NAIST-IS-DD0461045

Doctoral Dissertation

Active Camera Control for High-Resolution Imaging

Sofiane Yous

March 23, 2007

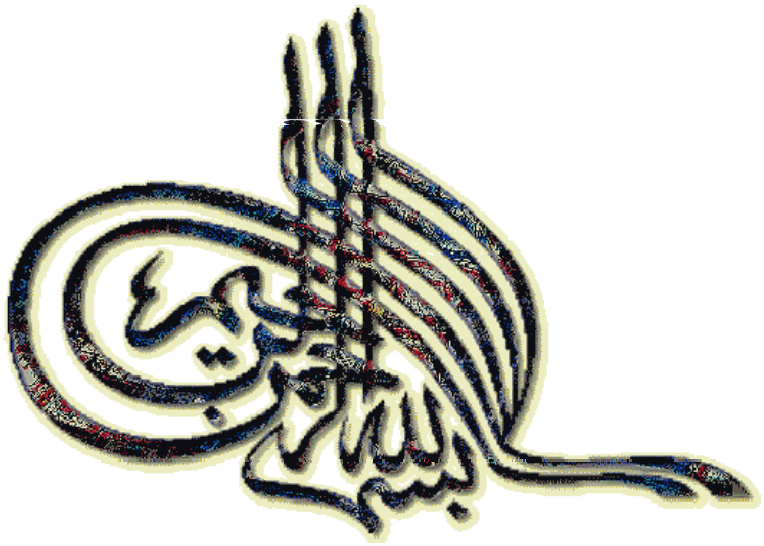
Department of Information Processing
Graduate School of Information Science
Nara Institute of Science and Technology

A Doctoral Dissertation
submitted to Graduate School of Information Science,
Nara Institute of Science and Technology
in partial fulfillment of the requirements for the degree of
Doctor of ENGINEERING

Sofiane Yous

Thesis Committee:

Professor Masatsugu Kidode	(Supervisor)
Professor Naokazu Yokoya	(member)
Associate Professor Atsuto Maki	(Member, Kyoto University)
Associate Professor Yasuyuki Kono	(Member)



To my family.

Active Camera Control for High-Resolution Imaging*

Sofiane Yous

Abstract

The basic problem in computer vision is to understand the real world given several images of it. These images are provided by different types of sensors each of which is suitable for a specific task. Fusing the information provided by different sensors provides a wealth of information allowing more understanding of the scene. In this thesis I will present a study on computer vision systems based on heterogeneous camera systems for high-resolution imaging. Mainly, I will consider 3D reconstruction applications. I will present a camera system combining static cameras with wide Fields Of View (FOV) and active Pan/Tilt (PT) cameras with narrow FOV . I will describe an assignment scheme to control the narrow-FOV PT cameras based on the analysis of a coarse shape reconstructed using the wide-FOV camera images. In order for the system to be applicable in 3D reconstruction of dynamic scenes, processing time is an important factor to be taken into account. I will present a hardware acceleration of the assignment scheme In order to speedup the processing. In addition, I will introduce a fast shape from silhouettes method that produces a surface-based representation of the shape as a preprocessing step for the assignment process. In order to show the applicability of the active camera control, I will present a networked heterogeneous camera system for high-resolution face images. Designed for visual surveillance applications, the proposed system combines static stereo cameras with wide FOV and monocular high-resolution PT cameras within a networked

*Doctoral Dissertation, Department of Information Processing, Graduate School of Information Science, Nara Institute of Science and Technology, NAIST-IS-DD0461045, March 23, 2007.

platform. The purpose is to provide high-resolution face images of in-motion targets while covering a wide area. The PT cameras are steered based of the 3D information provided by the stereo cameras.

Keywords:

Computer vision, Active camera control, 3D video, Shape from silhouettes, visual surveillance

Acknowledgements

First of all, I would like to express my deepest thanks to my supervisor, Professor Masatsugu Kidode. Without him, this thesis could never exist. He accepted me in his Laboratory and provided me with an unlimited support and guidance in my research and life in Japan.

I would like to thank Professor Naokazu Yokoya, Associate Professor Atsuto Maki, and Associate Professor Yasuyuki Kono, the committee members of my thesis, for their time reading and evaluating my thesis and for their invaluable comments and advices.

My grateful thanks to all the staff of Nara Institute of Science and Technology (NAIST) and the Graduate School of Information Science for their invaluable help. In particular, I am grateful to Professor Kunihiro Chihara for having accepted me in the 21st Century NAIST-COE program on Ubiquitous Networked Media Computing as a promoted researcher and for the grants and the support of this work.

I also gratefully acknowledge the Japanese Ministry of Education, Culture, Sport, Science and Technology for the scholarship they offered me during my stay in Japan, offering me a precious opportunity to experience a different life and a new culture.

Many thanks to my fellow students in the Artificial Intelligence Laboratory for their friendship and help. A particular thank goes to Ikuhisa Mitsugami, my tutor when I came first to NAIST, for his great help in facilitating my life in Nara.

Thanks to many Japanese friends, I was always feeling myself at home in Japan. I consider myself fortunate indeed to have met them.

Special thanks to Abdelaziz Khiat with whom I collaborated and co-authored two conference papers. Many thanks to Hamid Laga for his precious advices and for his presence in the crucial moments. Distinguished thanks to the best couple I have met in my life, Bassel and his wife Meriem, with whom I and my family have passed the best moments in Japan. I, definitely, consider them new members of my family. My Thanks to Abdelhafid Benterkia for his precious help and constant presence for me.

Last but not least, I would like to thank the persons to whom I awe all the

success in my life, my family, for being a constant source of love and affection. I can never thank enough my parents for their love and prayers for me, my brothers and sisters, my little boy and lovely daughter who are filling my life with joy and happiness, and my dearest wife, Fella, whose love, understanding and support are limitless.

Contents

Acknowledgements iii

1 Introduction **1**

1.1. Active Camera Control for High-Resolution Imaging 2

1.2. Outline of the Thesis 4

2 Active Camera Control for 3D Reconstruction applications **5**

2.1. Introduction 6

2.2. Multiple Active Camera Assignment 8

2.2.1 Preliminaries 8

2.2.2 Assignment Constraints 9

2.2.3 Assignment Constraints 9

2.2.4 Overview of the Assignment Scheme 10

2.3. Windowing Scheme 12

2.4. Visibility Quantification 15

2.4.1 Facet-wise Visibility Quantification 15

2.4.2 Local Visibility Quantification 15

2.4.3 Global Visibility Quantification 17

2.5. Global Assignment Scheme 18

2.6. Applicability of the Active Camera Assignment in 3D reconstruction of Dynamic Scenes 19

2.7. Experimental Results 21

2.7.1 Visibility Evaluation 22

2.7.2 Camera Movement Optimization 23

2.7.3 Processing Time 25

2.8. GPU-based Assignment Scheme 26

2.8.1 GPU: A Powerful Tool for General-Purpose Applications . 26
2.8.2 GPU-Based Assignment Scheme 27
2.8.3 Local Visibility = Multi-pass Pyramidal Filtering 29
2.8.4 Global Assignment 30
2.8.5 Implementation 31
2.9. Conclusion 33

3 Preprocessing For Active Camera control: An Accelerated Surface-Based Shape from Silhouettes 34
3.1. Introduction 35
3.2. Background and Related Work 36
3.2.1 Volume-Based VH Reconstruction 37
3.2.2 Surface-Based VH Reconstruction 38
3.2.3 Image Based Visual Hull Rendering (IBVH) 39
3.3. Overall Scheme 40
3.4. Viewing Edge Computing 41
3.4.1 Direct CSG-Based Veiw-Dependent Rendering 43
3.4.2 Our Depth Layer Traversal Method 45
3.4.3 Application to Viewing Edge Computing 47
3.4.4 Viewing Cone Projection 48
3.4.5 Viewing Edge Storage 49
3.4.6 Evaluation 51
3.5. VH Surface Construction 51
3.6. Experimental Results 54
3.7. Conclusion 59

4 Practicability of Active Camera Control: Heterogeneous Camera System for Visual Surveillance 61
4.1. Introduction 62
4.2. System Overview 63
4.2.1 Hardware Requirements 63
4.2.2 Fixed-Viewpoint Calibration 64
4.2.3 Global System Calibration 67
4.3. System Operation 67

4.3.1 3D Face Position Estimation 68
4.3.2 Active Camera Control 70
4.4. Experimental Results 72
4.4.1 System Operation 73
4.4.2 Accuracy of the 3D Face Position Estimation 74
4.4.3 Sensitivity to Lighting Changes 74
4.5. Conclusion 76

5 Conclusion and Future Work **78**

References **81**

List of Figures

1.1 System operation. 2

2.1 3D environment. 6
2.2 The configuration of the two cameras differs from (a) to (b), depending on the posture of the target. 7
2.3 Input data : 3D mesh surface. 9
2.4 Flowchart of the assignment scheme. 11
2.5 The first two steps of the proposed algorithm. 12
2.6 Windowing scheme: From (a) to (b) a flood filling is applied, and an adjustment is applied to get (c). The top cycle corresponds to the first windowing iteration for a given camera. The missing part in the depth image of the bottom cycle was windowed and deleted in previous iterations. 13
2.7 Assignment scheme: The different steps of our proposed scheme. $f(i)$, $f(i,k)$, and $L(c,k)$ refer respectively to the set of facet visible from camera c, these corresponding to a window w_k^c, and the local visibility corresponding to c and w_k^c. 17
2.8 Selected frames from the same viewpoint. 21
2.9 Facet-wise visibility histogram. 22
2.10 Local visibility evaluation. 23
2.11 Global visibility changes within frames with/without camera movement optimization. 24
2.12 The angular displacement each camera before and after camera movement optimization. 24
2.13 Scaled facet-wise rendering: Data flow through the GPU pipeline. 28

2.14 Multi-pass pyramidal filtering: The same operations are processed for each grid element. 29
2.15 Local visibility rendering. 30
2.16 Common texture: Local visibility accumulation. 31
2.17 GPU-based global assignment flowchart. 32

3.1 The VH reconstruction scheme 40
3.2 Viewing edge definition: The extracted viewing edges are shown in blue. 41
3.3 Direct CSG Vs. our method 2D: The traversed depth layers are drawn in green dashed lines. The points shown in red are the tested points. The Blue points are the saved points and those bounded by a red circle are tested and saved. The number of tested points is 5 in the direct-CSG method and 3 in ours. 44
3.4 The silhouette generalized cone: Each ray is represented as a cone face bounded by the faces of its immediate neighbors. The color of the faces refers to a unique id given to each face. 48
3.5 Viewing edge storage scheme: This processing is invoked at pixel (fragment) level. 50
3.6 Viewing edge extraction scheme. 52
3.7 Surface construction: (a) A view of the viewing edge merged together from 8 viewpoints. (b) The viewing edges are disconnected from each other after extraction. (c) The edges are connected to each other using the associated id. (d) The disconnected edges due to resolution difference between views are connected. 54
3.8 The VH reconstruction: Bunny dataset. 55
3.9 The VH reconstruction: Shark dataset. 55
3.10 The VH reconstruction: Real data (Maiko). 55
3.11 Reconstruction at lower scales. 57
3.12 Evaluation the proposed method for depth layer traversing: Comparison with the native direct CSG. 58
3.13 Variation of the number of extracted faces within iterations. . . . 58

4.1 Camera setup. 62

4.2 Custom Pan/Tilt unit allowing fixed-viewpoint calibration. 64
4.3 Fixed-viewpoint calibration process.. 65
4.4 Finite state machines for the system operation. 68
4.5 3D face position estimation process 69
4.6 The face's region concerned by the 3D face position estimation. . 70
4.7 Pan and tilt rotation angles 71
4.8 Top view of experiment environment 72
4.9 Sample images taken during the system operation experiments. . . 73
4.10 Estimated 3D face position while walking around a box. 75
4.11 The variation of the 3D face position estimation against the change in lighting conditions. 77

List of Tables

2.1 Summary of the gain in camera movement and the loss of visibility within the 26 frames. 25
2.2 Processing time for windowing, local visibility, and global assignment steps 25
2.3 Processing time for windowing, local visibility, and global assignment steps at different scales 26

3.1 Processing time evaluation: The processing time is calculated for each camera and for each dataset. The shown time concerns the viewing edge extraction and face generation. 'Point count' columns refer to the number of occluding contour points. 56
3.2 Comparison with the reconstruction using down-scaled images. . . 59

Chapter 1

Introduction

Computer vision is a science that focuses on understanding the surrounding world based on a set of images of it. These images are provided by different types of visual sensors such as stereo, monocular, active, and stationary cameras. Each visual sensor is suitable for a specific task and a specific situation. Fusing the information from these sensors is a precious tool for more understanding of the scene.

For instance, providing high-resolution images of targets in the scene requires the use of high-resolution cameras which have usually narrow Fields of View (FOV). If the observable area is wide, a huge number of these cameras are required for a full coverage of the scene. On the other hand, much less low resolution cameras (with wide FOV) are required for this coverage, but provide low resolution images. Though the images provided by these cameras are not useful as a final goal of the system, they can provide spatial information about the targets within the scene. These information can be used to steer the high-resolution active cameras to gaze the targets and capture high-resolution images of them. Hence, the full coverage of the scene with high-resolution cameras is no longer required.

Depending on the application, the active camera control is performed differently and based on different kind of data. In the literature, several systems have been described. Collins et al. [1] presented an active camera system for multi-view video. The goal was to control a set of active Pan/Tilt/Zoom (PTZ) cameras to keep a target within the monitored area centered in all views. Here we

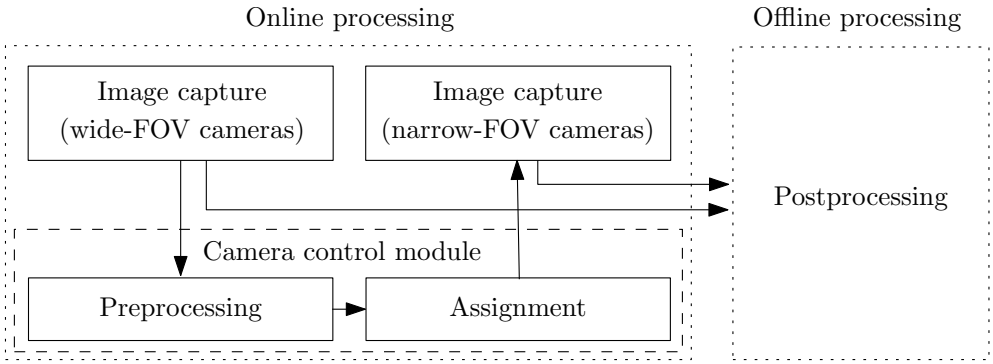


Figure 1.1. System operation.

are with a homogeneous cooperative camera system. The cameras communicate to fuse the target's location information into a 3D estimate. The camera that loses the track of the target can recover as long as other cameras are tracking it. Each camera adjusts its orientation and zoom based on the continuous tracking of the target using Mean Shift algorithm [2].

An image based PTZ camera control system was proposed in [3] for an automated surveillance system with multiple cameras. The positions of the targets are detected in one camera, called Master Camera, on which the detections and tracking are performed. The trajectories of the available active cameras are derived by homography with the master camera.

In this thesis I will present an active camera control for high-resolution imaging. I propose an active camera system that combines static wide-FOV cameras covering the entire observable area and sparsely-distributed active Pan/Tilt (PT) cameras with narrow FOV within a networked platform.

1.1. Active Camera Control for High-Resolution Imaging

The general operation scheme is illustrated in Fig. 1.1. Basically, the final output of the system is post-processed based of the high-resolution images. The narrow-FOV active cameras are controlled based on the processing of the wide-

FOV images. The Control module is composed of two sub-modules: 1) preprocessing, and 2)assignment. The preprocessing consists in processing the images provided by the wide-FOV cameras and provide useful information, to control the narrow-FOV cameras, to the assignment sub-module. The wide-FOV images are processed differently depending on the type of cameras and the final goal of the system.

In this thesis I mainly consider 3D reconstruction applications. I consider the general case where the narrow-FOV cameras can capture only partial views of a moving object, mainly human actor, but with high resolution. In such circumstances, one camera can get different visibility toward different parts of the object following the shape and the posture of the object. I will present an active camera assignment scheme based on the analysis of a shape reconstructed based on the wide-FOV camera images. The preprocessing consists in reconstructing the shape of the object based on the wide-FOV images. The major issue is the automatic assignment of each narrow-FOV camera to an appropriate part of the target in order to get high-resolution images of the whole object. The reconstructed shape is refined and textured in postprocessing step based on the captured high resolution images. In order for the system to be useful for real time applications, the processing time is an important factor to be taken into account. A hardware-accelerated camera assignment scheme will be presented. The first scheme is modified to run in the Graphics Processing Unit (GPU).

The scheme requires a 3D reconstruction of the target's shape. Most of real-time shape reconstruction methods provide voxel-based representation of the object. However, this representation is not suitable for visibility analysis. I will present a new shape from silhouettes method that provides a surface-based representation of the object. The proposed method runs on the GPU and recovers the 3D surface of the object in an interactive frame rate.

In order to show the practicability of the camera control of Fig. 1.1, I will present a networked heterogeneous camera system for high-resolution face images. This system is designed for visual surveillance applications where the goal is to monitor a wide area while providing high-resolution images of the faces of persons passing through. The proposed system combines static stereo wide-FOV cameras and high-resolution PT cameras within a networked platform. The preprocessing

consists in estimating the 3D positions of the faces within the monitored area based on stereo processing. These 3D positions are useful to steer the PT cameras in order to gaze the faces and capture close-up views of them while the targets are in motion. A custom PT unit allowing a static camera projection center was designed to hold the high-resolution cameras.

1.2. Outline of the Thesis

The rest of this thesis is organized as follows. In chapter 2, I will introduce the active camera assignment scheme. I will describe the camera system and detail the different steps of the proposed scheme. In section 4, I will present an accelerated surface based shape from silhouettes as a preprocessing step in the camera control module. In order to show the practicability of the proposed system, I will present, in chapter 5, a networked heterogeneous camera system for high resolution face images. The hardware and software components of this system will be presented in details. Finally, I will conclude this thesis and discuss the possible future work in chapter 6.

Chapter 2

Active Camera Control for 3D Reconstruction applications

In this chapter I present an assignment scheme to control multiple Pan/Tilt (PT) cameras for 3D reconstruction applications. The system combines static wide field of view (FOV) cameras and active Pan/Tilt (PT) cameras with narrow FOV within a networked platform. I consider the general case where the PT cameras have as high resolution as they can capture only partial views of the object. The major issue is the automatic assignment of each active camera to an appropriate part of the object in order to get high-resolution images of the whole object. I propose an assignment scheme based on the analysis of a coarse 3D shape produced in a preprocessing step based on the wide-FOV images. For each high-resolution camera, the visibility toward the different parts of the shape, corresponding to different orientations of the camera and with respect to its FOV, are evaluated. Then, each camera is assigned to one orientation in order to get high visibility of the whole object. The continuously captured images are saved to be used offline in the reconstruction of the object.

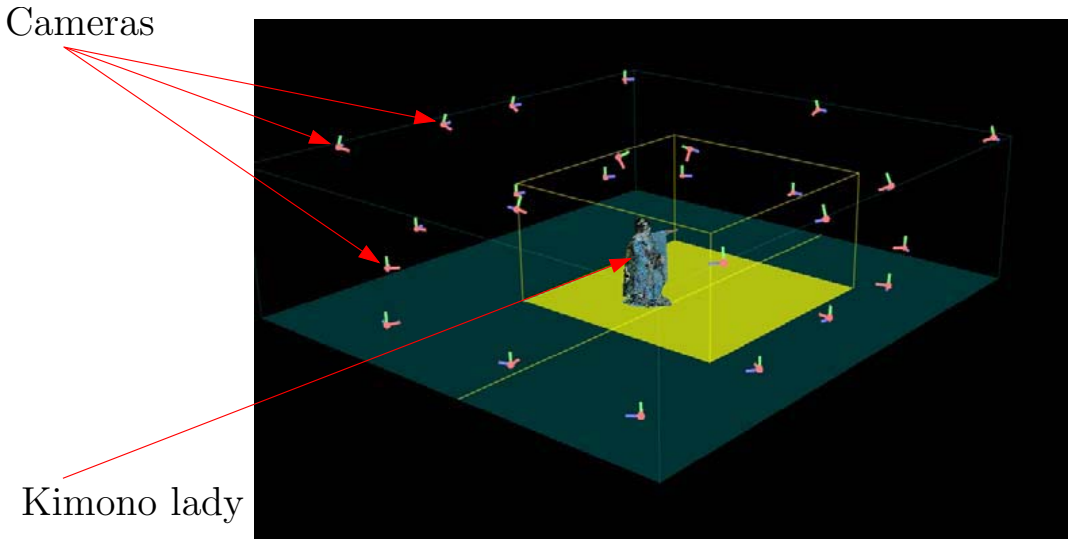


Figure 2.1. 3D environment.

2.1. Introduction

High-resolution images are a requirement for high-resolution visualization and 3D reconstruction applications. For texture mapping [24] and shape refinement [23], priority is given to closer and accessible (without occlusion) viewpoints and with adequate angles of view [26, 27]. An example of these applications is 3D video system (Figure 2.1). Several systems have been proposed such as [5], [9], [10], [11], [12], and [13]. These systems focus on capturing images of an acting human body and use a distributed static camera system for a real-time synchronized observation. While [10], [5], and [13] generate the final video offline, [8], [11], [12], and [14] employ a volume intersection method on a PC cluster in order to achieve a full 3D shape reconstruction in real time.

A static camera system requires a continuous coverage of the entire observable area. This prevents the resolution of the cameras to be increased without affecting the coverage of the scene. To overcome this limitation, I have been designing a 3D video based on an active camera system. With such a system, the area to be covered is the one occupied by the moving object and hence, it is unnecessary to continuously observe the entire scene. Thus, it becomes possible to increase the resolution of the camera with more freedom.

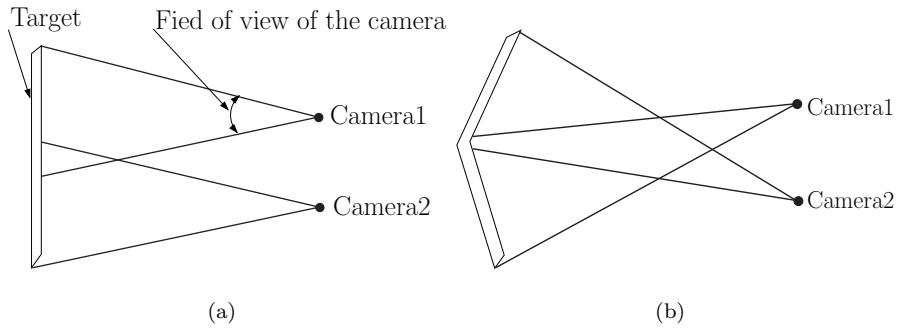


Figure 2.2. The configuration of the two cameras differs from (a) to (b), depending on the posture of the target.

The proposed system combines stationary cameras having wide fields of view and active PT cameras with narrow FOV (high resolution). The sparse distribution of the active PT wide-FOV cameras does not allow the continuous coverage of the entire scene. Thus it is necessary to control these cameras to gaze the object. The wide-FOV cameras, which can cover the entire scene with low resolution, are required to provide the necessary information about the object to steer the narrow-FOV cameras.

I consider the general case where the narrow-FOV cameras can capture only partial views of the object, but with high resolution. In such situations, a camera can have different visibility to different parts of the same object following its 3D shape, as shown in Figure 2.2. From this figure, we can notice that the camera system configuration needs to be adjusted depending on the position and the posture of the target. If we consider the texture mapping or the refinement of the reconstructed 3D shape, the camera is more useful in some regions than others. Therefore, I propose a multiple active PT camera assignment to assign each camera to one part in order to provide high-resolution images and with high visibility of the whole object [4, 6, 7]. The depth, occlusion, and angle of incidence constraints are taken into account in this assignment scheme. The wide-FOV cameras are charged to provide global information about the position and posture of the target, necessary for the assignment of wide-FOV cameras. According to the camera control module presented in Figure 1.1, the preprocessing corresponds to the reconstruction of the object's shape. As for the assignment sub-module, it

corresponds to the scheme that will be described in this chapter.

2.2. Multiple Active Camera Assignment

The goal is to control the PT wide-FOV cameras in order to provide high resolution images for 3D reconstruction. The quality of the reconstruction is related to the photometric consistency of the produced shape.

2.2.1 Preliminaries

In postprocessing, sophisticated 3D reconstruction algorithms, such as deformable mesh model[8] and space carving[23], are employed to produce the final 3D video using the images captured online. Basically, high-resolution images are used for this reconstruction. However, wide-FOV images serve to recover and reconstruct the non-covered areas when the narrow-FOV cameras do not cover the entire object. As an online processing, the cameras are controlled following the position and posture of the target.

The 3D reconstruction methods start by reconstructing the Visual Hull (VH) of the observed object. Using a set of images taken from different viewpoints, the VH is refined based on photometric consistency criteria. The quality of a reconstructed 3D shape is measured by how the shape is photometrically-consistent. The photometric consistency is related to three main factors: Occlusion, angle of incidence, and depth. This is the fact that the photometric consistency is directly related to the re-projection error which is inversely proportional to the projection area on a texture image, as reported in [26] and [27]. The projection area is related to the angle of incidence and the distance between the camera and the surface patch in question. That is, a surface that is far or viewed at an oblique angle from all the cameras will have only few pixels projected to it. However, the surface which is visible, parallel to the image plane and close to a camera will have more pixels that project to it. On the other hand, it is clear that the more the points that project to a given surface, the better it is reconstructed (see [26] and [27] for more details).

Starting from this, the multiple active camera assignment scheme should consider this photometric consistency in assigning each active PT camera to a specific

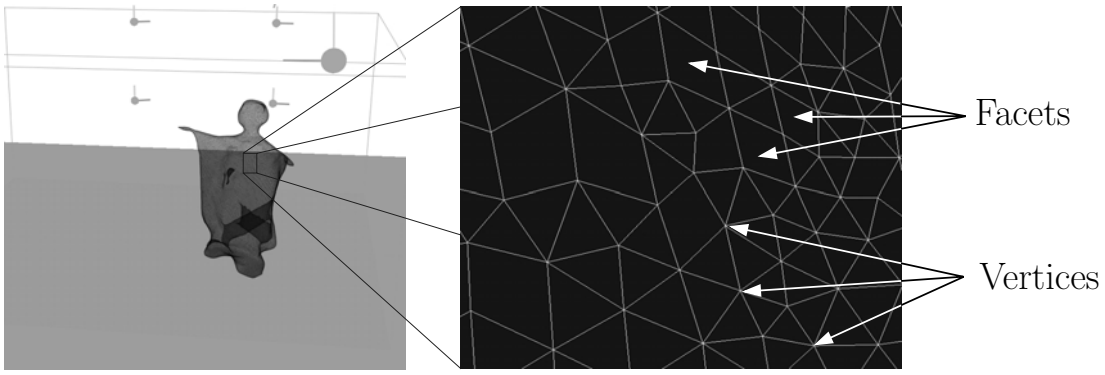


Figure 2.3. Input data : 3D mesh surface.

orientation toward the object. In other words, the goal is the automatic assignment of each narrow-FOV camera to an appropriate part of the target based on the photometric consistency requirements.

2.2.2 Assignment Constraints

The input is the 3D shape of the object in the form of a mesh surface $O(V, F)$; V and F are, respectively, the vertices and facets sets (Figure 2.3). Each facet is defined as a sequence of ordered vertices wherewith, the outward normal vector can be computed. In the rest of this chapter, we adopt the following notations:

$c \in \{1..N\}$: The N PT cameras of our system.

$f \in \{1..M\}$: The M facets composing the 3D mesh surface.

$\overrightarrow{n_f}$: The unit outward normal vector of a facet f.

o_c : The optical center of camera c.

t_f : The centroid of facet f.

2.2.3 Assignment Constraints

Based on the photometric consistency requirements, the constraints that should drive the assignment process are summarized in what follows:

1. Angle of incidence constraint: for one camera, only the facets whose outward faces are oriented to the camera are taken into account. To meet this

condition, a facet must have a negative dot product between its outward normal vector and the vector associated to the optical center of the camera and the centroid of the facet, as follows:

$$f \;\; visible\, from \;\; c \Rightarrow (\overrightarrow{n_f} \cdot \overrightarrow{o_c t_f}) < 0 \tag{2.1}$$

Furthermore, if both vectors are normalized, then the dot product can quantify the visibility (details in 2.4). That is, the higher the absolute value of the dot product, the better the visibility.

2. Occlusion constraint: for one camera, only non-occluded facets are considered. The set of facets $V(c)$ meeting the angle of incidence condition and not occluded with respect of a camera c, are provided by the related depth image D_c. We associate an index image I_c having as attributes the indices (identifiers) of these facets.

   $V(c)$ can be defined such that:

$$j \in V(c) \Leftrightarrow \exists_{x,y} I_c(x,y) = j \tag{2.2}$$

3. FOV constraint: only the region observed within the FOV of a given camera is taken into account while the whole surface is projected onto the image plane.

4. Depth constraint: the camera-object distance is taken into account in the assignment process. That is, a camera should have more chances to be assigned to closer regions.

5. 3D reconstruction constraint: this constraint requires us to have at least two views toward each surface point in order to allow the refinement of 3D shape .(e.g. shape by space carving[23]).

2.2.4 Overview of the Assignment Scheme

The preprocessing consists in reconstructing the shape of the object based on the wide-FOV images. The camera assignment is realized by analyzing the 3D shape and seeking the best camera set-up that allows the best visibility of all object parts, knowing that:

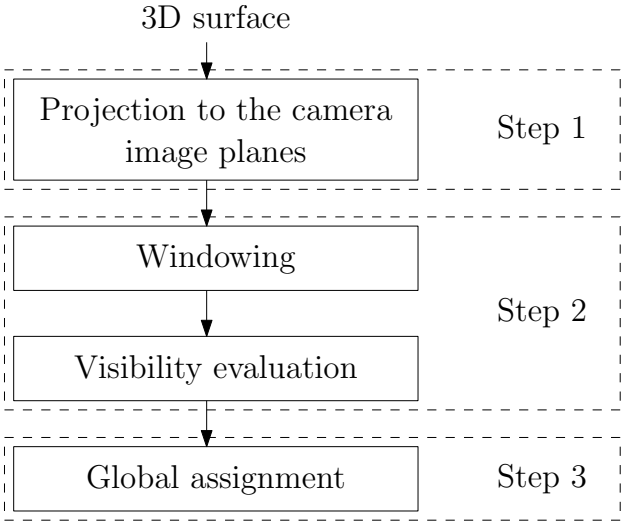


Figure 2.4. Flowchart of the assignment scheme.

- A camera can have a large number of possible orientations.
- A camera contributes in the visibility of the whole 3D surface depending of its orientation.
- For one orientation, the camera contributes in the global visibility through the subset of facets visible within its FOV.

In order to reduce the possibilities of one camera orientations, I introduce the windowing scheme that will be presented in section 2.3. As a result of this scheme, the facets concerned by each orientation are selected and involved in the evaluation of the orientation. For this evaluation, visibility quantification is introduced in section 2.4. As for the global assignment, it is the subject of section 2.5.

The proposed algorithm can be summarized in three steps (Figure 2.4):

1. Step1: the 3D shape is projected onto the panoramic [1] planes of all cameras as shown in the upper part of Figure 2.5.

[1] The panoramic plane of a given camera is a reference plane corresponding to a chosen camera orientation(e.g., (pan, tilt) = $(0, 0)$). All the images of this chapter correspond to this plane. See [22] for details.

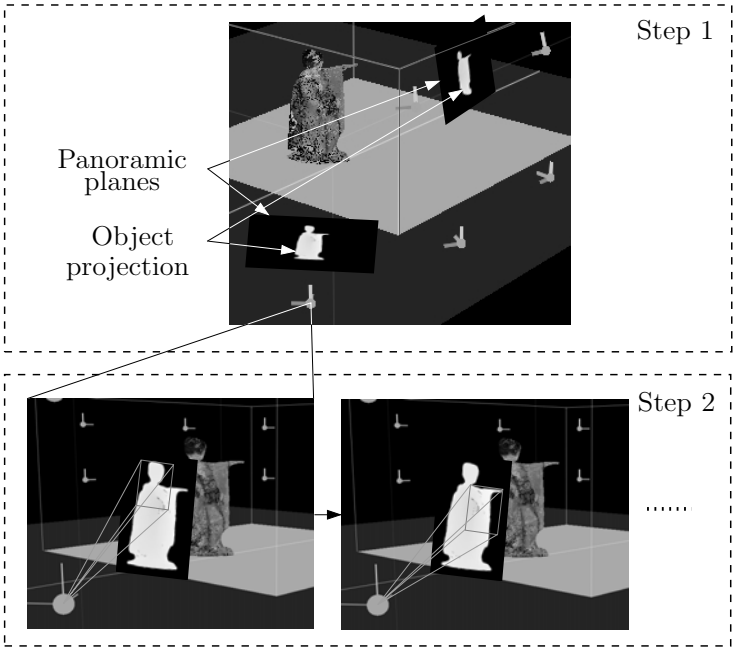


Figure 2.5. The first two steps of the proposed algorithm.

2. Step2: the orientations to the object regions are evaluated for each camera with respect to its FOV. In the panoramic image, each orientation corresponds to a window determined by the camera FOV. For a given window (orientation), the evaluation concerns the visibility to the corresponding 3D surface region (lower part of Figure 2.5).

3. Step3: the best assignment configuration between the cameras and the windows that maximizes the global visibility is sought.

2.3. Windowing Scheme

The goal of the windowing scheme is twofold: 1) select from the huge possible orientations, a small set that cover the object view based on the FOV constraints, and 2) select for each orientation the set of facets that satisfy the angle of incidence and occlusion constraints. In addition, we consider the fact that under self-occlusion conditions, a camera can have a view of regions with discontinu-

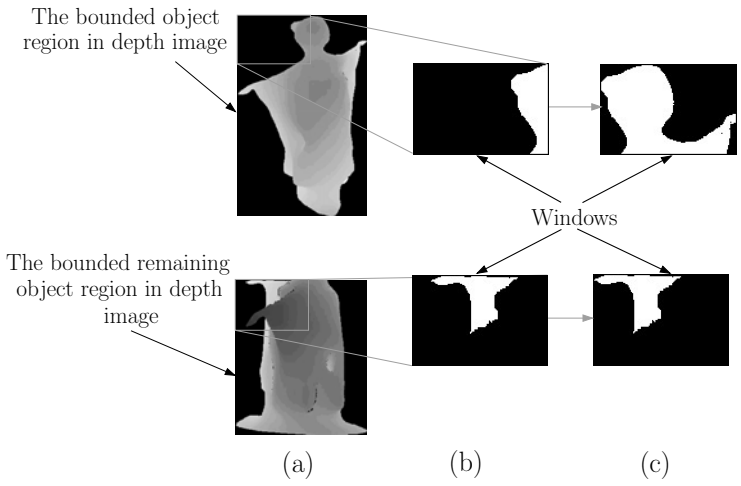


Figure 2.6. Windowing scheme: From (a) to (b) a flood filling is applied, and an adjustment is applied to get (c). The top cycle corresponds to the first windowing iteration for a given camera. The missing part in the depth image of the bottom cycle was windowed and deleted in previous iterations.

ous depths within the same FOV. We impose the connectivity as an additional constraint to assign a camera to one region with continuous depth.

To achieve such a goal, I propose to gradually splitting the depth image into several windows. After setting a window to the offset of the bounding rectangle of the object region in depth image, a flood-filling is applied to the region of interest defined by the window in order to extract one connected region. Then, the window position is repeatedly readjusted by shifting in order for the window to fit the best with the connected region. For each new position, the flood-filling is reapplied to find the new limits of the connected region. Finally, this region is withdrawn from the depth image and used as a mask for the index image in order to select the facets to be involved in the visibility evaluation.

Let us denote by $w_k^c|_{k=1..K}$ the K resulted windows, and (x_k, y_k) their respective off-set addresses with respect to the depth image. If a given point has (x, y) as coordinates in the window w_k^c, then its coordinates in the original image are $(x + x^k, y + y^k)$. If this point belongs to the connected region, then $w_k^c(x, y) > 0$ (The point (x, y) in the window w_k^c is strictly positive).

The set of facets $V(c)$ is splitted into $V(c,k)|_{k=1..K}$ such that:

$$\begin{cases} \bigcup_k V(c,k) = V(c) \\ \\ \wedge \\ f \in V(c,k) \Leftrightarrow \\ \quad \exists_{x,y} \begin{cases} w_k^c(x,y) > 0 \\ \wedge \\ I_c(x_k + x, y_k + y) = f \end{cases} \end{cases} \tag{2.3}$$

The proposed algorithm is summarized in what follows (Figure 2.6):

1. Define the bounding rectangle of the object region in the depth image if it is the first iteration or of the remaining regions of the object if not. The bounded region will be set as the region of interest (ROI) of the depth image, as shown in Figure 2.6-a.

2. Set a window to the offset of the ROI (Figure 2.6-a).

3. Repeat:

   - Apply a flood-filling starting from the top-left seed point of the object region in the window, as shown in Figure 2.6-b.
   - Shift the window along the bounded region in order to bring the outer contours of the connected region to the borders of the window. If we consider the top window of Figure 2.6-b, the window has to be shifted in the right direction. After applying the flood-filling, we get the top window of Figure 2.6-c.

   Until the top and left borders of the window meet the external contour of the connected component. If it is the last horizontal window, we consider the right border of the window instead of the left. Similarly, we consider the bottom border instead of the top if it is the last vertical window (Figure 2.6-c).

4. Save the window and delete the area corresponding to the connected component from the depth image. This window will serve as a mask to the index image in order to select the corresponding facets set.

5. If the depth image is not empty, then goto 1.

At the end of this process, a set of windows for each camera is obtained. Each camera orientation is evaluated using the selected facets, as explained in the following section.

2.4. Visibility Quantification

The possible orientations for each camera been defined, the next step will be the evaluation of each of them based on the visibility analysis. Based on the constraints derived from the photometric consistency requirements, I propose to quantify the visibility at three levels: Facet-wise, local, and global.

2.4.1 Facet-wise Visibility Quantification

The facet-wise visibility is the direct application of the angle of incidence constraint and concerns the visibility of a given facet from a given camera. It can be expressed by the absolute value of the dot product between the unit normal vector of the facet in question in one side and the normalized vector connecting the optical center of the camera and the centroid of the facet in the other side.

If the unit vector, corresponding to the optical ray of a camera c_i toward t_j, is:

$$\vec{r_{c,f}} = \frac{\overrightarrow{o_c t_f}}{\left\| \overrightarrow{o_c t_f} \right\|}$$

Then, the facet-wise visibility is given by:

$$F(c, f) = \left| \vec{r_{c,f}} \cdot \vec{n_f} \right| \tag{2.4}$$

2.4.2 Local Visibility Quantification

For one camera, the local visibility level concerns the visibility toward each of its windows selected by the windowing scheme. For a given window, it involves the facet-wise visibility of all facets related to the window. The formulation of

the local visibility is very important, since it is on which the global assignment is based. The straightforward way to quantify the local visibility is to sum the facet-wise visibility of the corresponding facets. Though simple, this solution has the following disadvantages:

1. A narrow region made of tiny facets is given similar evaluation as a wide region with large facets if the two regions have a similar number of facets.
2. One region is given the same evaluation whatever its distance from the camera.

In order to overcome these disadvantages, the proposed formulation should:

1. Respect the aforementioned depth constraint.
2. Involve the facet area. That is, the contribution of each facet in the local visibility should be proportional to its surface area.
3. Be normalized.

Let us denote by:

- $\hat{D}(c,k)$, the depth of the nearest point of the 3D surface within a window w_k^c with respect to a camera c such that:

$$\hat{D}(c,k) = MIN_{f \in V(c,k)} \left(D(c,f) \right)$$

where $D(c,f)$ denotes the depth of f with respect to c.

- $\bar{S}(c,k)$, the area of the 3D surface related to the window w_k^c of a camera c such that:

$$\bar{S}(c,k) = \sum_{f \in V(c,k)} S(f)$$

where $S(j)$ denotes the surface area of a facet f_j in the 3D space.

The local visibility of a window w_k^c from a camera c is given by:

$$L(c,k) = \frac{\hat{D}(c,k)}{\bar{S}(c,k)} . \sum_{f \in V(c,k)} \frac{F(c,f).S(f)}{D(c,f)} \tag{2.5}$$

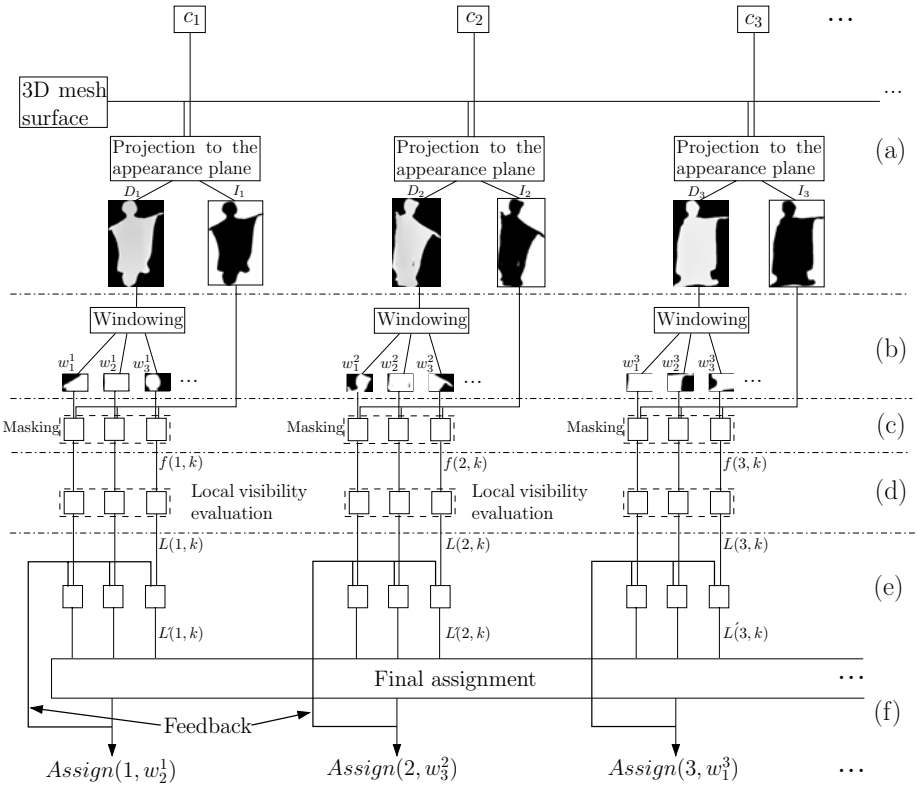


Figure 2.7. Assignment scheme: The different steps of our proposed scheme. $f(i)$, $f(i,k)$, and $L(c,k)$ refer respectively to the set of facet visible from camera c, these corresponding to a window w_k^c, and the local visibility corresponding to c and w_k^c.

For a set of facets $V(c,k)$, the local visibility is the normalization of the sum of the scaled facet-wise visibility of these facets. The scale is the ratio between the 3D surface area of the facet and the depth of its centroid with respect to the camera in question. As for the normalizing factor, it is the ratio between the minimum depth of the 3D surface and its 3D area.

2.4.3 Global Visibility Quantification

The purpose of the assignment process is to maximize the global visibility of the 3D surface. After the system is set to a given configuration, this global visibility

can be expressed by averaging the local visibility of all cameras. Let us assimilate the assignment scheme by a function A that associates to a camera c the assigned window w_q^c. $\dot{A}$ is the set of couples (c, q) such that:

$\dot{A} = \{(c, q)/A : c \rightarrow Assigned\ \ to\ \ w_q^c\}$

Then the global visibility is given by:

$$G = \frac{1}{N} \sum_{(i,j) \in \dot{A}} L\,(i, j) \tag{2.6}$$

2.5. Global Assignment Scheme

After having addressed the windowing and visibility quantification issues, we are now able to establish a global assignment scheme. The purpose is to direct each camera toward a specific part of the object in order to get a high visibility of the whole object. This can be reduced to select one window for each camera so as to maximize the global visibility.

The straightforward solution is to apply a full search by checking all possible configurations between the cameras and their respective windows and choose the one that maximize the global visibility. In addition to its complexity ($O(n^k)$: k windows for each of the n cameras), this solution does not handle an important requirement which is the coverage problem. The solution has to cover the largest possible region of the 3D shape. This requirement can be achieved by minimizing the overlapping area of the shape surface. With respect to the 3D reconstruction constraint, one region is no longer considered for assignment after it is assigned twice.

Therefore, I propose a method which, though yields a local optimum, handles the two requirements, namely, 1) high visibility and 2) coverage. For the first requirement, I propose to iteratively assign the camera to the window with the highest local visibility among all cameras and all their respective windows. This camera will not be considered in the following iterations. As for the second requirement, we impose the following condition:

- *After been assigned twice, a facet is removed from the 3D surface and accordingly, the local visibility for all windows of all cameras is updated.*

If no camera is left while the facet set is not empty, then camera deficiency is declared. This case does not influence the assignment process since the reconstruction is ensured by the wide-FOV cameras. In the offline processing, the non-covered areas are reconstructed using the wide-FOV cameras images, or estimated from the previous or next frame.

As a summary of the assignment algorithm, the assignment is repeatedly processed on the remaining cameras from the last step and the non-empty object region which has been assigned at most once as long as cameras are available, as follows:

1. For each camera c, compute the set $V(c)$: Project the 3D mesh surface to the panoramic plane and build the depth and index images D_c and I_c respectively, as shown in Figure 2.7-a.

2. Split $V(c)$ into $V(c,k)$ (Figure 2.7-b,c): Apply the above-mentioned windowing scheme to get for each camera c, the corresponding windows w_k^c (Figure 2.7-b). Then, mask the index image I_c using these windows in order to get the sets $V(c,k)$ (Figure 2.7-c).

3. For each couple (c, w_k^c), calculate $L(c,k)$: Evaluate the local visibility between all cameras and their respective windows (Figure 2.7-d).

4. Repeat:

   (a) Select the pair (c, w_k^c) having the highest local visibility and assign the camera c to the window w_k^c.

   (b) Delete all facets chosen twice and accordingly, update L (the local visibility) for all windows (of all cameras) comprising the deleted facets.

   until no camera or no facet left (Figure 2.7-f).

2.6. Applicability of the Active Camera Assignment in 3D reconstruction of Dynamic Scenes

For 3D reconstruction of dynamic scenes, such as 3D video, the processing time is very important. For this, it is necessary to consider the processing time for

the assignment as well as for the online shape reconstruction. In the assignment scheme described so far, the visibility information is temporally independent. Consequently, the cameras can undergo, repeatedly, large inter-frame angular displacements. This can have an influence on the performance of the system. For the sake of less large displacements and smooth camera movements while respecting the system requirements presented so far, the assignment at a given frame should be inferred from the last camera set-up. This process is illustrated by the feedback in Figure 2.7. As shown in this figure, I propose to update the local visibility of one camera using its last orientation. Therefore, I introduce a new parameter in the local visibility such that:

$$L'(c,k) = \lambda \cdot L(c,k) \tag{2.7}$$

As far as the purpose is an optimized camera movement, the parameter λ should favor the new orientations having smaller angles from the last orientation over the ones with larger angles. For this purpose, we set λ as follows:

$$\lambda = 1 - \beta(1 - \left|\overrightarrow{R_{c,w}} \cdot \overrightarrow{O_c^{t-1}}\right|) \tag{2.8}$$

Where:

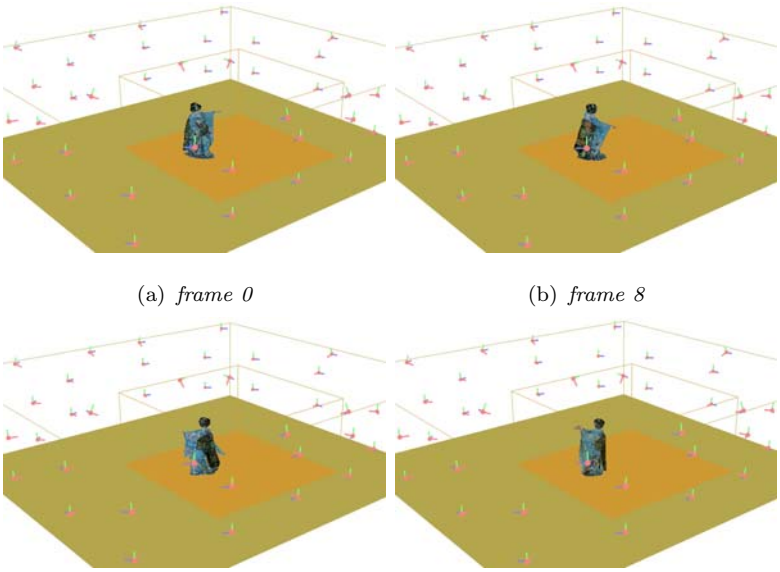
- $\overrightarrow{R_{c,k}}$ is the unit vector corresponding to the optical ray of a camera c toward a window w_k^c,
- $\overrightarrow{O_c^{t-1}}$ is the last orientation of camera c,
- And $\beta \in [0,1]$ is a predefined parameter.

Thus, the new local visibility can be written as:

$$L'(c,k) = L(c,k) \cdot (1 - \beta(1 - \left|\overrightarrow{R_{c,k}} \cdot \overrightarrow{O_c^{t-1}}\right|)) \tag{2.9}$$

β expresses the priority given by a user to the camera movement optimization. We can notice that:

- $\beta = 0 \Rightarrow L'(c,k) = L(c,k)$: optimization ignored.
- $\beta = 1 \Rightarrow L'(c,k) = L(c,k) \cdot (\left|\overrightarrow{R_{c,k}} \cdot \overrightarrow{O_c^{t-1}}\right|)$: the highest priority is given to camera movement optimization.



(a) *frame 0* (b) *frame 8*

(c) *frame 15* (d) *frame 26*

Figure 2.8. Selected frames from the same viewpoint.

The smoothness of the camera movement resulted from this optimization is related to the fact that a camera has more chances to keep assigned to the same region or to a close one than to farther regions over frames. A large transition is decided only when the gain in visibility is worth. This, in addition to the overall assignment scheme, will be evaluated in the next section.

2.7. Experimental Results

In order to evaluate the effectiveness of the assignment scheme, I conducted a set of experiments using the data provided by Matsuyama Laboratory of Kyoto University. The data are a sequence of 26 visual hulls of a dancing Kimono lady (Maiko) corresponding to 26 frames. Figure 2.8 shows selected frames (frame0, 8, 15, and 26). 25 wide-FOV cameras, with focal lengths varying between 5 and $11mm$ and spread around a room of $9m$ long, $8.5m$ large, and 2.7 height, were used for the reconstruction of the visual hulls of the target [8]. I built a simulation environment of the real scene, illustrated in Figure 2.1, with 25 high-resolution

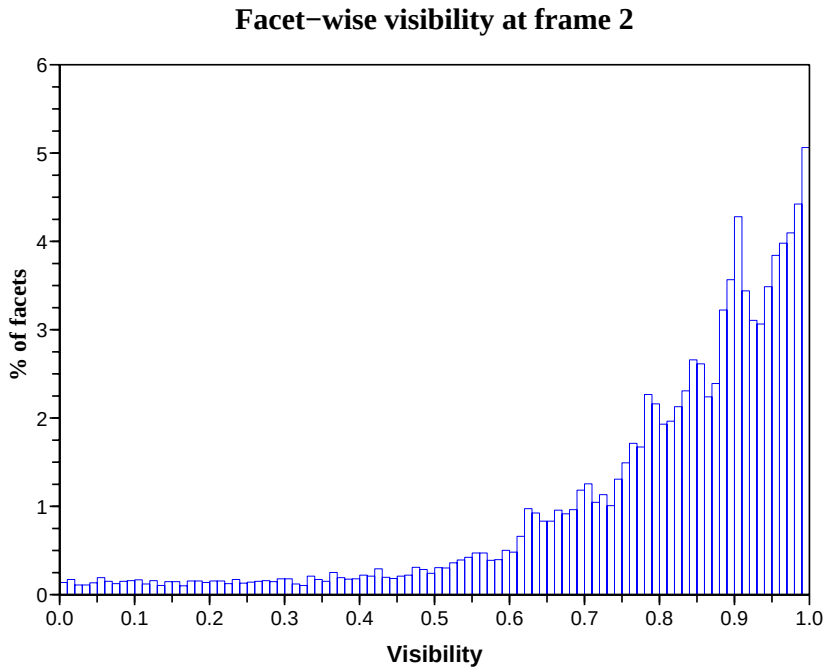


Figure 2.9. Facet-wise visibility histogram.

cameras with $35mm$ lenses. The assignment scheme is implemented on a PIV PC with 1 GB of RAM and running Windows. I evaluated the system according to 3 aspects: 1) visibility, 2) camera movement, and 3) processing time.

2.7.1 Visibility Evaluation

At each frame, I evaluated the three visibility levels, namely, facet-wise, local, and global. Figure 2.9 and Figure 2.10 show the result from a selected frame. The facet-wise visibility is shown in Figure 2.9 where we can notice that most of facets (91.56%) have a visibility higher than 0.5 which means with an angle of incidence smaller than 45 degrees. In Figure 2.10, the local visibility for the 25 cameras as well as the global visibility are presented. The local visibility varies between 0.65 and 0.94 while the global visibility is 0.82. If the average angle of view to a surface point can be approximated by arccos of the global visibility, then it is about 35 degrees.

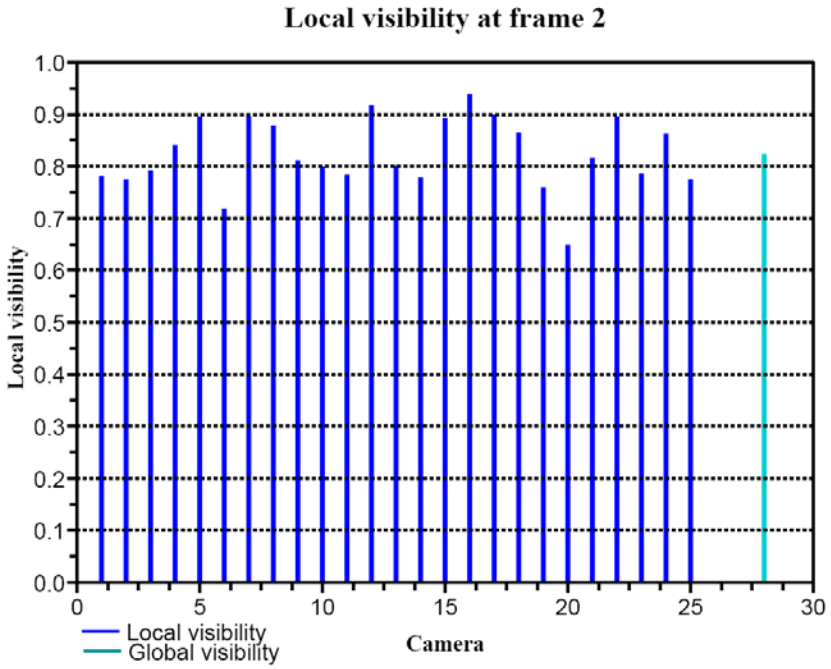


Figure 2.10. Local visibility evaluation.

2.7.2 Camera Movement Optimization

The second aspect of the evaluation concerns the influence of camera movement optimization introduced in section 2.6. In order to show the relevance of this optimization, I evaluated the gain in terms of camera movement against the loss in visibility. The loss of visibility within 26 frame is shown in Figure 2.11. The mean global visibility when the optimization is ignored, as shown in Table 1, is 0.725 while it is 0.723 when the optimization is considered. The difference is 0.003 which represents 0.414% of the mean global visibility of the first case. As for the gain in camera movement, I accumulated the angular displacement for each camera within the 26 frames in both cases and the result is shown in Figure 2.12 and Table 1. The mean angular displacement in the first case (without optimization) is 319.172 degrees, while it is 190.722 degrees in the second. The difference is 128.367 degrees which represents 40.22%. Thus, we can say that a clear optimization of camera movement has been obtained at the price of a minor loss in visibility.

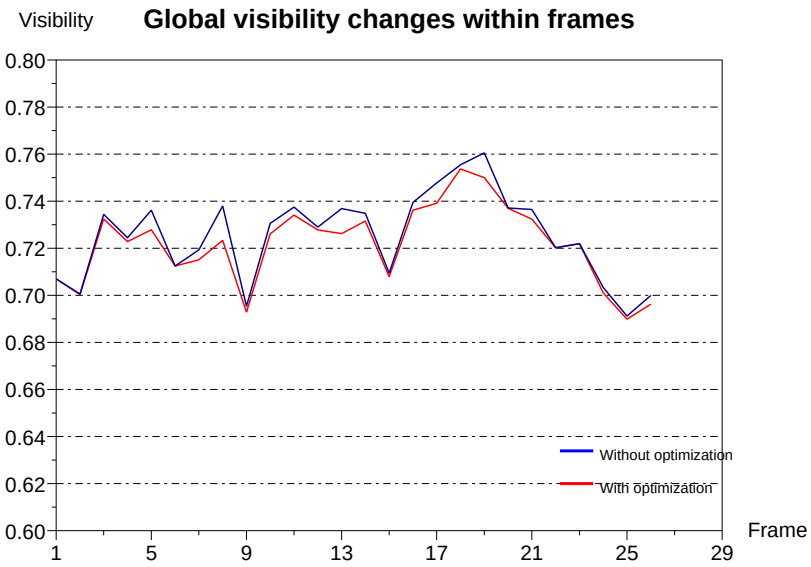


Figure 2.11. Global visibility changes within frames with/without camera movement optimization.

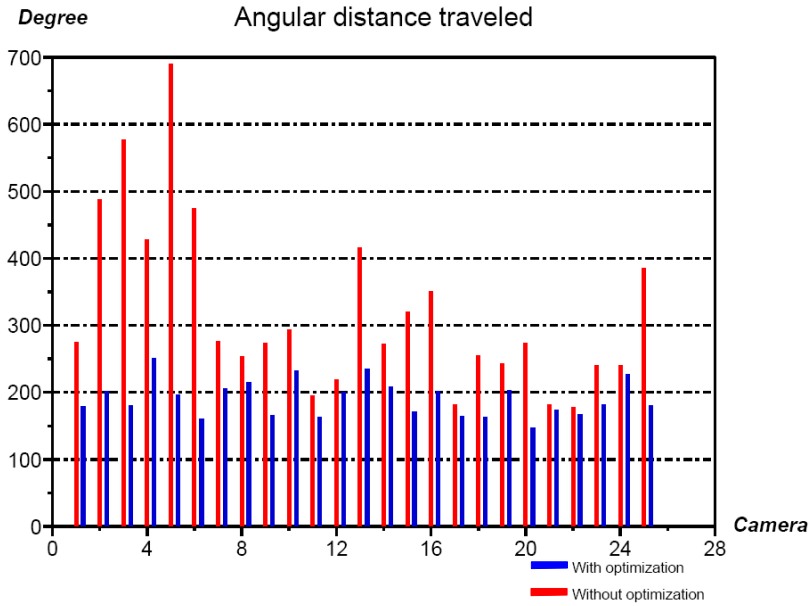


Figure 2.12. The angular displacement each camera before and after camera movement optimization.

Table 2.1. Summary of the gain in camera movement and the loss of visibility within the 26 frames.

	Mean global visibility	Mean angular displacement
Without optimization	0.725	319.172°
With optimization	0.722	190.805°
Difference	0.003	128.367°
%	0.414%	40.22%

Table 2.2. Processing time for windowing, local visibility, and global assignment steps

Step	Windowing	Local visibility	Global assignment	Total
Processing time (ms)	1073	21	47	1141

2.7.3 Processing Time

The third aspect of our evaluation is the processing time. Table 1 shows the cost in processing time for windowing scheme, local visibility quantification, and the global assignment. We can see that the most expensive step is the windowing scheme. The windowing scheme and the local visibility quantification are designed to run for each camera independently and in parallel. The global assignment is executed only once based on the results from all camera hosts. The estimated total processing time, without considering any additional factor (e.g. data flow,...), is the sum of the three entities shown in Table 2 which gives $1141ms$. It is clear that this processing time does not allow an implementation in a real system.

The assignment scheme in operated on the visual hull of the target. This visual hull is an approximation of the 3D object and does not represent its details. This means that it is possible to get the same assignment result if the shape is scaled down. If so, then the processing time might be shorter. I applied the

Table 2.3. Processing time for windowing, local visibility, and global assignment steps at different scales

	Processintg time (ms)			
Step	Windowing	Local visibility	Global assignment	Total
Scale =1	1073	21	47	1141
Scale =1/2	168	10	47	225
Scale =1/4	46	3	47	96

assignment process after scaling down the surface half and fourth the original size(1/2, 1/4) and in fact, the same camera orientations have been obtained. As for the processing time, it is summarized in Table 3. The processing time is shorter as the scale is lower.

2.8. GPU-based Assignment Scheme

The most time-consuming steps of the assignment process are windowing and local visibility evaluation. The windowing scheme was introduced to reduce the camera orientation space which is very large. We will see how the Graphics Processing Unit (GPU) can be used as a powerful coprocessor to evaluate all possible camera orientations in an interactive processing time.

2.8.1 GPU: A Powerful Tool for General-Purpose Applications

The GPU,which is no longer just for graphics, is becoming an alternative solution for many real-time applications. The huge memory bandwidth, computational power, and the programming flexibility have made of the latest GPUs real massively parallel co-processors. Taking advantages of that, many general-purpose (GP-GPU) applications have been implemented. The disadvantage of GPU re-

sides, mainly, in the difficulty of programming. This is due to the difficulty of grid-based mapping of sequential solutions, and the fact that GPU is accessed only through an API, such as OpenGL and Direct3D. GPU can be thought of as a pipeline, where the input is the 3D scene in the form of vertices and triangles (facets). The vertices and their attributes, such us 3D position, lighting, material, normal, color, etc., are first transformed with respect to the viewpoint (camera), using the projection matrix. Afterward, the transformed shape is rasterized to fragments each of which corresponds to a pixel on the screen (rendering buffer). Early Face-culling and depth test play the roles of incidence angle and occlusion constraints of our CPU-based scheme. The fragment shader has as input these fragments, and has access to several textures to build the final scene view to the rendering buffer.

Two programmable points are available: 1) Vertex level, and 2) fragment level. GP-GPU applications escape, usually, the vertex level to work only with the fragment shader for the following reasons:

1. More pipelines are available for fragment that for vertex levels. For instance, NVIDIA 7800 GTX is equipped with 8 vertex and 24 fragment pipelines.

2. More data are processed at vertex level (3D scene) than at fragment level (projected scene + face-culling + Z-culling).

3. The vertex shader has no access to textures, where these what are matrices to CPU.

4. GP-GPU applications tasks are grid-based implemented and no transformation is required. That is, there is no need for the vertex shader. A full-screen quad drawn in projective geometry is sufficient.

2.8.2 GPU-Based Assignment Scheme

The first step in the assignment scheme is the projection to the image planes of all cameras. On GPU, this can be done using a vertex shader. This shader has as input the camera matrix to transform the 3D positions and normals of the 3D object to the camera coordinates. The depth test should be enabled with

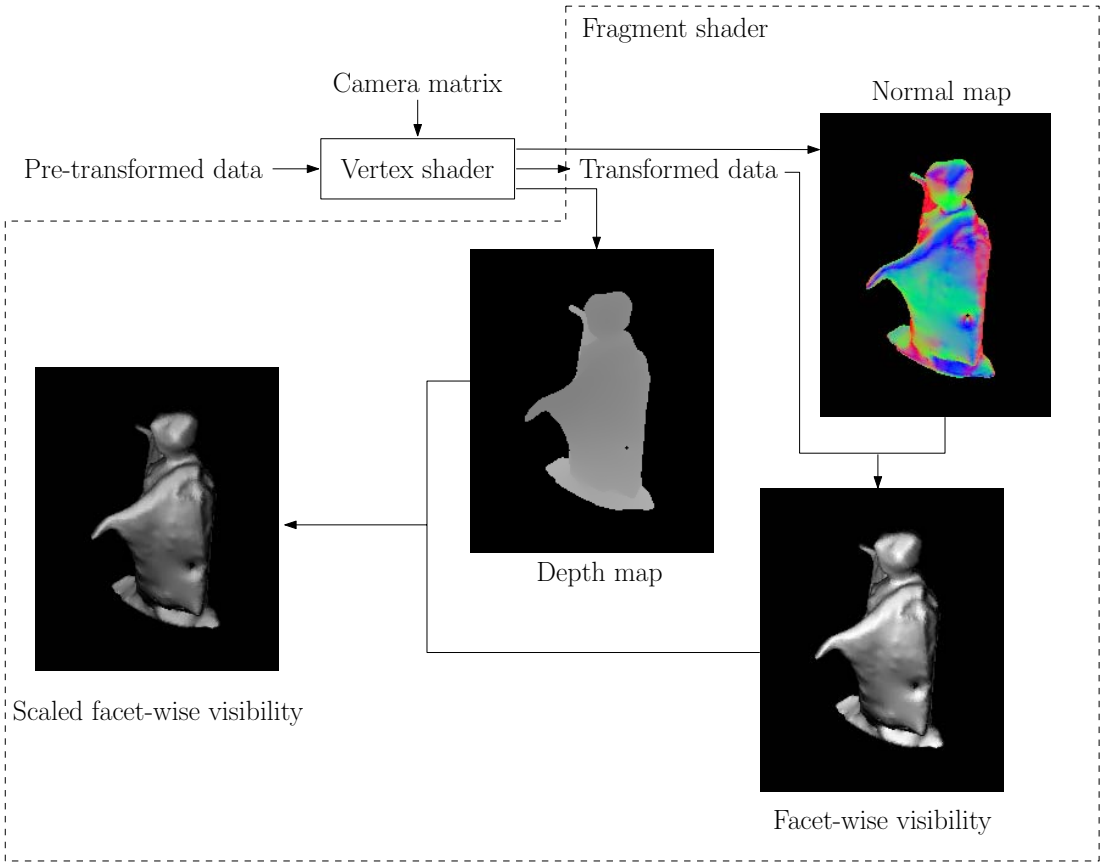


Figure 2.13. Scaled facet-wise rendering: Data flow through the GPU pipeline.

'less or equal' test function in order to cast away all occluded facets (occlusion constraint). The 3D, depth, and normal maps are, subsequently, passed to the fragment processor. Using these three maps, the facet-wise visibility scaled with the depth is returned, see Figure 2.13.

The computations at fragment level are invoked at pixel level. That is, all fragments (pixels) undergo, simultaneously, the same processing. As a result, the scaled facet-wise visibility is computed for all visible non-occluded facets. The surface area of a facet is taken into account implicitly since larger facets are rasterized to more fragments and considered as much. Moreover, the normalization is obtained implicitly with respect to the predefined depth range.

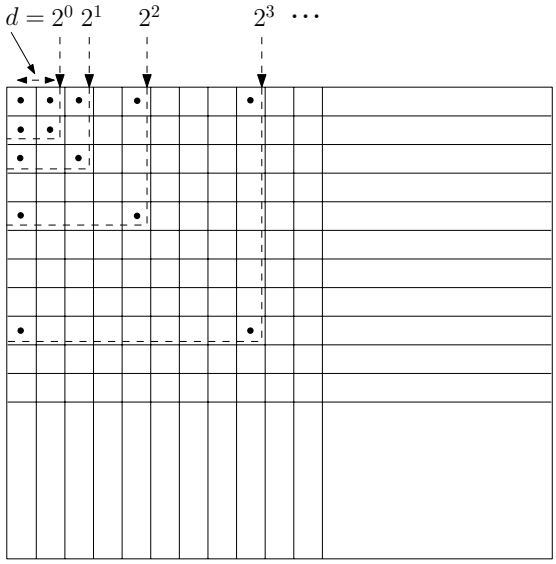


Figure 2.14. Multi-pass pyramidal filtering: The same operations are processed for each grid element.

2.8.3 Local Visibility = Multi-pass Pyramidal Filtering

The evaluation of one camera orientation corresponds to the sum of the scaled facet-wise visibility values within the FOV of the camera. The computation consists in applying a mean $w \times h$ filter to each image point. Applied to one point, the result is the average value of the $w \times h$ points surrounding the point under consideration. Though the filter is separable, the calculation sequentially is expensive in processing time. Also, no similar filter is predefined to achieve that on GPU. Therefore, I propose a multi-pass pyramidal filter. It consists in gradually averaging each image point (i, j) and its $(i, j + 2^k)$, $(i + 2^k, j)$, and $(i + 2^k, j + 2^k)$ neighbors. Where $k = 1..n$ is the pass index and n is the number of passes, see Figure 2.14.

The multi-pass pyramidal filter is useful only if the size of the FOV is power of 2. In order to generalize it to free size FOV, the viewport of the camera is scaled to render the FOV of the camera a power of 2 on the reference image. This can be achieved by shrinking the viewport of the camera if we want to scale up the FOV, or the other way around. If we don't want to lose the details of the shape, it is

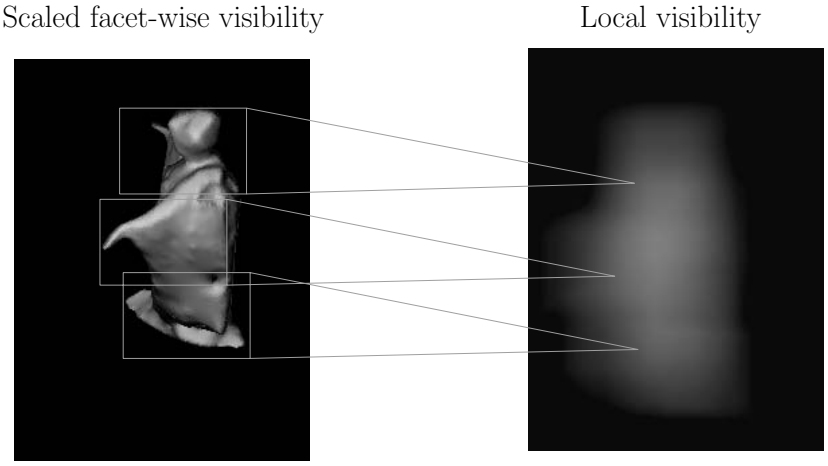


Figure 2.15. Local visibility rendering.

better to scale up the FOV. Since we are working with a coarse reconstruction, I chose the first way (scale down the FOV). If the resultant window size is $2^n \times 2^n$, n passes are required to achieve the local visibility evaluation.

After applying the multi-pass filtering, the top left $(W-w)\times(H-h)$ points of the rendered texture will refer to the evaluation of all camera orientations. The coordinate of the pixel refer to the off-set of the window corresponding to the FOV of the camera, see Figure 2.15. For each camera, the local visibility texture is saved in the GPU memory. The same texture is accumulated in two channels of an RGBA-texture such that: A value is saved in the R-channel if it is higher that the one already present. If so, the camera index in saved in the G channel, see Figure 2.16. This will help in what follows.

2.8.4 Global Assignment

After processing all cameras, the highest value of the common texture refers to the best orientation among all cameras. The camera indicated by the G-channel is assigned to the orientation determined by the off-set of its window retrieved from the R channel. Another rendering pass is needed in order to return the concerned facets. This is done through 3D surface rendering to the selected camera, with the facets indices as their colors. A returned index will be used to flag the facet as

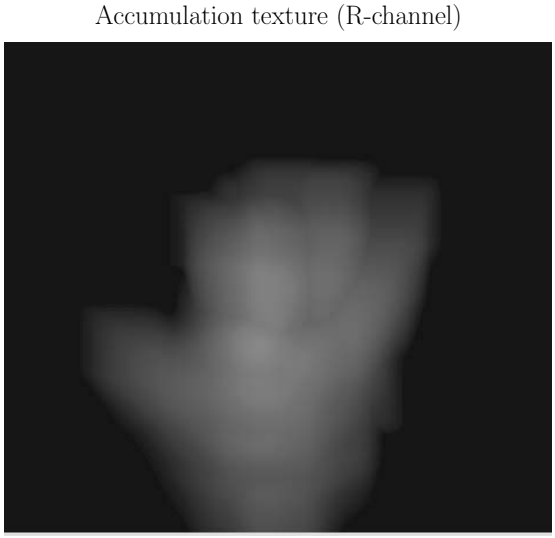


Figure 2.16. Common texture: Local visibility accumulation.

assigned once (it will be removed from the surface after the second assignment).

So far, one camera is assigned. The same process is repeated to assign the remaining cameras with the following considerations:

- After each assignment step, only the removed facets are rendered. This is significantly faster than rendering the remaining facets.
- The rendered local-visibility texture is used to update the saved one by subtracting the former from the later. Afterward, the resultant texture is accumulated to the common texture. The remaining steps remain unchanged.

Figure 2.17 summarized the GPU-based assignment scheme.

2.8.5 Implementation

For the implementation, I used OpenGL as an API and C-like shading language of NVIDIA (CG). Instead of rendering to the screen, I used a Frame Buffer Objects (FBO). FBO, also called off-screen, is an extension that allows the use of textures as rendering targets, instead of the buffer provided by the window system. FBO

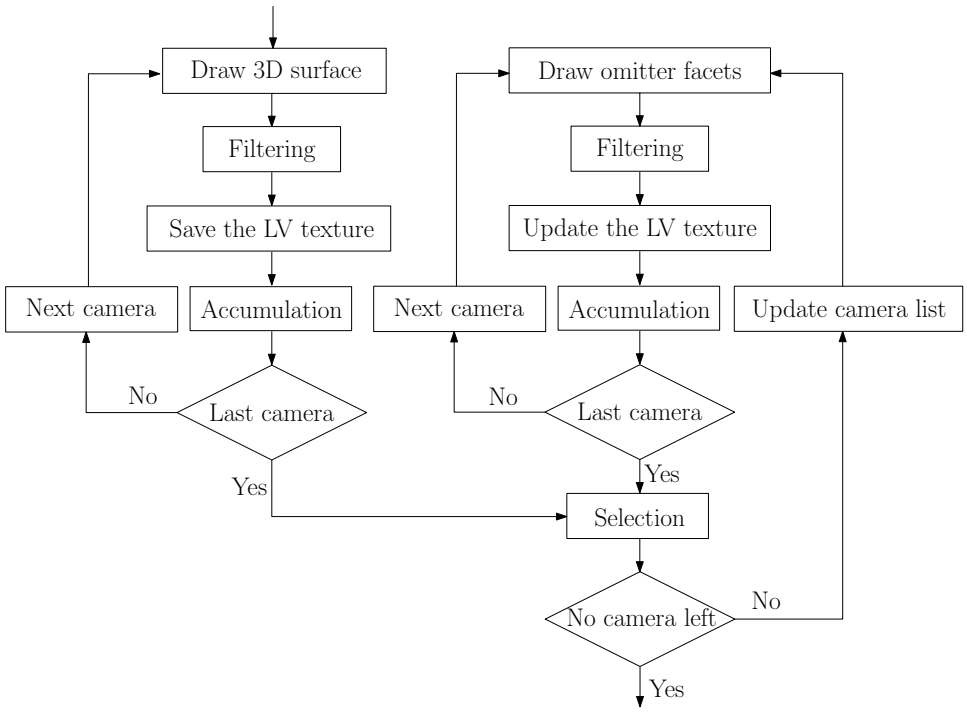


Figure 2.17. GPU-based global assignment flowchart.

has few attachment points for textures. In order to avoid GPU-CPU and CPU-GPU transfers, I made use of *pingpong* technique. This technique is very useful for multi-pass rendering and consists in attaching two textures to a FBO; one as input texture and the other as rendering target. After each pass, the two attachment points are swapped to inverse their roles.

I implemented the CPU part on a $P4$ PC with 1 GB of memory, running windows, and equipped with NVIDIA 7800 GTX. Under the same condition of the first experiment, I obtained a processing time of $574ms$ for the overall assignment (25 cameras). If all PC-cluster hosts are equipped with similar graphics boards, even much faster processing time could be obtained. Another alternative way to speedup the processing time is to use one assignment host equipped with several GPUs.

2.9. Conclusion

In this chapter, I presented an assignment scheme to control multiple active PT cameras for 3D reconstruction applications. I showed how active high-resolution cameras are controlled to gaze at different parts of an object based on the visibility analysis of a 3D shape reconstructed using a set of wide-FOV camera images. Since 1) a camera can have a large number of possible orientations, and 2) evaluating all these orientations results in a processing time not suitable for real-time applications, I presented a windowing scheme to reduce these possibilities to a small set. I showed how to evaluate one camera orientation based on the visibility of all facets of the 3D shape within the FOV of the camera. For this, I quantified the visibility at three levels; facet-wise, local and global visibility based on the photometric consistency requirements. Then, I presented a method to assign each camera to one orientation in such a way to get a high global visibility.

For practicability in 3D reconstruction of dynamic scenes, the processing time in an important factor to be considered. I introduced the last camera orientation in the local visibility evaluation in order to give more importance to short displacements if the gain in visibility is not significant. The goal was to get a smooth and optimized camera movement. Finally, I showed how to use the GPU as a general-purpose co-processor to make possible a fast evaluation of all possible camera orientations.

In addition to the assignment scheme, the 3D reconstruction of the shape in the preprocessing step must be done in nearly real time if we project an application in dynamic 3D reconstruction. Most of the real-time shape from silhouette methods produces a volumetric representation of the object. However, this representation is not suitable for visibility analysis. In chapter 3, I will introduce a new surface-based visual-hull reconstruction that can run in an interactive frame rate.

Chapter 3

Preprocessing For Active Camera control: An Accelerated Surface-Based Shape from Silhouettes

In this chapter, I consider the preprocessing sub-module of Figure 1.1. I describe a new method for shape reconstruction from a set of silhouette images. I propose a method for a fast traversing the depth layers of the projected viewing cones from all viewpoints based on the principle of the Constructive Solid Geometry (CSG). Taking advantage of the growing Graphics Processing Units (GPU), the proposed CSG-like method achieves a full reconstruction of VH, rather than rendering a looking of the VH from a virtual viewpoint, in an interactive frame rate. For each viewpoint, the viewing edges are computed separately in a parallel manner. Subsequently, the edges are merged together to generate the final surface-based Visual Hull (VH). The method was tested on several datasets including real data and the results will be presented in this chapter.

3.1. Introduction

Acquiring the 3D shape of a real object is a key issue in a variety of modeling and 3D multimedia applications. The straightforward estimation of this shape can be obtained from silhouette images of the object taken from different viewpoints. This concept was first introduced by Baumgart [32], and later given the name of Visual Hull (VH) by larentini [31]. Based on this concept, the VH is the maximal approximation of the object that reproduces the silhouettes of all viewpoints.

Estimating the 3D shape from silhouette images was motivated by the ease of obtaining silhouette images, especially in indoor environments with known camera parameters, and by the ease of implementation. Several methods have been proposed for VH reconstruction and/or rendering. Depending on the application, the VH hull is processed differently. For visualization applications, such as new viewpoint synthesis or interactive rendering, the exact reconstruction of the VH is not required. Image-based VH (IBVH) technique [33] suffices to generate the VH looking from a desired view. Hardware-acceleration of IBVH was proposed through texture mapping-based visual cone trimming [39] or direct CSG rendering [36, 37, 38] to speed up the processing.

Applications such as object modeling and 3D digital archiving require a full reconstruction of the object's shape. Voxel-based VH reconstruction tends to recover a volumetric representation of the object. Volume carving methods split up the 3D space into a 3D grid of voxels. These voxel are tested, later on, for their belonging to the all silhouette cones and labeled as volume voxels if so. The surface separating the inside and the outside of the volume can be computed by the mean of marching cube [30]. This class of methods suffers from the extensive computations and the memory overhead. The approach was first proposed by Martin and Aggarwal [41]. Later on, octree representation was proposed by Chien and Aggarwal [42] to substitute the voxel representation for less memory and computation demand, and efficient approaches were proposed for VH reconstruction [40, 43]. Marching Intersections (MI) was proposed by Rocchini [29] as a resampling method for surface management and adapted later by Tarini et al. [28] for volumetric shape reconstruction. From each silhouette image, the MI structure representing the trunked conoids is first obtained. Intersection of conoids is then computed performing an AND operation.

Surface-based methods also target an exact reconstruction of the VH, but as a 3D polyhedral surface. The surface vertices and faces are estimated by intersecting the generalized cones issued from the occluding contours of the silhouette images, rather than all the silhouette points. Baumgart [32] was the first to propose such a strategy, and later Koenderink [44], Cippola [45], and Boyer [46]. This class of methods produces visual artifact-free VH and requires much less computations and memory, comparing to the previous one. However, intersection in the 3D space is very sensitive to numerical instabilities, especially between complex objects.

In this chapter, I propose a new surface-based VH reconstruction from a set of silhouette images [25]. The reconstruction is based on the principle of CSG [36, 24]. I propose a fast depth layer traversing method based on which, the viewing edges issued from the occluding contours are computed for each view in a parallel manner. For each viewpoint, the generalized cones from the remaining viewpoint are drawn in the 3D space. A multi-pass rendering using an off-screen is performed to traverse the depth layers of the drawn scene and extract only the line segments which lie to all cones. The viewing edges are computed only where the local occluding points occur.

This scheme is designed to run completely in the Graphics Processing Unit (GPU) and only one readback from the GPU memory is required at the end of the process to recover the viewing edges. I propose a storage method to allow the accumulation of the extracted edges at each iteration in a shared buffer allocated on the GPU memory. Next, the viewing edges from all viewpoints are merged together to construct the final 3D shape after rectification of the 3D positions of their vertices.

3.2. Background and Related Work

The visual hull H with respect to a set of N silhouettes S_n ($n = 1..N$) can have two definitions.

Definition 1 : H is the intersection of the projections of the N silhouettes in 3D space with respect to the respective camera centers C_n.

Definition 2 : H is the maximal approximation of the 3D volume that repro-

duces the silhouettes of all viewpoints, such that:

$$p \subset H \Rightarrow \forall_n \Pi^n(p) \subset S_n \tag{3.1}$$

Where p is a 3D point in the space and $\Pi^n(p)$ is the projection of p to the image plane of the camera C_n.(I refer to a camera and its center with the same symbol)

Though the two definitions are completely equivalent, they can show two opposite ways to reconstruct a VH. All polyhedral-based reconstruction and image based rendering methods are based on the first definition while the volumetric-based method are based on the second.

3.2.1 Volume-Based VH Reconstruction

The volume-based methods start by discretizing the 3D space into voxels and then carve the ones that do not verify (1). The voxel size is predefined by the user and determines the resolution of the final VH. This reconstruction approach, proposed by Martin and Aggarwal. [41], has the drawback of memory overhead and also requires an extensive processing load, since all the space voxels nead to be checked whether or not they lie to all visual cone using (1). Cheung et al. [40] tried to speed up the processing by proposing a distributed system. The system they propose could achieve a fast processing but with a very low resolution ($2m \times 2m \times 2m$ split up into $64 \times 64 \times 64$ voxels). To overcome these disadvantages, Chien and Aggarwal[42] proposed a more optimized representation based on octree structure instead of voxels. Szeliski et al. [43] used this representation and proposed a rapid method to reconstruct an object rotating on a turnable table. However, despite the time consumption, the result of these methods suffers from the aliasing artifact. To give more pleasant looking to the already reconstructed VH, Marching Cube method was proposed by Lorensen [30] to compute the mesh surface separation the bounded volume from the outside space. In order to merge volume reconstruction and surface computing, Marching Intersections (MI) was proposed by Rocchini [29] as a resampling method for surface management and adapted later by Tarini et al. [28] for volumetric shape reconstruction. From each silhouette image, first the MI structure representing the trunked conoid is

obtained. Subsequently, intersection of conoids from all silhouettes are computed performing an AND operation.

3.2.2 Surface-Based VH Reconstruction

Surface-based class of methods is based on the first definition. These methods tend to recover a polyhedral surface of the VH by intersecting the generalized cones from all viewpoints. Baumgart [32] was the first to propose this approach in his PHD thesis and later Larentini [31] gave it the name of Visual Hull and use it as a basis for scene understanding. Earlier studies were conducted by Koenderink [44] on how the intrinsic and radial curvature of the shape is related to the curvature of the occluding contours, and by Cippola [45] on the computation of local surface curvature along the corresponding rims based on the known image motion of a silhouette or apparent contours. Later on, Boyer [46] proposed a 3D surface reconstruction using occluding contours. The methods of this class start with approximating the occluding contour of each silhouette by a polygon. The union of all faces originated from the camera center and the polygon edges represents the generalized cone corresponding to the occluding contour. The intersection of these cones from all viewpoints generates a polyhedral representation of the VH.

Each silhouette can generate cones representing the external occluding contours and the holes. Let us denote by O_n, the union of the generalized cones generated by the K outer contours and by I_n, the union of cones generated by the T inner contours(holes), such that:

$$O_n = \bigcup_{k=1..K} O_n^k \qquad I_n = \bigcup_{t=1..T} I_n^t$$

The polyhedral surface is given by:

$$H = (\bigcap_n O_n) \cap (\overline{\bigcup_n I_n}) = \underbrace{(\bigcap_n O_n)}_{1} \cap \underbrace{(\bigcap_n \overline{I_n})}_{2} \tag{3.2}$$

The second term of (3.2) refers to the complement of the space occupied by the union of the cones.

Due to the difficulty to perform intersections in the 3D space and its sensitivity to computational instabilities, Matusik et al.[34] proposed a more robust and

efficient way to compute the polyhedral faces by reducing the calculation to the 2D space. The idea is to project each face issued from one viewpoint to each of the image planes of the other viewpoints and to compute the intersection with the respective silhouettes. The intersections are then back-projected to the 3D space while applying a booleen operation to keep only the face region intersecting all silhouettes. Franco and boyer[49] used the same idea to extract the viewing edges by intersecting the rays associated with the occluding contour vertices, instead of edges, with the other silhouettes. The next step is to walk through these edges to reconstruct the oriented polyhedral faces after recovering the missing vertices and connections. To avoid the extensive computations, the methods of this category approximate the occluding contours by polygons. However this approximation yields a loss of details in the reconstructed VH.

3.2.3 Image Based Visual Hull Rendering (IBVH)

For application such as visualization and interactive rendering, where the VH geometry is not needed, IBVH rendering proposed by Matusik et al. [33] is a cheaper and lighter way. This method is based on finding the first intersection of the ray associated with each image point of the desired view with the VH based on ray casting approach [48]. With advances in graphics hardware, accelerated IBVH methods, based on the growing Graphics Processing unit (GPU), were recently proposed. The direct CSG approach was proposed by Goldfeather [47] and used later by Guha [36] and Li et al. [24] for GPU-based view-dependent VH rendering. Given a desired view (different from all reference views), the principle of this approach is to draw the generalized cones from all reference views and traverse all depth layers of the front facing cone fragments. The goal is to find for each image pixel, the depth related of the first visible VH point. A front-facing fragment belongs to the VH if the difference between the front-facing and the back facing fragments separating the camera center and the fragment in question is equal to the number of reference views, which corresponds exactly to (1). This method achieves a fast processing and haven't the memory overhead problem. Furthermore, they produce a rendered view of the VH without aliasing artifact, since the resolution is dependent only on the 2D resolution of the image and the relative position of the desired view with respect to the VH. However, they

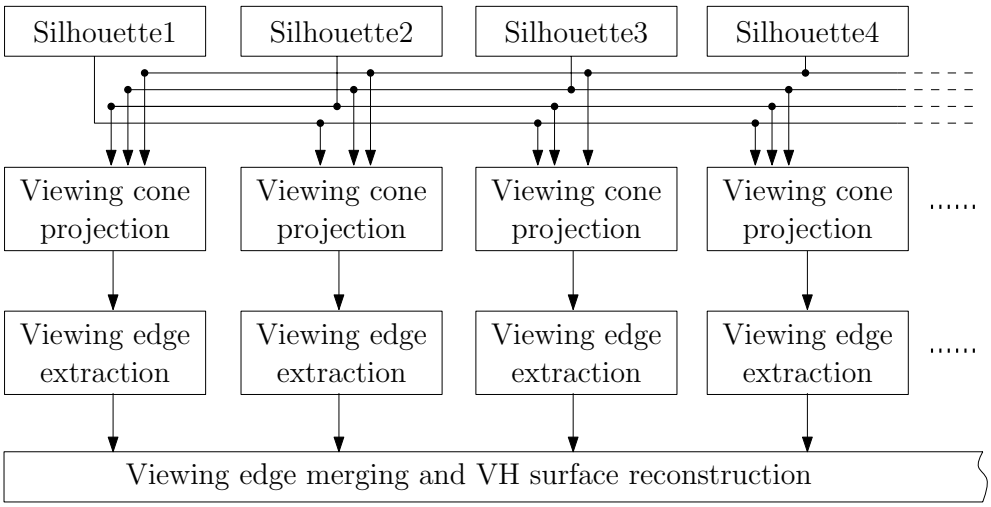


Figure 3.1. The VH reconstruction scheme

do produce only a view of the VH from a virtual view without any geometric information. This is also true for texture-mapping based visual cone trimming proposed by Li et al. [39].

3.3. Overall Scheme

The method I propose in this chapter is a surface-based VH reconstruction. However, the viewing edges are computed directly on GPU based on a CSG-like method. Each camera behaves as a target view in a scene of $N-1$ reference views. For a parallel design, each camera broadcasts the contours of its silhouette to all other cameras, which is much lighter than broadcasting the silhouette image. The local camera after receiving the occluding contours from the other views, draws the cones (except those issued from its own contours) and traverses the depth layers only where the occluding contour points occur, in order to extract the viewing edges for each point. We need to include the ray identification (id) in the related drawn cone face, so that it can be identified from the other views.

A viewing edge is defined by two intersections, the first with a front face and the second with a back face, see Figure 3.2. Direct CSG rendering method was proposed to estimate a novel view from a set of reference views without any

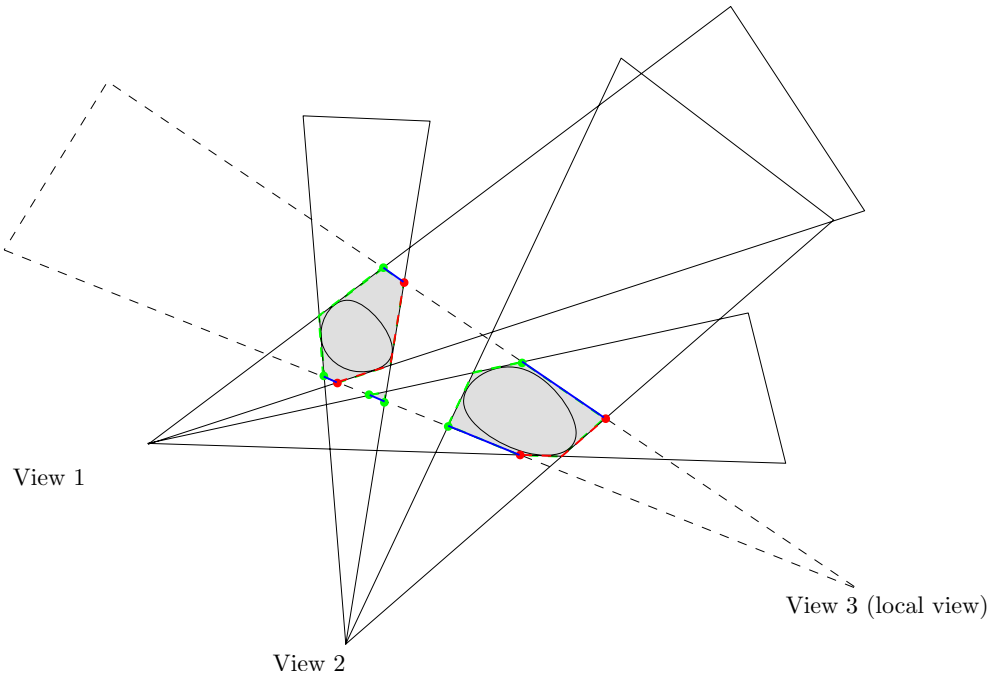


Figure 3.2. Viewing edge definition: The extracted viewing edges are shown in blue.

geometric information. This method performs a traversal of all depth layers for each point in the target view and keeps only the first intersections of the rays with the VH. The CSG-like method I propose in this chapter achieves a faster depth layer traversal to compute the viewing edges for each occluding point. For each viewing edges, the proposed method returns the 3D coordinates of the two associated vertices in addition to the viewing edges from other viewpoints sharing each vertex. These information will be used later on to construct the polyhedral surface which is performed using the computed edges from all viewpoints. The overall scheme is illustrated in Figure 3.1.

3.4. Viewing Edge Computing

Given a set of N silhouette images associated to a set of N calibrated cameras C_n, with (x_0^n, y_0^n, z_0^n) the coordinates of their centers, the viewing edges for each view are the line segments parts of the rays associated with the occluding contour

points and passing through the VH, and hence, lie to the silhouette images of all other views (Figure 3.2). Usually, the viewing edges are extracted by projecting each ray from each view to the silhouettes of all other views and find the line segments that intersect all silhouettes. This method is expensive in terms of processing time especially when the number of occluding contour points is large. To speedup the computations, often one starts by approximating the occluding contours by polygons to reduce the number of points. However, this approximation yields a loss of details in the reconstructed VH. In the method I propose, no approximation is applied. Each occluding contour point is drawn as a separate cone face bounded by the cone faces of its immediate neighbors, see Figure 3.4.

Let us consider the pinhole camera model and refer by A_n to the camera matrix of the camera C_n and by fl_n to its focal length. If o_n^m is a point of a contour O_n of M points ($m = 1..M$) with the coordinates (x_i, y_i) in the image plane, then its local coordinates are (x_i, y_i, fl_n) in the local camera coordinates. Each point v_n^m of the ray r_n^m has the following form:

$$v_n^m = kc_n^m \tag{3.3}$$

Its coordinates in the world coordinate system are given by:

$$\begin{aligned} & v_n^m = O_n + \alpha A^{-1} c_n^m \\ \Longrightarrow \quad & \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} x_0^n \\ y_0^n \\ z_0^n \end{pmatrix} + \alpha A^{-1} \begin{pmatrix} x_i \\ y_i \\ z_i \end{pmatrix} \end{aligned} \tag{3.4}$$

where α is a real constants.

Instead of projecting the rays to all cameras, I employ a CSG-like method. For each viewpoint, the viewing cones from the other cameras are projected to the 3D space. Next, we traverse the depth layers of the drown scene and keep only those lying to all viewing cones. The intersections of these layers with the ray issued from the occluding contour points of the local camera, with respect to the camera center, define vertices in the VH surface. A viewing edge is a line segment which is part of a ray and defined by two vertices intersecting two adjacent opposite layers. The starting point in the outgoing direction from the camera belongs to a front facing, while the other point belongs to a back facing

layer. A front fragment is a region of a cone projecting to a pixel in the local image plane, where the dot product between its outward normal vector and vector joining the camera to the fragment is negative. If this dot product is positive, then the fragment in question is a back fragment.

Traversing the depth layers and checking the intersections is the basic idea of the direct-CSG rendering methods. In the following, I will present this concept.

3.4.1 Direct CSG-Based Veiw-Dependent Rendering

The direct-CSG rendering method was proposed by Goldfeather[47] and used later by Guha[36] and Li et al.[24] for GPU-based view-dependent VH rendering. This method is based on the representation of a complex 3D object as the result of a set of primitive shapes related by a set of operations $(\cap, \cup, \setminus)$. A CSG tree is used to define the operation to perform between the primitive objects to get the complex object. The tree is *normalized* or (*sum of product*) if it is written as a union of intersections/subtructions, knowing that a subtraction between two primitives is the intersection of the first with the complement of the second. Suppose the following tree:

$$(O_1 \cup (O_2 \cap O_3)) \setminus O_4 \tag{3.5}$$

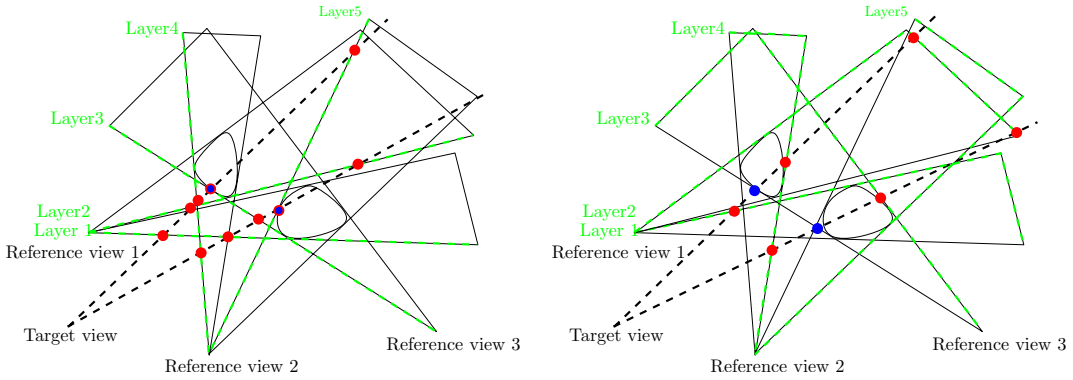
This tree is normalized as follows:

$$\underbrace{O_1 \cap \overline{O_4}}_{P_1} \cup \underbrace{O_2 \cap O_1 \cap \overline{O_4}}_{P_2} \tag{3.6}$$

P_1 and P_2 are two products of the tree that can be processed in a parallel way, and merged later on.

A point p takes part of a product if the difference between the number of front layers and the back layers having smaller depths than the point with respect to any point outside the product, is equal to the number of primitives in the product. This point takes part of the CSG tree (object) if it belongs to, at least, one of its products. Suppose P is a product of $|P|$ objects. If we refer by $f(d,p)$ and $b(d,p)$ to number of, respectively, front and back faces with smaller depth than the a point p with respect to a desired viewpoint d. Then, p belongs to the product if:

$$f(d,p) - b(d,p) = |P| \tag{3.7}$$



(a) Direct CSG. (b) Our method.

Figure 3.3. Direct CSG Vs. our method 2D: The traversed depth layers are drawn in green dashed lines. The points shown in red are the tested points. The Blue points are the saved points and those bounded by a red circle are tested and saved. The number of tested points is 5 in the direct-CSG method and 3 in ours.

In Figure 3.3(a), only the blue points verify this conditionand, hence, belongs to the product. For an uncomplemented primitive object, $f(c,p)$ is always equal to 1 and $b(d,p)$ to 0, while for a complemented primitive object, $f(c,p) = d(c,p) = 1$. For a complex object, however, $f(c,p)$ and $b(c,f)$ can take different values for different viewpoints without violating the equation 3.7.

A VH reconstructed from a set of viewpoints each of which has a full view of the same object(s) can be expressed by the intersections of all unions of cones, each of which is generated by the outer contours of one silhouette, and the complements of unions of cones, each of which is issued from the inner contours (holes) of one silhouette, as expressed by (3.2). This expression can be regarded as a product involving complex objects.

The view-dependent rendering proposed in [47, 36, 24] is based on the principle of CSG trees in order to generate the looking of a VH from a desired view. They make use of two-sided depth test [36] and depth peeling [50] techniques in order to traverse the front facing layers of the drawn scene and for each layer, count the front and back facing fragments separating the desired view center and the layer in question. These operations are processed for each image point to compute the first intersection of the ray generated from the desired view center and passing through the image point. The method can be summarized in the following steps:

- Repeat for all depth layers:

    1. Project the next (first for the first iteration) depth layer of front faces.
    2. Count the front faces separating the traversed depth layer and the desired view position.
    3. Count the back faces separating the traversed depth layer and the desired view position.
    4. Save the depth of the points that verify equation (3.7).

After traversing all front layers, we get a depth map of the object with respect to the desired view. Figure 3.3(a) shows the selected intersections for two rays using this method.

3.4.2 Our Depth Layer Traversal Method

With respect to a given target view, the number of depth layers associated with each image point is given as follows:

Let us denote by G the target image with M rays r_m associated to the M points g_m ($m = 1..M$). To each ray r_m are associated N epipolar lines el_m^n ($n = 1..N$) representing its projections to the N reference image planes. If int_m^n is the number of intersections of the epipolar line el_m^n with the occluding contours of a silhouette image S_n, then the depth layers corresponding to each images point t_m is given by:

$$l_m = \Sigma_n int_m^n \tag{3.8}$$

Half of these layers are front-facing and the other half are back-facing the reference view center, such that:

$$lb_m = lf_m = \frac{1}{2} l_m \tag{3.9}$$

The number of iterations needed to traverse all layers is given by:

$$it = lf_f \quad \wedge \quad f = \arg\max_m (lf_m) \tag{3.10}$$

The problem is that we do not know exactly the number of intersections int_m^n. In other words, it not easy to fix the number of iterations lf_m needed to

traverse all depth layers, since the product involves complex objects. This fact can induce missed layers and false rendering. By examining the illustrations of Figure 3.3(a), we can see that a front face candidate to be a valid intersection is always immediately preceding a back face in the outgoing direction from the target view. This means that only the last of a succession of front faces can be a candidate, all the remaining can be skipped. This can be done by traversing the back faces instead of the front faces. Also, only the first of a list of back faces is traversed, all the others can be skipped by rendering the first front face before two back face traversing. The new algorithm is as follows:

1. Render the next (or first in the first iteration) depth layer of back faces (skip all front faces).

2. Count the back faces separating the traversed depth layer and the desired view position.

3. Count the front faces separating the traversed depth layer and the desired view position and keep the depth of the last depth in the outgoing direction from the camera.

4. Save the depth of the front layer that verify (3.7).

5. Render the first front layer having a depth greater than the current back layer (skip all back layers separating the two layers).

6. Finish the process if no layer is returned, otherwise go to 1.

The advantage from using this new algorithm is to reduce the number of needed renderings and reduce the probability of missing some layers. In the first algorithm, the number of traversed layers is fixed and given by (3.9). However, in the proposed algorithm, the number can vary from 1 to $it - M + 1$. This is due to the fact that, since one front face is an intersection only if it is preceded by $M - 1$ front face non-immediately followed by a back faces. This means that we can skip at least $M - 1$ front faces. Furthermore, if we add more cameras, it is not necessary to increase the number of iterations, unless the complexity of the scene changes. If the number of iteration is set to $k < it$, then the probability of

missing one depth layer is 1 in the first scheme while in the proposed scheme, it is given by the equation below:

$$P = \frac{it - k}{it} \tag{3.11}$$

This can be noticed in Figure 3.3. In the direct-CSG rendering method, 5 layers were traversed and tested while only 3 in our method which respects (3.10) $(3 = 5 - 3 + 1)$.

3.4.3 Application to Viewing Edge Computing

I use the afore-presented depth layer traversing method to compute the viewing edges for each viewpoint while considering what follows:

1. The concern are the reference views instead of a virtual view for image-based rendering: for each camera, the number of views used to test the validity of an intersection is $N - 1$ instead of N.

2. More information concerning valid intersections than the depth returned by CSG method are needed: The returned depth in the case of image-based rendering is useful for multi-view texture mapping. For viewing edges, however, more information are needed. Mainly, the 3D position of the two vertices of each edge and the edge(s) from the other viewpoints sharing each vertex rae needed.

3. The viewing edges are extracted using only the occluding contour points instead of the whole silhouettes: optionally, the remaining image points can be masked during the rendering if this can speedup the processing.

4. We are concerned with all valid intersections instead of only the first one: No further renderings are needed since we are traversing all layers. Nevertheless, information about the intersections need to be saved when a valid intersection is met.

The first two points are taken into account while drawing the cone for a given viewpoint (see section 3.4.4). As for the last point, we need to modify the traversing algorithm in such a way to allow the storage of all the positively tested layers.

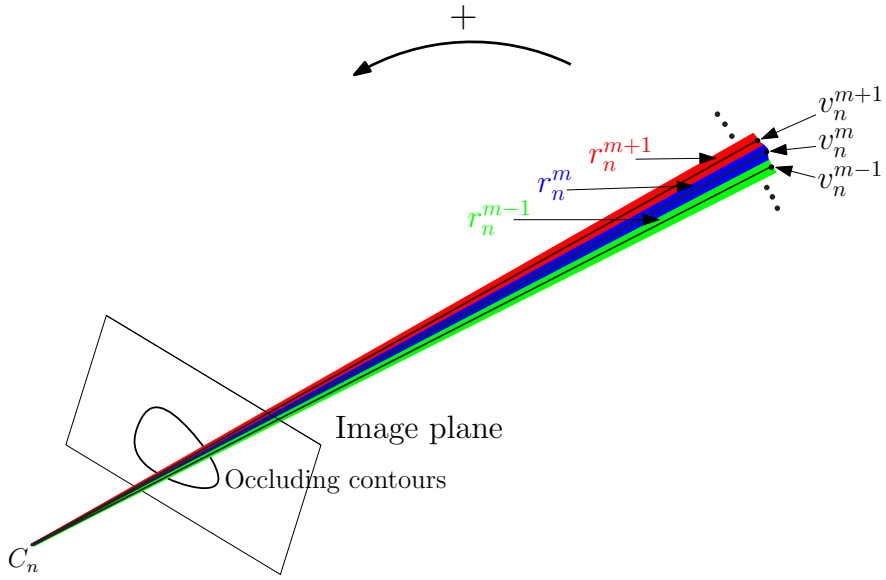


Figure 3.4. The silhouette generalized cone: Each ray is represented as a cone face bounded by the faces of its immediate neighbors. The color of the faces refers to a unique id given to each face.

The straightforward solution is to read back the data from the GPU, since it is on which the algorithm run, after each traversal. However, reading back from the GPU represents the main bottleneck of all GPU-based application. The depth peeling [50] was proposed to just overcome this limitation by offering the possibility to update at each iteration the depth for only those intersections which haven't been set yet. However, not only the first intersections are targeted, but rather all intersections. Thus, finding a way to skip this read-back bottleneck, while storing all edges, will be highly benefic for the performance of the proposed method. The new storage scheme will be explained in section 3.4.5.

3.4.4 Viewing Cone Projection

To draw the viewing cones, we start by classifying the occluding contours into outer (external) contours and inner contours representing the holes. The points are ordered in counterclockwise order for the outer contours (Figure 3.4) and in the opposite order for the inner contours. This order allows distinguishing the front from the back of a face. Also it will help in constructing the VH surface

faces later (the same order is respected between the vertices of a face). Each ray will be drawn as a cone face bounded by those of its neighbors. We set α in the Equation 3.4 to an appropriate value that determines the depth of each cone face to be drawn. We take into account the distance D between the camera and the farthest point of the 3D area covered by the cameras. α is set as:

$$\alpha = D/fl \tag{3.12}$$

This setting ensures that each ray intersects all viewing cones whatever the position of the object in the covered scene. After setting α, it becomes possible to find for each ray r_n^m, the farthest point v_n^m from the camera center C_n. The cone face associated to the ray r_n^m is defined by the ordered vertices $(C_n, v_n^{(m-1)\%M}, v_n^{(m+1)\%M})$. In order to be able to identify the viewing edges sharing the same vertices, we label each ray with a unique id (cone face), as shown in Figure 3.4. This id is passed to the cone face during the drawing step as color information. We make use of the R, G, B and A channels to store the id, which gives us the possibility to define 255^4 id.

3.4.5 Viewing Edge Storage

As explained, we are interested in only the occluding contour points. These points are few as compared to the image points. The idea I propose is to save the edges passing the test to a storage buffer allocated as a RGBA texture in the GPU memory. This buffer is read-back once all edges extracted. We need for that to add one more rendering pass after each iteration. The rendering pass consists in drawing a full screen quad in a projective geometry. Five textures are attached as inputs: two textures for each vertex containing the 3D position and the id of the corresponding intersecting ray, and one texture loaded once at the beginning and serving as a lookup table for each point to get the coordinates of the texture point to store. Let us refer by *3DMap1* and *IdMap1* the 3D and color maps of the first vertex, and by *3DMap2* and *IdMap2* to those of the second vertex of the edge. The color map contains the id of the intersecting edges. Also I refer by *lut* to the lookup table texture, by *width* and *high* to the texture and image size, by M to the number of occluding contour points, and by *it* to the number of iterations. *lut* is initialized once and load loaded to the GPU memory. It contains

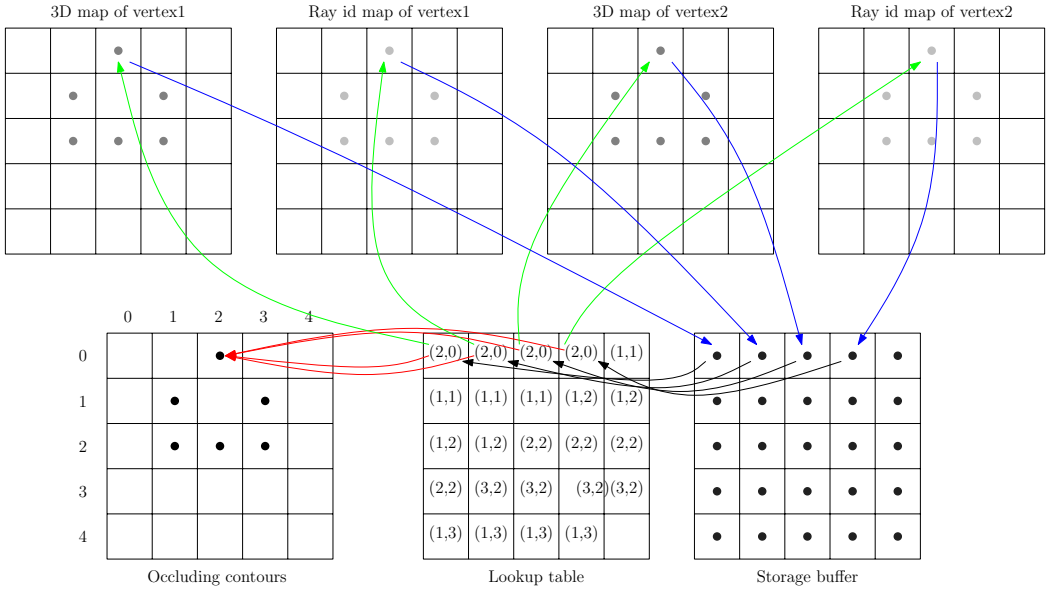


Figure 3.5. Viewing edge storage scheme: This processing is invoked at pixel (fragment) level.

a list a subsequent occurrences of the list of the occluding points, each of which is duplicated four times, as shown is Figure 3.5. The lookup table initialization is ilustrated by Algorithm 1.

The kernel (fragment shader) invoked at point level, to store the edge vertices, reads the coordinates from *lut* and uses them to locate the information to store from one of the four vertex textures. This is done only if the invoking point is located within the region concerned by the current iteration, as shown in Figure 3.5. If the coordinates of this point in the storage buffer are (x, y), the storage is as in Algorithm 2: The maximum number of iteration that can be processed within the storage capacity of one buffer is given by:

$$MaxIt = \frac{width \times high}{4 \times M} \tag{3.13}$$

Suppose the image size is 640×480 and each image size contains less than 2000 contour points. In this case, one storage buffer can support until 38 iterations. If more iterations are needed, then, more buffers must be allocated.

Algorithm 1 Lookup table initialization

```
for i = 0 to it do
  for j to M do
    for k = 0 to 4 do
      lut[(i * j + k) mod width, (i * j + k) div width] ← coordinates(C[j]);
    end for
  end for
end for
```

3.4.6 Evaluation

I implemented the described edge extraction scheme as a multi-pass rendering on the GPU. I made use of OPENGL as an *API* and C-like shading language (CG) of NVIDIA to write the shaders (kernels). I made use of a Frame Buffer Object (FBO) as an off-screen rendering target instead of the screen. To this FBO, I bind a depth buffer, a stencil buffer, and a shadow buffer. the depth and shadow buffer serve to the two-sided buffer test [36], while the stencil buffer is for counting the layers. I bind also a storage buffer and a lookup texture to the FBO. The lookup texture is initialized first, using Algorithm 1, before it is loaded to GPU memory. At each rendering step, appropriate textures are attached as input(s) and output(s). In addition, one fragment and/or one vertex shaders are loaded to the programable vertex and fragment processors in order to achieve one step of the extraction algorithm. The storage buffer is read back only once at the last rendering pass to retrieve the edges. Figure 3.6 shows the edges using four silhouette images of a rabbit taken from 4 viewpoints.

3.5. VH Surface Construction

After been extracted from all views, the viewing edges are merged together to construct the VH surface as shown Figure 3.7(a). A vertex, being the intersection of two or more edges issued from different cameras, can be detected with slightly different 3D position in each camera. This fact makes the extracted edges disconnected from each other as can be noticed in Figure 3.7(b). Thus, we need to recover a unique 3D position for each vertex. This could be done by computing

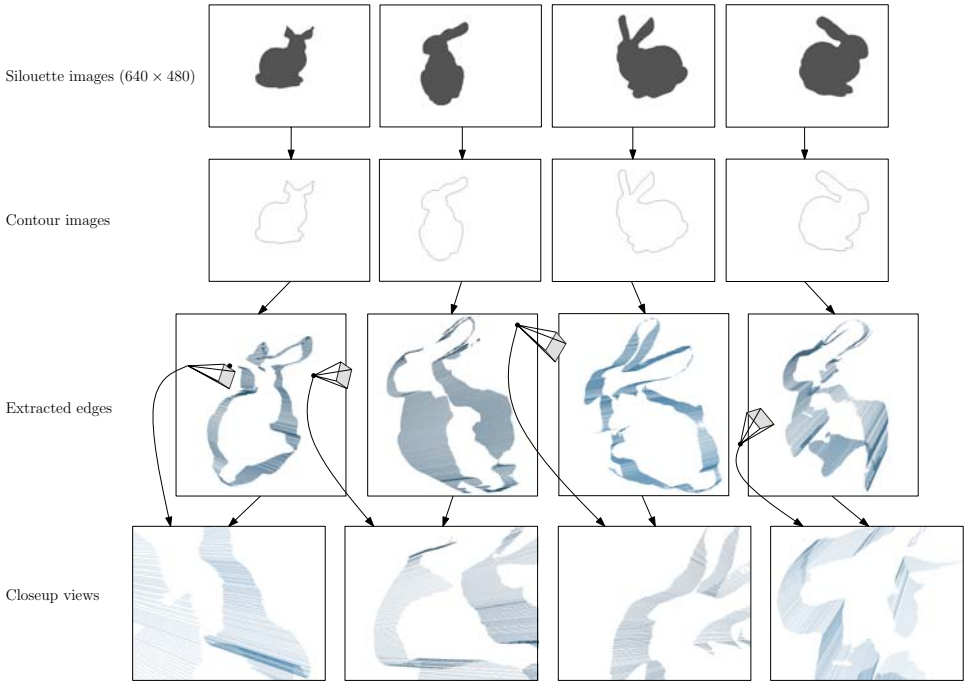


Figure 3.6. Viewing edge extraction scheme.

this 3D position as a mean of its coordinates estimated by all views, as follows:

$$v = \frac{1}{n}\Sigma_n v_n \tag{3.14}$$

The result is shown in Figure 3.7(c). Even after this step, still some edges remains disconnected. This fact is due to the resolution difference between the cameras. In fact, the projection of a 3D object occupies more points in closer or higher-resolution views. We have two options to connect these edges; either we 1) create new edges in lower resolution views, or 2) join the edges to the closest neighboring vertex (issued from a neighboring point of the same contour). These two choices lead to different resolution of the final VH. In fact, In the case of cameras with the same optics, the maximal resolution is related to the closest view from the object. In the case of cameras with different optics, three parameters influence the resolution; 1) the distance from the object, 2) the focal

Algorithm 2 Storage kernel

```
if M * it * 4 <= y * width + x < M * (it + 1) * 4 then
  (a, b) ← lup[x, y];
  if (y * width + x) mod 4 = 0 then
    storage[x,y]=3DMap1[a,b];
  end if
  if (y * width + x) mod 4 = 1 then
    storage[x,y]=IdMap1[a,b];
  end if
  if (y * width + x) mod 4 = 2 then
    storage[x,y]=3DMap2[a,b];
  end if
  if (y * width + x) mod 4 = 3 then
    storage[x,y]=IdMap2[a,b];
  end if
end if
```

length, and 3) the pixel size for each camera.

Let us denote by d_n the distances of a point p from a camera C_n, by f_n the focal length of this camera, and by ps_n the pixel size in its image plane. The maximal resolution is given by:

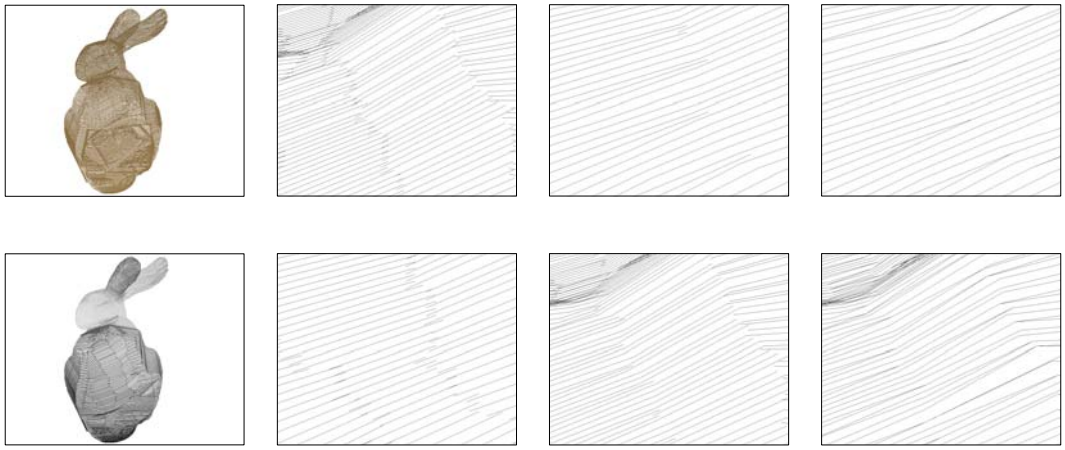
$$R_{max} = \frac{d_{\hat{n}}}{f_{\hat{n}}} ps_{\hat{n}} \quad \wedge \quad \hat{n} = \arg\max_n \frac{d_n}{f_n} ps_n \tag{3.15}$$

Similarly, the minimal resolution is:

$$R_{min} = \frac{d_{\check{n}}}{f_{\check{n}}} ps_{\check{n}} \quad \wedge \quad \check{n} = \arg\min_n \frac{d_n}{f_n} ps_n \tag{3.16}$$

Thus, the 3D resolution in the first choice is equal to R_{max} and in the second to R_{min}. I chose the second choice, namely connecting the missed vertex to the nearest neighbor, and show the result in Figure 3.7(d)

The VH face generation can be processed by each separately in a step prior to the rectification of the 3D positions of the vertices. The faces are generated by connecting the appropriate edges generated by neighboring contour points.



(a) The merged edges and a depth view.

(b) Closeup view: Initial state

(c) Closeup view: After unification of vertex positions.

(d) Closeup view: After connection of missed edges.

Figure 3.7. Surface construction: (a) A view of the viewing edge merged together from 8 viewpoints. (b) The viewing edges are disconnected from each other after extraction. (c) The edges are connected to each other using the associated id. (d) The disconnected edges due to resolution difference between views are connected.

We consider the predefined order of the contours in generating the faces. The reconstruction results will be presented in the next section in addition to the evaluation of the overall VH reconstruction.

3.6. Experimental Results

The presented scheme was implemented on a *PIV* PC with $1GB$ RAM and equipped with a *NVIDIA GeForce 9700* graphics card. I tested the reconstruction scheme on 3 synthetic datasets. I used 2 shapes provided by Princeton Shape Benchmark [52] to generate the silhouettes from 8 viewpoints. I also tested the reconstruction scheme on a real dataset provided by Matsuyama Laboratory of Kyoto University in the form of 8 silhouette images of a Kimono Lady (Maiko) and the related camera parameters. The results of reconstruction are shown in Figures 3.8, 3.9, and 3.10 where the upper row shows the silhouettes and the last row, four virtual views of the reconstructed VH. The silhouette images sizes vary between 652×490 and 728×549.

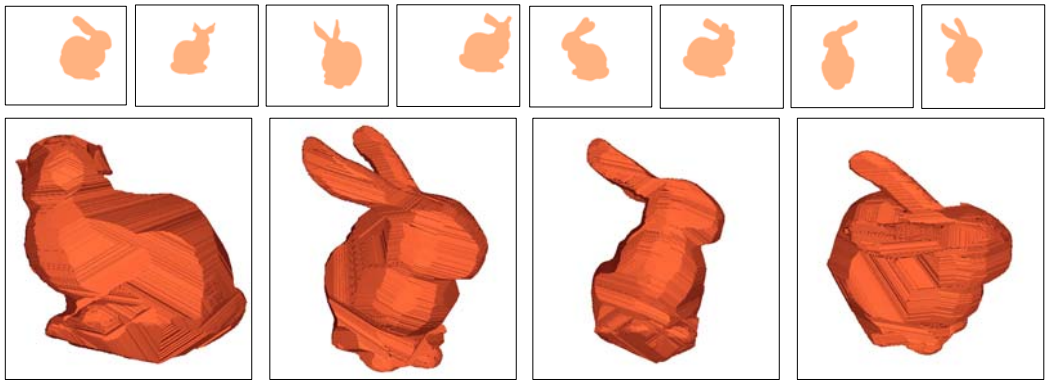


Figure 3.8. The VH reconstruction: Bunny dataset.

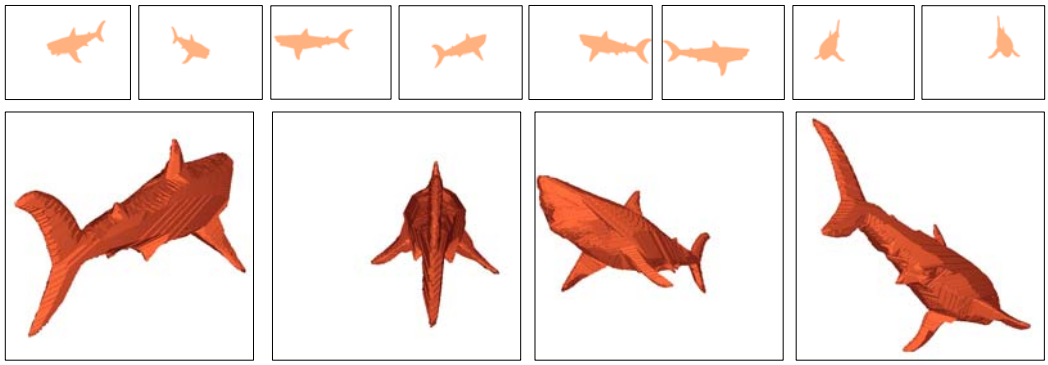


Figure 3.9. The VH reconstruction: Shark dataset.

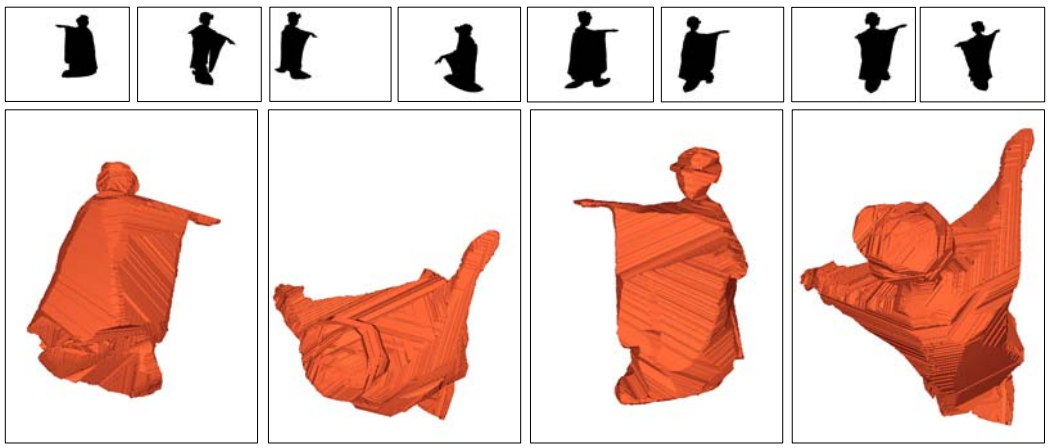


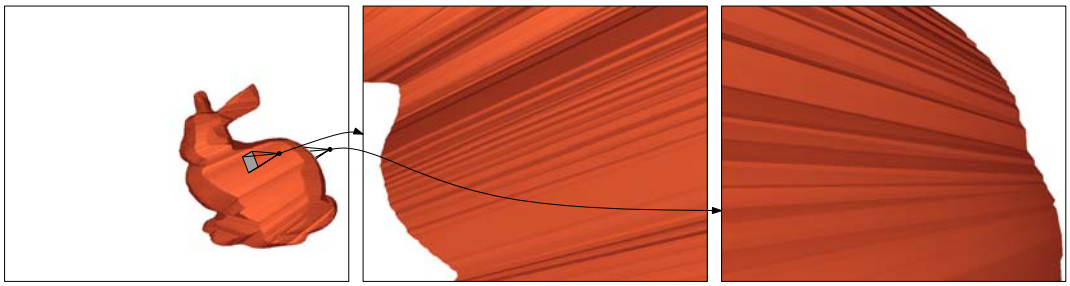
Figure 3.10. The VH reconstruction: Real data (Maiko).

	Bunny		Shark		Maiko	
Camera	Point count	Time (ms)	Point count	Time (ms)	Point count	Time (ms)
Camera1	887	110	720	109	1109	156
Camera2	1062	140	680	109	1306	172
Camera3	960	125	1256	140	1209	172
Camera4	971	125	887	125	1075	156
Camera5	1052	140	703	109	1316	172
Camera6	1066	140	1069	125	1565	156
Camera7	1024	141	1159	140	1185	156
Camera8	1060	140	966	125	1413	172

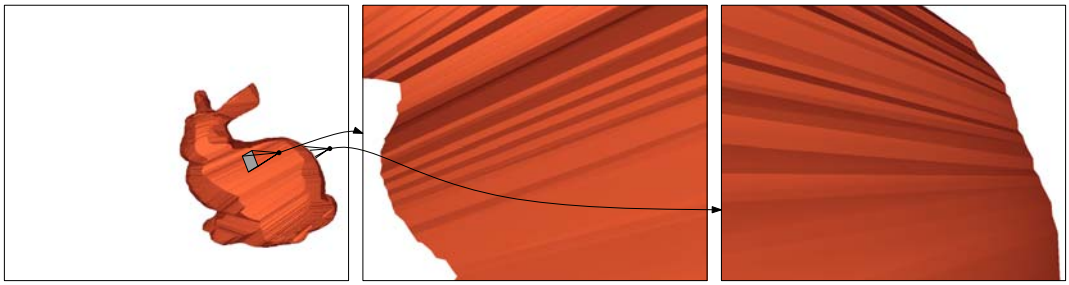
Table 3.1. Processing time evaluation: The processing time is calculated for each camera and for each dataset. The shown time concerns the viewing edge extraction and face generation. 'Point count' columns refer to the number of occluding contour points.

Table I, summarizes the processing time for each camera and for each dataset. This can allow us to get an idea about the processing time when the scheme is distributively implemented on multiple PCs, each of which is connected to one camera. In this case, the processing time is the largest time among all cameras, added to the time needed for vertex unification, which is $31ms$. The processing time varies from one camera to another. This fact is related to the complexity of the scene that varies with respect to each viewpoint, yielding different number of depth layers. Also it is related to the area occupied by each silhouette. In fact, I used scissoring technique to speedup the rendering time. Thus, the rendering is allowed only in the region defined by the bounding rectangle of the silhouette.

As to evaluate the processing time of the proposed algorithm, I considered the algorithm proposed by Matusik [35] which is supposed to be the first to compute the VH polyhedral representation in an interactive frame rate. Implemented on a 1GHz Pentium III machine with 1GB of RAM, this method reconstruct the VH in 2 sec for 8 viewpoints with 641 contour points in each view. From Table I and is we consider an implementation on one PC, the processing time varies between $1012ms$ for the 'shark' and $1343ms$ for 'Maiko', with much more contour points for each view. In the case of the Matusik method, as for all similar methods, the silhouette is approximated for a faster processing. If we want to do the same



(a) Initial scale (640 × 480)



(b) Lower scale (320 × 240)

Figure 3.11. Reconstruction at lower scales.

thing, we scale down the silhouette images. This will results in more natural approximation. Figure 3.11 shows the reconstructed VH using 8 640×480 images and using the same images but scaled down to 320×240, and Table II summarizes the processing time. The down scaled images contain comparable contour point counts to this of Matusik's example (641). If we consider an implementation on 1 PC, then the processing time is $1250ms$ in the initial scale, while it is $312ms$ in the lower scale. I could speedup the process 4 times by down scaling the image to the half size (in each direction). Also, I can cite the method proposed by Franco and boyer in [49] for exact VH reconstruction. Implemented on a 1.8GHz PC, their method achieve the reconstruction using 4 silhouette images with 250 contour points each in 142 ms.

In order to prove the effectiveness of the depth layer traversal method, I computed the number of the traversed depth layers of the drawn cones with different number of cameras using the direct-CSG method and ours. the graph presented in Figure 3.12 show the result where we can notice that our method requires less iterations than the native CSG method do to visit all candidate

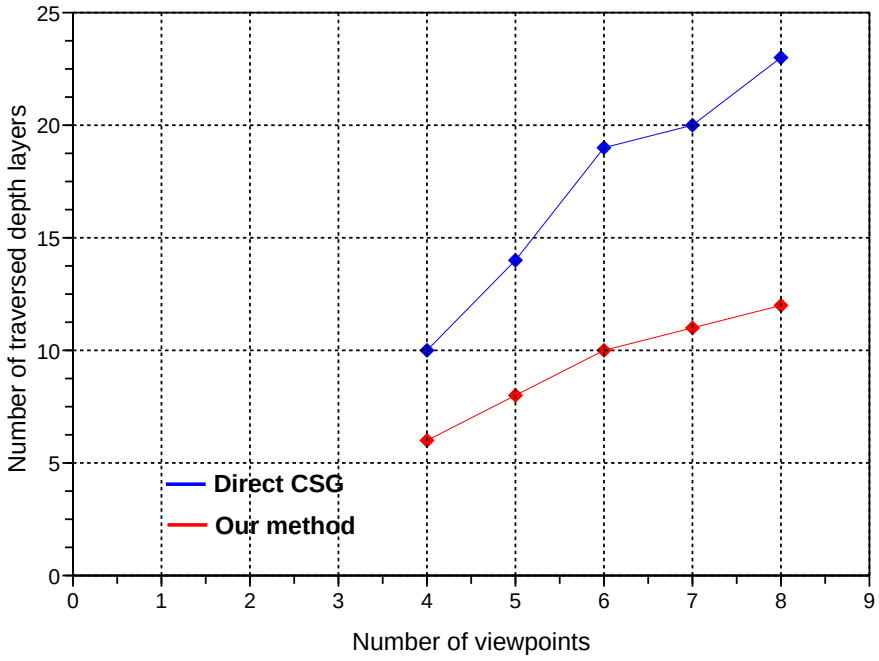


Figure 3.12. Evaluation the proposed method for depth layer traversing: Comparison with the native direct CSG.

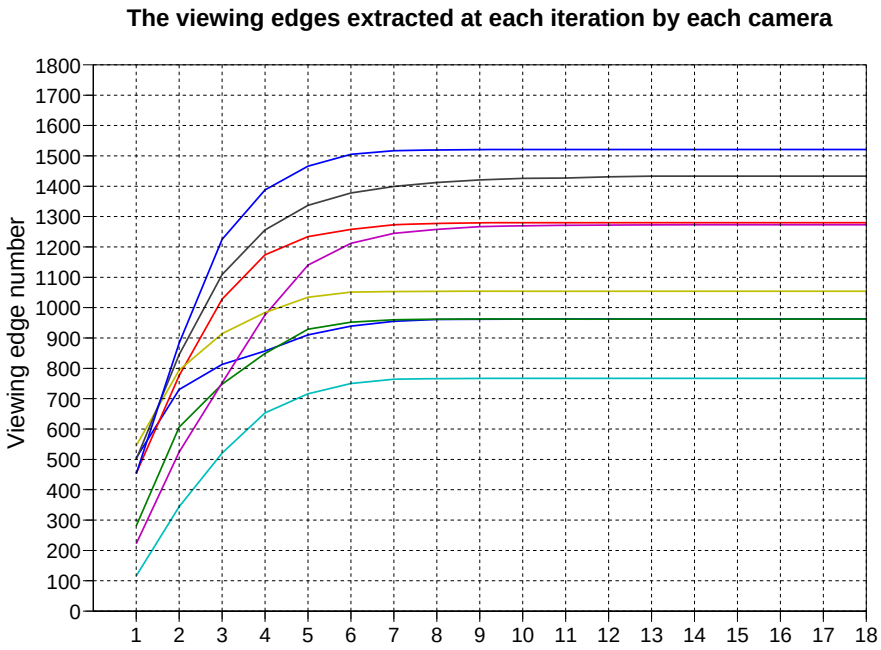


Figure 3.13. Variation of the number of extracted faces within iterations.

	640 × 480		320 × 240	
Camera	Point count	Time(ms)	Point count	Time(ms)
Camera1	1334	172	666	46
Camera2	1028	140	508	31
Camera3	973	141	482	31
Camera4	1242	156	611	31
Camera5	1302	172	648	47
Camera6	1121	156	559	32
Camera7	1093	141	543	31
Camera8	1061	141	531	32

Table 3.2. Comparison with the reconstruction using down-scaled images.

depth layers. Furthermore, the more are the cameras the bigger is the difference. Also, the number of viewing edges extracted after each iteration are shown in the graph of Figure 3.13. 12 iterations are required to permit all cameras to recover all viewing edges. In these tests, I set the number of iterations to 15 for all models.

3.7. Conclusion

In this chapter, I presented a new method for shape from occluding contours. I proposed a CSG-like method for a fast depth layer traversing and viewing edge computing, rather than just rendering the depth of the shape from a desired view. The viewing edges are extracted for each camera separately without camera-camera projection. This fact allows the system to be implemented in a distributed system where each camera is connected to one PC and operates independently of the rest. This design will provide a faster processing. The proposed reconstruction scheme doesn't need any approximation of the silhouette and, hence, preserves the details of the shape.

The possible future extensions of this scheme are:

1. The parallel implementation of this system and more and new evaluations.

2. Texture mapping. This can also be implemented on GPU using the register

combiner mechanism of the GPU to achieve a fast processing time.

Chapter 4

Practicability of Active Camera Control: Heterogeneous Camera System for Visual Surveillance

In order to show the practicability of the camera control module presented in Figure 1.1, I present in this chapter describes a system that combines static stereo cameras with wide field of view (FOV) and high resolution active Pan/Tilt (PT) cameras into a networked platform. Designed for security applications, the purpose is to provide high resolution face images while covering a wide area. A custom PT unit allowing a fixed camera projection center was designed to hold the high resolution camera. Two such cameras in addition to one stereo camera were implemented in a real scene in order to test the effectiveness of the system. Experimental results demonstrate the efficiency of the proposed system.

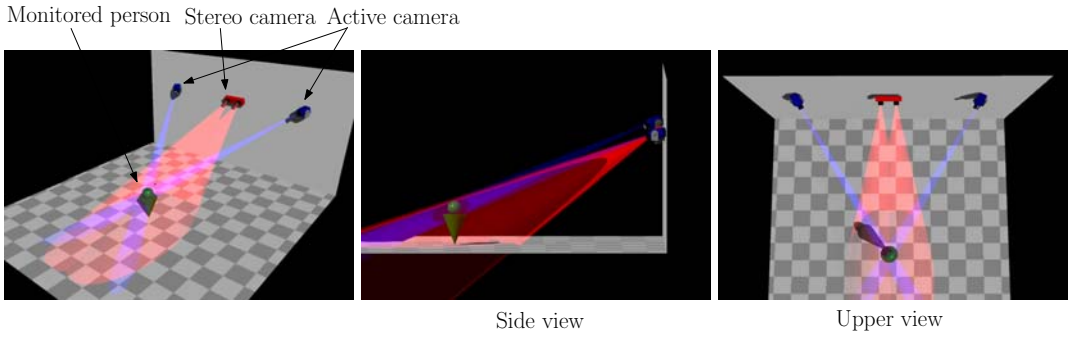


Figure 4.1. Camera setup.

4.1. Introduction

Visual surveillance is getting an increasing interest as security issues are becoming a major concern in our society. There is extensive literature about surveillance systems that try to cover different situations. The main shortcoming is that when these systems cover large areas, the task becomes that of detecting the existence of people [58], or that of tracking them [60] with almost no possibility of identifying the person being tracked. If consideration is given to identification these systems generally cannot cover wide areas, resulting in difficulties when applying the technology in actual situations of public spaces.

Few systems have been proposed in the literature to overcome these limitations. The face cataloger system [59] combines two stationary cameras with wide Fields Of View(FOV) and two active Pan/Tilt/Zoom (PTZ) cameras, in order to provide high resolution face images while covering a wide space. Foreground segmentation followed by a model-based head tracking is performed, simultaneously, on the two stationary cameras. The 3D head position, obtained by triangulation, and the velocity of the person are used for steering the PTZ cameras. These cameras wait for the face to be stable to zoom up and capture a closeup view images of it. This constraint, however, limits the usability of the system to situations where people are aware of its existence.

Earlier, a similar system was proposed by Stillman et al. [57] for tracking and recognizing multiple people with multiple cameras. Using the same camera configuration as the first system, the person's face is detected in the static cameras based on skin color information. Color correlation is performed on both

static cameras to locate the target's face and steer the PTZ cameras toward it. Subsequently, the detected face is continuously tracked by the PTZ camera with the support of the static cameras when the face is lost. The limitation of this system resides in the restriction made on the color of the person's clothes which must differ from the skin color.
The design of new security systems useful in outdoor environments implies the consideration of the following requirements:

- Real-time operation.
- Coverage of wide areas.
- Accuracy in face detection.
- Consistency against lighting changes.

I propose in this chapter a heterogeneous camera system made up of static wide FOV stereo and active high resolution Pan/Tilt (PT) cameras in order to provide high resolution face images while covering a wide area [54, 55]. According to Figure 1.1, the preprocessing consists in estimating the 3D positions of the faces within the monitored area based on stereo processing. These 3D position are used by the active cameras to change their orientations to gaze the face and display it in high resolution, as shown in Figure 4.1. In order for an active camera to gaze any indicated face, the camera should have access to the entire region covered by the stereo camera. The camera hosts communicate through a network platform.

4.2. System Overview

4.2.1 Hardware Requirements

The proposed system involves static wide FOV stereo and active high resolution PT cameras. The stereo camera used is a Videre Design MEGA-D^{TM}model with 20[cm] baseline. The active cameras are PtGrey XGA Flea^{TM}cameras mounted on custom designed pan-tilt units. Panning and tilting are achieved with two digitally controlled ROBOTIS Dynamixel ^{TM}motors.

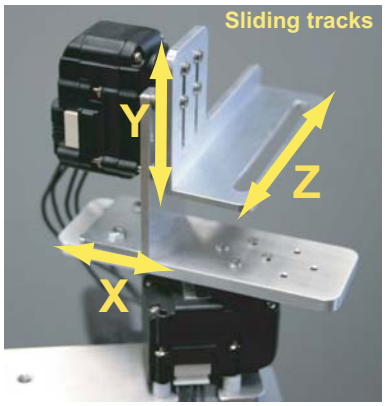
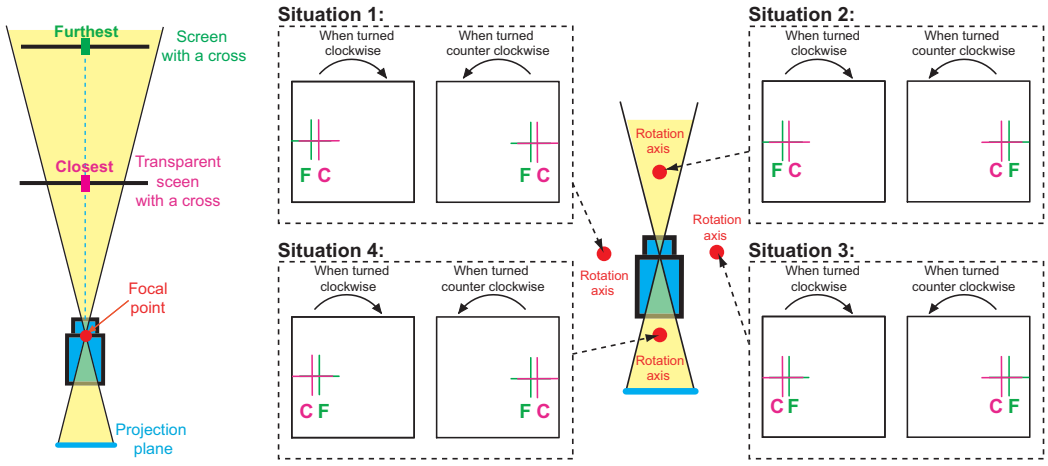


Figure 4.2. Custom Pan/Tilt unit allowing fixed-viewpoint calibration.

The pan-tilt unit is designed to allow its rotation center to meet the projection center of the mounted camera. As shown in Figure 4.2, the unit parts have sliding tracks allowing 3 degrees of freedom (DOF). With this design, the projection center of the camera can be adjusted to the rotation center of the PT unit by sliding the camera along the unit axis. By doing so, the translation of the camera center can be neglected and only its orientation is considered. The unit has a maximum rotation speed of 70[rpm] and a resolution of 0.37[deg]. Each camera is attached to one PC host that communicates with the other hosts through a TCP/IP network connection.

4.2.2 Fixed-Viewpoint Calibration

The target of this procedure is to bring the projection center of the camera to coincide with the rotation center of the PT unit whereon, the camera is attached. In many computer vision applications, the camera internal and external parameters are needed. For a stationary camera, these parameters are set once and for all as a beforehand setup. For an active camera, however, the extrinsic parameters have to be reset after each movement. If the projection center of the camera is itself the rotation center of the unit, then only the camera rotation matrix needs to be calculated. Moreover, if we know with precision the pan and tilt rotation



(a) Initial setup. (b) Possible situations during the calibration process

Figure 4.3. Fixed-viewpoint calibration process..

angles of the PT unit, then even the rotation matrix could be calculated directly, since the rotation matrices of the unit and the camera are identical.

Fixed viewpoint calibration, inspired by [61], assumes that if the focal point is on the rotation axes, then any point on the same line of sight should be projected at the same point on the camera projection plane regardless of camera rotation. Here, we use two parallel planes in front of the camera. The planes are screens on which cross signs are drawn and the screen closest to the camera is transparent. First, the camera and the two planes are set such that the center of the camera image and those of the two crosses are on the same line of sight, as shown in Figure 4.3(a). Then, pan rotation is applied in two directions (left and right) and two images are taken. Depending on the position of the rotation center with respect to the projection center, different images can be obtained and different camera position situations estimated, as shown in Figure 4.3(b). The goal is then to get as close as possible to the assumption of having the cross signs on the same line of sight. The fixed viewpoint calibration procedure is summarized in Algorithm 3:

The fixed viewpoint calibration method can be expressed as an optimization problem that tries to satisfy the following condition:

$$\arg\min_{x,y,z} \sum_{i\in\{1,2\}} |X_{i1} - X_{i2}|, \tag{4.1}$$

Algorithm 3 Fixed-viewpoint calibration

repeat
  Rotate(CW); {Clockwise pan rotation}
  $X_{11} \Leftarrow CrossCenter(image1)$ {Cross center on the near plane}
  $X_{21} \Leftarrow CrossCenter(image2)$ {Cross center on the far plane}
  Rotate(CCW); {Counterclockwise pan rotation}
  $X_{12} \Leftarrow CrossCenter(image1)$
  $X_{22} \Leftarrow CrossCenter(image2)$
  if $X_{11} > X_{12}$ and $X_{21} < X_{22}$ **then**
    Move the camera backward
  end if
  if $X_{11} < X_{12}$ and $X_{21} > X_{22}$ **then**
    Move the camera forward
  end if
  if $X_{11} < X_{12}$ and $X_{21} < X_{22}$ **then**
    Move the camera leftward
  end if
  if $X_{11} > X_{12}$ and $X_{21} > X_{22}$ **then**
    Move the camera rightward
  end if
until $X_{11} = X_{12}$ and $X_{21} = X_{22}$
repeat
  Rotate(Upward); {Up rotation}
  $Y_1 \Leftarrow CrossCenter(image1)$
  $Y_2 \Leftarrow CrossCenter(image2)$
  if $Y_1 > Y_2$ **then**
    Move the camera upward
  end if
  if $Y_1 < Y_2$ **then**
    Move the camera downward
  end if
until $Y_1 = Y_2$

where x, y, and z are the positions of the fixtures on the pan-tilt unit sliding tracks.

4.2.3 Global System Calibration

In order for the active cameras to gaze the face indicated by the stereo camera, all camera, regardless of their characteristics, must be calibrated together to the same world coordinates system. The calibration is done for each camera separately using the method presented in [56]. We begin by one common reference plane (pattern), to which corresponds the world coordinate system. With respect to this plane, the translation and rotation matrices are derived for all cameras. Also the pan and tilt angles of PT unit are recorded as the camera's home orientation. Finally, we complete the multi-plane calibration for each camera separately to estimate the internal parameters.

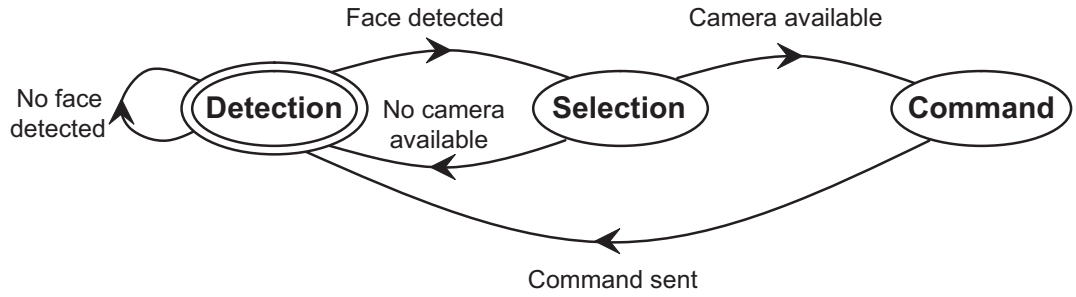
4.3. System Operation

As explained so far, the stereo host is in charge of:

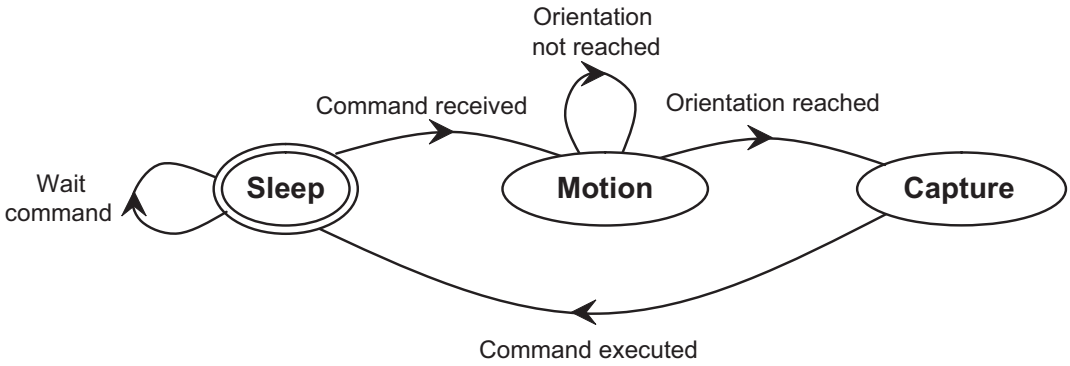
- detecting a face within the covered space,
- estimating its 3D position,
- and pass this 3D position to the available active cameras.

The stereo host, as summarized by the finite state machine of Figure 4.4(a), keeps detecting,continuously, a face in the scene. It leaves this state when a face is detected and comes back to it if no active camera is available. If one or more cameras are available, the stereo host passes to them the 3D face position and goes back to the detection state. As for the active hosts, they:

- receive the 3D face position from the stereo host,
- adjust the local parameters accordingly,
- and control the PT unit in order for the camera to gaze the face.



(a) Operation of the stereo camera host.



(b) Operation of the active camera host.

Figure 4.4. Finite state machines for the system operation.

When an active camera host receives the 3D face position from the stereo one, it changes its state to busy, processes the necessary transformations, and executes the rotation. When the 3D position is reached, the camera host executes a release notification and goes back to the sleeping state to wait for a new command, as shown in Figure 4.4(b).

4.3.1 3D Face Position Estimation

As shown in the diagram of Figure 4.5, the stereo camera provides two images. The left stereo camera is set as principal with respect to which, and using both images (left and right), the 3D map is obtained by stereo processing. continuously, faces are detected in the left image. In the literature, many face detection

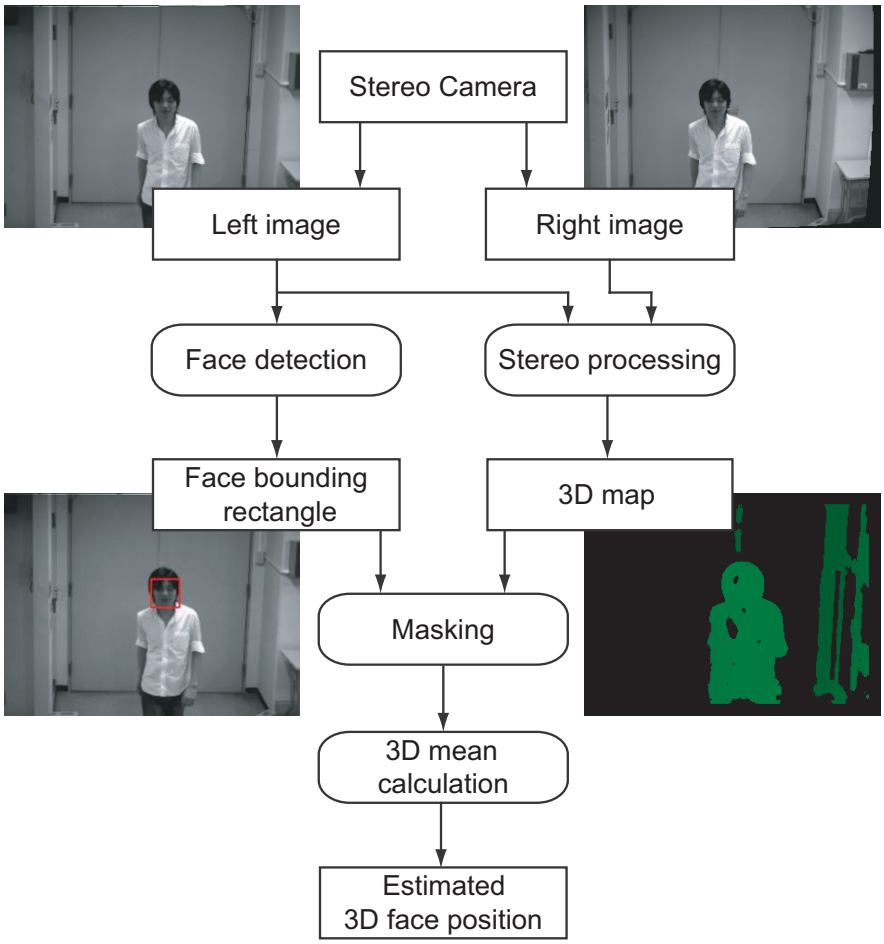


Figure 4.5. 3D face position estimation process

methods have been proposed [53]. I made use of the appearance-based method described in [62] due to its reliability and processing speed. Once a face is detected, the offset and the size of its bounding square are returned. The image region bounded by the face square is, then, set as the region of interest (ROI), and the corresponding points from the 3D map are extracted. Finally, the mean position of the extracted points is set as the estimated 3D face position. In the preliminary implemented system, processing speed was about 15[fps] on an Intel ^{TM}Celeron 2.8GHz processor.

It happens that the square bounds a region larger than the face itself. In such cases, 3D points other than those contained in the face are taken into account, and

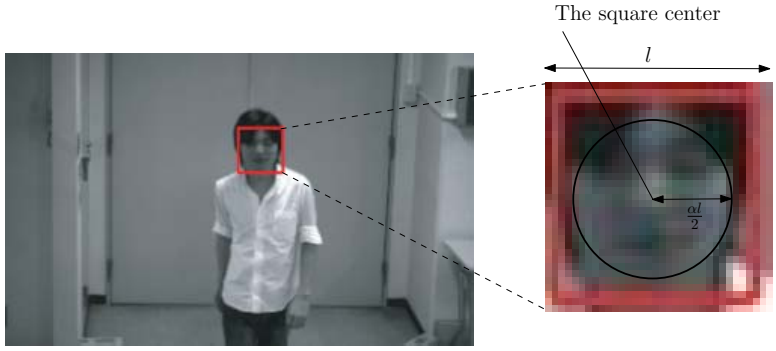


Figure 4.6. The face's region concerned by the 3D face position estimation.

consequently, the estimation error is large. To cope with this problem, the region from the square concerned by the mean calculation is limited by a circle whose diameter is shorter than the square size, see Figure 4.6. The circle diameter, denoted d, is defined as as a scale of the square size, see Figure 4.5. If the square size is $l \times l$ then:

$$d = \alpha l \quad \alpha \in]0, 1] \tag{4.2}$$

Let us denote by $O = (x_o, y_o)$, the offset of the bounding square with respect to the image. The set V of the 3D points to be averaged is given by:

$$\begin{aligned} V &= \{map(x,y) / \left(|x - x_o + \frac{l}{2}| < \frac{\alpha l}{2} \right) \\ &\quad \wedge \left(|y - y_o + \frac{l}{2}| < \frac{\alpha l}{2} \right) \wedge (valid(x,y))\} \end{aligned} \tag{4.3}$$

where map(x,y) returns the 3D point related to the image point (x,y), and $valid(x, y)$ returns *true* if the image point corresponds to a valid disparity (successfully mapped).

4.3.2 Active Camera Control

An active camera receives the 3D position $\mathbf{p}_s$ to gaze to from the stereo camera. However, the position $\mathbf{p}_s$ is expressed in the left stereo camera coordinates. Thus, before any control starts, the active camera host has to transform this position into its own coordinates. Let's denote the projection matrices of the left stereo camera and the active camera by $\mathbf{A}_s$ and $\mathbf{A}_a$, respectively. Then

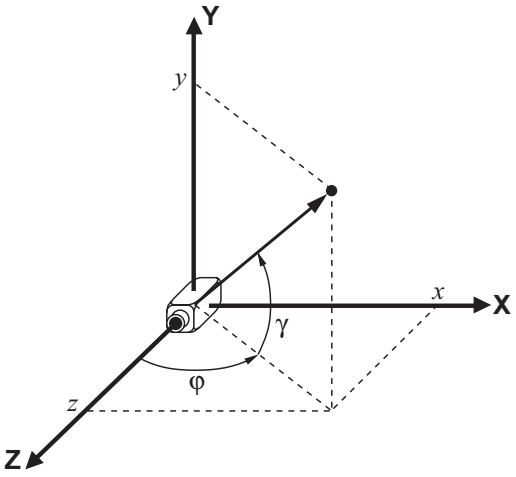


Figure 4.7. Pan and tilt rotation angles

$\mathbf{p}_a = (x_a, y_a, z_a)^T$, the same point in the active camera coordinates, is given by the following:

$$\mathbf{p}_a = \mathbf{A}_a \mathbf{A}_s^{-1} \mathbf{p}_s \tag{4.4}$$

Using $\mathbf{p}_a$, the active camera host can calculate φ and γ, respectively, the pan and tilt rotation angles, as shown in Figure 4.7. First, we denote by $\overrightarrow{\mathbf{n}} = (x_n, y_n, z_n)^T$ the unit vector of the considered position obtained by normalizing $\mathbf{p}_a$.

$$\overrightarrow{\mathbf{n}} = \frac{1}{\sqrt{x_a^2 + y_a^2 + z_a^2}} \mathbf{p}_a \tag{4.5}$$

Then φ and γ are given by:

$$\varphi = arctan(\frac{x_n}{z_n}) \qquad \gamma = arcsin(y_n). \tag{4.6}$$

The PT unit executes a rotation in pitches units. The value of a pitch defines the angular resolution of any PT unit, and is 0.37° for our designed unit. Having a target 3D position, one has to find the appropriate rotation angles and convert them to pitch units. Furthermore, the obtained orientations are with respect to the camera coordinates. However, it is not necessary that the camera and the corresponding PT unit have the same coordinate system. Therefore, we have to add to these rotation angles the rotations of the reference orientations of the

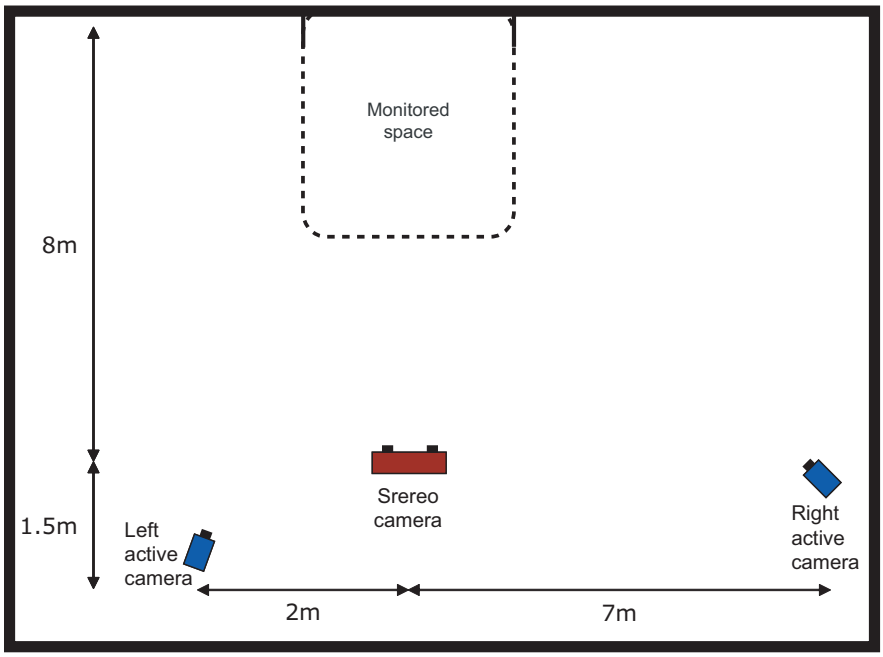


Figure 4.8. Top view of experiment environment

camera with respect to the PT unit. The reference orientation for each camera, denoted (φ_0, γ_0), is saved once the global system calibration is done.

The final rotation angles φ_f and γ_f are given by:

$$\varphi_f = \frac{\varphi + \varphi_0}{res} \qquad \gamma_f = \frac{\gamma + \gamma_0}{res} \tag{4.7}$$

where res is the PT resolution.

Each of the two angle is sent, subsequently, to the appropriate PT unit motor through a rotation command. The time spent by the PT unit to reach the target rotation, referred to as *control delay*, depends on the motor speed and the wideness of the angles. The rotation speed is as high as 70[rpm] for our designed unit.

4.4. Experimental Results

To evaluate the proposed system, I conducted preliminary experiments where three aspects were considered: 1) System operation, 2) 3D face position estimation accuracy, and 3) Robustness against lighting changes.

Figure 4.9. Sample images taken during the system operation experiments.

As shown in Figure 4.8, the monitored area has a size of $3 \times 2[m^2]$ and located at the entrance of our experiment room. The cameras were fixed at the ceiling, high of about $3m$. The active cameras were fixed at different distances from the stereo camera and at opposite sides. The purpose was to test the influence of spatial configuration over the performance of the system. The camera on the left was set at a distance of 2[m] from the stereo camera, while the one on the right was set as much as 7[m] further. 25[mm] lenses were used on the stereo camera, and 50[mm] lenses on the active cameras.

4.4.1 System Operation

The goal of this experiment is to show how the different components of our system behave when a person is within the monitored area, and also to test the influence of the spatial camera placement on the system performance. A person was invited to access and walk through the covered area while facing the stereo camera.

In the column (ii) of Figure 4.9, the left stereo camera images on which are

superimposed the respective bounding squares indicating the detected faces, are shown. In (i) and (iii) columns of this figure, the two active camera images taken at the same moment are shown. As for the last column, it shows the positions of the subject within the covered space. The difference in the position of the face with respect to the higher resolution image centers from one raw to one another, is due essentially to the camera control delay while the subject is in motion, in addition to the resolution of the PT unit (0.37°). It can be noticed that this variation is more important in the camera with longer distance from the stereo camera (right camera). This means that the control error for an active camera becomes more important as the angle between its projection axis and this of the stereo camera increases. We can conclude that this angle should be taken into account in the spatial design of the proposed system. In general systems with several stereo and active cameras, every stereo camera should be associated the closest active cameras with the narrowest angles.

4.4.2 Accuracy of the 3D Face Position Estimation

In this experiment, a person was asked to walk through a rectangular trajectory within the covered space while facing the stereo camera. The goal was to see if the box's shape was reflected correctly by the result of the process shown in Figure 4.5. To evaluate the accuracy of the 3D face position estimate, the result is plotted in the 3D graph of Figure 4.10. The Z axis represents the estimated height, while the plane defined by X and Y axes represents the room's floor. It is clear from the figure that the shape of the box was significantly well reflected, although there is oscillation about the data. This is due to the fact that the subject's head motion naturally oscillates when walking.

4.4.3 Sensitivity to Lighting Changes

The most important factor, for an application of the proposed system in outdoor environments, is the lighting changes. To evaluate the sensitivity to lighting changes, I conducted the following experiment. A subject sits still on a chair and faces the stereo camera. I execute the face detection and 3D position estimation under different lighting conditions. For each condition, the estimation

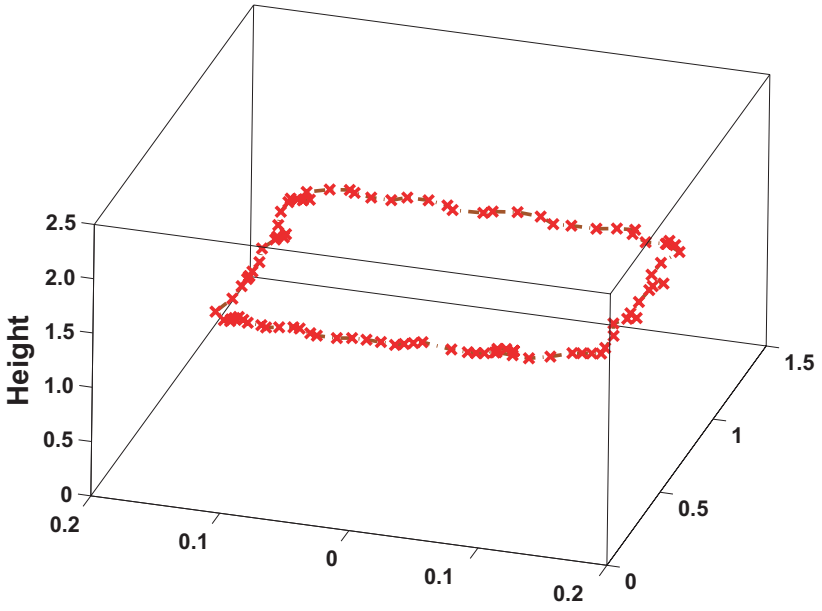


Figure 4.10. Estimated 3D face position while walking around a box.

is executed continuously for a certain duration with a time step equal to 1[sec]. The result is shown in Figure 4.11. The upper row shows four images of the subject under four lighting conditions. The color below each image indicates the corresponding plotting color in the four graphs. The X, Y, and Z-coordinates of the estimated positions are plotted in the graphs of Figure 4.11(b), 4.11(c), and 4.11(d) respectively. These coordinates are expressed in the left stereo camera coordinates. In addition, the depth of the position with respect to the left stereo camera is plotted in the graph of Figure 4.11(e). We can notice that the variation for all coordinates is less then 2[cm]. Moreover, this variation is the same under the same lighting condition and between the four. This means that changing the lighting condition does not affect the system and the same variation remains as it is under the same light.

The result of this experiment is very important since it indicates that allows the system would to work correctly in natural outdoor environments.

4.5. Conclusion

In this chapter, I presented a system that combines static stereo cameras with wide FOV and active high resolution PT cameras into a network platform. Useful for visual surveillance applications, the goal is to provide high resolution face images while covering a wide area. The experiments have shown a good performance of the proposed system in terms of 3D face position estimation and consistency against the lighting changes. These characteristics allow the system to work correctly in natural environments. What I have proposed can be regarded as an atomic unit of what could be implemented in a real system.

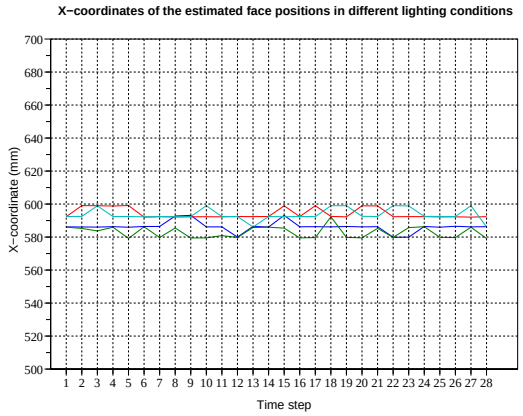
As a future work, more evaluations and an extension to a larger system involving more cameras and covering wider areas, are planned. This extension will require further considerations such as:

- System architecture: depending on the wideness and nature of area to cover different architectures should be adopted. i.e. the system designed to cover a gate differs from the one to cover a waiting space.

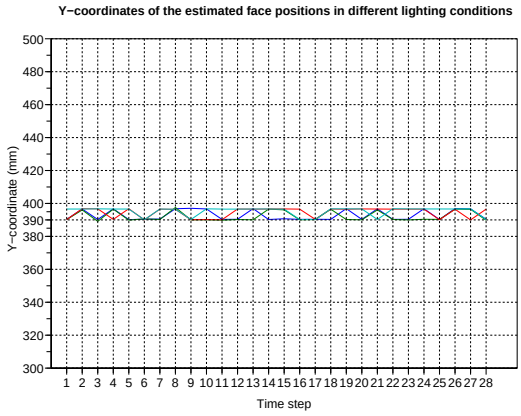
- Active camera assignment: face orientation if known can be taken as a selection criterion to find the best camera facing the face. This is especially important when a stereo camera is associated more than one active camera.

- Connection to other applications: Face recognition and registration modules are planned to be integrated to the system. The goal is twofold: 1) discriminates the already registered faces, 2) recognize the detected faces.

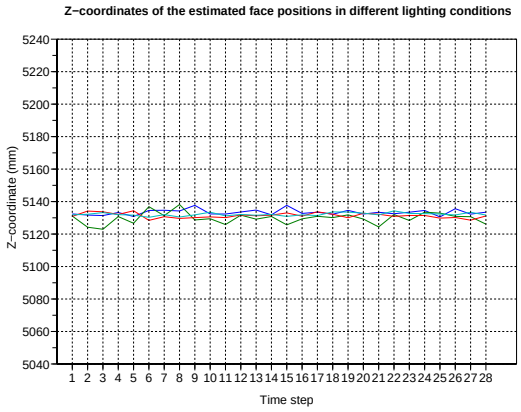
(a) The images and their respective graph color.



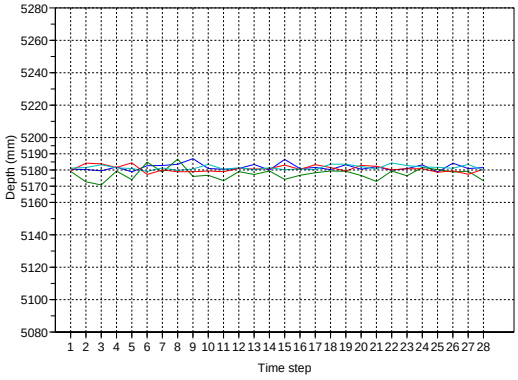
(b) Plot of X-coordinates.



(c) Plot of Y-coordinates.



(d) Plot of Z-coordinates.



(e) Plot of depth.

Figure 4.11. The variation of the 3D face position estimation against the change in lighting conditions.

Chapter 5

Conclusion and Future Work

The ultimate goal of the thesis is to study how information from different types of visual sensors are fused to achieve a common task in a computer vision application. I presented an active camera control for high-resolution imaging. The proposed camera system combines Active PT high-resolution and stationary wide-FOV cameras within a networked platform. The high-resolution cameras can access the whole scene but cannot cover it entirely at a given time, though they provide high-resolution images. On the other hand, the wide-FOV cameras can cover, continuously, the entire scene while providing low resolution images. These low resolution images are not suitable as outputs of the system. However, they can provide useful geometric information about the scene. These information are used to steer the high-resolution cameras to specific targets in the monitored area.

In this thesis, I mainly considered 3D reconstruction applications. The final 3D reconstruction is produced in postprocessing using sophisticated 3D reconstruction algorithms such as deformable mesh model [8] and space carving [23] based on photometric consistency. Basically, high-resolution images are used for this reconstruction. However, wide-FOV images are useful when the narrow-FOV cameras do not cover the entire object, to recover and reconstruct the non-covered areas.

I considered the general case where the narrow-FOV cameras can capture only partial views of an object in the scene but with high resolution. In such circumstances and based on photometric consistency criteria, one camera can get

different visibility toward different parts of the object following the shape and the posture of the object. I presented an active PT camera assignment scheme based on the analysis of a shape reconstructed based on the wide-FOV camera images as a preprocessing step. The major issue was the automatic assignment of each narrow-FOV camera to an appropriate part of the target in order to get high-resolution images of the whole object. The shape analysis is based on several constraints derived from the requirements of photometric consistency.

In order for the system to be useful for real time applications, the processing time is an important factor to be taken into account. I presented a hardware-accelerated camera assignment scheme to speedup the assignment scheme. This by modifying the first scheme so that it can run in the Graphics processing unit.

The assignment scheme is based on a visibility analysis of a shape reconstructed in real-time. Most of real-time shape reconstruction methods produce voxel-based representations. However, the voxel-based representation is not suitable for visibility analysis. I proposed, in chapter 4, a new method for shape from silhouettes. The proposed methods takes benefits of the modern GPU to achieve a fast processing time and produce a surface-based representation of the shape.

In order to show the practicability of the proposed system, I presented a networked heterogeneous camera system for high resolution face images useful for visual surveillance applications. The goal was to cover a wide area while providing high-resolution images of the scene. We proposed to combine wide-FOV static stereo and high-resolution active PT cameras within a networked platform. Similarly to the first system, the full coverage of the scene cannot be ensured continuously by the high-resolution cameras, unless a huge number of them is used. The stereo cameras, thanks to their wide FOV and stereo processing, can cover the scene and locate the visible faces within it. The information provided by stereo cameras are useful to steer the active cameras to gaze the faces and capture closeup views of them.

As a future work, we are planning to implement the surface reconstruction and assignment scheme in a real system. This will allow the identification of the limitations, especially the physical limitations, of the PT units of the active camera and investigate how these constraints caould be considered in the assignment process.

The heterogeneous camera system for high resolution face images presented in this thesis can be regarded as an atomic unit of what could be implemented in a real visual surveillance system. We are working on an extension of this system to a support system for visual monotoring of public spaces using a sparsely distributed camera system. We will investigate the semantic human appearance description, indexing, and recognition for visual monitoring of public spaces.

References

[1] R. Collins, O. Amidi, and T. Kanade, "An active camera system for acquiring multi-view video", *In Proceedings of the International Conference on Image Processing (ICIP'02)*, pp. 517-520, September, 2002

[2] D. Comaniciu, V. Ramesh, and P. Meer, "Real-time tracking of non-rigid objects using mean shift", *IEEE Computer Vision and Pattern Recognition*, Vol. 22(8), pp. 142.149, 2000.

[3] S.-N. Lim, A. Elgammal, L. S. Davis, "Image-based Pan-Tilt Camera Control in a Multi-Camera Surveillance Environment", *IEEE International Conference on Multimedia and Expo (ICME 2003)*, pp. 205-212, July, 2003.

[4] S. Yous, N. Ukita, M. Kidode, "Multiple Active Camera System for High Resolution 3D Video", *Academy Publisher, Journal of Multimedia*, Vol. 2(1), pp. 10-19, February 2007.

[5] T. Kanade, P. Rander, and P.J. Narayanan, "Virtualized Reality: Constructing Virtual Worlds from Real Scenes", *IEEE Multimedia, Immersive Telepresence*, Vol. 4, pp. 34-47, 1997.

[6] S. Yous, N. Ukita, and M. Kidode, "Multiple Active Camera Assignment for High Fidelity 3D Video", *To appear in the proceeding of the 4th IEEE International Conference on Computer Vision Systems (ICVS2006), NY-USA*, 2006.

[7] S.Yous, N. Ukita, and M. Kidode, "Toward a High Fidelity 3D Video: Multiple Active Camera Control Assignment", *IPSJ SIG Technical Report, 2006-CVIM-152*, Vol. 5, pp. 85-92, 2006.

[8] T. Matsuyama, X. Wu, T. Takai, S. Nobuhara, "Real-Time 3D Shape Reconstruction, Dynamic 3D Mesh Deformation, and High Fidelity Visualization for 3D Video", *International Journal on Computer Vision and Image Understanding*, Vol. 96, pp.393-434, 2004.

[9] X. Wu and T. Matsuyama, "Real-Time Active 3D Shape Reconstruction for 3D Video", *In the proceeding of the 3rd International Symposium on Image and Signal Processing and Analysis,Rome, Italy*, pp. 186-191, 2003.

[10] S. Moezzi, L. Tai, P. Gerard, "Virtual view generation for 3d digital video", *IEEE Multimedia*, Vol. 4(1), pp. 18-26, 1997.

[11] E. Borovikov, L. Davis, "A distributed system for real-time volume reconstruction", *In Proceedings of International Workshop on Computer Architectures for Machine Perception, Padova, Italy*, pp. 183-189, 2000.

[12] G. Cheung, T. Kanade, "A real time system for robust 3d voxel reconstruction of human motions", *In Proceedings of Computer Vision and Pattern Recognition, South Carolina, USA*, pp. 714-720, 2000.

[13] J. Carranza, C. Theobalt, M. A. Magnor, H.-P. Seidel, "Free-viewpoint video of human actors", *ACM Transactions on Computer Graphics*, Vol. 22(3), pp. 569-577, 2003.

[14] M. Li, M. Magnor, H.-P. Seidel, "Hardware-accelerated visual hull reconstruction and rendering", *In Proceedings of Graphics Interface (GI'03), Halifax, Canada*, pp. 65-71, 2003.

[15] M. Christie, R. Machap, J. M. Normand, P. Olivier, J. Pickering, "Virtual Camera Planning: A Survey", *In proceedings of the 5th International Symposium on Smart Graphics, Frauenworth Cloister, Germany*, pp. 40-52, 2005.

[16] J. Blinn, "Where am I? what am I looking at?", *IEEE Computer Graphics and Applications*, Vol 8(4), pp. 76-81, 1988.

[17] C. Ware and S. Osborne, "Exploration and virtual camera control in virtual three dimensional environments", *In proceedings of the Symposium on Interactive 3D Graphics, New York, NY, USA*, ACM Press, pp. 175-183, 1990.

[18] N. Courty and E. Marchand, "Computer animation: A new application for image-based visual servoing", *In Proceedings of IEEE International Conference on Robotics and Automation (ICRA'2001)*, Vol 1, pp. 223-228, 2001.

[19] W. H. Bares, J. P. Gregoire, and J. C. Lester, "Realtime Constraint-Based Cinematography for Complex Interactive 3D Worlds", *In Proceedings of AAAI-98/IAAI-98*, pp. 1101-1106, 1998.

[20] K. Yachi, T. Wada, and T. Matsuyama, "Human Head Tracking Using Adaptive Appearance Models with a Fixed-Viewpoint Pan-Tilt-Zoom Camera", *In the proceeding of the Fourth IEEE International Conference on Automatic Face and Gesture Recognition*, pp. 150-155, 2000.

[21] G.K. Cowan and P.D. Kovesi, " Automatic sensor placement from vision task requirements", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 10(3), pp. 407-416, May 1988.

[22] T. Matsuyama, "Cooperative Distributed Vision - Dynamic Integration of Visual Perception, Action, and Communication", *In Proceedingd of DARPA Image Understanding Workshop*, pp. 365-384, 1998.

[23] KN Kutulakos, SM Seitz, "A Theory of Shape by Space Carving", *International Journal of Computer Vision*, Vol. 38(3), pp. 199-218, 2000.

[24] M. Li, M. Magnor, and H.P. Seidel, "A Hybrid Hardware-Accelerated Algorithm for High Quality Rendering of Visual Hulls", *In Proceedings of Graphics Interface (GI'04)*, pp. 41-48, 2004.

[25] S. Yous, M. Kidode, "A Fast Surface-Based Visual Hull Reconstruction", *In the Proceedings of the IAPR Conference on Machine Vision Applications (MVA'07), Tokyo-Japan*, May 2007.

[26] J. Isidoro, S. Sclaroff, "Stochastic mesh-based multiview reconstruction", *In Proceedings of the 1st International Symposium on 3D Data Processing Visualization and Transmission (3DPVT'02),Padova,Italy*, Vol. 1, pp 568-577, 2002

[27] J. Isidoro and S. Sclaroff, "Stochastic refinement of the visual hull to satisfy photometric and silhouette consistency constraints, ", *In Proceedings of the International Conference on Computer Vision (ICCV'03)*, pp. 1335-1342, 2003.

[28] M. Tarini, M. Callieri, C. Montani, C. Rocchini, "Marching Intersections: An Efficient Approach to Shape-from-Silhouette", *In Proceedings of the Vision, Modeling, and Visualization Conference*, pp. 255-262, 2002.

[29] C. Rocchini, P. Cignoni, F. Ganovelli, C. Montani, P. Pingi, R. Scopigno, "Marching Intersections: an Efficient Resampling Algorithm for Surface Management", *In Proceedings of the International Conference on Shape Modeling and Applications (SMI'01)*, pp. 296-305, 2001.

[30] W. E. Lorensen and H. E. Cline, "Marching cubes: A high resolution 3D surface construction algorithm", *In Proceedings of the ACM Computer Graphics (SIGGRAPH'87)*, vol. 21, pp. 163-170, 1987.

[31] A. Laurentini, "The visual hull concept for silhouette-based image understanding", *IEEE Transactions on Pattern Analysis and Machine intelligence*, Vol. 16(2), pp.150-162, 1994.

[32] B.G. Baumgart, "Geometric Modeling for Computer Vision", *PhD thesis, Stanford University*, 1974.

[33] W. Matusik, C. Buehler, R. Raskar, S. J. Gortler, and L. McMillan, "Image-based visual hulls", *In Proceedings of the ACM Computer Graphics (SIGGRAPH'00)*, pp. 369-374, 2000.

[34] W. Matusik, C. Bueler, and L. McMillan, "Polyhedral visual hulls for real-time rendering", *In Proceedings of the 12th Eurographics Workshop on Rendering*, pp. 115-125, 2001.

[35] W. Matusik, C. Buehler, L. McMillan, and S. Gortler, "An Efficient Visual Hull Computation Algorithm", *Technical Memo 623, LCS, MIT*, 2002.

[36] S. Guha, S. Krishnan, K. Munagala, and S. Venkat, "Application of the two-sided depth test to CSG rendering", *In Proceedings of Symposium on Interactive 3D Rendering*, pp. 177-180, 2003.

[37] N. Stewart, G. Leach, and S. John, "An improved Z- buffer CSG rendering algorithm", *In Proceedings of the SIGGRAPH/Eurographics workshop on graphics hardware*, pp. 25-30, 1998.

[38] T. F. Wiegand, "Interactive rendering of CSG models", *Computer Graphics Forum*, Vol. 15(4), pp. 249.261, 1996.

[39] M. Li, M. Magnor, and H.P. Seidel, "Hardware-accelerated visual hull reconstruction and rendering", *In Proceedings of Graphics Interface (GI'03)*, pp. 65-71, 2003.

[40] K.M. Cheung, T. Kanade, J.Y. Bouguet, and M. Holler, "A real time system for robust 3d voxel reconstruction of human motions", *In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR'00)*, Vol. 2, pp. 714-720, 2000.

[41] W.N. Martin and J.K. Aggarwal, "Volumetric description of objects from multiple views", *IEEE Transactions on Pattern Analysis and Machine intelligence*, Vol. 5(2), pp. 150-158, 1983.

[42] C.H. Chien and J.K. Aggarwal, "Volume/surface octress for the representation of three-dimensional objects", *Computer Vision, Graphics and Image Processing*, Vol.36(1), pp. 100-113, 1986.

[43] R. Szeliski, "Rapid Octree Construction from Image Sequences", *Computer Vision, Graphics and Image Processing*, vol. 58(1), pp. 23-32, 1993.

[44] J.J. Koenderink, "What Does the Occluding Contour Tell us About Solid Shape?", *Perception*, Vol.13, pp. 321-330, 1984.

[45] R. Cipolla and A. Blake, "Surface Shape from the Deformation of Apparent Contours", *International Journal of Computer Vision*, Vol. 9, pp. 83-112, 1992.

[46] E. Boyer and M.-O. Berger, "3D surface reconstruction using occluding contours", *International Journal of Computer Vision*, Vol. 22(3), pp. 219-233, 1997.

[47] J. Goldfeather, J. P. M. Hultquist, and H. Fuchs, "Fast constructive-solid geometry display in the pixelpowers graphics system", *In Proceedings of the ACM Computer Graphics (SIGGRAPH'86)*, pp. 107-116, 1986.

[48] S. D. Roth, "Ray Casting for Modeling Solids", *Computer Graphics and Image Processing*, Vol. 18, pp. 109-144, 1982.

[49] J. S. Franco and E. Boyer, "Exact Polyhedral Visual Hulls", *In Proceedings of the 14th British Machine Vision Conference*, pp. 329-338, 2003.

[50] C. Everit, "Interactive order-independent transparency", *Technical report, Nvidia Corporation*, 2002.

[51] I. Buck, "Gpgpu: General-purpose computation on graphics hardware", *ACM SIGGRAPH Course Notes*, 2004.

[52] Prinston Shape Retrieval and Analysis Group, "Prinston Shape Benchmark", http://shape.cs.princeton.edu/benchmark/

[53] M. Yang, D. Kriegman and N. Ahuja, "Detecting faces in images: A survey", *IEEE Transactions on Pattern Analysis and Machine intelligence*, Vol. 24(1), pp.34-58, 2002.

[54] S. Yous, A. Khiat, M. Kidode, T. Ogasawara, "Networked Heterogeneous Camera System for High Resolution Face Images", *In the Proceedings of 2nd International Symposium on Visual Computing (ISVC'06), Lake Tahoe, NV-USA*, LNCS, Vol. 4292(2), pp.88-97, Springer, 2006.

[55] A. Khiat, S. Yous, T. Ogasawara, M. Kidode, "Combining Fixed Stereo and Active Monocular Cameras into a Platform for Security Applications", *In the Proceedings of the IEEE International Conference on ROBOTICS and BIOMIMETICS, Kunming, China*, December 17-20, 2006.

[56] Z. Zhang, "A flexible new technique for camera calibration", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 22(11), pp 1330-1334, 2000.

[57] S. Stillman, R. Tanawongsuwan, and T. Essa, "A System for Tracking and Recognizing Multiple People with Multiple Cameras", *In Proceedings of the 2nd International Conference on Audio and Vision-based Person Authentication, Washington DC*, pp. 96-101, 1999.

[58] T. Ahmedali and J.J. Clark, "Collaborative Multi-Camera Surveillance with Automated Person Detection", *In Proceedings of the Canadian Conference on Computer and Robot Vision*, pp. 3-10, 2006.

[59] A. Hampapur, S. Pankanti, A. Senior, Y-L. Tian, L. Brown, and R. Bolle, "Face Cataloger: Multi-Scale Imaging for Relating Identity to Location", *In Proceedings of IEEE Conference on Advanced Video and Signal Based Surveillance*, pp.13-20, 2003.

[60] J. Krumm, S. Harris, B. Meyers, B. Brumitt, M. Hale and S. Shafer, "Multi-Camera Multi-Person Tracking for EasyLiving", *In Proceedings of IEEE Workshop on Visual Surveillance*, pp. 3-10, 2000.

[61] T. Wada and T. Matsuyama, "Appearance Sphere: Background Model for Pan-Tilt-Zoom Camera", *In Proceedings of IEEE International Conference on Pattern Recognition*, pp. 718-722, 1996.

[62] P. Viola and M. Jones, "Rapid Object Detection using a Boosted Cascade of Simple Features", *In Proceedings of IEEE Computer Vision and Pattern Recognition*, pp. 511-519, 2001.

List of publications

Journal Paper

1. S. Yous, N. Ukita, M. Kidode, "Multiple Active Camera System for High Resolution 3D Video", *Academy Publisher, Journal of Multimedia*, Vol. 2(1), pp. 10-19, February 2007.

International Conferences

1. S. Yous, M. Kidode, "A Fast Surface-Based Visual Hull Reconstruction", *In the Proceedings of the IAPR Conference on Machine Vision Applications (MVA'07), Tokyo-Japan*, May 2007.

2. A. Khiat, S. Yous, T. Ogasawara, M. Kidode, "Combining Fixed Stereo and Active Monocular Cameras into a Platform for Security Applications", *In the Proceedings of the IEEE International Conference on ROBOTICS and BIOMIMETICS, Kunming, China*, December 17-20, 2006.

3. S. Yous, A. Khiat, M. Kidode, T. Ogasawara, "Networked Heterogeneous Camera System for High Resolution Face Images", *In the Proceedings of 2nd International Symposium on Visual Computing (ISVC'06), Lake Tahoe, NV-USA*, LNCS, Vol. 4292(2), pp.88-97, Springer, 2006.

4. S. Yous, N. Ukita, and M. Kidode, "Multiple Active Camera Assignment for High Fidelity 3D Video", *To appear in the proceeding of the 4th IEEE International Conference on Computer Vision Systems (ICVS2006), NY-USA*, 2006.

National Conferences/Meetings

1. S. Yous, N. Ukita, M. Kidode, "Toward a High Fidelity 3D Video: Multiple Active Camera Control Assignment", *Information Processing Society of Japan, Computer Vision and Image Media, No.2005-CVIM-152, Osaka, Japan*, pp. 85-92, 2006.

2. S. Yous, N. Ukita, M. Kidode, "Epipolar sweeping and color-shape consistency criterion for Dense 3d reconstruction and occlusion detection", *Image Recognition and Understanding Symposium (MIRU'05), Awaji, Japan*, pp.1247-1254, 2005.


الحمد لله رب العالمين