4 Proposed Reconstruction Method

We use an iterative graph-cut approach to recover the shape of the object. In each iteration, first the surface visibility is computed. Using this information the normals to the surface are optimized and used to compute a reliable consistency score. These scores are used in a volumetric graph-cut over narrow bands around the current surface estimate.

10.4.1 Surface Normal Optimization

As explained in section 10.3.1, we need the normal to the surface to compute the consistency score. The accuracy of the normal plays an important role in obtaining good reconstruction

results. If the used normals are inaccurate the computed consistency score will be low which in turn will lead to overcarving. This is especially true for surface regions with high curvature, because a small change in the normal can have a big effect on the consistency score. We therefore find the optimal normals which lead to the highest consistency score.

Since we use a volumetric representation, we first compute the gradient in the volume for all surface voxels, giving us the initial normals. However, the normals obtained this way are usually not very accurate due to the discretization and because the current surface estimate can be wrong. In particular, there are problems in surface regions with high curvature. Therefore, we only use these estimated normals as a starting point for an optimization over the surface orientation. The optimization allows us to obtain a better normal estimate and consequently a better consistency measure.

We represent the orientation of the surface through the rotation \mathbf{R} of a reference plane with normal $\mathbf{n}^* = [0, 0, 1]^\top$ (see figure 10.1(b)). The rotation is represented using a (3×3) rotation matrix \mathbf{R} . The consistency score for a voxel \mathbf{x} between images i and j depending on the rotation \mathbf{R} is given by

$$\eta_{ij}(\mathbf{x}, \mathbf{R}) = NCC(I_i(\mathbf{P}_i\mathbf{M}(\mathbf{x}, \mathbf{R})\mathbf{X}), I_j(\mathbf{P}_j\mathbf{M}(\mathbf{x}, \mathbf{R})\mathbf{X})) \quad (10.4)$$

where \mathbf{P}_i and \mathbf{P}_j are the projection matrices for images I_i and I_j respectively. \mathbf{X} are 3D points (expressed in homogeneous coordinates) defined on the reference plane. The matrix $\mathbf{M}(\mathbf{x}, \mathbf{R})$ is a (4×4) matrix that describes the transformation of these points given the orientation \mathbf{R} and the voxel center \mathbf{x} . It is defined as

$$\mathbf{M}(\mathbf{x}, \mathbf{R}) = \begin{bmatrix} \mathbf{R} & (\mathbf{I} - \mathbf{R})\mathbf{x} \\ \mathbf{0} & 1 \end{bmatrix} \quad (10.5)$$

The optimal normal and the highest consistency measure are found by determining the optimum of the cost function

$$\rho(\mathbf{x}) = \min_{\mathbf{R}} \left\{ \Psi \left(\frac{2}{n(n-1)} \sum_i \sum_{j>i} a_{ij} \eta_{ij}(\mathbf{x}, \mathbf{R}) \right) \right\} \quad (10.6)$$

where Ψ is the normalization function defined in section 10.3.1. In practice we optimize this cost function by discretely sampling a dense set of normals in the hemisphere into which the initial normal is pointing. The optimization over the rotation \mathbf{R} is restricted such that it provides a normal vector to the tangent plane inside a cone defined by the initial normal and an opening angle θ . In our experiments, we set θ to 60 degrees. Empirically we found that the normal optimization is a very important factor in achieving good reconstruction results. Without it protrusions and thin structures are carved much more often.

10.4.2 Robust Visibility Test

Determining in which input images a voxel is visible is essential for computing reliable consistency scores. However, the surface of the object usually contains a lot of small bumps and dents. This leads to problems when using a straightforward visibility test. Surface parts which should be considered visible are occluded by little bumps in their vicinity and are therefore marked as invisible. If a point is lying in a small dent it might not be visible in any

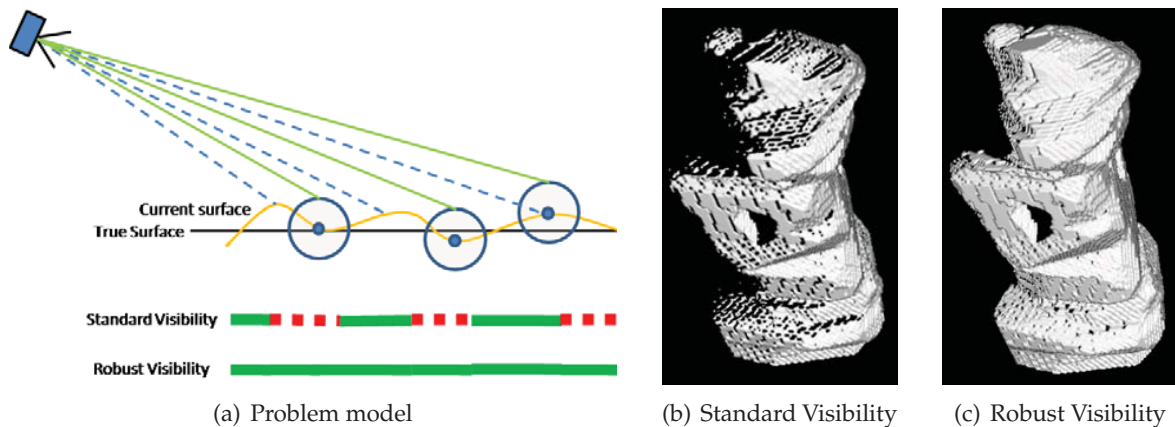


Figure 10.2: During the optimization, the object surface is usually not very smooth. This leads to bad visibility estimates due to self-occlusion using a standard visibility test (dashed lines). To avoid this problem, we compute the visibility for small spheres around the considered points (solid lines). This rendering of the visible points shows the difference between the standard and the robust visibility test applied to the visual hull of the *gargoyle* data set. The standard version contains a lot of holes while the proposed robust version contains much less. Note that this is not the actual vantage point of the camera. The view was rotated after the visibility calculation to highlight the differences in the visibility test.

of the cameras. A typical example of this problem and its effect on the visibility computation is shown in figure 10.2. Any exact visibility test will mark these points as invisible. To deal with this problem, we propose a robust visibility test.

We consider small spheres of radius r around every point and test if some part of the sphere is visible. To reduce the computational complexity, we approximate the sphere by 6 sample points located at the intersections of the sphere with the positive and negative coordinate axes. In our implementation, we choose r as the side length of a voxel. Figure 10.2 shows the difference between the results obtained using a standard visibility test and the proposed robust visibility test.

10.4.3 Narrow Band Graph-Cut

Existing volumetric graph-cut-based methods usually perform only one cut in a fairly big band around the initial surface estimate. By doing this, they assume that the maximum displacement of the true surface from the initial estimate is known. This is in particular not true if only rough silhouette estimates are available. Another disadvantage of one-shot graph-cuts over big bands is that it is hard to reason about visibility for voxels far from the initial surface estimate. Therefore, the consistency score computed for these voxels can be wrong. Another problem is setting the normals for the voxels in the band. Typically, either the normal of the closest surface point is chosen or the normal is computed using the surface given by the layer. However, this approximation can quickly become wrong when going further away from the original surface. Moreover, a big band can completely fill out protrusions and narrow parts of the object. As graph-cuts prefer short cuts, these protrusions

are easily carved away. In addition, the danger of overcarving increases since, in a big band, there are more possibilities to find a "shortcut" than in a narrow band.

We construct a narrow band by expanding the current surface estimate to the outside and the inside. The inside layers carve inconsistent voxels while the outside layers allow us to correct segmentation errors in the initial silhouette images. This iterative approach makes the visibility reasoning and the normal propagation much more stable as we stay close to the current surface estimate. Using narrow bands makes it possible to avoid the carving of narrow parts and protrusions since they are not totally filled out. Assuming that there is a clear low-cost path through the band (if there is one the normal optimization will find it), the graph-cut does not deviate from this path. In fact, the differences in path length through the band are so small, that they cannot outweigh the penalty of cutting through a high-cost region. Using narrow bands also allows us to recover concavities. Indeed, we are able to recover even deep concavities, as can be seen from the results we obtained on real-world data sets. While we cannot guarantee to reach a global minimum (which might not be useful due to the minimal surface bias of the graph-cut), we overcome local minima inside the band, which is sufficient for most practical scenes. This is an advantage over using level sets since by choosing the size of the band we can overcome local minima within the band and achieve a better solution than level sets.

In our implementation, we apply the graph-cuts iteratively until the consistency score of the surface converges. The number of iterations depends on how close the initial estimate approximates the true surface. In most cases, less than 20 iterations are needed to converge to a stable reconstruction.

10.5 Results

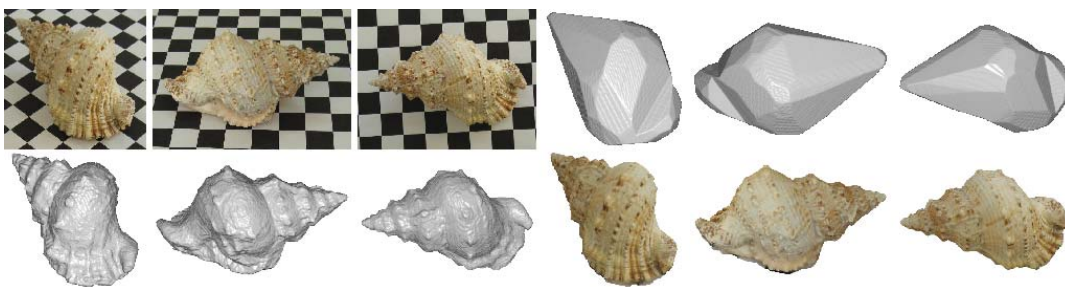
We present the evaluation of our method on two standard data sets and one custom data set that illustrate the advantages of our algorithm. The first data set is the *Gargoyle* set provided by K. Kutulakos. It shows a stone sculpture and contains 16 images with a resolution of 719×485 pixels. The difficulty in this data set is that it contains holes which introduce a fair amount of self-occlusion. The input images were roughly segmented to obtain an initial estimate of the visual hull (see figure 10.3(a)). The resolution used is 200^3 voxels. We used a 5×5 grid to sample the plane used for estimating the voxel score. As in all of our experiments, we allowed a movement of one voxel layer outside and inside the current estimate for every iteration of the graph-cut. Therefore the width of the band is three layers. The width of the band stays unchanged during the optimization. The reconstruction time was 60 minutes. As shown in figure 10.3(a), we obtain an accurate reconstruction of the gargoyle. In particular, we correctly reconstruct protrusive parts of the object like the ears and the nose without the need of an 'ad-hoc' ballooning term. At the same time, we recover the concavities of the object, for instance around the eyes and in particular in the region between its body and the stick it is holding. The reconstruction converges after 19 iterations, while most of the surface details are already recovered after 10 iterations. Only the deep concavity between the belly and the stick requires more carving cycles. This shows that our method is able to recover even deep concavities. Our reconstruction is of at least the same quality as the one presented in [BL06] where the photoflux is used to preserve thin structures.



(a) *Gargoyle*



(b) *DinoSparseRing*



(c) *Shell*

Figure 10.3: Reconstruction results. The first row of each result shows three input images and three views of the visual hull which is used as a starting point for the reconstruction. The second row shows the untextured and the textured reconstruction results.

The second data set we used our method on is the *DinoSparseRing* data set provided by [SCD⁺06]. This set contains 16 images of a plaster dinosaur model with a resolution of 640×480 pixels. The reconstruction time was 62 minutes. The particular difficulty of this data set is the lack of texture. We used a resolution of $200 \times 234 \times 200$ and a 5×5 grid for this reconstruction. The algorithm converged after 21 iterations and recovered most of the surface details as seen in figure 10.3(b). The concavities between the legs and the scales on the back were both recovered. The surface on the back of the model is fairly smooth while the surface on the front is a little rough due to the very uniform intensity. When compared to the reconstruction result of [VTC05] on the same data set, we can see a clear improvement especially around the tail area and around the legs. Compared to [TD06], we obtain a more accurate surface estimate with more details especially around the legs. Our reconstruction achieved an accuracy of 0.89mm (defined as the distance that brings 90% of the reconstruction result within the ground-truth surface) and a completeness of 95.0% (defined as the percentage of the ground-truth that lies within 1.25mm of the reconstruction results) in the Middlebury multi-view evaluation. These results reflect that our method provides accurate reconstructions which surpass the results of [VTC05] (using a ballooning term) (1.18mm/90.8%) and [TD06] (using silhouette constraints) (1.26mm/89.3%) in both accuracy and completeness.

The third data set consists of 24 images of a shell at a resolution of 1600×1200 . The reconstruction time was 112 minutes. Since the shell has many small protrusions and concave regions, it is a perfect test object to show the properties of our reconstruction method. We roughly segmented the shell by drawing a bounding polygon around the object. We used a voxel grid of size $415 \times 276 \times 200$ and sampled the tangent plane with an 11×11 grid due to the bigger image resolution. The reconstruction converged after 35 iterations because we started far from the true surface. This also shows that the method is not critically dependent on the initial surface estimate. Figure 10.3(c) shows our reconstruction results. We correctly recover the concavities and the small protrusions on the shell. The reconstruction also recovers the undulations on the base of the shell. This shows that our method can successfully preserve protrusions and at the same time recover concavities without relying on a ballooning term or on exact silhouette images.

10.6 Conclusion

We presented a novel iterative reconstruction algorithm designed to overcome problems resulting from the graph-cut inherent bias for shorter cuts. At each iteration, we optimize the surface normals of the current surface and apply a volumetric graph-cut over narrow bands around the current surface estimate.

Experimental results obtained on standard and custom data sets show that our method preserves protrusions and at the same time recovers concavities. We applied the proposed algorithm on ground-truth data and compared the obtained results with the results of existing volumetric graph-cut-based methods that rely on a global ballooning term [VTC05] and on silhouette constraints [TD06]. We obtained a reconstruction with higher accuracy and completeness.

Chapter 11

Spectral Camera Clustering

Having looked at obtaining high-quality reconstructions from moderately sized image sets, we now want to address the problem of reconstructing large image collections efficiently since it is not possible to directly use traditional reconstruction methods such as the one presented in the previous chapter on large image collections due to memory and run time constraints. We therefore propose an algorithm for clustering large sets of images of a scene into smaller subsets covering different parts of the scene suitable for 3D reconstruction. Unlike the canonical view selection of [SSS07], we do not focus only on the visibility information, but introduce an alternative similarity measure which takes into account the relative camera orientations and their distance from the scene. This allows us to formalize the clustering problem as a graph partitioning and solve it using spectral clustering. The obtained image clusters bring down the amount of data that has to be considered by the reconstruction algorithms simultaneously, thereby allowing traditional algorithms to take advantage of large multi-view data sets processing them significantly faster and at smaller memory costs compared to using the full image data sets. We tested our approach on a number of multi-view data sets and demonstrated that the clustering we obtain is suitable for 3D reconstruction and coincides with what a human observer would consider as a good clustering.

11.1 Introduction

The availability of high quality digital cameras at reasonable prices allows an easy acquisition of large numbers of high-quality images. This motivated Computer Vision researchers to intensify their research on 3D reconstruction problems, proposing new solutions which exploit those rich sources of information. The recent advances in static 3D reconstructions from calibrated cameras exploit the spatial redundancy among the images and produce high-quality reconstruction results [SCD⁺06, FP07, PKF07, HS04, HK06b]. This usually requires using all available images of the objects of interest at once. This is reasonable when a limited number of images is used, but becomes prohibitive when the number of available images is large and they are high-resolution. This is especially the case with large scenes such as those of figure 11.5. For this reason breaking the collection of images into meaningful clusters, ideally covering different parts of the object, has to be addressed. The community has already addressed the problem of handling huge collections of images available on the

Internet [SSS06]. This required addressing several problems such as structure from motion [SSS08a], summarizing images by finding canonical views [SSS07] or finding paths through the photos and turning them into controls for image based rendering [SSS08b]. Unlike these approaches, and similar to [ZCI⁺08], our objective is to cluster a collection of images of a particular object into smaller image subsets suitable for 3D reconstruction as shown in figure 11.1. In contrast to [SSS07] and [ZCI⁺08], we do not focus only on the visibility of the scene in the cameras, but introduce an alternative similarity measure which takes into account the relative camera orientations and their distance from the scene. This measure prefers camera configurations suitable for 3D reconstruction, i.e. those with a reasonable baseline and similar scale. We represent the collection of images as a fully connected graph with the nodes corresponding to the cameras and the edges between them corresponding to the introduced similarity measure. This formulation of the problem allows us to define the clustering problem as a graph partitioning and solve it using spectral clustering. The clusters we obtain bring down the amount of data that has to be considered by the reconstruction algorithms simultaneously. We tested our approach on a number of multi-view data sets and demonstrated that the clusters we obtain are suitable for 3D reconstruction and coincide with what a human observer would consider as a good clustering. We also show that the total reconstruction time spent when using the camera clusters to reconstruct the scene is significantly smaller than the time spent when using all images at once. In addition, it is obvious that the memory requirements are much smaller when clusters of cameras are used, compared to using the full image dataset with all cameras.

The remainder of this chapter is structured as follows: In section 11.2 we discuss previous work dealing with camera clustering. Section 11.3 introduces our proposed clustering approach, while results on clustering in general and multi-view reconstruction in particular are presented in section 11.4. We conclude with section 11.5.

11.2 Related Work

The problem of processing large sets of images for the purpose of 3D reconstruction has in particular been brought to the attention of the community through the Photo Tourism project [SSS06]. This led to improved solutions for many related problems, such as structure from motion from unorganized image collections [SSS08a], scene navigation [SSS08b], object segmentation in the scene [SS08] and scene summarization [SSS07]. In all those problems camera clustering has been considered. The SFM problem of [SSS08a] concentrates on reducing a full set of available images to a skeletal subset which is sufficient to provide a full sparse reconstruction, where other image views can be quickly added using pose estimation. The skeletal image set is computed by a maximum leaf t -spanner [ADD⁺93] of the graph with cameras at its nodes and the edges being joint position covariances between the pairs of cameras. In the other works, especially the one of Simon *et al.* [SSS07], the goal is typically not finding a good partitioning of the cameras, but instead the selection of canonical views. These views are desired to be orthogonal to each other, while at the same time showing a representative view of the object. The proposed approach of [SSS07] was used for scene summarization through a set of orthogonal canonical views. They use point visibility information in an ad-hoc greedy method to automatically find their canonical views. While this seems to give good results they only make use of the point visibility, disregarding

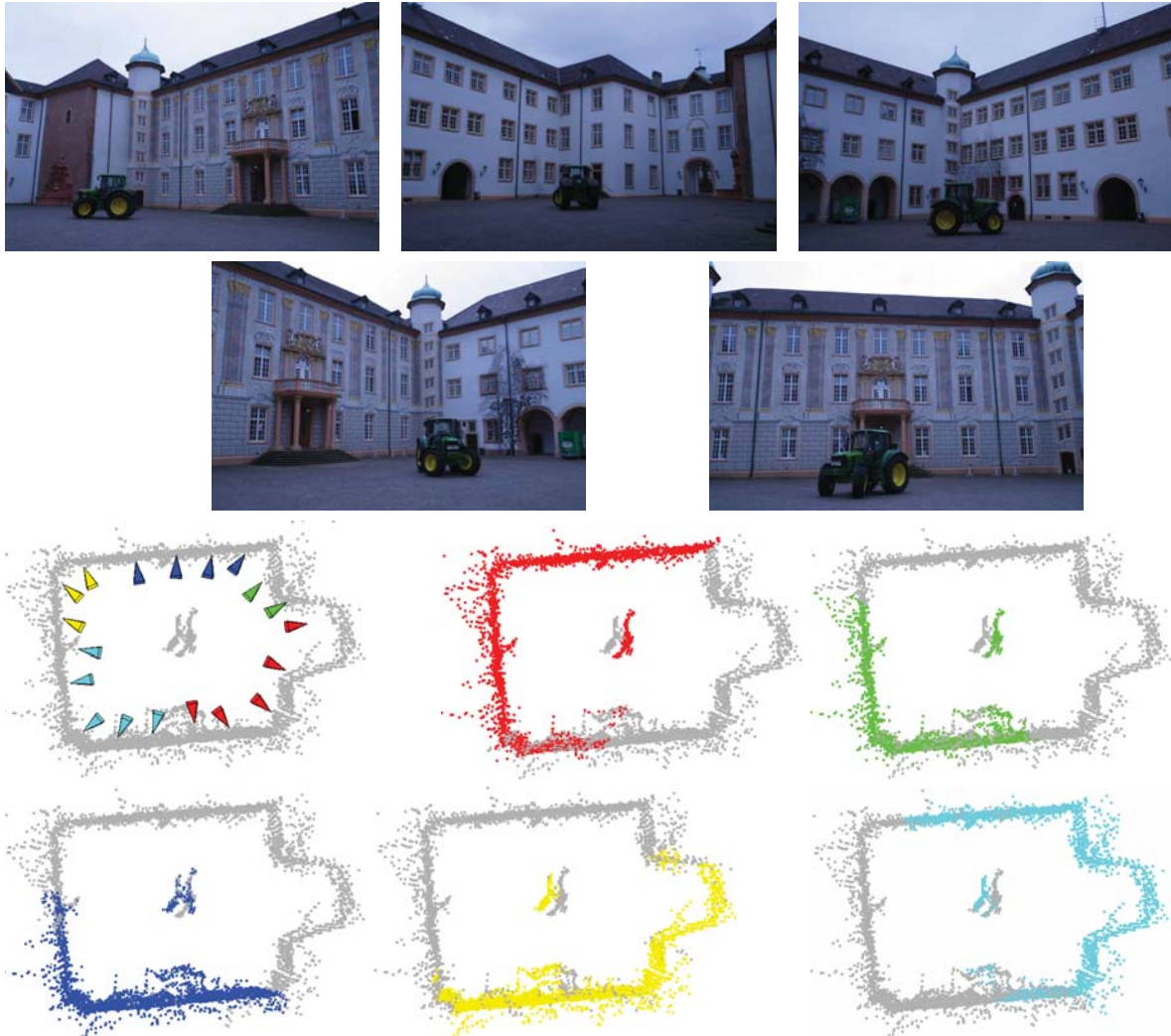


Figure 11.1: Clustering on the *Castle* sequence. The first two rows contain images showing representative views for each cluster. In the first image of the third row the camera clusters found by our algorithm are depicted. The following images show the object points visible in each cluster.

information about camera configurations. In our implementation of their greedy algorithm the number of clusters is highly dependent on the choice of some weighting parameters proposed also in the original algorithm. In [DDAS04] Denton *et al.* propose a method to find canonical views for a set of silhouette images of an object, in order to use them for view-based 3D object recognition. To solve the problem the authors use semidefinite programming (SDP) on a graph which models the similarity between the views. In the context of robot localization in [BZK06], the environment is represented by a large number of collected images stored in a database. To speed up the search of the most similar image, the database is reduced to the set of representative canonical views. This is done using a graph representation of the database images with weights measuring image similarity. The problem is solved

using the graph pruning technique known as the Connected Dominant Set problem in graph theory. While we also use a graph for representing the relation between different views, we are looking for a partitioning of the graph and not for canonical views. In addition we use spectral clustering [vL07, SM00] based on spectral graph theory, which is significantly easier to implement than related approaches and yields good results. Zaharescu *et al.* [ZCI⁺08] also consider the topic of camera clustering. Similar to our approach they directly find the clusters. However, they only use visibility information and apply the k-means algorithm for clustering requiring advance knowledge about the number of clusters. Hornung *et al.* [HZK08] also select images for multi-view reconstruction, but their focus is on removing less useful views instead of clustering them. To be more precise they add views maximally contributing to the quality of the initial rough reconstruction.

Compared to previous work, we perform camera clustering based on geometric information about both the scene and the camera positions. We do not need to know the number of clusters in advance and only need a rough proxy geometry, which can be obtained either by a sparse bundle adjustment or the use of the visual hull computed from camera images. Especially in the context of multi-view reconstruction silhouettes are typically available, making our algorithm easy to integrate into an existing reconstruction pipeline.

11.3 Camera Clustering

We represent the camera clustering problem as a graph partitioning problem by modeling the relationship between the cameras as a graph. This allows us to apply powerful methods from graph theory to find a solution to our problem. Each vertex in the graph represents a camera, while the edges connecting two vertices represent the similarity between these two views. Hence, the first step in our algorithm is to define an appropriate similarity measure. While some existing clustering algorithms only use point visibility, we also incorporate the

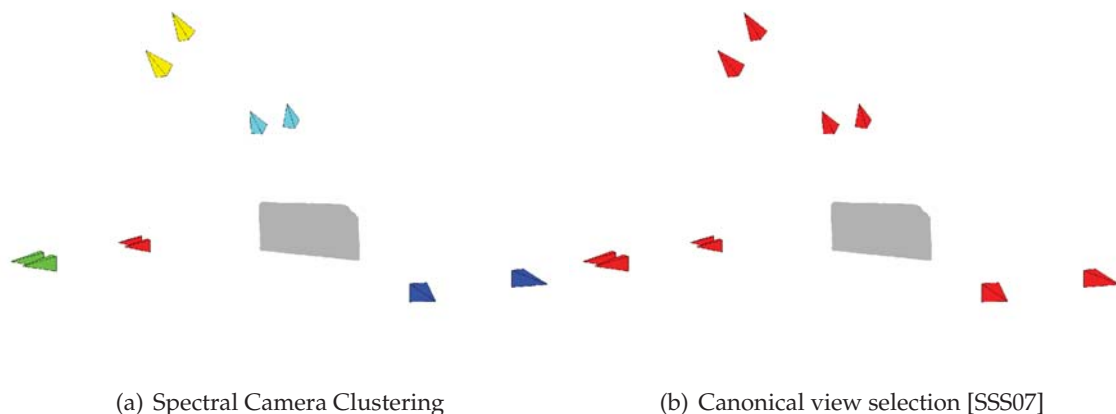


Figure 11.2: Comparison of the clustering given by our method (left) and the one given by the canonical views algorithm [SSS07] (right). The canonical views fail in this case, since the scene is planar and all cameras see the same points. Our method takes the positions of the cameras into account and therefore performs a more reasonable clustering.

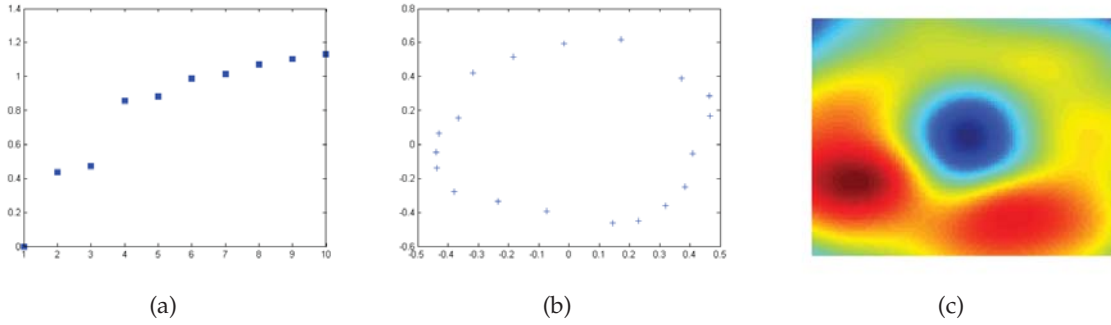


Figure 11.3: Eigenvalues and the clustering in the feature space of the *Castle* sequence of figure 11.1. (a) The 10 smallest eigenvalues. Note the jump between eigenvalues λ_3 and λ_4 indicating that only two eigenvectors are needed for classification. (b) Plot of the entries of the eigenvectors corresponding to the two smallest eigenvalues (excluding the one being zero). (c) Same as (b) but overlaid with the Gaussian kernel used in mean-shift. Note that the layout of (b) resembles the camera positions, and that the maxima of (c) represent the clusters obtained by the mean-shift algorithm.

viewing angle between camera pairs and the distance to the scene to obtain more meaningful clusters. This is demonstrated in figure 11.2 where we compare the performance of our method to that of the canonical views algorithm [SSS07], which we extended to provide clusters instead of only canonical views by assigning each view to its most similar canonical view. Since the scene is mostly planar and every camera can see it, we only obtain one cluster using the canonical views. The cameras are spatially quite well clustered, but due to the use of the non-discriminate visibility information the algorithm cannot take advantage of this. Our method on the other hand successfully finds the clusters, since it also takes the camera geometry into account.

Before we introduce our similarity measure, let us first define some notation. For each camera we have its projection matrix P_i , a set of surface points X and the visibility matrix V , which collects the information about the visibility of X in each view, i.e. the row-vector V_i contains the visibility of the scene points with respect to camera i . In addition we also use V_i to represent the set of points which are visible in camera i . The 3D scene points X can be obtained in one of two ways. The first possibility is to run a bundle-adjustment procedure (e.g. [FP07] or [SSS06]). However, this can be time consuming, especially for large scenes. In order to speed this process up and since we only need a rough object proxy geometry we downsample the images and use a sparse sampling (as it is possible in the Furukawa method). The second possibility is to make use of silhouette images when available by building the object's visual hull and then using the mesh vertices for X . Once this information has been obtained we can compute our scene similarity measure. We want to be able to take into account both scene geometry and camera configuration. This is achieved by using the visibility information (scene geometry) and the viewing angles and scene distances for each camera-point pair (camera geometry). Our similarity measure between view i and j is given by:

$$S_{ij} = \frac{\|V_i \cap V_j\|}{\sum_{X \in (V_i \cap V_j)} \alpha S_{angle} + \beta S_{distance}}$$

$$S_{angle} = \text{abs}(\text{abs}(\arccos(\frac{(C_i - X)^\top (C_j - X)}{\|C_i - X\| \|C_j - X\|})) - \gamma)$$

$$S_{distance} = \text{abs}(\|C_i - X\| - \|C_j - X\|)$$

where C_i and C_j are the camera centers of view i and j respectively. The term $\|V_i \cap V_j\|$, computed as the dot product between rows i and j of the visibility matrix, gives the number of scene points visible in both cameras i and j at the same time. The parameters α and β are weights to control the influence of the angular and the distance term (both α and β are set to 1 in our experiments). The angular term is representing the angle between the viewing directions of the two cameras C_i and C_j with respect to the point X . A small value means that less perspective distortion will occur between the points seen in the two cameras. This in turn will allow a more accurate and stable reconstruction. However, a too small angle is not desired since then an accurate reconstruction becomes problematic due to the small baseline. We therefore include the threshold γ . Angles smaller than γ will be penalized. The distance term is meant to help with scene parts seen at different scales. Since many image-based similarity measures are not very robust with respect to scale changes, we try to group the cameras in such a way, that their distances to the scene are similar. This avoids unnecessary resampling of the images leading to more accurate reconstruction results. The nominator normalizes the score with respect to the number of common points visible in both cameras, in order to avoid any bias towards camera pairs which have a lot of points in common. This is desirable, since the proxy geometry we use might be more densely sampled (i.e. contain more points) in certain scene regions. By normalizing the similarity measure we are independent of the sampling density in the proxy geometry.

Our goal is to find the optimal partitioning of the graph, so that the nodes of the graph, i.e. the cameras, are separated in different groups according to their similarity. In other words, we want to find the partitioning of the graph such that the edges between the different graph parts have very small weight (similarity), and the edges within the same part have high weight, i.e. the views are very similar. This problem is NP-hard, but there exist approximative algorithms based on spectral graph theory, such as the normalized cuts [SM00]. It can be shown that the graph partitioning can be computed based on the solution to the generalized eigenvalue problem of the graph Laplacian [vL07]. Let W be the symmetric weighting matrix describing the edge weights, i.e. $W_{ij} = S_{ij}$, and D a diagonal matrix with $D_{ii} = \sum_{j=1}^N W_{ij}$. The Graph Laplacian is then defined as $L = D - W$. We solve the generalized eigenvalue problem $Lv = \lambda Dv$ and take the eigenvectors u_1, u_2, \dots, u_k associated to the k smallest eigenvalues $\lambda_1, \lambda_2, \dots, \lambda_k$. Typically k is determined by observing all eigenvalues and taking the k smallest of them after which there is a visible jump in the eigenvalues (see

Algorithm 4 Method Outline

- 1: Compute the graph laplacian $L = D - W$.
 - 2: Solve the generalized eigenvalue problem $Lv = \lambda Dv$.
 - 3: Select the eigenvectors corresponding to the k smallest non-zero eigenvalues after which there is a visible jump in the eigenvalues, disregarding the eigenvector corresponding to $\lambda = 0$.
 - 4: Construct the matrix E from the k smallest non-zero eigenvectors and perform a clustering on its rows using the mean-shift algorithm.
-

figure 11.3). This is known as the eigengap heuristic. Note that the smallest eigenvalue is zero and its eigenvector is constant, since every row of L adds up to zero. Therefore, we count from the first non-zero eigenvalue. In practice, we use $k = 2$ or $k = 3$ depending on when the jump in the eigenvalues occurs. We then use the eigenvectors corresponding to those eigenvalues, as shown in figure 11.3 for the sequence of figure 11.1, and perform a clustering on them. As it can be seen this representation resembles the real 3D positions of the cameras and already indicates the existence of separate clusters in this representation. Let E be the matrix containing the eigenvectors corresponding to the k smallest non-zero eigenvalues as its columns. Then, usually, the final step in the literature is to perform k -means clustering on the rows of E . However, this would require us to know the number of clusters in advance. Therefore we chose to adapt the mean-shift algorithm [CM02], which does not need to know the number of clusters in advance. The only parameter we have to choose is the width of the kernel. To choose the best value we applied a heuristic approach, i.e. we choose the kernel width as the mean of the k smallest eigenvalues. This heuristic gives very reasonable results. Our method is summarized in algorithm 4.

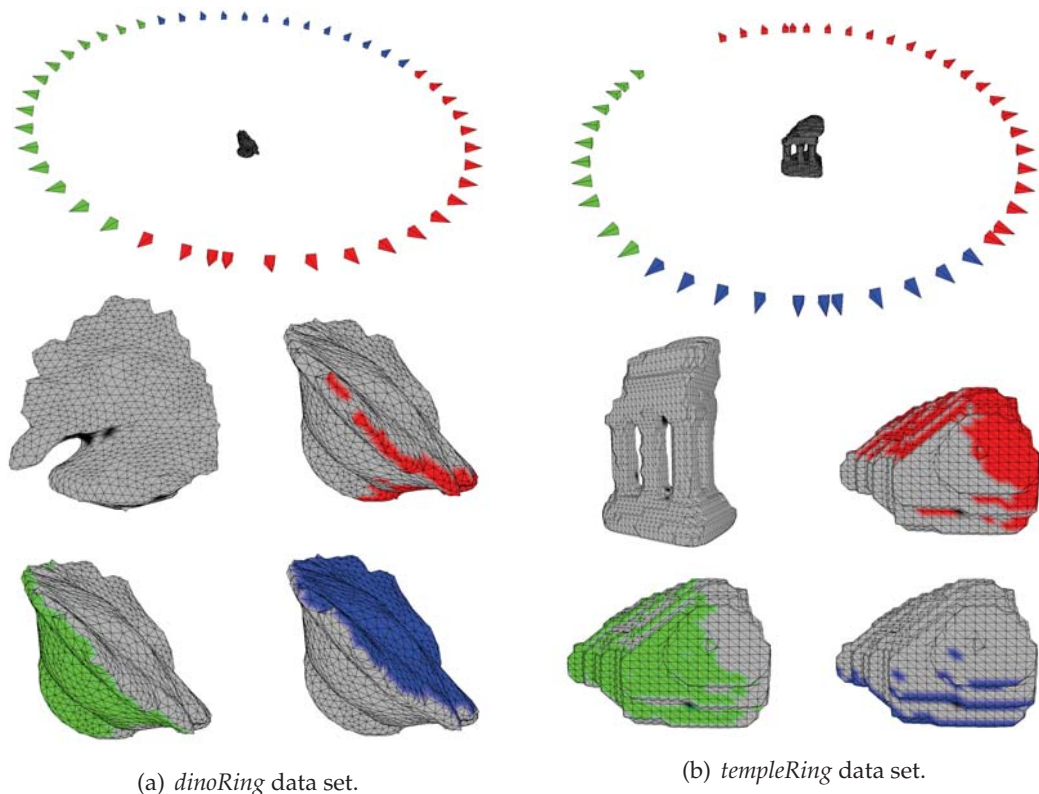


Figure 11.4: Clustering on the *dinoRing* and the *templeRing* dataset from the Middlebury multi view evaluation. In this example we use the visual hull as a proxy geometry.



Figure 11.5: Clustering on the *La Sarraz* dataset. The scene is clustered into 7 separate clusters, which mostly follow the streets. See figure 11.6 for representative views of each cluster and our reconstruction results.

11.4 Results

We performed several experiments on real and synthetic datasets to validate our approach. Since there is no commonly agreed on definition of what constitutes a "good" clustering, we can only evaluate the obtained clusters in terms of their suitability for 3D reconstruction. Therefore, we desire the clusters to be coherent and local with respect to the part of the scene they are viewing. In addition, we check our results by reconstructing some datasets using the obtained clusters and compare the results to a full reconstruction using all the images.

The first dataset we used is the *Castle* sequence provided by Christoph Strecha [SvHG⁺08] consisting of 19 high-resolution images. We obtained a proxy geometry by reconstructing the scene using the method of Furukawa [FP07] after downsampling the images in order to speed up the run-time of the procedure. Our clustering algorithm finds five clusters as shown in figure 11.1. To help assess the quality of the clustering we also colored the scene points. A point is considered to belong to a cluster when it is seen by at least two cameras

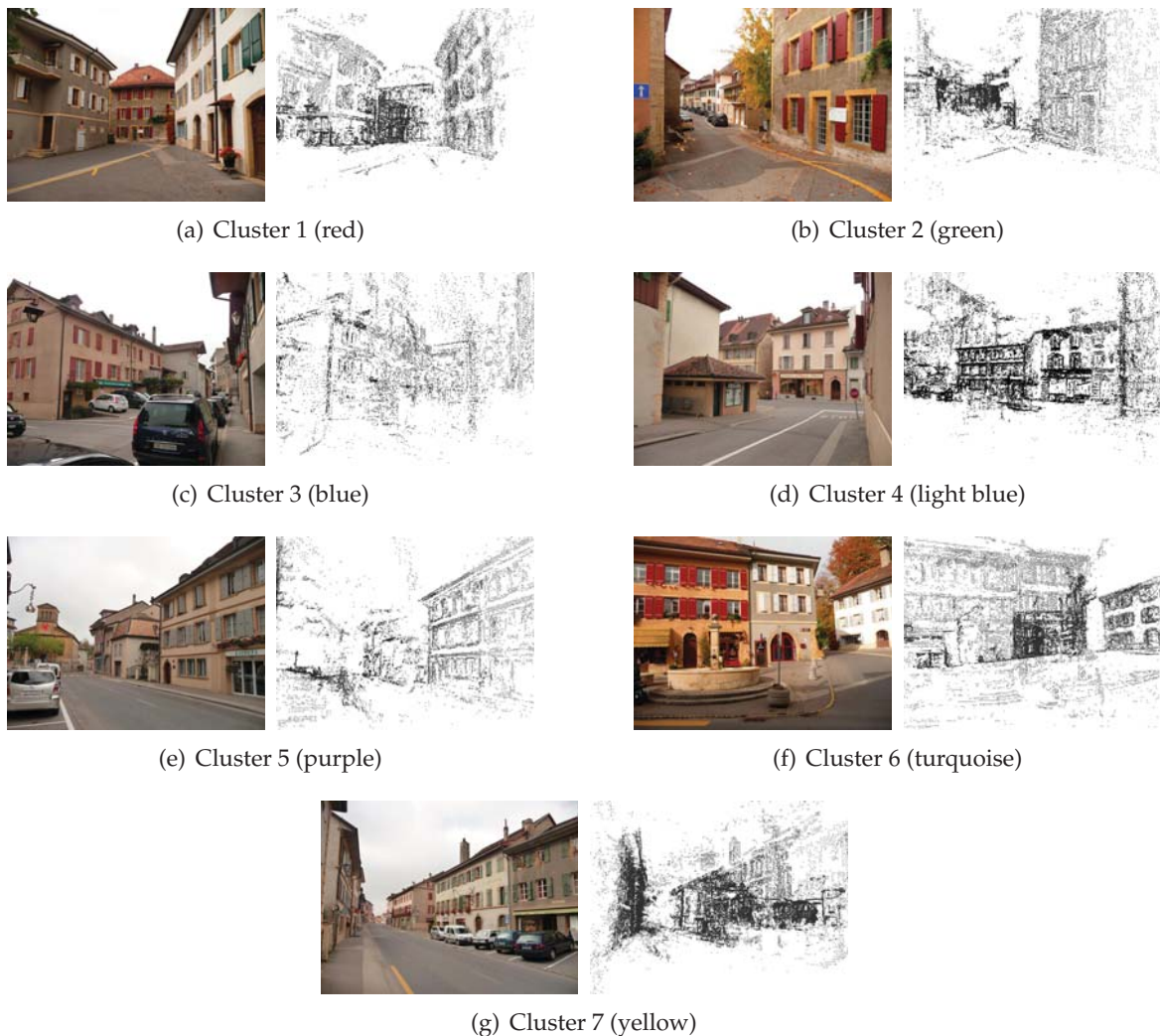


Figure 11.6: Representative images for each cluster in the *La Sarraz* dataset shown in figure 11.5 together with our reconstruction results.

in the cluster. This gives a good visual indication of which parts of the scene can be reconstructed using a cluster. The clusters exhibit a fair amount of overlap in the scene points which is important in order to obtain a good fitting between the reconstructed scene parts.

We also ran tests on the well-known *Dino* and *Temple* sequences from the Middlebury multi-view evaluation [SCD⁺06] as shown in figure 11.4(a). Here we used the visual hull, shown in the images as a proxy geometry. This is significantly faster than performing a multi view stereo reconstruction. For the *dino* we obtain three clusters, one covering each side and one looking at the head of the dino. This makes sense because the head part contains high curvatures and therefore should be reconstructed independently from the sides which are mostly planar. We reconstructed the *dino* model using the Furukawa method [FP07] and obtained a runtime of 59 minutes for the full reconstruction and 42 minutes for reconstructing the three clusters independently. This is a significant increase in performance while a visual compar-

ison of the two reconstruction results shows no significant differences. For the temple we also obtain three clusters, one covering the front, one for the back and one for the narrow side.

To show the validity on large scale scenes, we used a sequence of 125 high-resolution images provided by Christoph Strecha who also provides a multi view stereo reconstruction of the scene which we use as our proxy geometry. The sequence shows streets in the town of La Sarraz in Switzerland, where some streets have been traversed in opposite directions as indicated by the camera orientations in figure 11.5. As you can see from our results in figure 11.5 and figure 11.6 we get a reasonable clustering, which mostly follows the streets. This result would not have been possible to obtain with the canonical views algorithm, since there is no point overlap between the first and the last views of most clusters, so that they would not have been clustered together. Also the running time of the canonical views is prohibitive due to the large amount of points in the scene (50,433 points). After 15 minutes of computation it still hadn't converged to a solution. Our method on the other hand took only a few seconds to cluster the scene. We also performed a reconstruction on this dataset using the method of Furukawa [FP07] as shown in figure 11.6. We reconstructed each cluster we found independently and merged the results. This took a total of 624 min. Reconstructing the full sequence on the other hand took over 25 hours and yielded an incomplete reconstruction with a lot of outliers. We suspect that this is due to the large number of images which lead to a much higher probability of having incorrect matches during the feature matching stage of the algorithm. This clearly shows that the use of good clusters not only speeds up the reconstruction times significantly, but can also help to obtain better results without having to deal with the higher complexity of the reconstruction task at the level of the reconstruction algorithm.

11.5 Conclusion

We presented a method for camera clustering of large image sets into subsets. We modeled the camera clustering as a graph partitioning problem and introduced a new similarity measure which is used to weight the edges in the graph. It is based on the relative orientations between the cameras and their distance to the scene, and naturally favors configurations with a short baseline and similar distance from the scene. These properties are important for obtaining a good 3D reconstruction. We then use spectral clustering to partition our graph into suitable camera clusters. We showed that our algorithm provides reasonable results and runs very fast on large data sets.

Chapter 12

Region Graphs

In the previous chapter we considered the topic of reconstructing large image collections. In doing this we assumed that all images show the scene and are relevant for the reconstruction task. This is however not the case when collecting images from Internet photo sharing sites like Flickr since a simple keyword search will also return many irrelevant images. We therefore need a way of organizing such image collections in order to find clusters of related images and to remove redundant images. We tackle this problem by organizing image collections into meaningful data structures upon which applications can be build (e.g. navigation or reconstruction). In contrast to structures that only reflect local relationships between pairs of images we propose to account for the information an image brings to a collection with respect to all other images. Our approach builds on abstracting from image domains and focusing on image regions, hereby reducing the influence of outliers and background clutter. We introduce a graph structure based on these regions which encodes the overlap between them. The contribution of an image to a collection is then related to the amount of overlap of its regions with the other images in the collection. We demonstrate our graph based structure with several applications: image set reduction, canonical view selection and image-based navigation. The data sets used in our experiments range from small examples to large image collections with thousands of images.

12.1 Introduction

Dealing with large image collections has recently become a subject of interest in the vision community. It includes such diverse topics as 3D reconstruction [SSS08a, ASS⁺09], canonical view selection [SSS07, LWZ⁺08], image-based navigation [SGSS08] and image retrieval [JDS08, PCI⁺08] among others. While applications in this domain can be very different, a key issue that all must address is how to efficiently organize and handle the available and often redundant data. In image retrieval, for instance, state-of-the-art approaches deal with image datasets containing up to one million images [JDS08] and even in 3D reconstruction applications the sizes of the image sets grow rapidly, reaching up to 150,000 images [ASS⁺09]. Most approaches in this field organize images with graphs where edges relate images that share information and edge weights depend on the application. For instance [SSS08b] uses a graph where the edge weight is based on the covariance of the camera posi-

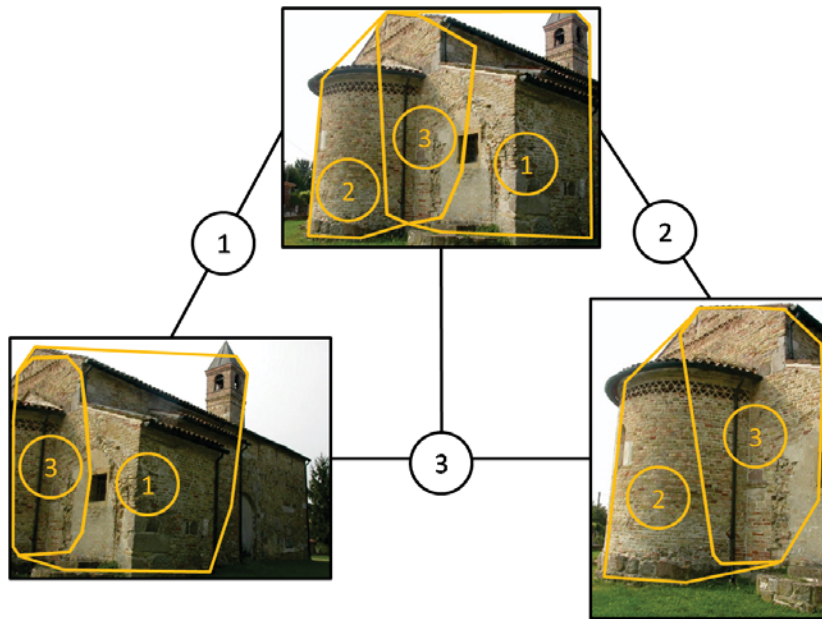


Figure 12.1: Region graph for three images. Overlapping regions are denoted as 1, 2 and 3.

tions, while [LWZ⁺08] weights the edges by the number of inlier matches between images. The resulting data structures reveal little on how informative an image is with respect to all other images. In this work, we take a different strategy and propose a data structure, region graphs, that encodes spatial relationships between an image and a collection of images. This provides a basis upon which various applications can be build, navigation or reconstruction for instance, where not all but only the most informative images are of interest.

The central idea behind our approach is to use image regions and their redundancies over an image set to define a global hierarchy in the set. More precisely, we consider the overlap between images, where we consider the overlap to be the image regions which contain the same part of the scene. This is a natural criterion, based on objective evidence, that does not require any information about the 3D structure of the scene. It also adapts to the sampling of the scene given by the images. The overlapping regions are then used to build a graph relating all images spatially. The graph contains two kinds of nodes, one representing images and the other representing overlapping regions. Each region is connected with an edge to the images it is contained in. This means that region and image nodes are alternating on any given path through the graph. Using this graph we can efficiently represent the spatial relationship between the regions and images and identify redundancies over regions. This allows us to model the importance of regions shared by many images and to identify less important regions shared only by few or even no images. These less important regions are often small and not very essential to the scene. They typically contain background or other irrelevant information.

Figure 12.1 shows an exemplary region graph constructed from a three image data set. There are three image nodes and three region nodes in the graph, representing the images and the distinct overlapping regions, i.e. regions visible in a set of images. Using the region graphs as a basis we build applications for image set reduction, canonical view selection and image-

based navigation. We tested our method on several real data sets ranging from a few dozen to thousands of images. The results obtained show that the data structure we propose reveals intrinsic properties of an image set that are useful for various applications.

In the remainder of the chapter we first discuss the related work in section 12.2. We then proceed to describe the construction of the region graphs in section 12.3 and show some exemplary applications built on them in section 12.4. We present results in section 12.5 and conclude with section 12.6.

12.2 Related Work

In the last years many papers dealing with the issue of large image collections have been published. Most of them focus on specific applications, for instance image retrieval [JDS08, PCI⁺08, WSZ09] or 3D reconstruction [SSS08a, LWZ⁺08, ASS⁺09]. In the 3D reconstruction literature one of the first major works on this topic was the Photo Tourism project [SSS08a]. In that paper a large set of images taken from Internet photo collections is used for performing a point-based 3D reconstruction of the scene. An exhaustive pairwise matching followed by an incremental bundle adjustment phase have been used both for the reduction of the image set and for 3D reconstruction. Follow-up work focused on navigating through large image collections [SGSS08], summarizing the scene by selecting canonical views [SSS07] and speeding up the initial reconstruction process by building skeletal graphs over the image set [SSS08b]. While an image graph was used for instance in [SSS08b] it was designed for the goal of finding a better subset of images for the initial reconstruction. Li *et al.* [LWZ⁺08] presented an application for performing reconstruction and recognition on large image sets. They construct a so called iconic scene graph which relates canonical views of the scene and use it for 3D reconstruction. The edge weights used are the number of inlier matches. Recently Farenzena *et al.* [FFG09] proposed a hierarchical image organization method based on the overlap between images. The overlap is used as an image similarity measure used to assemble the images into a dendrogram. The hierarchy given by the dendrogram is then used for a hierarchical bundle adjustment phase. In this regard that work is interesting, because it also considers a global criterion. However, it is focused on Structure-from-Motion and not on defining global representations of image collections. Schaffalitzky [SZ02] *et al.* also present some work dealing with handling large unordered data sets. They focus on the task of performing a 3D reconstruction from unordered image sets and only briefly mention image navigation, which they base on homographies.

Most existing work organizes images with respect to the application, which is often 3D reconstruction. We follow a different strategy and organize images with respect to the regions they share only. This allows us to score images according to the information they bring and without 3D reconstruction. Subsequent applications can then easily build on the region graph structure, even navigation as shown later in section 12.4. We are not aware of any attempt to build such an intermediate structure based on 2D cues only. We think that these structures will become a key component when dealing with large and highly redundant image datasets.

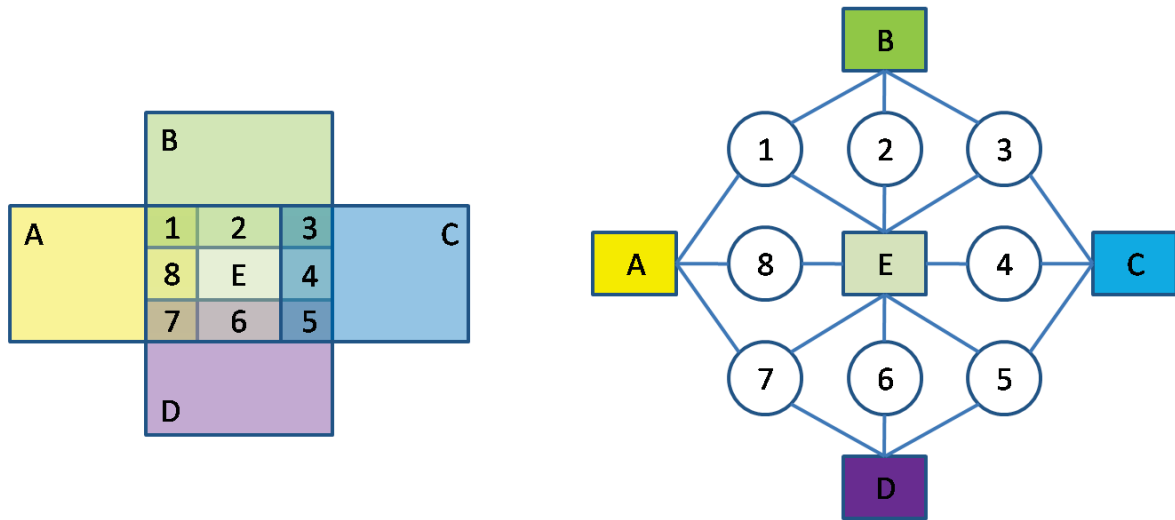


Figure 12.2: Graph construction for a synthetic example containing five images (A to E) which create 8 different overlap regions.

12.3 Building Region Graphs

In this section we describe how to construct region graphs. The most important construction principle is to identify overlapping regions in the images. Overlapping regions are regions in different images showing the same part of the scene. Figure 12.2 gives an example. For instance region 1 is an overlapping region shared by images A, B and E. To identify this overlapping region, the intersection of the overlap between image A and E and the overlap between image B and E has to be computed. Each overlapping region is represented as a *region node* in the graph. The images are represented in the graph as *image nodes*. Each region node is connected to the images in which it is detected. In the example of figure 12.2 this means that node 1 representing region 1 is connected to the nodes of images A, B and E. In the following sections the graph construction process is described in more detail. The construction process is summarized in algorithm 5.

12.3.1 Identifying Overlap Between Images

The first step in the graph construction is to identify the overlap between the images. This is accomplished in a multi-step process. First we extract features using a scale-invariant interest-point detector on all input images [Low04]. We then match the features among the images. Since we are dealing with very large image sets, performing an exhaustive pairwise matching is computationally infeasible. Therefore we use vocabulary trees [NS06] to perform a preselection among the images (in our experiments we use the implementation provided by [FWFP08]). For every image we retrieve the k (we use $k = 10$ in all our experiments) most similar images using the vocabulary tree.

This preprocessing step significantly reduces the size of the set of image pairs which have to be matched. The matching is performed using the standard SIFT distance ratio on the descriptors and the resulting putative matches are pruned using epipolar constraints in a

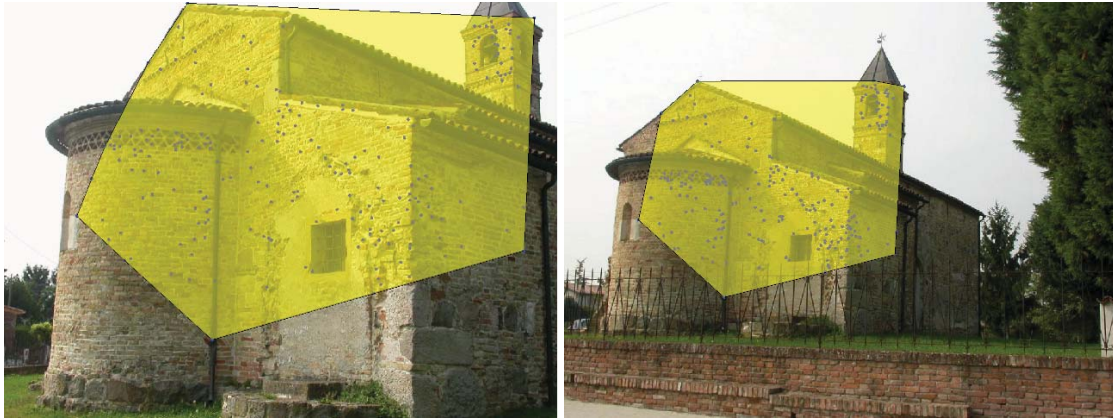


Figure 12.3: The convex hull of the set of matched features between two images defines the regions considered during graph construction.

RANSAC framework. Given the feature correspondences between two images we compute the convex hull spanned by the matched features in each image. This is illustrated in figure 12.3. The area enclosed by the convex hull in each image is the overlap between the two images.

12.3.2 Identifying Overlapping Regions

After performing the matching, we generally obtain several different convex hulls per image, one per matched image. In general these convex hulls will overlap with each other. We want to identify each overlapping region created by the intersection of these convex hulls. In the following let CH_i^j be the convex hull spanned in image i by the features matching image j . To determine unique overlapping regions we assign each CH_i^j a label (i, j) to indicate that this region is shared by images i and j . When two regions CH_i^k and CH_i^l overlap, the common region will receive the label $L = (i, k, l)$. After performing this labeling for all convex hulls every intersection will have an associated label L . The image is then subdivided into regions sharing the same label. While it is possible to perform these computations directly on the image by discretizing the convex hulls, we chose to perform the computations purely geometrically by representing the convex hulls as polygons and using CGAL to perform the intersection operations. This has the advantage of being image resolution independent and does not require to allocate a discretization space for every image, which would be very memory intensive for large image data sets. Finally every identified region is merged into a region list storing its label and the images in which it was detected.

12.3.3 Constructing the Region Graph

After all overlapping regions have been identified, the region list contains all the information needed to build the region graph. It is constructed by inserting one image node per image and one region node for every entry in the region list. The region nodes are subsequently connected to the image nodes specified in the region list. The weight of the edges connecting the region nodes to the image nodes is application specific. One generic choice is to assign

Algorithm 5 Graph Construction

- 1: Extract feature points on all images I
 - 2: Use a vocabulary tree to select the k most similar images for each image
 - 3: Perform robust matching
 - 4: **for** each image i in I **do**
 - 5: **for** each image j matched to i **do**
 - 6: Compute the convex hull CH_i^j and assign it the label (i, j)
 - 7: **end for**
 - 8: Intersect the convex hulls in image i to obtain the overlapping regions
 - 9: Add overlapping regions into region list
 - 10: **end for**
 - 11: **for** each image i in I **do**
 - 12: Create an image node in the graph
 - 13: **end for**
 - 14: **for** each region entry l in the region list **do**
 - 15: Create a region node and connect it to the image nodes of the images in which it was detected
 - 16: Set the weight of the outgoing edges according to the application criteria, e.g. the normalized size of the region
 - 17: **end for**
-

the normalized size of the region, defined as the size of the overlapping region divided by the image size, as an edge weight. This is the edge weight which is used in most of our experiments.

12.4 Using Region Graphs

In this section we discuss several applications based on the proposed region graphs. The first application is image-based navigation which allows the user to traverse the image set in a spatially consistent way. The second application is image set reduction. Its goal is to reduce the size of the data set while retaining as much information as possible. The final application we are considering is canonical view selection. In this application we want to find a small orthogonal subset of images which summarizes the whole image set.

12.4.1 Image Set Reduction

The goal of image set reduction is to remove redundant and non-contributing images from the data set. In [SSS08b] for instance a subset of an image set is selected for performing a 3D reconstruction. However, the graph structure and edge-measure were application specific and based on the covariance of the camera positions. We would like to define a more general measure for the information content of an image. Intuitively an image which contains many regions shared with other images is more important for the data set than an image having little overlap with the other images in the data set. We therefore formalize an information

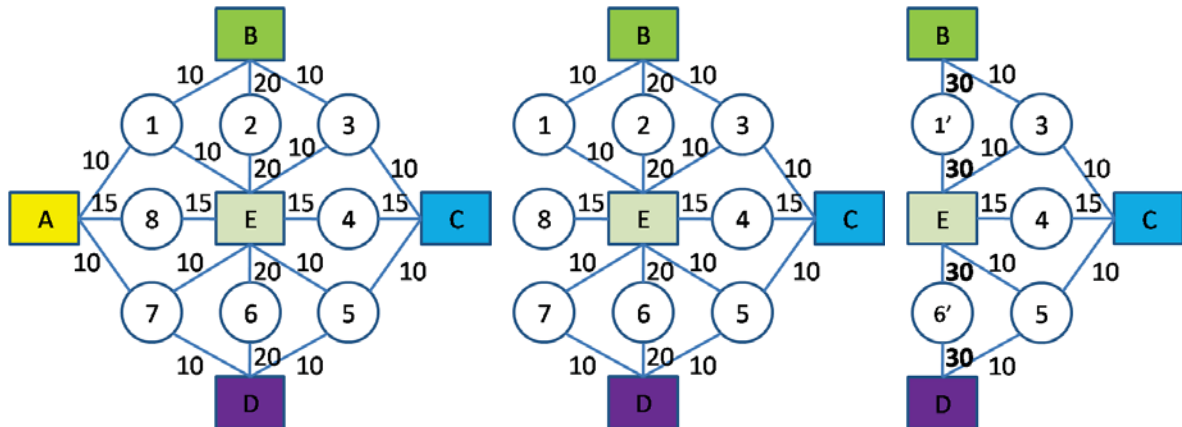


Figure 12.4: Removal process on the synthetic example given in figure 12.2. The image with the least score among all images is removed first (left). Leaf nodes created by the removal of the image are removed (middle). Newly created duplicate paths are joined (right).

criterion for an image i and its associated image node v_i in the region graph as

$$\rho(v_i) = \sum_{r \in N(v_i)} \sum_{e \in E(r)} w(e) \quad (12.1)$$

where $N(v_i)$ is the set of neighbor regions of image node v_i , $E(r)$ is the set of edges in the region graph connected to node r and $w(e)$ is the weight of edge e . The intuition behind this information criterion is that an image which contains many regions which are also present in many other images is more important than an image which only contains few regions shared with few other images. The choice of which images to remove is directly related to this criterion. At each step of the removal process the image with the smallest image score is removed.

In figure 12.4 we give an example of the image removal process in the graph. Once the image to be removed has been identified, its corresponding node and all incident edges are removed from the graph. The resulting graph might then contain leaf nodes (node 8 in the example) which are also removed. Due to the removal of an image it can also happen that two previously distinct regions collapse into one. This can be seen in the graph through the existence of several identical paths between two image nodes (paths $E \rightarrow 1 \rightarrow B$ and $E \rightarrow 2 \rightarrow B$ in figure 12.4 (middle)). These paths are joined to obtain a region node representing the new region. All these computations can purely be based on the graph. No recomputations are needed. This is due to the explicit representation of regions in the graph. If only images were represented in the graph it would have to be recomputed after every image removal.

12.4.2 Canonical Views

Canonical views are views which are of high importance in a given image set. They show parts of the scene which are captured in many images (e.g. because they are considered to be very important). We want to automatically find these important parts of the scene and

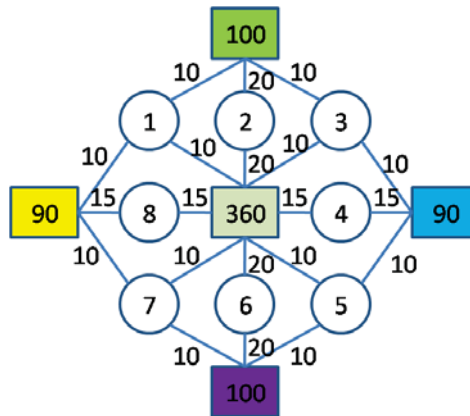


Figure 12.5: Canonical view selection on the synthetic example given in figure 12.2. The numbers inside the image nodes indicate the image score computed according to equation 12.1. The central image has a maximum score in its neighborhood and is therefore selected as the canonical view.

select one representative view, i.e. the canonical view, for each of them. Some previous work on this subject was done in [SSS07]. In that work the criterion for selecting a canonical view was based on the visibility of the points in the scene. A canonical view was defined to be an image which is very different from all other canonical views in terms of the scene points it observes. This criterion was optimized by a greedy approach. We have a similar definition of canonical views. However, we do not assume any explicit visibility information to be available. We also do not perform a greedy optimization, but instead deduce the canonical views directly from the region graphs.

Intuitively the images having the highest amount of overlap with the image set should be selected as canonical views. However, we would like to avoid selecting multiple images of the same part of the scene. One natural way of including this constraint is to find maxima over the graph. Each image node v_i is assigned a weight using the score function given in equation 12.1. Only the nodes which have a score bigger than all their neighboring image nodes are selected. These nodes then constitute the canonical views. The neighboring image nodes are defined to be all the image nodes which are connected to the node under consideration over exactly one region node, i.e. two images are considered to be neighbors in the graph when they share a common region. Figure 12.5 gives an example.

12.4.3 Image-based Navigation

The goal of image-based navigation is to allow the user to traverse the image set in a spatially consistent order. For instance the user can choose to view the image to the right or to the left of the current image. In order to allow such a navigation the spatial relationship among the images has to be determined. While some prior work [SGSS08] assumes the availability of a 3D scene reconstruction we base the navigation purely on the images. This is achieved by considering the spatial positions of matching regions in the images. To represent this information in the graph we augment the edges with information about the spatial relationship of the associated nodes. In practice we assign each edge in the region

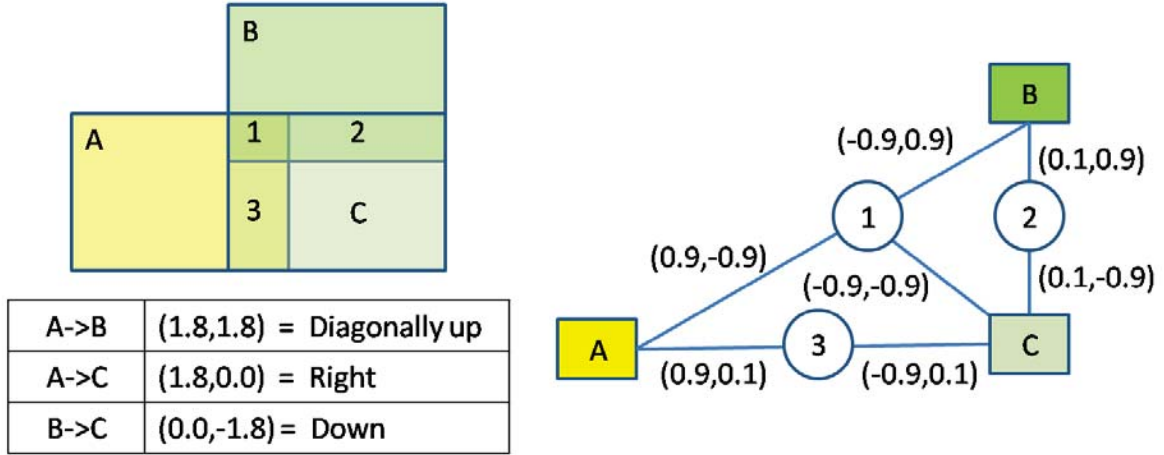


Figure 12.6: Illustration of image navigation. The region graph is augmented with the relative positions of the regions within the image. For determining the relative motion between two images all shared regions are considered and their relative motions are averaged. The resulting relative motions between the images are shown in the table. Note that for clarity scale is not considered in this example.

graph a three-dimensional vector $[x \ y \ z]^\top$ which describes the relative position and scale of the region within the image. The position inside the image is specified with respect to the image center and normalized to the range $[-1; 1] \times [-1; 1]$. The first two components of the vector describe the horizontal and vertical position, while the third one represents the scale. They are computed by considering the position of the center of gravity of the convex hull in the image. Let $\mathbf{g}_i = [g_i^x \ g_i^y]^\top$ represent the center of gravity of the region in image i and let w_i and h_i be the size of the image in pixels. Then the relative position of the convex hull inside the image is given by

$$\mathbf{p}_i = \begin{pmatrix} \frac{2g_i^x - w_i}{w_i} \\ \frac{2g_i^y - h_i}{h_i} \end{pmatrix} \quad (12.2)$$

To represent the scale we consider the relative area of the region with respect to the image area. This makes us independent of the image resolution. Let A_i be the number of pixels in the convex hull and I_i the total number of pixels in image i . Then the scale is given by

$$s_i = \frac{A_i}{I_i} \quad (12.3)$$

The region movement (position and scale) for a region shared by images i and j is computed as

$$x_{i \rightarrow j} = -(p_j^x - p_i^x) \quad (12.4)$$

$$y_{i \rightarrow j} = p_j^y - p_i^y \quad (12.5)$$

$$z_{i \rightarrow j} = s_j - s_i \quad (12.6)$$

To navigate the user specifies a spatial movement in the image plane (two dimensions) and a zoom-in/zoom-out movement (one dimension). This results in the desired movement vector. To find the next image to move to, the movement between the current and all neighboring images is computed. Given two images the relative movement is given by the average of

the region movement of the regions shared by the images. The image whose region movement agrees most with the user motion (in the sense of the dot-product) is then displayed to the user. Figure 12.6 gives an example of how the relative movement between images is computed using the shared regions. Since we explicitly represent the regions in our graph it would also be possible for the user to select a specific region of interest inside the image and to perform the navigation with respect to this region instead of the whole image.

12.5 Experimental Results

To validate our approach we performed experiments on several data sets of different sizes. In the following we will first briefly describe each data set used and then show results for the different applications we are proposing. The first two data sets we used were provided by [FFG09]. The *pozzoveggiani* data set contains 54 images of a church and the *piazzaerbe* data set contains 259 images of a big town square. The other data set we used was the *Notre Dame* data set provided by [LWZ⁺08]. It contains 6,248 images of the Notre Dame cathedral in Paris collected from Flickr.

The first step common to all application is the construction of the region graph. The construction times (excluding feature extraction and matching) were 1 minute for *pozzoveggiani*, 3 minutes for *piazzaerbe* and 38 minutes for *Notre Dame* on a 2.66 GHz Intel QuadCore CPU (only one core was used). Most of the time was spent on intersecting the convex hulls.

12.5.1 Image Set Reduction

To show the validity of the reduction we first perform a 3D reconstruction with the full data set and then compare it to a reconstruction on the reduced data set. Figure 12.7 shows the results for the *pozzoveggiani* data set. The first row shows two views of the reconstruction obtained on the full data set, while the next two rows show the results obtained after removing 12 and 24 images respectively. While the point cloud does get sparser the whole structure is still present.

Figure 12.8 shows the results we obtained on the *Notre Dame* data set. We computed the connected components of the region graph and used the biggest one (907 images). The first image shows the full reconstruction. Each of the following reconstructions was obtained by removing 150 images from the previous one. Again the point cloud gets sparser, but the overall structure of the scene is retained.

12.5.2 Canonical Views

The results of the canonical view selection on the *pozzoveggiani* data set are shown in figure 12.9. One view is selected for each side of the church. To compare to previous work we implemented the canonical view selection method described by Simon *et al.* [SSS07]. The results of their method are shown in figure 12.10. They are comparable to ours. The first four canonical views are virtually identical, while the last two are not very essential to the scene. Since Simon's method uses two tuning parameters, it was necessary to manually adjust them until a reasonable result was obtained. Our method on the other hand is parameter free.

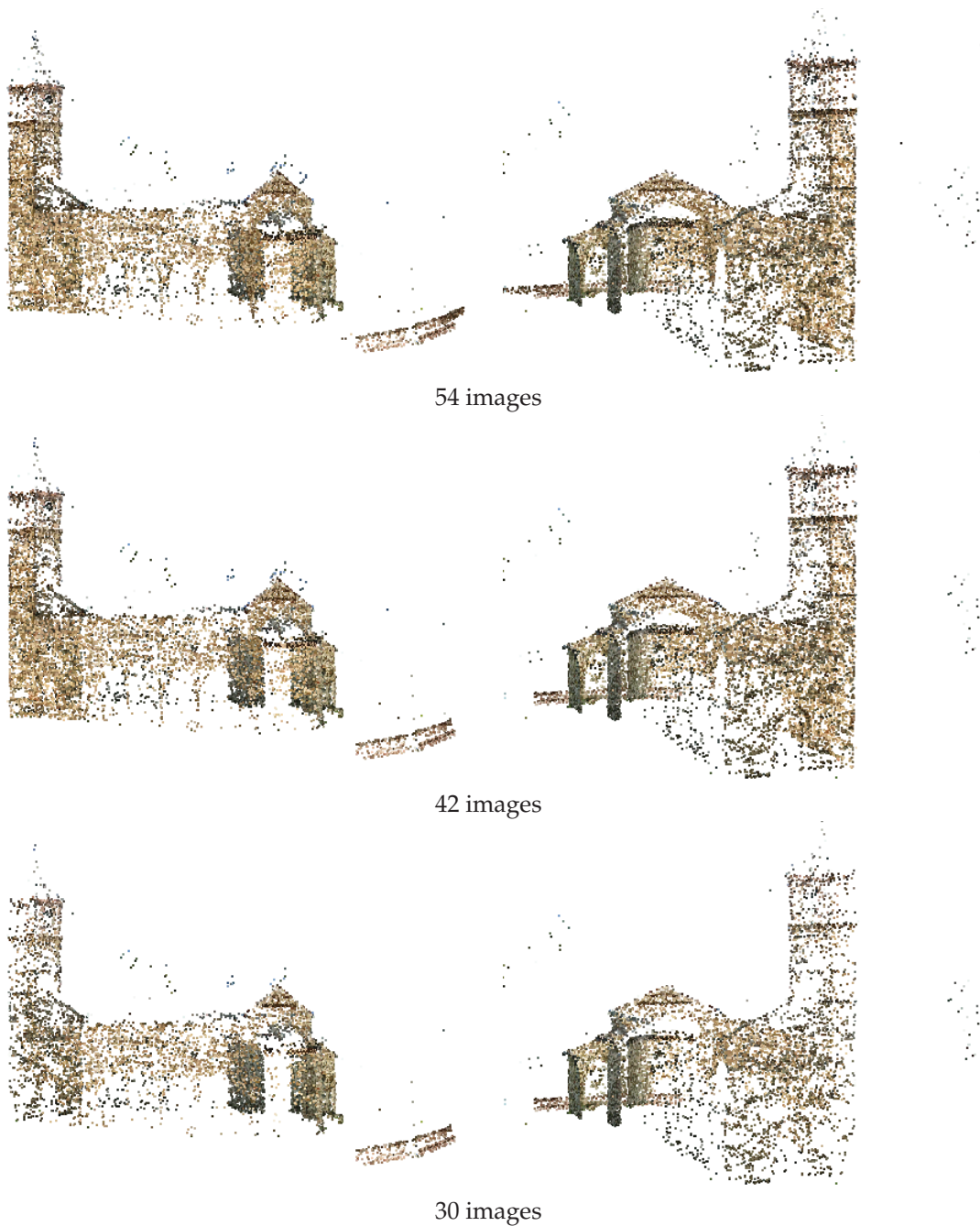


Figure 12.7: Image set reduction for the *pozzoveggiani* data set. The first row shows the full reconstruction, while the second and third row show the results after removing 12 and 24 of the 54 images respectively.

The results of the canonical view selection on the *piazzaerbe* data set are shown in figure 12.11. The selected images are very distinct from each other. Only the fountain and the pagoda are seen twice in the images. However, they are pictured from approximately opposite sides and have a completely different background.

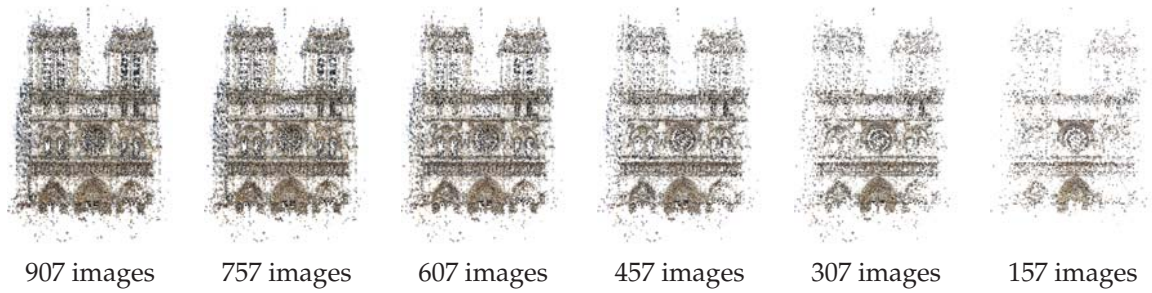


Figure 12.8: Image set reduction for the *Notre Dame* data set. The first image shows the full reconstruction (907 images). Each following image shows the result after 150 images were removed from the previous reconstruction.



Figure 12.9: Canonical views for the *pozzoveggiani* data set. One image was selected for each side of the church.



Figure 12.10: Canonical views for the *pozzoveggiani* data set as produced by [SSS07]. The parameters for obtaining this result had to be manually adjusted until a reasonable result was obtained.

Since we initially only use a sparse set of matches (i.e. we do not match every image to every other image), the region graph is also only sparsely connected. This means that similar images might not be connected in the region graph. The effect of this is that similar images might be selected as canonical views. Therefore we apply the canonical view selection twice. Once on the initial sparse graph and then on the obtained canonical views after performing



Figure 12.11: Canonical views for the *piazzaerbe* data set.

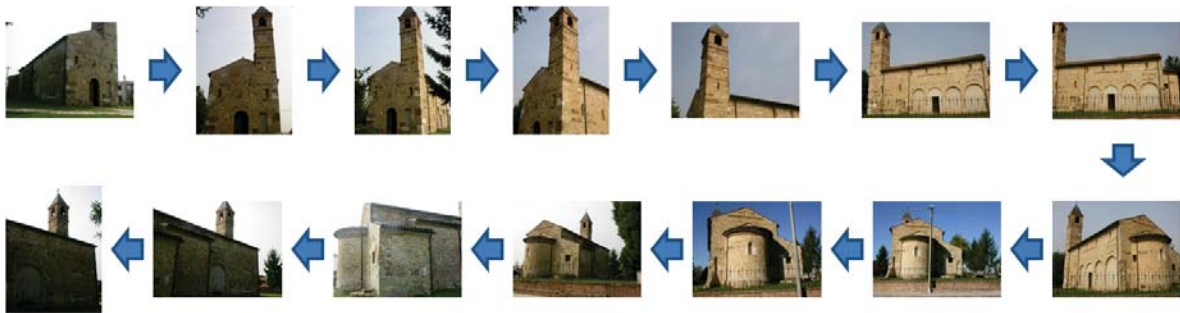


Figure 12.12: Image-based navigation on the *pozzoveggiani* data set. Starting from the top left image the user always moves to the right, thereby circling the church once.

an exhaustive pairwise matching on them. This is generally not very computationally expensive, since the number of canonical views is comparatively small compared to the size of the original data set. Optionally a vocabulary tree could be used to speed up the matching.

12.5.3 Image-based Navigation

Figure 12.12 shows the results for image-based navigation obtained on the *pozzoveggiani* data set. The user starts with the top left image and then continues to move to the right, circling the church once.

Figure 12.13 shows the results of an image-based navigation on the *Notre Dame* data set. On the left the user starts with the highlighted image and then navigates in the direction of the arrows (left, right, up and down). On the right the user performs a zoom-in and a zoom-out movement respectively. Note the number of scale levels traversed during the zoom-in and

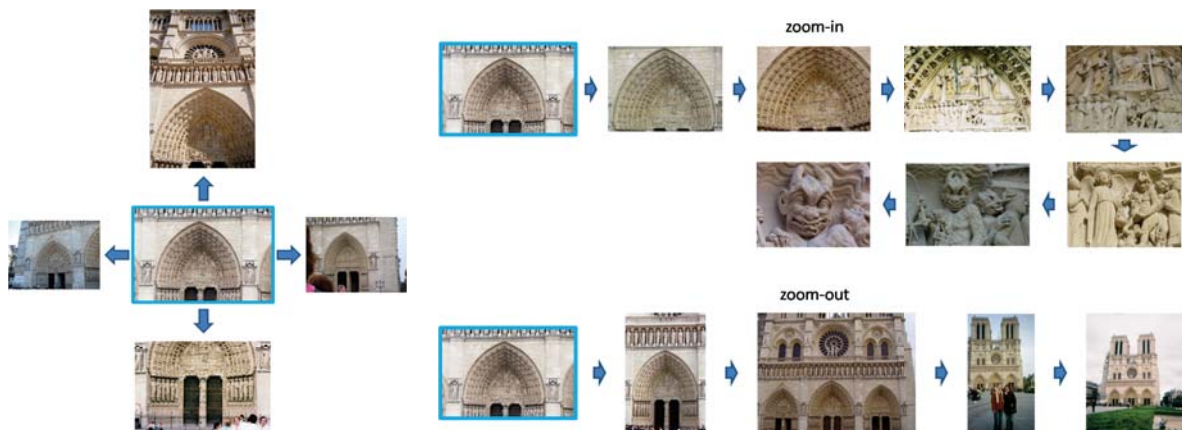


Figure 12.13: Image-based navigation on the *Notre Dame* data set. The user starts with the highlighted image and then performs several navigation operations resulting in the shown images. The images on the left show the results of a spatial navigation (left, right, up and down) while the images on the right show the results of zooming in and out respectively.

zoom-out operation.

12.6 Conclusion

We presented a novel framework for organizing large spatially related image collections. Our approach is based on the overlapping regions between multiple images. We represent these regions and the images in a graph and use this graph as a foundation for several different applications related to organizing large image collections, such as image-based navigation, image set reduction and canonical view selection. Using these applications we presented results on several image sets of different sizes, showing the validity of our image organization approach.

Chapter 13

Conclusion

We presented several methods related to reconstructing and organizing image collections. First we proposed a method for obtaining high-quality reconstructions using volumetric graph-cuts while reducing the effect of the minimum surface bias by using narrow bands and surface normal optimization. We then continued to investigate reconstructing large image collections efficiently by dividing them into smaller subsets based on scene and camera geometry criteria. Each subset is reconstructed independently making the reconstruction significantly faster than reconstructing all images together. Finally, we introduced a new data structure for organizing large image collections by focusing on the relation between image regions. This is more powerful than purely focusing on images since it allows a finer grained control over image relations and puts the actually important parts of the images into the focus of interest disregarding background and clutter. Based on this data structure we presented several applications related to organizing large image collections such as image-based navigation, image set reduction and canonical view selection.

13.1 Discussion and Future Work

The presented work is a first step towards the goal of reconstructing sites automatically from Internet photo collections. In order to get to this point more work especially on organizing and automatically processing Internet photo collections for use in reconstruction needs to be performed. The main challenges at this point in time are related to the size of the image collections and the huge runtime induced by the amount of data present. One possible solution to this issue, namely organizing and clustering image collections, was presented in the last two chapters.

One of the most time consuming aspects of reconstructing from Internet photo collections is establishing initial point correspondences between images. A simple minded exhaustive pairwise matching is not reasonable on large image collections since the runtime grows quadratically with the number of images. In the past several authors have proposed methods for grouping images based on color and edge descriptors [LWZ⁺08] and vocabulary trees [NS06]. This already drives down the number of image-pair matchings which need to be performed and increases the performance of the reconstruction process. One interesting issue to consider would be to integrate such methods into the region graph structure to

automatically obtain graphs which are sparser and therefore better suited for reconstruction.

Another interesting aspect which warrants further investigation is the matter of occlusion handling in the region graph structure. Due to occlusions created for instance by people standing in front of a building, the region graph creates more regions than necessary. By automatically detecting occluding objects and discarding them the region graph would become more efficient. This would also allow to perform image-based inpainting in order to remove the occluders from the image as for instance done in [HE07, WSZ09]. The detection of occluding objects could for instance be based on a photometric comparison between matched regions which would identify the parts of the regions which are not photometrically consistent and therefore possibly occluded. Considering photometric consistency between regions could also be used to improve the shapes of the regions since the polygonal hull created by the intersection of the convex hulls of matched features is only a first approximation to the actual region shape.

Furthermore, it would be interesting to consider reconstructing only certain image regions as represented in the region graph. This would allow a user to only reconstruct the part of the scene he is interested in instead of reconstructing the whole scene at once. This could possibly also speed up a global reconstruction since each region is much smaller than the whole scene and therefore more efficient to reconstruct.

List of (Co-)Authored Publications

Related to this Thesis

A. LADIKOS, C. CAGNIART, R. GOTHBI, M. REISER, and N. NAVAB, *Estimating Radiation Exposure in Interventional Environments*, in International Conference on Medical Image Computing and Computer-Assisted Intervention (MICCAI), 2010.

A. LADIKOS, E. BOYER, N. NAVAB, and S. ILIC, *Region Graphs for Organizing Image Collections*, in ECCV Workshop on Reconstruction and Modeling of Large-Scale 3D Virtual Environments (RMLE), 2010.

A. LADIKOS and N. NAVAB, *Real-Time 3D Reconstruction for Occlusion-aware Interactions in Mixed Reality*, in International Symposium on Visual Computing (ISVC), 2009.

A. LADIKOS, S. ILIC, and N. NAVAB, *Spectral Camera Clustering*, in ICCV Workshop on Emergent Issues in Large Amounts of Visual Data (LAVD), 2009.

A. BIGDELOU, A. LADIKOS, and N. NAVAB, *Incremental Visual Hull Reconstruction*, in British Machine Vision Conference (BMVC), 2009.

A. LADIKOS, S. BENHIMANE AND N. NAVAB, *Multi-View Reconstruction using Narrow-Band Graph-Cuts and Surface Normal Optimization*, in British Machine Vision Conference (BMVC), 2008.

A. LADIKOS, S. BENHIMANE AND N. NAVAB, *Real-time 3D Reconstruction for Collision Avoidance in Interventional Environments*, in International Conference on Medical Image Computing and Computer-Assisted Intervention (MICCAI), 2008.

A. LADIKOS, S. BENHIMANE AND N. NAVAB, *Efficient Visual Hull Computation for Real-Time 3D Reconstruction using CUDA*, in CVPR Workshop on Computer Vision on GPUs (CVGPU), 2008.

Other Publications

A. LADIKOS, S. BENHIMANE, and N. NAVAB, *High Performance Model-Based Object Detection and Tracking*, in Computer Vision and Computer Graphics. Theory and Applications, vol. 21 of Communications in Computer and Information Science, Springer, 2008.

LIST OF (CO-)AUTHORED PUBLICATIONS

A. LADIKOS, S. BENHIMANE AND N. NAVAB, *Model-Free Markerless Tracking for Remote Support in Unknown Environments*, in International Conference on Computer Vision Theory and Applications (VISAPP), 2008.

S. BENHIMANE, A. LADIKOS, V. LEPETIT, and N. NAVAB, *Linear and Quadratic Subsets for Template-Based Tracking*, in IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR), 2007.

A. LADIKOS, S. BENHIMANE, and N. NAVAB, *A Real-Time Tracking System Combining Template-Based and Feature-Based Approaches*, in International Conference on Computer Vision Theory and Applications (VISAPP), 2007.

G. PANIN, A. LADIKOS, and A. KNOLL, *An Efficient and Robust Real-Time Contour Tracking System*, in IEEE International Conference on Computer Vision Systems (ICVS), 2006.

Bibliography

- [ADD⁺93] I. ALTHÖFER, G. DAS, D. DOBKIN, D. JOSEPH, and J. SOARES, *On sparse spanners of weighted graphs*, *Discrete Comput. Geom.*, 9 (1993), pp. 81–100.
- [AFM⁺06] J. ALLARD, J.-S. FRANCO, C. MÉNIER, E. BOYER, and B. RAFFIN, *The GrImage Platform: A Mixed Reality Environment for Interactions*, in *IEEE International Conference on Computer Vision Systems*, 2006.
- [AKA⁺04] D. ASAI, S. KATOPO, J. ARATA, S. WARISAWA, M. MITSUISHI, A. MORITA, S. SORA, T. KIRINO, and R. MOCHIZUKI, *Micro-Neurosurgical System in the Deep Surgical Field*, in *Medical Image Computing and Computer-Assisted Intervention*, 2004.
- [AMR⁺07] J. ALLARD, C. MENIER, B. RAFFIN, E. BOYER, and F. FAURE, *Grimage: Markerless 3D Interactions*, in *SIGGRAPH - Emerging Technologies*, 2007.
- [APSK07] E. AGANJ, J. PONS, F. SÉGONNE, and R. KERIVEN, *Spatio-Temporal Shape from Silhouette using Four-Dimensional Delaunay Meshing*, in *International Conference on Computer Vision*, IEEE, 2007.
- [ASS⁺09] S. AGARWAL, N. SNAVELY, I. SIMON, S. M. SEITZ, and R. SZELISKI, *Building Rome in a Day*, in *International Conference on Computer Vision*, 2009.
- [AW87] J. AMANATIDES and A. WOO, *A fast voxel traversal algorithm for ray tracing*, in *Eurographics*, 1987.
- [BBH08] D. BRADLEY, T. BOUBEKEUR, and W. HEIDRICH, *Accurate Multi-View Reconstruction Using Robust Binocular Stereo and Surface Meshing*, in *IEEE Conference on Computer Vision and Pattern Recognition*, 2008.
- [BBT08] E. BOGAERT, K. BACHER, and H. THIERENS, *A large-scale multicentre study in Belgium of dose area product values and effective doses in interventional cardiology using contemporary X-ray equipment*, *Radiation Protection Dosimetry*, 128 (2008), pp. 312–323.
- [Bel96] P. N. BELHUMEUR, *A Bayesian approach to binocular stereopsis*, *International Journal of Computer Vision*, 19 (1996), pp. 237–260.
- [Ber97] M. BERGER, *Resolving Occlusion in Augmented Reality : a Contour Based Approach without 3D Reconstruction*, in *IEEE Conference on Computer Vision and Pattern Recognition*, 1997.

BIBLIOGRAPHY

- [BF03] E. BOYER and J. S. FRANCO, *A Hybrid Approach for Computing Visual Hulls of Complex Objects*, in IEEE Conference on Computer Vision and Pattern Recognition, 2003.
- [BK03] Y. BOYKOV and V. KOLMOGOROV, *Computing Geodesics and Minimal Surfaces via Graph-Cuts*, in International Conference on Computer Vision, 2003.
- [BL06] Y. BOYKOV and V. LEMPITSKY, *From Photohulls to Photoflux Optimization*, in British Machine Vision Conference, 2006.
- [BM04] S. BAKER and I. MATTHEWS, *Lucas-Kanade 20 Years On: A Unifying Framework*, International Journal of Computer Vision, 56 (2004), pp. 221 – 255.
- [BM06] O. BATCHELOR and R. MUKUNDAN, *Ray Traversal for Incremental Voxel Coloring*, 2006. M.Sc Thesis.
- [BMG05] O. BATCHELOR, R. MUKUNDAN, and R. GREEN, *Ray Casting for Incremental Voxel Colouring*, in International Conference on Image and Vision Computing, 2005.
- [BMS98] J. BATLLE, E. MOUADDIB, and J. SALVI, *Recent progress in coded structured light as a technique to solve the correspondence problem: a survey*, Pattern Recognition, 31 (1998), pp. 963 – 982.
- [Bou] J.-Y. BOUGUET, *Camera calibration toolbox*, http://www.vision.caltech.edu/bouguetj/calib_doc.
- [BSD03] E. BOROVNIKOV, A. SUSSMAN, and L. DAVIS, *A High Performance Multi-Perspective Vision Studio*, in ACM International Conference on Supercomputing, 2003.
- [BVZ01] Y. BOYKOV, O. VEKSLER, and R. ZABIH, *Fast Approximate Energy Minimization via Graph-Cuts*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 23 (2001), pp. 1222–1239.
- [BWD⁺09] O. BOTT, M. WAGNER, C. DUWENKAMP, N. HELLRUNG, and K. DRESSING, *Improving education on C-arm operation and radiation protection with a computer-based training and simulation system*, International Journal of Computer Assisted Radiology and Surgery, 4 (2009), pp. 399–407.
- [BZK06] O. BOOIJ, Z. ZIVKOVIC, and B. KRÖSE, *Sparse appearance based modeling for robot localization*, in International Conference on Intelligent Robots and Systems, 2006.
- [CBI10a] C. CAGNIART, E. BOYER, and S. ILIC, *Free-From Mesh Tracking: a Patch-Based Approach*, in IEEE Conference on Computer Vision and Pattern Recognition, 2010.
- [CBI10b] C. CAGNIART, E. BOYER, and S. ILIC, *Probabilistic Deformable Surface Tracking From Multiple Videos*, in European Conference on Computer Vision, 2010.

BIBLIOGRAPHY

- [CBK05a] K. CHEUNG, S. BAKER, and T. KANADE, *Shape-From-Silhouette Across Time Part I: Theory and Algorithms*, International Journal of Computer Vision, 62 (2005), pp. 221 – 247.
- [CBK05b] K. CHEUNG, S. BAKER, and T. KANADE, *Shape-From-Silhouette Across Time: Part II: Applications to Human Modeling and Markerless Motion Tracking*, International Journal of Computer Vision, 63 (2005), pp. 225 – 245.
- [CKBH00] G. CHEUNG, T. KANADE, J. Y. BOUGUET, and M. HOLLER, *A Real-Time System for Robust 3D Voxel Reconstruction of Human Motions*, in IEEE Conference on Computer Vision and Pattern Recognition, 2000.
- [CM02] D. COMANICIU and P. MEER, *Mean Shift: A Robust Approach Toward Feature Space Analysis*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 24 (2002), pp. 603–619.
- [DDAS04] T. DENTON, M. F. DEMIRCI, J. ABRAHAMSON, and A. SHOKOUFANDEH, *Selecting Canonical Views for View-Based 3-D Object Recognition*, in International Conference on Pattern Recognition, 2004.
- [DMB07] B. D. DECKER, T. MERTENS, and P. BEKAERT, *Interactive Collision Detection for Free-Viewpoint Video*, in International Conference on Computer Graphics Theory and Applications, 2007.
- [FB03] J. S. FRANCO and E. BOYER, *Exact Polyhedral Visual Hulls*, in British Machine Vision Conference, 2003.
- [FB05] J. S. FRANCO and E. BOYER, *Fusion of Multi-View Silhouette Cues Using a Space Occupancy Grid*, in International Conference on Computer Vision, 2005.
- [FFG09] M. FARENZENA, A. FUSIELLO, and R. GHERARDI, *Structure-and-Motion Pipeline on a Hierarchical Cluster Tree*, in IEEE International Workshop on 3-D Digital Imaging and Modeling, 2009.
- [FFK⁺07] G. FICHTINGER, J. FIENE, C. KENNEDY, G. KRONREIF, I. IORDACHITA, D. SONG, E. BURDETTE, and P. KAZANZIDES, *Robotic Assistance for Ultrasound Guided Prostate Brachytherapy*, in Medical Image Computing and Computer-Assisted Intervention, 2007.
- [FII⁺06] S. FUKUI, Y. IWAHORI, H. ITOH, H. KAWANAKA, and R. WOODHAM, *Robust Background Subtraction for Quick Illumination Changes*, in Pacific-Rim Symposium on Image and Video Technology, 2006.
- [FK98] O. FAUGERAS and R. KERIVEN, *Variational Principles, Surface Evolution, PDE's, Level Set Methods, and the Stereo Problem*, IEEE Transactions on Image Processing, 7 (1998), pp. 336–344.
- [FLB06] J. S. FRANCO, M. LAPIERRE, and E. BOYER, *Visual Shape of Silhouette Sets*, in International Symposium on 3D Data Processing, Visualization and Transmission, 2006.

BIBLIOGRAPHY

- [FMBR04] J. S. FRANCO, C. MENIER, E. BOYER, and B. RAFFIN, *A Distributed Approach to Real Time 3D Modeling*, in IEEE Conference on Computer Vision and Pattern Recognition, 2004.
- [FP06] Y. FURUKAWA and J. PONCE, *Carved Visual Hulls for Image-Based Modeling*, in European Conference on Computer Vision, 2006.
- [FP07] Y. FURUKAWA and J. PONCE, *Accurate, Dense, and Robust Multi-View Stereopsis*, in IEEE Conference on Computer Vision and Pattern Recognition, 2007.
- [FWFP08] F. FRAUNDORFER, C. WU, J.-M. FRAHM, and M. POLLEFEYS, *Visual Word based Location Recognition in 3D models using Distance Augmented Weighting*, in International Symposium on 3D Data Processing, Visualization and Transmission, 2008.
- [GBMR09] B. D. GIORDANO, J. F. BAUMHAUER, T. L. MORGAN, and G. R. RECHTINE, *Patient and Surgeon Radiation Exposure: Comparison of Standard and Mini-C-Arm Fluoroscopy*, *Journal of Bone & Joint Surgery*, 91 (2009), pp. 297–304.
- [GLV⁺05] J.-Y. GAUVRITA, X. LECLERCA, M. VERMANDELB, B. LUBICZA, D. DESPRETZD, J.-P. LEJEUNEC, J. ROUSSEAU, and J.-P. PRUVOA, *3D Rotational Angiography: Use of Propeller Rotation for the Evaluation of Intracranial Aneurysms*, *American Journal of Neuroradiology*, 26 (2005), pp. 163–165.
- [GM04] B. GOLDLÜCKE and M. MAGNOR, *Space-time Isosurface Evolution for Temporally Coherent 3D Reconstruction*, in IEEE Conference on Computer Vision and Pattern Recognition, 2004.
- [GSC⁺07] M. GOESELE, N. SNAVELY, B. CURLESS, H. HOPPE, and S. SEITZ, *Multi-View Stereo for community photo collections*, in International Conference on Computer Vision, 2007.
- [GSdA⁺09] J. GALL, C. STOLL, E. DE AGUIAR, C. THEOBALT, B. ROSENHAHN, and H.-P. SEIDEL, *Motion capture using joint skeleton tracking and surface estimation.*, in IEEE Conference on Computer Vision and Pattern Recognition, 2009.
- [GSFP06] L. GUAN, S. SINHA, J.-S. FRANCO, and M. POLLEFEYS, *Visual Hull Construction in the Presence of Partial Occlusion*, in International Symposium on 3D Data Processing, Visualization and Transmission, 2006.
- [GTH⁺07] O. GRAU, G. A. THOMAS, A. HILTON, J. KILNER, and J. STARCK, *A Robust Free-viewpoint Video System for Sport Scenes*, in Proceedings of the 3DTV Conference, 2007.
- [GWN⁺03] M. GROSS, S. WÜRMLIN, M. NAEF, E. LAMBORAY, C. SPAGNO, A. KUNZ, E. KOLLER-MEIER, T. SVOBODA, L. V. GOOL, S. LANG, K. STREHLKE, A. V. MOERE, OLIVER, E. ZÜRICH, and O. STAADT, *blue-c: A Spatially Immersive Display and 3D Video Portal for Telepresence*, in *ACM Transactions on Graphics*, 2003, pp. 819–827.
- [HE07] J. HAYS and A. A. EFROS, *Scene Completion Using Millions of Photographs*, *ACM Transactions on Graphics*, 26 (2007).

BIBLIOGRAPHY

- [HK06a] M. HABBECKE and L. KOBBELT, *Iterative Multi-View Plane Fitting*, in International Workshop on Vision, Modeling and Visualization, 2006.
- [HK06b] A. HORNING and L. KOBBELT, *Hierarchical Volumetric Multi-View Stereo Reconstruction of Manifold Surfaces based on Dual Graph Embedding*, in IEEE Conference on Computer Vision and Pattern Recognition, 2006.
- [HK06c] A. HORNING and L. KOBBELT, *Robust and Efficient Photo-Consistency Estimation for Volumetric 3D Reconstruction*, in European Conference on Computer Vision, 2006.
- [HK07] M. HABBECKE and L. KOBBELT, *A Surface Growing Approach to Multi-View Stereo Reconstruction*, in IEEE Conference on Computer Vision and Pattern Recognition, 2007.
- [HLGB03] J.-M. HASENFRATZ, M. LAPIERRE, J.-D. GASCUEL, and E. BOYER, *Real-Time Capture, Reconstruction and Insertion into Virtual World of Human Actors*, in Vision, Video and Graphics, 2003.
- [HLS04] J. M. HASENFRATZ, M. LAPIERRE, and F. SILLION, *A Real-Time System for Full Body Interaction with Virtual Worlds*, Eurographics Symposium on Virtual Environments, (2004).
- [HS04] C. HERNANDEZ and F. SCHMITT, *Silhouette and Stereo Fusion for 3D Object Modeling*, Computer Vision and Image Understanding, 96 (2004), pp. 367–392.
- [HS09] H. HIRSCHMÜLLER and D. SCHARSTEIN, *Evaluation of Stereo Matching Costs on Images with Radiometric Differences*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 31 (2009), pp. 1582–1599.
- [HSJS08] B. HUHLE, T. SCHAIRER, P. JENKE, and W. STRASSER, *Robust Non-Local Denoising of Colored Depth Data*, in IEEE CVPR Workshop on Time of Flight Camera based Computer Vision, 2008.
- [HVC07] C. HERNANDEZ, G. VOGIATZIS, and R. CIPOLLA, *Probabilistic Visibility for Multi-View Stereo*, in IEEE Conference on Computer Vision and Pattern Recognition, 2007.
- [HZ04] R. HARTLEY and A. ZISSERMAN, *Multiple View Geometry in Computer Vision*, Cambridge University Press, second ed., 2004.
- [HZK08] A. HORNING, B. ZENG, and L. KOBBELT, *Image Selection For Improved Multi-View Stereo*, in IEEE Conference on Computer Vision and Pattern Recognition, 2008.
- [iP] IVIEW PROJECT, <http://www.bbc.co.uk/rd/projects/iview>.
- [JDS08] H. JÉGOU, M. DOUZE, and C. SCHMID, *Hamming embedding and weak geometric consistency for large scale image search*, in European Conference on Computer Vision, 2008.

BIBLIOGRAPHY

- [KCHD05] K. KIM, T. CHALIDABHONGSE, D. HARWOOD, and L. DAVIS, *Real-time foreground-background segmentation using codebook model*, *Real-Time Imaging*, 11 (2005), pp. 172–185.
- [Keh05] R. KEHL, *Markerless Motion Capture of Complex Human Movements from Multiple Views*, PhD thesis, ETH Zurich, 2005.
- [KN07] T. KANADE and P. J. NARAYANAN, *Virtualized Reality: Perspectives on 4D Digitization of Dynamic Events*, *IEEE Computer Graphics and Applications*, (2007), pp. 32 – 40.
- [KRN97] T. KANADE, P. RANDEK, and P. J. NARAYANAN, *Virtualized Reality: Constructing Virtual Worlds from Real Scenes*, *IEEE Multimedia, Immersive Telepresence*, 4 (1997), pp. 34–47.
- [KS00] K. KUTULAKOS and S. SEITZ, *A Theory of Shape by Space Carving*, *International Journal of Computer Vision*, 38 (2000), pp. 199–218.
- [Kut00] K. KUTULAKOS, *Approximate N-View Stereo*, in *European Conference on Computer Vision*, 2000.
- [KYS03] H. KIM, S. YANG, and K. SOHN, *3D Reconstruction of Stereo Images for Interaction between Real and Virtual Worlds*, in *IEEE International Symposium on Mixed and Augmented Reality*, 2003.
- [KZ01] V. KOLMOGOROV and R. ZABIH, *Computing Visual Correspondence with Occlusions via Graph Cuts*, in *International Conference on Computer Vision*, 2001.
- [KZ02] V. KOLMOGOROV and R. ZABIH, *Multi-Camera Scene Reconstruction via Graph-Cuts*, in *European Conference on Computer Vision*, 2002.
- [KZ04] V. KOLMOGOROV and R. ZABIH, *What Energy Functions Can Be Minimized via Graph Cuts?*, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26 (2004), pp. 147–159.
- [Lau94] A. LAURENTINI, *The Visual Hull Concept for Silhouette-Based Image Understanding*, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16 (1994), pp. 150–162.
- [LBI06] V. LEMPITSKY, Y. BOYKOV, and D. IVANOV, *Oriented Visibility for Multiview Reconstruction*, in *European Conference on Computer Vision*, 2006.
- [LBN08] A. LADIKOS, S. BENHIMANE, and N. NAVAB, *Efficient Visual Hull Computation for Real-Time 3D Reconstruction using CUDA*, in *CVPR Workshop on Computer Vision on GPUs*, 2008.
- [LC87] W. E. LORENSEN and H. E. CLINE, *Marching cubes: A high resolution 3D surface construction algorithm*, *SIGGRAPH*, 21 (1987), pp. 163–169.
- [Lev88] M. LEVOY, *Display of Surfaces from Volume Data*, *IEEE Computer Graphics and Applications*, 8 (1988), pp. 29–37.

BIBLIOGRAPHY

- [LFP07] S. LAZEBNIK, Y. FURUKAWA, and J. PONCE, *Projective Visual Hulls*, *International Journal of Computer Vision*, 74 (2007), pp. 137–165.
- [LMS03a] M. LI, M. MAGNOR, and H. SEIDEL, *Hardware accelerated visual hull reconstruction and rendering*, in *Graphics Interface*, 2003.
- [LMS03b] M. LI, M. MAGNOR, and H. SEIDEL, *Improved hardware-accelerated visual hull rendering*, in *International Workshop on Vision, Modeling, and Visualization*, 2003.
- [LMS04] M. LI, M. MAGNOR, and H.-P. SEIDEL, *A hybrid hardware-accelerated algorithm for high quality rendering of visual hulls*, in *Graphics Interface*, 2004.
- [LNA⁺06] J. LEONG, M. NICOLAOU, L. ATALLAH, G. MYLONAS, A. DARZI, and G.-Z. YANG, *HMM Assessment of Quality of Movement Trajectory in Laparoscopic Surgery*, in *Medical Image Computing and Computer-Assisted Intervention*, 2006.
- [LNWB03] B. LOK, S. NAIK, M. WHITTON, and F. BROOKS, *Incorporating dynamic real objects into immersive virtual environments*, in *Symposium on Interactive 3D graphics*, 2003.
- [Low04] D. LOWE, *Distinctive Image Features from Scale-Invariant Keypoints*, *International Journal of Computer Vision*, 60 (2004), pp. 91–110.
- [LPC⁺00] M. LEVOY, K. PULLI, B. CURLESS, S. RUSINKIEWICZ, D. KOLLER, L. PEREIRA, M. GINZTON, S. ANDERSON, J. DAVIS, J. GINSBERG, J. SHADE, and D. FULK, *The Digital Michelangelo Project: 3D Scanning of Large Statues*, in *SIGGRAPH*, 2000.
- [LSM⁺05] H. LIN, I. SHAFRAN, T. MURPHY, A. OKAMURA, D. YUH, and G. HAGER, *Automatic Detection and Segmentation of Robot-Assisted Surgical Motions*, in *Medical Image Computing and Computer-Assisted Intervention*, 2005.
- [LWZ⁺08] X. LI, C. WU, C. ZACH, S. LAZEBNIK, and J.-M. FRAHM, *Modeling and Recognition of Landmark Image Collections Using Iconic Scene Graphs.*, in *European Conference on Computer Vision*, 2008.
- [MBM01] W. MATSUIK, C. BUEHLER, and L. MCMILLAN, *Polyhedral Visual Hulls for Real-Time Rendering*, in *Eurographics Workshop on Rendering*, 2001.
- [MBR⁺00] W. MATSUIK, C. BUEHLER, R. RASKAR, S. GORTLER, and L. MCMILLAN, *Image-Based Visual Hulls*, in *SIGGRAPH*, 2000.
- [ME96] M. MARX and J. ELLIS, *Radiation protection of the hand in interventional radiology: should it fit like a glove?*, *Radiology*, 200 (1996), pp. 24–25.
- [MH06] G. MILLER and A. HILTON, *Exact View-Dependent Visual Hulls*, in *International Conference on Pattern Recognition*, 2006.
- [MHK06] T. B. MOESLUND, A. HILTON, and V. KRUEGER, *A survey of advances in vision-based human motion capture and analysis*, *Computer Vision and Image Understanding*, 104 (2006), pp. 90–126.

BIBLIOGRAPHY

- [MP02] D. MARTINEC and T. PAJDLA, *Structure From Many Perspective Images with Occlusions*, in European Conference on Computer Vision, 2002.
- [MSE06] C. MÜLLER, M. STRENGERT, and T. ERTL, *Optimized Volume Raycasting for Graphics-Hardware-based Cluster Systems*, in Eurographics Symposium on Parallel Graphics and Visualization, 2006.
- [MWTN04] T. MATSUYAMA, X. WU, T. TAKAI, and S. NOBUHARA, *Real-time 3D shape reconstruction, dynamic 3D mesh deformation, and high fidelity visualization for 3D video*, *Computer Vision and Image Understanding*, 96 (2004), pp. 393–434.
- [NS06] D. NISTÉR and H. STEWÉNIUS, *Scalable Recognition with a Vocabulary Tree.*, in IEEE Conference on Computer Vision and Pattern Recognition, 2006.
- [Nvi] NVIDIA CUDA, <http://www.nvidia.com/cuda>.
- [OSYT98] T. OHSHIMA, K. SATOH, H. YAMAMOTO, and H. TAMURA, *AR2 Hockey: A Case Study of Collaborative Augmented Reality*, in Virtual Reality Annual International Symposium, 1998.
- [PBE+07] N. PADOY, T. BLUM, I. ESSA, H. FEUSSNER, M.-O. BERGER, and N. NAVAB, *A Boosted Segmentation Method for Surgical Workflow Analysis*, in Medical Image Computing and Computer-Assisted Intervention, 2007, pp. 102–109.
- [PCF+02] S. PRINCE, A. CHEOK, F. FARBIZ, T. WILLIAMSON, N. JOHNSON, M. BILLINGHURST, and H. KATO, *3D Live: Real Time Captured Content for Mixed Reality*, in IEEE International Symposium on Mixed and Augmented Reality, 2002.
- [PCI+08] J. PHILBIN, O. CHUM, M. ISARD, J. SIVIC, and A. ZISSERMAN, *Lost in Quantization: Improving Particular Object Retrieval in Large Scale Image Databases*, in IEEE Conference on Computer Vision and Pattern Recognition, 2008.
- [PKF07] J. P. PONS, R. KERIVEN, and O. FAUGERAS, *Multi-View Stereo Reconstruction and Scene Flow Estimation with a Global Image-Based Matching Score*, *International Journal of Computer Vision*, 72 (2007), pp. 179–193.
- [PLBR09] B. PETIT, J.-D. LESAGE, E. BOYER, and B. RAFFIN, *Virtualization Gate*, in SIGGRAPH - Emerging Technologies, 2009.
- [PLM+10] B. PETIT, J.-D. LESAGE, C. MÉNIER, J. ALLARD, J.-S. FRANCO, B. RAFFIN, E. BOYER, and F. FAURE, *Multi-Camera Real-Time 3D Modeling for Telepresence and Remote Collaboration*, *International Journal of Digital Multimedia Broadcasting*, (2010).
- [PMW+09] N. PADOY, D. MATEUS, D. WEINLAND, M.-O. BERGER, and N. NAVAB, *Workflow Monitoring based on 3D Motion Features*, in ICCV IEEE Workshop on Video-oriented Object and Event Classification, 2009.
- [Pot87] M. POTMESIL, *Generating octree models of 3D objects from their silhouettes in a sequence of images*, *Computer Vision, Graphics and Image Processing*, 40 (1987), pp. 1–29.

BIBLIOGRAPHY

- [SAHML07] C. P. SHORTT, H. AL-HASHIMI, L. MALONE, and M. J. LEE, *Staff Radiation Doses to the Lower Extremities in Interventional Radiology*, *Cardiovascular Interventional Radiology*, 30 (2007), pp. 1206–1209.
- [SCD⁺06] S. SEITZ, B. CURLES, J. DIEBEL, D. SCHARSTEIN, and R. SZELISKI, *A Comparison and Evaluation of Multi-View Stereo Reconstruction Algorithms*, in *IEEE Conference on Computer Vision and Pattern Recognition*, 2006.
- [SCM⁺04] G. SLABAUGH, B. CULBERTSON, T. MALZBENDER, M. STEVENS, and R. SCHAFER, *Methods for Volumetric Reconstruction of Visual Scenes*, *International Journal of Computer Vision*, 57 (2004), pp. 179–199.
- [SD99] S. SEITZ and C. DYER, *Photorealistic Scene Reconstruction by Voxel Coloring*, *International Journal of Computer Vision*, 25 (1999), pp. 151–173.
- [SG98] C. STAUFFER and W. E. L. GRIMSON, *Adaptive Background Mixture Models for Real-Time Tracking*, in *IEEE Conference on Computer Vision and Pattern Recognition*, 1998.
- [SGSS08] N. SNAVELY, R. GARG, S. SEITZ, and R. SZELISKI, *Finding paths through the world's photos*, *ACM Transactions on Graphics*, 27 (2008), pp. 1–11.
- [SH07a] J. STARCK and A. HILTON, *Correspondence labelling for wide-timeframe free-form surface matching*, in *International Conference on Computer Vision*, 2007.
- [SH07b] J. STARCK and A. HILTON, *Surface Capture for Performance-Based Animation*, *IEEE Computer Graphics and Applications*, 27 (2007), pp. 21–31.
- [SHZO07] S. SENGUPTA, M. HARRIS, Y. ZHANG, and J. D. OWENS, *Scan Primitives for GPU Computing*, in *Graphics Hardware*, ACM, 2007, pp. 97–106.
- [SI85] K. SATO and S. INOKUCHI, *Three-dimensional surface measurement by space encoding range imaging*, *Journal of Robotic Systems*, 2 (1985), pp. 27–39.
- [SKB⁺06] M. STRENGERT, T. KLEIN, R. BOTCHEN, S. STEGMAIER, M. CHEN, , and T. ERTL, *Volume surface octrees for the representation of 3D objects*, *The Visual Computer*, 22 (2006), pp. 550–561.
- [SM00] J. SHI and J. MALIK, *Normalized Cuts and Image Segmentation*, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22 (2000), pp. 888–905.
- [SMN⁺09] J. STARCK, A. MAKI, S. NOBUHARA, A. HILTON, and T. MATSUYAMA, *The multiple-camera 3-D production studio*, *IEEE Transactions on Circuits and Systems for Video Technology*, 19 (2009), pp. 856–869.
- [SMP05] T. SVOBODA, D. MARTINEC, and T. PAJDLA, *A Convenient Multi-Camera Self-Calibration for Virtual Environments*, *Presence: Teleoperators and Virtual Environments*, 14 (2005), pp. 407–422.
- [SMP07] S. SINHA, P. MORDOHAI, and M. POLLEFEYS, *Multi-View Stereo via Graph Cuts on the Dual of an Adaptive Tetrahedral Mesh*, in *International Conference on Computer Vision*, 2007.

BIBLIOGRAPHY

- [SMRR07] L. SOARES, C. MÉNIER, B. RAFFIN, and J.-L. ROCH, *Parallel Adaptive Octree Carving for Real-time 3D Modeling*, in IEEE Virtual Reality, 2007.
- [SP05] S. SINHA and M. POLLEFEYS, *Multi-View Reconstruction Using Photo-consistency and Exact Silhouette Constraints: A Maximum-Flow Formulation*, in International Conference on Computer Vision, 2005.
- [SS02] D. SCHARSTEIN and R. SZELISKI, *A Taxonomy and Evaluation of Dense Two-Frame Stereo Correspondence Algorithms*, International Journal of Computer Vision, 47 (2002), pp. 7–42.
- [SS03] D. SCHARSTEIN and R. SZELISKI, *High-Accuracy Stereo Depth Maps Using Structured Light*, in IEEE Conference on Computer Vision and Pattern Recognition, 2003.
- [SS08] I. SIMON and S. M. SEITZ, *Scene Segmentation Using the Wisdom of Crowds*, in European Conference on Computer Vision, 2008.
- [SSS06] N. SNAVELY, S. M. SEITZ, and R. SZELISKI, *Photo tourism: Exploring photo collections in 3D*, in SIGGRAPH, 2006.
- [SSS07] I. SIMON, N. SNAVELY, and S. M. SEITZ, *Scene Summarization for Online Image Collections.*, in International Conference on Computer Vision, IEEE, 2007, pp. 1–8.
- [SSS08a] N. SNAVELY, S. M. SEITZ, and R. SZELISKI, *Modeling the World from Internet Photo Collections.*, International Journal of Computer Vision, 80 (2008), pp. 189–210.
- [SSS08b] N. SNAVELY, S. M. SEITZ, and R. SZELISKI, *Skeletal graphs for efficient structure from motion.*, in IEEE Conference on Computer Vision and Pattern Recognition, 2008.
- [ST96] P. STURM and B. TRIGGS, *A Factorization Based Algorithm for Multi-Image Projective Structure and Motion*, in European Conference on Computer Vision, 1996.
- [SvHG⁺08] C. STRECHA, W. VON HANSEN, L. J. V. GOOL, P. FUA, and U. THOENNESSEN, *On benchmarking camera calibration and multi-view stereo for high resolution imagery.*, in IEEE Conference on Computer Vision and Pattern Recognition, 2008.
- [SVZ00] D. SNOW, P. VIOLA, and R. ZABIH, *Exact Voxel Occupancy With Graph-Cuts*, in IEEE Conference on Computer Vision and Pattern Recognition, 2000.
- [SWM⁺04] N. SUGITA, S. WARISAWA, M. MITSUISHI, M. SUZUKI, H. MORIYA, and K. KURAMOTO, *Development of a Novel Robot-Assisted Orthopaedic System Designed for Total Knee Arthroplasty*, in Medical Image Computing and Computer-Assisted Intervention, 2004.

BIBLIOGRAPHY

- [SZ02] F. SCHAFFALITZKY and A. ZISSERMAN, *Multi-view Matching for Unordered Image Sets, or "How Do I Organize My Holiday Snaps?"*, in European Conference on Computer Vision, 2002.
- [Sze93] R. SZELISKI, *Rapid Octree Construction from Image Sequences*, Computer Vision, Graphics and Image Processing: Image Understanding, 58 (1993), pp. 23–32.
- [TD06] S. TRAN and L. DAVIS, *3D Surface Reconstruction Using Graph-Cuts with Surface Constraints*, in European Conference on Computer Vision, 2006.
- [TLMpS03] C. THEOBALT, M. LI, M. A. MAGNOR, and H. PETER SEIDEL, *A Flexible and Versatile Studio for Synchronized Multi-view Video Recording*, in Vision, Video, Graphics, 2003.
- [TTK⁺08] I. A. TSALAFOUTAS, V. TSAPAKI, A. KALIAKMANIS, S. PNEUMATICOS, F. TSORONIS, E. D. KOULENTIANOS, and G. PAPACHRISTOU, *Estimation of Radiation Doses to Patients and Surgeons from various fluoroscopically guided orthopedic surgeries*, Radiation Protection Dosimetry, 128 (2008), pp. 112–119.
- [VBMP08] D. VLASIC, I. BARAN, W. MATUSIK, and J. POPOVIĆ, *Articulated Mesh Animation from Multi-view Silhouettes*, ACM Transactions on Graphics, 27 (2008), pp. 97:1–97:9.
- [VETC07] G. VOGIATZIS, C. H. ESTEBAN, P. H. S. TORR, and R. CIPOLLA, *Multiview Stereo via Volumetric Graph-Cuts and Occlusion Robust Photo-Consistency*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 29 (2007), pp. 2241–2246.
- [VKLP09] H.-H. VU, R. KERIVEN, P. LABATUT, and J.-P. PONS, *Towards high-resolution large-scale multi-view stereo*, in IEEE Conference on Computer Vision and Pattern Recognition, 2009.
- [vL07] U. VON LUXBURG, *A tutorial on spectral clustering*, Statistics and Computing, 17 (2007), pp. 395–416.
- [VTC05] G. VOGIATZIS, P. H. S. TORR, and R. CIPOLLA, *Multi-View Stereo via Volumetric Graph-Cuts*, in IEEE Conference on Computer Vision and Pattern Recognition, 2005.
- [VZBH08] K. VARANASI, A. ZAHARESCU, E. BOYER, and R. HORAUD, *Temporal Surface Tracking Using Mesh Evolution*, in European Conference on Computer Vision, 2008.
- [WDDDB09] M. WAGNER, C. DUWENKAMP, K. DRESING, and O. J. BOTT, *An Approach to Calculate and Visualize Intraoperative Scattered Radiation Exposure*, in MIE, 2009.
- [WM03] X. WU and T. MATSUYAMA, *Real-time active 3D shape reconstruction for 3D video*, in 3rd International Symposium on Image and Signal Processing and Analysis, 2003.
- [Woo80] R. WOODHAM, *Photometric method for determining surface orientation from multiple images*, Optical Engineering, 19 (1980), pp. 139–144.

BIBLIOGRAPHY

- [WSV91] M. W. WALKER, L. SHAO, and R. A. VOLZ, *Estimating 3-D location parameters using dual number quaternions*, CVGIP: Image Understanding, 54 (1991), pp. 358–367.
- [WSZ09] O. WHYTE, J. SIVIC, and A. ZISSERMAN, *Get Out of my Picture! Internet-based Inpainting*, in British Machine Vision Conference, 2009.
- [WTM06] X. WU, O. TAKIZAWA, and T. MATSUYAMA, *Parallel Pipeline Volume Intersection for Real-Time 3D Shape Reconstruction on a PC Cluster*, in IEEE International Conference on Computer Vision Systems, 2006.
- [XBA03] N. XU, R. BANSAL, and N. AHUJA, *Object segmentation using graph cuts based active contours*, in IEEE Conference on Computer Vision and Pattern Recognition, 2003.
- [YAC06] T. YU, N. AHUJA, and W.-C. CHEN, *SDG Cut: 3D Reconstruction of Non-lambertian Objects Using Graph Cuts on Surface Distance Grid*, in IEEE Conference on Computer Vision and Pattern Recognition, 2006.
- [ZCI⁺08] A. ZAHARESCU, C. CAGNIART, S. ILIC, E. BOYER, and R. P. HORAUD, *Camera Clustering for Multi-Resolution 3-D Surface Reconstruction*, in ECCV Workshop on Multi Camera and Multi-modal Sensor Fusion Algorithms and Applications, 2008.
- [Zha00] Z. ZHANG, *A Flexible New Technique for Camera Calibration*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 22 (2000), pp. 1330–1334.
- [ZPQS07] G. ZENG, S. PARIS, L. QUAN, and F. SILLION, *Accurate and Scalable Surface Representation and Reconstruction from Images*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 29 (2007), pp. 141–158.
- [ZvdH06] Z. ZIVKOVIC and F. VAN DER HEIJDEN, *Efficient adaptive density estimation per image pixel for the task of background subtraction*, Pattern Recognition Letters, 27 (2006), pp. 773–780.