AALBORG UNIVERSITY
DEPARTMENT OF HEALTH SCIENCE AND TECHNOLOGY

Fredrik Bajers Vej 7 ▪ DK-9220 Aalborg East Phone +45 96 35 86 00

TITLE:	Handling Occlusions in Augmented Reality
THEME:	Image Analysis and Computer Graphics
PROJECT PERIOD:	9/10th Semester. From September 1st 2001 to June 10th 2002
PROJECT GROUP:	1025

Abstract

This report documents the development of a system to resolve occlusions in augmented reality. The system is built as an extension to a reconstruction system made by the VISICS department at K.U. Leuven, Belgium, which is capable of reconstructing a textured 3D surface model of a static scene from a sequence of un-calibrated images.
The developed system consists of 3 parts: Generation of initial occlusion mask using background subtraction, refinement of occlusion mask using visual hull with post processing, and visualization.
The initial occlusion mask is created using background subtraction after an image warping step which transforms images into a reference viewpoint.
Refinement is done in two steps: First edges are refined using a modified visual hull algorithm, and secondly interior holes are classified as real holes or is caused by camouflage effects using an algorithm developed by the project group.
Visualization produces the final augmented image using the calculated occlusion mask. The method presented is optimized for the capabilities of modern 3D hardware.
A prototype has been implemented in C++ using OpenGL and The Visualization Toolkit (VTK) on a Linux platform.
Test of the prototype show that the background subtraction process is able to generate a good initial occlusion mask even in cases with complex silhouettes. Test on synthetic data show that the two-step refinement process is able to eliminate errors in the initial occlusion mask.
It is concluded that the augmented images generated by the developed system contain less visual artifacts than using the raw depth information of the reconstruction system, even without the refinement step. Adding the refinement step will most likely decrease the number of artifacts caused by camouflage effects in the interior of the silhouettes.

PARTICIPANTS:
Peter Korsgaard
Lars Jochumsen Kristensen

SUPERVISOR:
Claus B. Madsen,
Aalborg University

Kurt Cornelis,
Katholieke Universiteit Leuven

PUBLICATIONS:	6
NUMBER OF PAGES:	94
FINISHED:	June 10th 2002

AALBORG UNIVERSITET
INSTITUT FOR SUNDHEDSTEKNOLOGI

Fredrik Bajers Vej 7 ▪ DK-9220 Aalborg Øst   Telefon +45 96 35 86 00

TITEL:	Handling Occlusions in Augmented Reality
TEMA:	Billedanalyse og Computer Grafik
PROJEKTPERIODE:	9/10. semester. Fra d. 1. september 2001 til d. 10. juni 2002
PROJEKTGRUPPE:	1025

DELTAGERE:
Peter Korsgaard
Lars Jochumsen Kristensen

VEJLEDER:
Claus B Madsen,
Aalborg University

Kurt Cornelis,
Katholieke Universiteit Leuven

Synopsis

Denne rapport dokumenterer udviklingen af et system til at håndtere okklusioner i augmented reality. Systemet er bygget som en udvidelse af et rekonstruktionssystem udviklet af VISICS afdelingden på Katholieke Universiteit Leuven, Belgien, som kan rekonstruere en tekstureret 3D model af en statisk scene ud fra en sekvens af ukalibrerede billeder.
Det udviklede system består af 3 dele: Beregnelse af indledende okklusionsmaske via background subtraction, forfinelse af okklusionsmaske ved brug af visual hull med efterprocessering, og visualisering.
Den indledende okklusionsmaske laves med background subtraction efter et billedewarpings-trin, som transformerer billeder til et reference synspunkt.
Forfinelse sker i to trin: Først forfines kanter med en modificeret visual hull algoritme, og dernæst bliver interne huller klassificeret som rigtige huller eller som værende på grund af camouflageeffekter med en algoritme udviklet af projektgruppen.
Visualisering producerer det endelige augmented billede ved hjælp af den beregnede okklusionsmaske. Den præsenterede metode er optimeret til mulighederne i moderne 3D hardware.
En prototype er udviklet i C++ og gør brug af OpenGL og The Visualization Toolkit (VTK) på en Linux Platform.
Test af prototypen viser at background subtraction kan lave en god indledende okklusionsmaske, også i tilfælde med komplekse silhuetter. Test på syntetisk data viser at den to-delte forfinelsesproces kan fjerne fejl i den indledende okklusionsmaske.
Det konkluderes at de augmentede billeder beregnet af det udviklede system indeholder færre visuelle fejl end hvis den rå dybdeinformation fra rekonstruktionssystemet bruges, selv uden brug af forfinelsesprocessen. Tilføjes forfinelsesprocessen til systemet vil antallet af fejl grundet camouflageeffekter indenfor silhuetterne sandsynligvist falde.

OPLAGSTAL:	6
SIDEANTAL:	94
AFSLUTTET:	10. juni 2002

Preface

This report is the result of a 9/10th semester Computer Vision and Graphics project, developed during the period from September 1st 2001 to June 10th 2002. During the period from November 1st 2001 to June 1st 2002 the project group was stationed at the Katholieke Universiteit Leuven, Belgium.

Figures, tables and equations used in the report are referred to as figure/table/equation X.Y, where X denotes the chapter number and Y represents the figure/table/equation number in the chapter.

References are in the form [⟨author⟩, ⟨year⟩], i.e. (Surname(s), 2000). All references are listed in the bibliography located before the appendices in the report.

The project group would like to thank Kurt Cornelis and Maarten Vargauwen from VISICS, Katholieke Universiteit Leuven, Belgium and everyone else who made it possible for us to do this project in Leuven.

Aalborg, June 10th, 2002.

Peter Korsgaard

Lars Jochumsen Kristensen

Contents

List of Figures

List of Tables

CHAPTER 1

Introduction

Recently there has been a lot of development in computer graphics. There has especially been an increasing interest in combining live video with computer-synthesized objects (augmented reality). This approach opens a new world of applications where the user interacts with virtual objects in a real scene.

The task of an augmented reality system is to make the virtual object appear as a real object in the scene. The appearance problem can be divided into three main computer vision/graphics problems,

Registration is dealing with the registration of the cameras and the geometry of the scene. Recently there has been conducted a lot of research in reconstructing a real scene using inexpensive camcorders. Registration is important as it makes it possible to position virtual objects in a real scene.

Light parameters of the real scene must be estimated and approximated during rendering of virtual objects in order to create a convincingly augmented reality system.

Occlusion is when virtual objects are located between real objects in the scene and the camera. The objects in front of the virtual objects must occlude the virtual objects and virtual objects must occlude real objects positioned behind them.

In order to make a successful augmented reality system all three problems must be solved. This is illustrated in figure 1.1 where a recording of a man is combined with computer generated objects attached to his hand. It is important that all three problems are solved as it othwerwise would result in visible artifacts in the transition between the hand and the attached objects.

This report describes a possible solution to the occlusion problem in an augmented reality system. Estimation of light sources in the real scene is not part of the project. Registration of camera parameters and the geometry of the scene is done using a reconstruction system made by the VISICS department of Katholieke Universiteit Leuven, Belgium (Pollefeys, 2000). This reconstruction system estimates camera parameters and a 3D textured surface model of a scene from a sequence of uncalibrated images. More details on the reconstruction system can be found in chapter 2.

Figure 1.1: An example of augmented reality where a recording of a man is combined with computer generated objects attached to his hand (image from Inspector Gadget (Berton, 2002))

1.1 Initial Problem

Based on the above the initial problem for this project can thus be formulated as:

> How can an augmented reality system, which solved the occlusion problem, be built on top of the reconstruction system from VISICS?

1.2 Report Structure

The rest of the report is divided into the following parts:

Pre-analysis: Gives an overview of the reconstruction system from VISICS as well as problems with their approach.

System Specification: The extension to be developed is specified based upon the pre-analysis and external demands.

Method Analysis: Analysis of the examined methods and the theory behind.

Test: Results of tests on real images.

Conclusion: Conclusion and discussion of the results and the project.

Appendices: Background information on subjects dealt with in the report and images which supports the chapters.

CHAPTER 2

Pre-analysis

This chapter introduces the reconstruction system developed at VISICS, K.U Leuven, Belgium. The system is able to estimate camera parameters and generate a textured 3D polygonal model from a sequence of images of a static scene recorded with an uncalibrated camera. No special calibration objects or fiducial markers in the images are required. This chapter is based on Pollefeys (2000).

Figure 2.1 illustrates the different steps of the reconstruction system.

The different steps of the reconstruction are briefly described in the following sections.

2.1 Input

The input to the reconstruction system is a sequence of grayscale images. The disparity between consecutive images has to be small as the system relies on matching features within a small window in the images. This is especially important for the first two images as they are used to initialize the system, whereas it is not required that features in the first two images are visible in the entire sequence. Because of these requirements video sequences are optimal as input.

2.2 Feature Extraction/Matching and Relating Images

The first step of the reconstruction is relating consecutive images to each other. This consist of three steps:

1. Extract features
2. Match features in the two images
3. Estimate the fundamental matrix

Features are extracted using the Harris corner detector. Subpixel precision is achieved using quadratic interpolation of the Harris response function.

Matching is performed using correlation on 7x7 pixel windows. Only features nearby each other are matched against each other.

Based on the found matches the fundamental matrix between the two views is estimated. The RANSAC (RANdom SAmpling Consensus) algorithm is used to remove outliers.

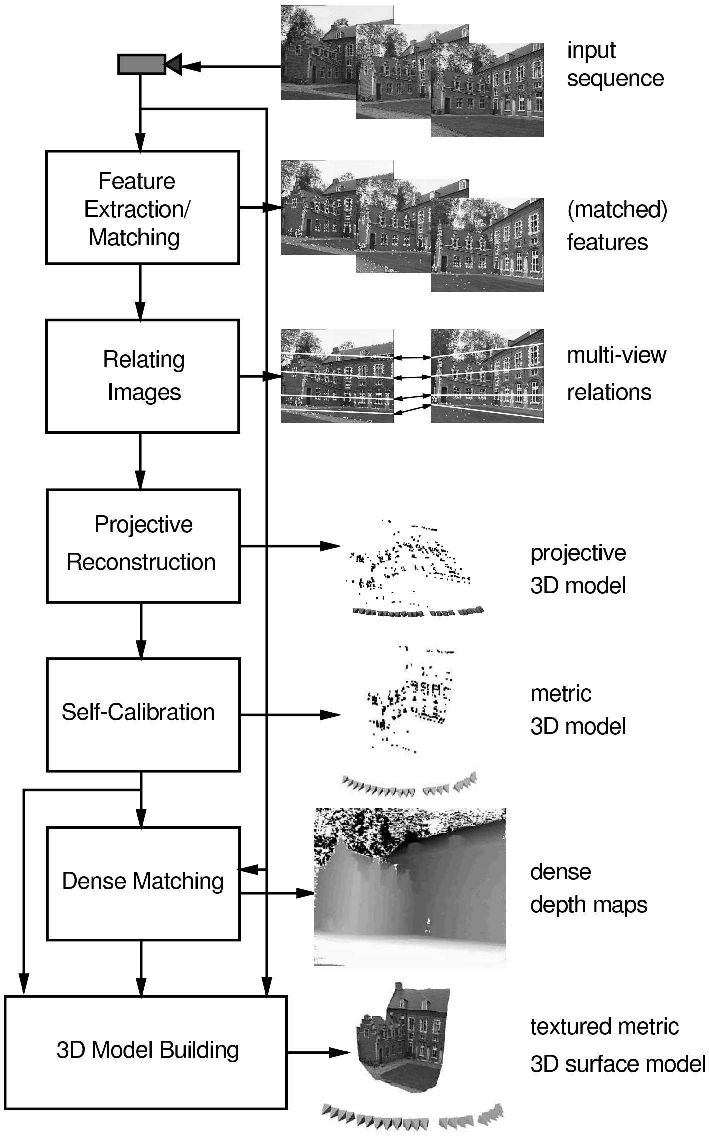


Figure 2.1: Block diagram of the reconstruction system from VISICS. The figure illustrates the different steps of the reconstruction process. (figure 1.7 Pollefeys (2000))

2.3 Projective Reconstruction

The second step is to estimate the structure of the scene and the motion of the camera throughout the sequence up to an arbitrary projective transformation based on the relation between views and correspondences between features. This is performed in four steps:

1. Construct initial frame from the first two images
2. Reconstruct initial structure using triangulation
3. For every additional view:
    3.1. Compute pose for view
    3.2. Refine position of existing structure points
    3.3. Initialize new structure points
4. Refine structure and motion globally through bundle adjustment

The two first images of the sequence are used to determine a reference world frame aligned with the first camera. The second camera is estimated so that the epipolar geometry corresponds to the retrieved fundamental matrix between first and second image.

The structure is reconstructed using triangulation of the corresponding features. For every new view with a correspondence for a given point the point is refined using an Iterated Extended Kalman Filter. New features not already reconstructed are added to the structure as correspondences are found.

2.4 Self-Calibration

The third step is to upgrade the projective structure to an Euclidean structure.

The self-calibration approach used in the reconstruction system estimates intrinsic camera parameters which are unknown but fixed throughout the sequence, therefore zoom and autofocus should not be used. The following parameters are estimated:

- Focal length (f_x, f_y)
- Optical center (o_x, o_y)
- Radial distortion parameters (k_1, k_2, k_3)

The self-calibration process is performed in two steps:

1. Initialization
2. Refinement

Initialization is performed using a linear approach which optains an approximate calibration. The purpose is to estimate the focal length of the camera. It assumes that pixels are rectangular, without skew, and the principal point is close to the center of the image.

Refinement is done using a non-linear optimization process (non-linear least squares). The purpose is to compute the remaining intrinsic parameters.

Some critical camera motions does not result in unique solutions to the self-calibration problem and should therefore be avoided. Examples of critical motions are pure rotation and pure forward motion.

2.5 Dense Depth Estimation

After the Euclidean camera parameters have been estimated, dense depth values can now be calculated using a calibrated structure from motion algorithm. Estimation is performed in three steps:

1. Dense Disparity matching between two views
2. Multiple-viewpoint linking
3. Depth estimation

2.5.1 Disparity matching

Disparity values are calculated using correlation with a window size of 5x5 pixels.

Scene constraints in a calibrated environment can be used to improve the performance of the correlation process. The constraints used in the reconstruction system are:

Epipolar: Correspondences are located on the corresponding epipolar line.

Ordering: For opaque surfaces the order of neighboring correspondences on the epipolar line are preserved.

Uniqueness: Correspondence between any two corresponding points are bidirectional.

Disparity Limit: Maximum disparity distance is limited by the maximum depth range of the observed scene.

Continuity: Disparity values vary mostly continuously.

Correlation is performed with a dynamic programming algorithm using the above constraints.

To improve performance the images are rectified so the epiopolar lines follow scan lines in the images before performing disparity matching .

The reconstruction system uses a method to generate a rectified image using polar coordinates around the epipole and which is guaranteed to create the smallest possible image without compressing any parts of the image.

2.5.2 Multiple-viewpoint linking

The correlation process performs best on image pairs with a small baseline, but triangulation requires a large baseline to perform well. To solve this problem the system uses an approach to track the same point through multiple images by creating links up and down from the current image using the disparity values. The linking process is ended when either no disparity value is found or the disparity is outside the valid depth range of the earlier views in the link. Figure 2.2 illustrates the process.

2.5.3 Depth estimation

Finally depth values are estimated using triangulation. The quality of the depth estimation can be controlled by varying the amount of linked views required to generate a depth estimate. Setting a low threshold gives a high fill rate in the depth map but low accuracy, setting a high threshold does the opposite.

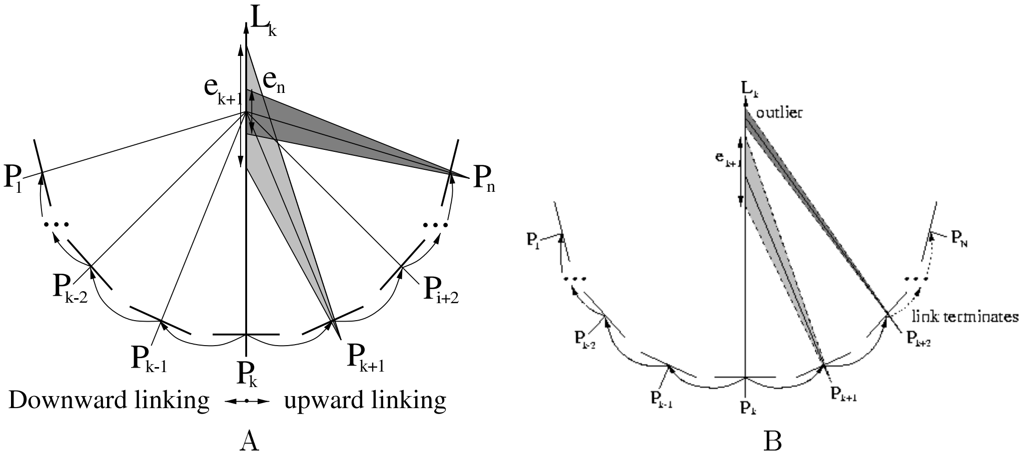


Figure 2.2: Viewpoint linking. The upwards and downwards linking of correspondences (A), and a correspondence outlier (B). (Figure 7.17 Pollefeys (2000))

2.6 3D Model Construction

The last step is to create a 3D mesh suitable for visualization from the depth maps. This can be done in several ways. Currently the reconstruction system is using a very simple method, but a volumetric approach is currently under development.

2.6.1 2D mesh

This approach overlays a 2D triangular mesh on top of a single depth map and then builds a corresponding 3D mesh by positioning the vertices in 3D using the depth values of the depth map. The image which corresponds to the depth map is used as texture with the 2D mesh coordinates as texture coordinates. This approach gives a very dense 3D model as every reconstructed pixel in the depth map becomes a vertice in the 3D model, but it is not able to create a 3D model of areas not visible in the depth map.

2.6.2 Volumetric model

This approach uses a 3D voxel space enclosing the scene. For each depth map all voxels are marked as either between camera and surface (A), behind surface (B) or not seen (C). After all depth maps have been considered the volumetric representation of the surface is the voxels which have been marked at least T_B times as being behind the surface and less than T_A times as being in front, where T_A and T_B are segmentation thresholds. Figure 2.3 illustrates the process. Finally the surface of the volume is converted into a 3D mesh using the marching cubes algorithm. Unfortunately the memory requirements of this algorithm makes it unfeasible for high resolution models.

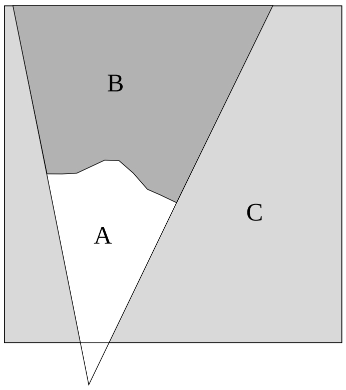


Figure 2.3: Segmentation of voxels in three categories.

2.7 Problems With the Reconstruction System

This section deals with problems with the reconstruction system if it is to be used in an augmented reality system.

(A) (B)

Figure 2.4: Figure (A) shows input image 10 of the botanical sequence. Figure (B) illustrates a reconstruction of the scene from the same point of view. The black areas are regions which are not reconstructed

Figure 2.4 illustrates a scene reconstructed from 37 images recorded in the botanical garden in Leuven, Belgium which is used throughout this section to illustrate the problems with the reconstruction system. Every third image of the complete sequence of input images and rendered views can be seen in appendix B.

The problems can be divided into two sections, problems with:

- Missing reconstruction
- Inaccurate reconstruction

2.7.1 Missing Reconstruction

The three main causes of areas with no reconstruction are:

- Occlusion shadows
- Depth discontinuities

- Missing / Repetitive texture

As the 3D model is created from a single depth map areas occluded in the depth map are not reconstructed as no depth information exist for those areas. Figure 2.5 illustrates the problem.

Figure 2.5: Rendering from viewpoint 15. Occlusion shadows leads to areas with no reconstruction. Also notice the missing reconstruction of the interior of the sign on the right

Areas with depth discontinuities breaks the disparity tracking algorithm as windows containing depth discontinuities change content when seen from different angles. Figure 2.6 illustrates the problem.

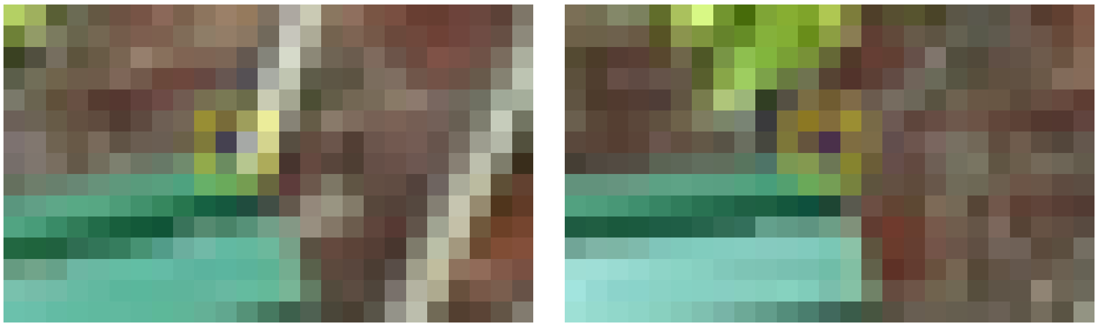


Figure 2.6: Zoomed in views of image 10 and 15. Depth discontinuities are problematic for disparity tracking

If an area does not contain any unique texture then the disparity tracking algorithm will not be able to correctly track the point throughout the sequence, and the viewpoint linking process will quickly reject the disparity values as outliers. This is illustrated in the white sign on the right of figure 2.5.

2.7.2 Inaccurate Reconstruction

Inaccuracy of the reconstruction is mainly caused by two things:

- Camouflage
- Occlusions

Camouflage effects occur when the background color is the same for a area around a 3D point seen from different viewpoints. This can cause the disparity tracking algorithm to track the area around a depth discontinuity as a foreground object even though the point is not. This has

Figure 2.7: Zoomed in view of figure 2.5. The white area is caused by camouflage effects as these parts of the background are reconstructed as foreground

the effect that the silhouette of the foreground object will contain pixels that are part of the background. Figure 2.7 illustrates the problem.

The viewpoint linking procedure described in section 2.5.2 works by tracking the same 3D point throughout the sequence using the disparity values. If no disparity value is found because the 3D point is occluded the link is broken and no further search is done even if the point will no longer be occluded in the next image. If a lot of occlusions occur the accuracy of depth estimation will suffer.

CHAPTER 3

Problem Specification

In the following sections the problem, requirements, and delimitation of the project are specified. First the specific problem of the project is described in detail. Next requirements regarding the accuracy and working environment of the developed system are specified. Finally a delimitation of the project is done.

3.1 Specification

The system to be developed is an extension to an existing system made by the Department of Vision for Industry Communications and Services (VISICS) at K.U. Leuven (Pollefeys, 2000), which is capable of constructing a textured 3D surface model of a scene from a sequence of un-calibrated images. The result of the system to be developed, as specified by Kurt Cornelis (Cornelis, 2001), is to resolve occlusions in augmented reality using the system developed by VISICS.

The task of an augmented reality system is to augment synthetic objects (virtual objects) into a scene based on images acquired of a real environment. The combination of the controlled environment (virtual objects) and the uncontrolled environment (real scene) issues problems like extracting the part of the real scene which is in front of the virtual objects inserted in the scene, especially if working in an un-calibrated environment.

The problem of extracting the part in front of the virtual object is difficult because the 3D surface model is estimated from a series of images acquired using a single non-range camera. As the 3D surface model is not accurate enough to directly extract the part in front of the virtual object without including parts of the background additional techniques is required, see figure 3.2.

The complete system is illustrated in figure 3.1.

As the system uses the acquired images of the real scene to estimate the 3D surface model, noise in the images results in an inaccurate 3D surface model and the estimation of the camera position for each image. Figure 3.2 illustrates the problems of using only the 3D surface model to segment the part in front of the virtual object. Figure 3.2 (a) illustrates an image of a real scene acquired using a hand-held camera. The image serves as a reference to figure 3.2 (b) & (c). A virtual object is placed behind the lantern hanging on the wall to the right of the scene. The virtual object to be inserted is a green box just for illustration. Figure 3.2 (b) illustrates the image from figure 3.2 (a) merged with the visible parts of the virtual object. The visible parts of the virtual object is determined using the 3D surface model to test for occlusion. Figure 3.2 (c) illustrates

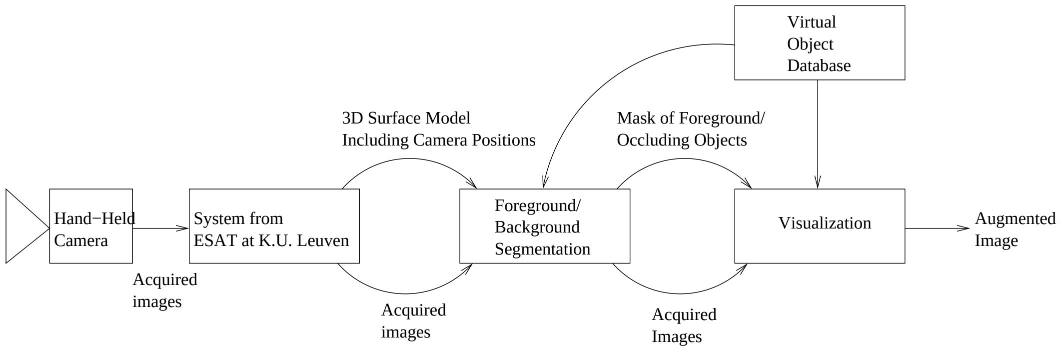


Figure 3.1: Illustrates an overview of the complete system including the extension to the system.

a close up of the part of the scene where the virtual object is merged. As seen in the figure there are problems around the edges of the foreground object, in this case the lantern. Parts of the background is considered foreground (part of the lantern), and parts of the foreground is considered background. Especially when parts of the background is considered foreground it introduces visible problems. This is very clear in the figure 3.2 (c) as the background color is very different from the foreground.

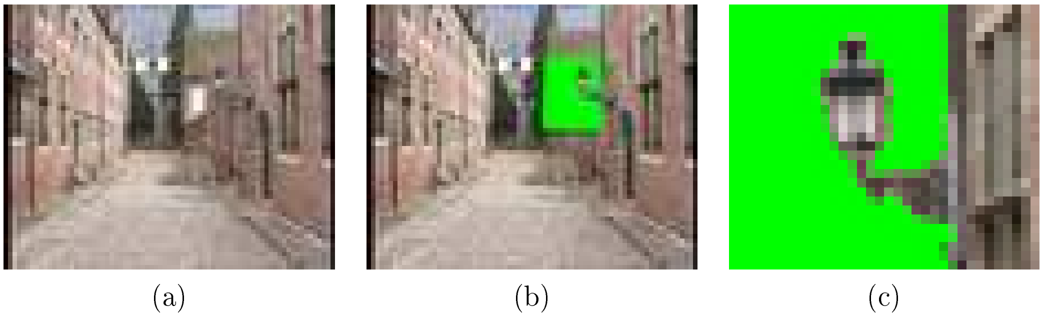


Figure 3.2: Figure (a) illustrates an acquired image of a real scene. Figure (b) illustrating figure (a) merged with the virtual object. Figure (c) illustrates a close up of the merged virtual object. As seen there are problems around the edges of the lantern in front the virtual object, where parts of the background are considered foreground introducing visible artifacts.

The environment the system should be able to operate in is itemized and briefly described below.

Scenes that are used in augmented reality are in this case represented as multiple images acquired using a hand-held camera. The composition of the scene is restricted to an everyday scene including a large range of textures, and the angles of the acquired images toward the scene has to cover the areas of interest of the scene. Meaning the area where the virtual objects is going to be inserted has to be acquired from different angles. The more different and wide-ranging angles the better results are possible.

Motion within the scene causes some problems as the sequence of images is acquired over time. Therefore a moving object is located at a different location in each acquired image. The direct outcome of this is an exclusion of the moving object from the 3D surface model, because the object is considered an outlier by the system from VISICS at K.U. Leuven (Pollefeys, 2000, p. 72).

Light in the scene can take different forms. As everyday scenes are permitted (both indoor and outdoor) the lighting might consist of multiple sources, shadows, specular highlight from e.g. reflecting windows, and varying intensity of the light in the scene. These factors have a large influence on the accuracy of the algorithms used for extracting the part of the scene in front of the virtual object.

3.2 Requirements

Most augmented reality systems are expected to run in real-time so that a user will be able to move about freely within the scene and see a properly rendered augmented images synchronized to his movements. This places two performance criteria to this extension of the system:

- Accuracy of the segmentation of foreground/occluding objects.
- Minimum update rate for the augmentation of the scene.

Accuracy of the Segmentation of Foreground/Occluding Objects

The accuracy of the segmentation is of great importance, because errors are easily visible in the final rendering. In the case of a light foreground object and a darker background, pixel from the background registered as a part of the foreground object will easily be spotted in the final rendering with the virtual object. As the 3D surface model does not have the accuracy to be used directly, it is necessary to support the 3D surface model with a number of other techniques.

Minimum Update Rate for the Augmentation of the Scene

The requirement of real-time is not as important as the accuracy of the segmentation of the foreground/occluding objects. The system from VISICS at K.U. Leuven (Pollefeys, 2000) is running off-line and this issues the necessity of a off-line phase for initializing the system for real-time use. As the extension only operates on the parts of the scene occluded by the virtual object in the actual rendering it is necessary to acquire a good basic image material in the off-line phase. The basic image material consists of multiple images covering the scene from different and wide-ranging angles. As some techniques does not work for real-time use, it is a trade off between accuracy and update rate. The focus of this extension is on accuracy.

3.3 Delimitation

The goal is to resolve occlusions in augmented reality. It has been decided to delimit the project by prioritizing the criteria specified in section 3.2:

1. Accuracy of the segmentation of foreground/occluding objects.
2. Minimum update rate for the augmentation of the scene.

The delimitation of the system can be divided into three types of constraints. The constraints are based on Harville, Gordon, and Woodfill (2001) and listed below:

Scene Constraints

- Multiple images acquired showing the area behind the occluding object from different and wide-ranging angles.
- Transparent objects are not handled.
- The background in the scene has to be of a diffuse material.

Motion Constraints

- Static background behind the virtual object.

Light Constraints

- Specular highlight within the area of the virtual object is not handled.
- Shadows within the area of the virtual object must have properties that classifies them as background.

3.4 The Specific Problem

In summary, and by omitting some insignificant delimitation and constraints, the specific problem addressed in this project can be stated as:

> How can a robust extension be constructed, which by use of an estimated 3D surface model and multiple images acquired from a hand-held camera, resolves occlusions in augmented reality ?

CHAPTER 4

Analysis Overview

The extension to be developed is a post-processing step to the reconstruction system made by the VISICS department at Katholieke Universiteit Leuven, Belgium (Pollefeys, 2000). The result of the reconstruction system is not accurate enough to directly solve occlusions in an augmented reality system. As the result of the reconstruction system is not feasible for augmented reality an extension is developed to refine the result.

The idea is to create a 2D mask of (the projection of) 3D structure in front of the virtual object and cut the virtual object against it. This occlusion mask is used instead of the depth map provided by the reconstruction system.

The extension consists of two steps,

1. Segmentation
2. Visualization

4.1 Segmentation

Segmentation of the scene into foreground and background is not possible using the depth map from the reconstruction system directly as it is not accurate enough. Another approach is to use the color information in the recorded images of the scene and only use the depth maps as initialization.

Different ways of extracting the mask of objects in front of virtual objects exists. The two types investigated in this report are,

Extraction of the edge of objects in front of the virtual objects , this approach is used in active contours, in this case snakes. Snakes follows information in the images which describes the boundary between objects.

Extraction of the parts not behind the virtual objects , this approach is used in background subtraction. Background subtraction uses information in the images which describes the part behind the virtual objects.

The two types focus on different parts of the scene. The snakes focus on the part in front whereas background subtraction focus on the part behind the virtual objects.

Both approaches depend on difference in color between the foreground and background in order to determine which parts are in front of the virtual objects, and which are not. This property makes them sensitive to camouflage effects in the scene as neither of the approaches are able to distinguish between camouflage effects and real structure because they both work on a single image.

Camouflage effects causes different problems in the two approaches. Camouflage effects at the edge of the foreground object causes the snake to add pixels to the occlusion mask, whereas camouflage effects inside the foreground object causes the background subtraction to remove pixels in the occlusion mask. A more detailed description on the problems of camouflage is found in chapter 5 for snakes and chapter 6 for background subtraction.

An approach based on visual hull is used to handle problems caused by camouflage effects.

The approach of using snakes to refine the occlusion mask has already been tried by Kurt Cornelis from K. U. Leuven. The results are not usable for generating the occlusion mask as it is difficult to find a snake that is general enough to cope with problems in real scenes. Chapter 5 contains a detailed description of snakes and some results on how snakes performs in real scenes.

The other approach, using background subtraction for extracting the occlusion mask, is investigated in this project.

4.2 Visualization

The visualization of the virtual objects in the scene is done by clipping the virtual objects against the occlusion mask and merging the result with an image of the real scene.

4.3 Design of the Extension

The extension is divided into four main tasks. They are as follows:

1. Background subtraction
2. Visual hull
3. Post processing of the visual hull representation
4. Visualization

The tasks and their relationship are illustrated in Figure 4.1

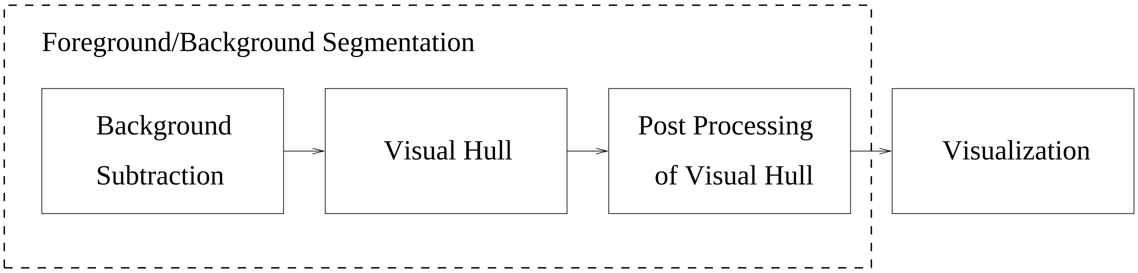


Figure 4.1: The Figure illustrates the subtasks of the extension and their relationship. Figure 3.1 shows where the extension fits into the complete system

The tasks in Figure 4.1 are briefly described in the following,

Background subtraction handles segmentation of the scene in foreground and background based on the location of the virtual object. In order to perform segmentation of the scene a model of the part of the scene behind the virtual object (background model) is estimated. The output is a initial occlusion mask based on the pixel-wise differences between the recorded image and the estimated background model.

A detailed description of the background subtraction is located in chapter 6.

Visual Hull handles the problem of missing outer boundaries caused by camouflage effects. Structure is added to the initial occlusion mask if multiple views agree on the structure. The output is a refined occlusion mask.

A detailed description of computing the visual hull representation is located in section 7.3.

Post processing of Visual Hull handles the problem of holes inside the refined occlusion mask. The cause of the holes are determined (camouflage or real holes), and the occlusion mask is updated depending on if they are caused by camouflage effects or are real holes.

A detailed description of the post process of the visual hull representation is located in section 7.4.

Visualization handles visualization of the result.

A detailed description of visualization is located in chapter 8.

CHAPTER 5

Active Contours

5.1 Overview

Active contours is a technique which can be used to locate a more accurate location of the edges of the foreground object. The problem is that the edge of the foreground object, based on the 3D reconstruction of the scene, is not accurate enough as pixels from the background are considered as part of the foreground and vice versa.

Active contours works in 2D images and is a possible method for extract the edges surrounding a foreground object. This chapter describes the results of the experiments performed by Kurt Cornelis from K. U. Leuven and is based on Xu and Prince (1998) and Cornelis (2000)

5.2 Method

Snakes are a type of active contours and are used for locating object boundaries. They are curves defined within an image domain that can move under influence of internal forces within the curve itself and external forces that are defined in such a way that the snakes will move towards desired features in the image, in this case the edges.

There are typically two problems when using traditional snakes: First, the initial curve must be placed close to the true boundary or else it most likely converges to a wrong result. Second, it has difficulty converging into U-shaped concave boundary regions.

A new class of external forces can be used to overcome most of these problems at the same time. These new external forces are called *gradient vector flow (GVF) fields* and are derived from the images by minimizing an energy functional in a variational framework. The advantages of a snake using GVF (*GVF snakes*) over a traditional snake are its insensitivity to initialization and its ability to move into U-shaped concave boundary regions.

5.2.1 The Traditional Snake Model

A traditional snake is a curve $\mathbf{x}(s) = [x(s), y(s)], s \in [0, 1]$, that moves through an image to minimize the energy functional (Thodberg, 2002),

$$E(\mathbf{x}) = \int_0^1 \left[\frac{1}{2}\alpha(s)\mathbf{x}_s^2 + \frac{1}{2}\beta\mathbf{x}_{ss}^2 + E_{ext}(\mathbf{x}(s))\right] \, ds \quad \text{, where } \mathbf{x}_s \equiv \frac{\partial \mathbf{x}(s)}{\partial s} \tag{5.1}$$

This is a variational formulation where α and β are weighting parameters that controls the snakes tension and rigidity, respectively. The parameter s defines the arc-length when running over the curve. $\mathbf{x}_s$ (length) and $\mathbf{x}_{ss}$ (curvature) denotes the first and second derivatives of $\mathbf{x}(s)$ with respect to s. The two parameters α and β are free and chosen using a heuristic process. The external energy function $E_{ext}(\mathbf{x})$ is derived from the image so that it takes on its smallest values at the features of interest, such as the boundaries.

Typical external energies $E_{ext}(\mathbf{x})$ are designed to pull snakes towards steep edges.

$$E_{ext}^1(x,y) = -|\nabla I(x,y)|^2 \tag{5.2}$$
$$E_{ext}^2(x,y) = -|\nabla(G_\sigma(x,u) * I(x,y))|^2 \tag{5.3}$$

where $G_\sigma(x,u)$ is a two-dimensional Gaussian function with standard deviation σ, ∇ is the gradient operator, and $I(x,y)$ is the gray-scale value of the image at position (x,y).

The boundaries becomes more blurry with increasing σ which in most cases are necessary in order to increase the capture range of the active contour.

External energies for locating lines in a binary image include

$$E_{ext}^3(x,y) = I(x,y) \tag{5.4}$$
$$E_{ext}^4(x,y) = G_\sigma(x,u) * I(x,y) \tag{5.5}$$

The energy function $E(\mathbf{x})$ in equation 5.1 has a stationary value (a critical point in this case a minimum), which means that the dericatives of the function $\mathbf{x}(s)$ vanish ($\mathbf{x}_s = 0$, and $\mathbf{x}_{ss} = 0$) (Weisstein, 2001). A snake minimizing the energy *E(x)* in equation 5.1 must satisfy the Euler-Lagrange differential equation (Weisstein, 2001) which for this case is,

$$\frac{\partial}{\partial s}(\alpha\, \mathbf{x}_s) - \frac{\partial^2}{\partial s^2}(\beta\, \mathbf{x}_{ss}) - \nabla E_{ext}(\mathbf{x}) = \mathbf{0} \tag{5.6}$$

This can be viewed as a force balance equation,

$$\mathbf{F}_{int}(\mathbf{x}) + \mathbf{F}_{ext}(\mathbf{x}) = \mathbf{0} \tag{5.7}$$

where $\mathbf{F}_{int}(\mathbf{x}) = \frac{\partial}{\partial s}(\alpha\, \mathbf{x}_s) - \frac{\partial^2}{\partial s^2}(\beta\, \mathbf{x}_{ss})$ and $\mathbf{F}_{ext}(\mathbf{x}) = -\nabla E_{ext}(\mathbf{x})$. The internal force $\mathbf{F}_{int}(\mathbf{x})$ counteracts stretching and bending of the snake while the external force $\mathbf{F}_{ext}(\mathbf{x})$ pulls the snake towards the desired image boundary.

The energy functional $E(\mathbf{x})$ is minimized or stable when the internal forces of the snake equals the external force of the image,

In order to find a solution to the Euler equation 5.6, the snake is made dynamic by treating $\mathbf{x}(s)$ as a function of time t (iterations) as well as s, i.e. $\mathbf{x}(s,t)$. Then, the partial derivative of $\mathbf{x}(s,t)$ with respect to t is then set equal to the left hand side of equation 5.6 as follows

$$\frac{\mathbf{x}(s,t)}{\partial t} = \frac{\partial}{\partial s}(\alpha\ \mathbf{x}_s) - \frac{\partial^2}{\partial s^2}(\beta\ \mathbf{x}_{ss}) - \nabla E_{ext}(\mathbf{x}) \quad \text{, where } \mathbf{x}_s \equiv \frac{\partial \mathbf{x}(s,t)}{\partial s} \tag{5.8}$$

When the solution $\mathbf{x}(s,t)$ stabilizes, the term $\mathbf{x}_t(s,t)$ vanishes and it is possible to achieve a solution of equation 5.6. A numerical solution to equation 5.8 can be found by discretizing the equation and solving the discrete system iteratively.

Behavior of Traditional Snakes

An example of the behavior of a traditional snake is shown in figure 5.1. Figure 5.1 (a) shows a 64 $\times$ 64 pixel image of line-drawing of a U-shaped object (shown in gray). The sequence of black curves in the image illustrates the iterative progression of a traditional snake (α = 0.6, β = 0.0). The snake is initialized outside the object but within the capture range of the potential force field. Figure 5.1 (b) illustrates the potential force field $\mathbf{F}_{ext} = -\nabla \mathbf{E}_{ext}^{(4)}$ where σ = 1.0 pixel in equation 5.5. Figure 5.1 (c) illustrates the reason for the poor convergence of the snake in form of a close-up of the potential force field. Although the potential forces correctly points towards the boundary of the object, but within the concave part of the object the forces point horizontally in opposite directions. Therefore the snake is pulled apart and does not progress downward into the concavity. No choice of α and β will correct this problem.

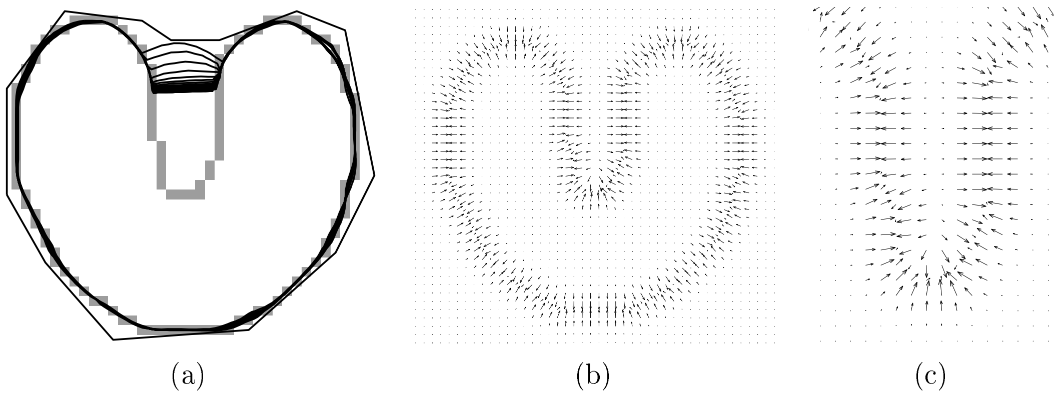


Figure 5.1: Figure (a) illustrates the convergence of a snake using traditional potential forces (α = 0.6, β = 0.0)(figure 1 (a) Xu and Prince (1998)). Figure (b) illustrates the traditional potential forces (figure 1 (b) Xu and Prince (1998)). Figure (c) illustrates an close-up of the potential forces within the concave part of the U-shaped contour (figure 1 (c) Xu and Prince (1998))

5.2.2 Gradient Vector Flow

The problems of initialization and convergence into an U-shape concave boundary regions can to a certain extend be solved using a new external force. The new force field is called the *gradient vector flow* and will replace the potential force field in the traditional snake framework.

The actual GVF field is defined as the vector field $\mathbf{v}(x,y) = (u(x,y), v(x,y))$ that minimizes the following energy function

$$\mathcal{E} = \int\int \mu(u_x^2 + u_y^2 + v_x^2 + v_y^2) + |\nabla f|^2 |\mathbf{v} - \nabla f|^2 \ dx\ dy \tag{5.9}$$

This variational formulation follows the standard regulation principle, that makes the result smooth when there is no data. When $|\nabla f|$ is small, the energy $\mathcal{E}$ is dominated by the sum of the squares of the partial derivatives of the vector field $\mathbf{v}(x,y)$. When $|\nabla f|$ is large, the energy $\mathcal{E}$

is dominated by the second term and minimized by setting $\mathbf{v} = \nabla f$. This produces the desired effect of keeping $\mathbf{v}$ nearly equal to the gradient of the edge map when it is large, but forcing the field to be slowly-varying in homogeneous regions. The parameter μ is a regularization parameter trading off the two different contributions to the energy $\mathcal{E}$. This parameter should be set according to the amount of noise present in the image (more noise, increase μ).

Using the *calculus of variations* (Courant and Hilbert, 1989), it can be shown that the GVF field can be found by solving the following Euler equations

$$\begin{aligned} \mu\nabla^2 u - (u - f_x)(f_x^2 + f_y^2) = 0 \\ \mu\nabla^2 v - (v - f_y)(f_x^2 + f_y^2) = 0 \end{aligned} \tag{5.10}$$

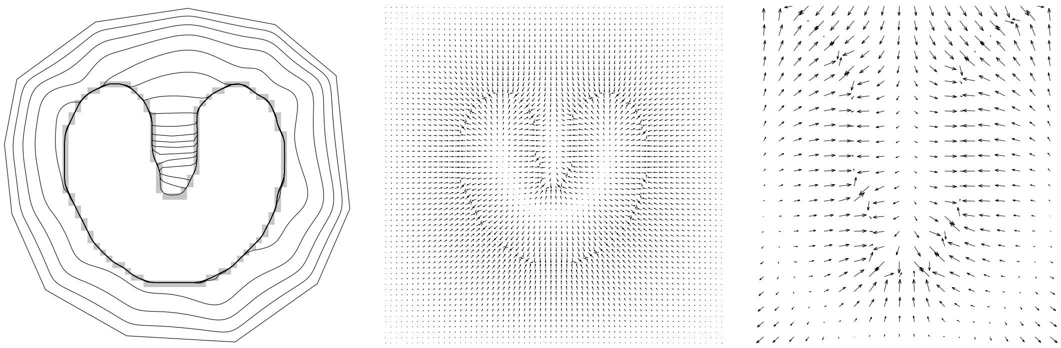
Where ∇^2 is the Laplacian operator. The equations 5.11 can be solved by making them dynamic and treating u and v as a function of time t in addition to space (x, y),

$$\begin{aligned} u_t(x,y,t) = \mu\nabla^2 u(x,y,t) - (u(x,y,t) - f_x(x,y))(f_x(x,y)^2 + f_y(x,y)^2) \\ v_t(x,y,t) = \mu\nabla^2 v(x,y,t) - (u(x,y,t) - f_y(x,y))(f_x(x,y)^2 + f_y(x,y)^2) \end{aligned} \tag{5.11}$$

The convergence of the iterative of these equations after discretization will yield the desired solution of the Euler equations. It is important to notice that these two dynamic equations are decoupled and can therefore be solved as separate equations in u and v. An implementation is available in Xu and Prince (1998).

Behavior of GVF Snakes

An example of the behavior of a GVF snake is shown in figure 5.2. Figure 5.2 (a) shows a 64 $\times$ 64 pixel image of line-drawing of a U-shaped object (shown in gray). The sequence of black curves in the image illustrates the iterative progression of a GVF snake ($\alpha = 0.6$, $\beta = 0.0$). The snake is initialized outside the object but within the capture range of the GVF force field. Figure 5.2 (b) illustrates the GVF force field with $\mu = 0.2$. Figure 5.2 (c) illustrates the reason for the convergence of the snake in form of a close-up of the GVF force field. The GVF forces correctly points towards the boundary of the object, and within the concave part of the object the forces point into the concavity. Therefore the snake progresses downward into the concavity.



(a) (b) (c)

Figure 5.2: Figure (a) illustrates the convergence of a snake using GVF external forces ($\alpha = 0.6$, $\beta = 0.0$) (figure 1 (a) Xu and Prince (1998)). Figure (b) illustrates the GVF external forces (figure 1 (b) Xu and Prince (1998)). Figure (c) illustrates an close-up of the potential forces within the concave part of the U-shaped contour (figure 1 (c) Xu and Prince (1998))

5.3 Results

This section contains the results of a series of test cases where GVF snakes are used for locating an edge in an image of a real recorded scene. The results in this section are made by Kurt Cornelis at VISICS, K.U. Leuven, Belgium.

The results shown i figure 5.3 illustrated the iterative process of converging the snake towards the desired edge in the image.

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

Figure 5.3: Figure (a) illustrates the initial position of a GVF snake next to a roof (figure 3, left, in Cornelis (2000)). Figure (b) illustrates the final position of the GVF snake from (a) after 5 iterations (figure 3, right, in Cornelis (2000)). Figure (c) illustrates the initial position of the GVF snake next to a depth discontinuity (Figure 4, left, in Cornelis (2000)). Figure (d) illustrates the final position of the GVF snake from (c) after 5 iterations (figure 4, right, in Cornelis (2000))

The following three cases illustrates the performance of the GVF snake in different real scenes. In order to visualize the result of the GVF snake, a virtual object (green box) is placed behind the desired edge in order to emphasize the reconstructed edge, which is the edge in question. The reconstruction of the scene is done using the software from VISICS at K. U. Leuven.

The parameters for all three test cases are as follows,

$$\alpha = 0.4$$
$$\beta = 0.05$$
$$\mu = 1$$
$$\textbf{iterations} = 5$$

The snakes are initialized using the depth maps of the reconstruction software from VISICS.

First Test Case

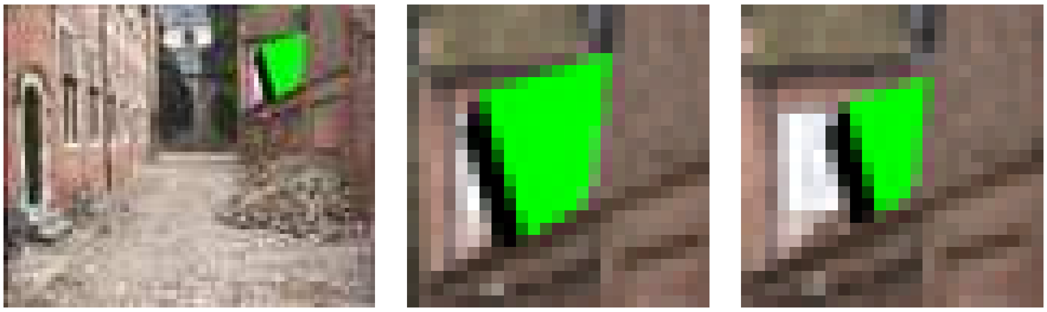
The first test case is shown in figure 5.4 (a). The desired edge is the boundary of the lantern in the right of the image. Figure 5.4 (b) shows an enlargement of the area around the lantern which illustrates that the reconstruction include some of the background as part of the foreground (lantern). The task of the GVF snake is in this case to converge towards the real boundary of the lantern. Figure 5.4 (c) illustrates the result of the GVF snake. In this particular case the snake did not bend enough which results in a lot of the small details on the lantern being cut away. There is a clear visible difference between the real edge of the lantern and the one cut by the GVF snake.

(a) (b) (c)

Figure 5.4: Figure (a) shows an image of the scene in question with an overlayed virtual object which is clipped to the depth map. Figure (b) shows an enlargement of the area around the lantern. The reconstruction system misclassifies parts of the background around the lantern as foreground. Figure (c) illustrates the result of clipping the virtual object to the result of the GVF snake after 5 iterations. It is clear that the GVF snake removes a lot of detail because it has problems with the combination of sudden change in direction and long straight distances.

Second Test Case

The second test case is shown in figure 5.5 (a). The desired edge is the boundary between the roof and the wall. Figure 5.5 (b) shows an enlargement of the area around the wall which illustrates that the reconstruction include some of the background as part of the foreground (the wall). The task of the GVF snake is in this case to converge towards the real boundary of the wall. Figure 5.5 (c) illustrates the result of the GVF snake. In this particular case the snake did a good job, but the desired edge did not contain a lot of twisting and turning which would have made it more difficult.



(a) (b) (c)

Figure 5.5: Figure (a) shows an image of the scene in question with an overlayed virtual object which is clipped to the depth map. Figure (b) shows an enlargement of the area around the wall. The reconstruction system misclassifies parts of the background around the wall as foreground. Figure (c) illustrates the result of clipping the virtual object to the result of the GVF snake after 5 iterations. It is clear that the GVF snake gives a better result in this case than in first test case. This is because the edge is almost straight.

Third Test Case

The third test case is shown in figure 5.6 (a). The desired edge is the boundary between the roof and the sky/trees. Figure 5.6 (b) shows an enlargement of the area around the roof which illustrates the resulting edge from the reconstruction. As it can be seen the edge is relatively accurate but jaggy. Figure 5.6 (c) illustrates the result of the GVF snake. In this particular case the snake did a good job along the roof top, but got the same problems at house end as in test case one. It cut away to much and a lot of details are lost.

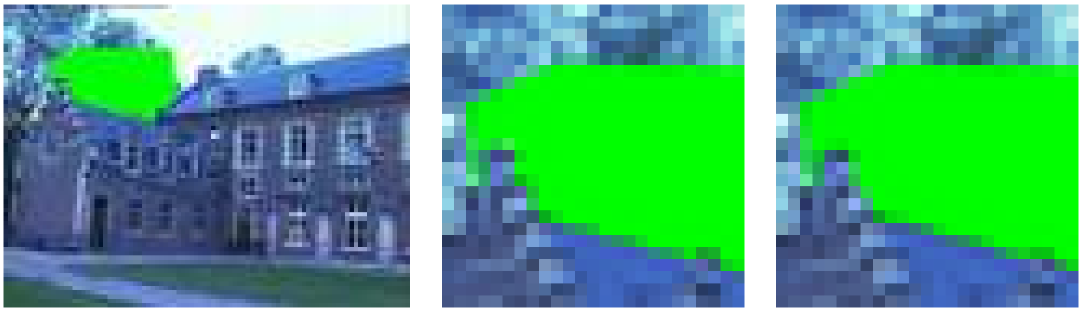


Figure 5.6: Figure (a) shows an image of the scene in question with an overlayed virtual object which is clipped to the depth map. Figure (b) shows an enlargement of the area around the roof. The reconstructed edge is relatively accurate but jaggy. Figure (c) illustrates the result of clipping the virtual object to the result of the GVF snake after 5 iterations. The GVF snake gives a better result along the roof top in this case than in first test case, but got the same problem at the house end as in the first test case.

5.4 Summary

The purpose of the snake is to converge towards a boundary in an image (high gradient, edge). In the case of reconstruction, there is a tendency to have inaccuracy around the edges which causes the edges from the depth map to be jaggy and not at the position of the real edge of the object in question. The goal of the snake is to converge towards the real edge of the object in question using information from the image.

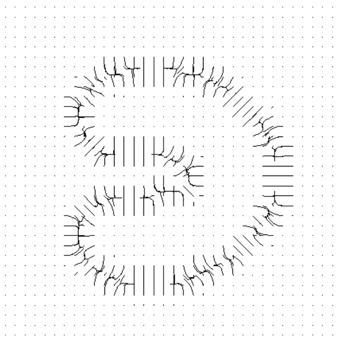
The snake works in two steps. They are as follows,

1. **Initialization** of the snake is especially important if there are multiple edges near by the desired edge. A good initialization of the snake is the inaccurate reconstruction of the edge that needs to be corrected. In this case, the edge is found by the depth values of the reconstruction from VISICS of the scene. Figure 5.7 shows how close the initial snake has to be in order to be able to converge to the desired edge. Figure 5.7 (a) illustrates for a traditional potential force field, and figure 5.7 (b) illustrates for a GVF force field.
2. **Convergence** of the snake towards the desired edge is done by iterating over the force field. As the snake is initialized near the desired edge only a few iterations are needed to converge to the desired edge.

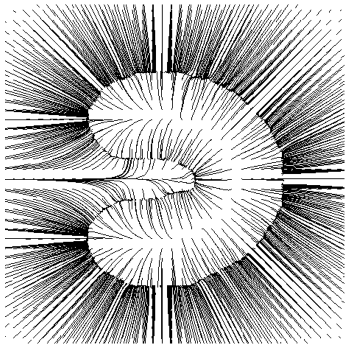
Based on figure 5.1 and figure 5.2 it is possible to conclude that GVF snakes is to prefer compared to traditional snakes. The problem of using snakes in especially an image of a real scene is to select parameters (α, β, and μ) which works in various scenes which includes everyday objects like houses, trees, people etc. Edges which consist of a lot of detail requires the snake to be flexible whereas straight edges requires the snake to be less flexible. The problem is to find a trade off which is robust to noise in the image and still preserves the contours of the edges.

To overcome the problems of selecting the parameters α, β, and μ, that works in the general case, it will help to divide the snake into a number of snakes along the edge. By splitting the snake into a number of local snakes, that covers a part of the edge with certain characteristics like straight lines, high amount of local details, etc., it is possible to select different parameters for each local snake that is optimal for that particular part of edge. One way of dividing the global snake into a number of local snakes is to select parameters that gives a less flexible snake. This will fit the snake to the straight parts of the edge. The second step is to run through the snake measuring how much the external force influences the snake at a given point. If there is a region with a large influence, it will result in a local snake for this region, with parameters that matches the requirement of the actual region.

Other methods might be developed in order to overcome the problems of selecting the parameters of the snake. But as this approach (using snakes) is all-ready tried by the people at VISICS with less convincing results, it will be more reasonable to try out a completely different approach.



(a)



(b)

Figure 5.7: Figure (a) illustrates how a free particle will move in the traditional potential force field from the case shown in figure 5.1. The streamlines indicates where and how the particle is influenced in the force field (figure 4 (a) Xu and Prince (1998)). Figure (b) illustrates the case of a GVF forces field in the case shown in Figure 5.1(figure 4 (c) Xu and Prince (1998)).

CHAPTER 6

Background Subtraction

6.1 Overview

Background subtraction is a technique to build up a model of the background from an image sequence. Each new image frame is then compared to the model, and based on this comparison, segmented into foreground and background pixels. A flowchart of this process can be seen in figure 6.1.

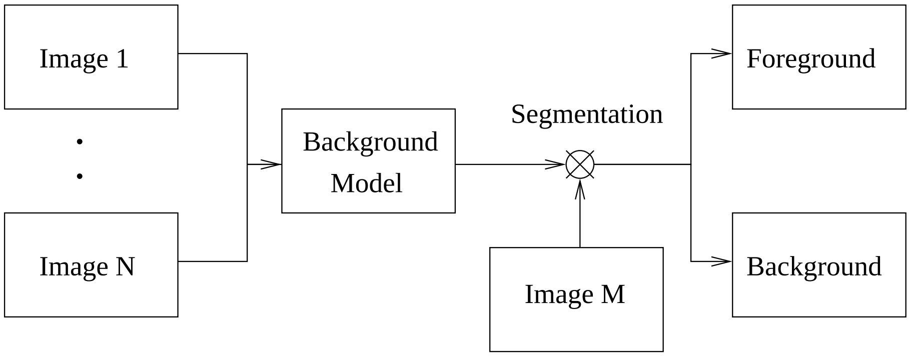


Figure 6.1: Flowchart of the background subtraction process with a fixed camera.

Background subtraction requires that the background is static, but does not put any constraints on motion of the foreground. It does however require that foreground objects have different color or luminosity than the background, I.E. there must not be any camouflage effects.

Background subtraction is often used in systems with a fixed camera, but can be used with a movable camera if an image warping step is performed on the images previous to the background model construction.

6.1.1 Image Warping

In order to create a model of the background in systems with a moving camera it is first necessary to transform the images into the viewpoint of a reference image using knowledge of the scene structure and camera parameters. Figure 6.2 illustrates the process.

When images are transformed from one viewpoint to another occlusion shadows (areas visible in

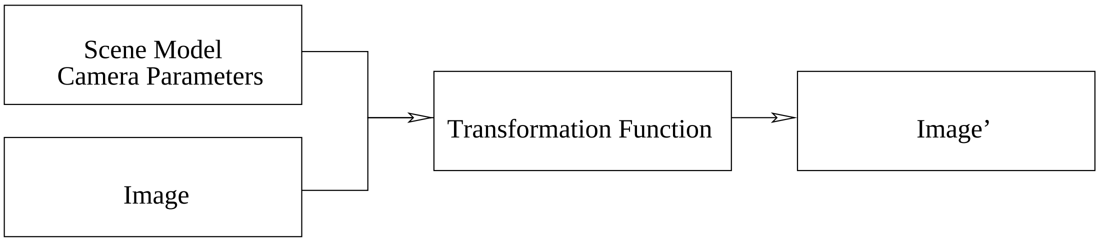


Figure 6.2: Flowchart of the Image Warping Process.

the reference image but not in one or more of the other views because of an occluding object) must be taken into consideration. Figure 6.3 illustrates the problem. An image warping algorithm can never reconstruct areas of the scene occluded by the cylinder given the two images.

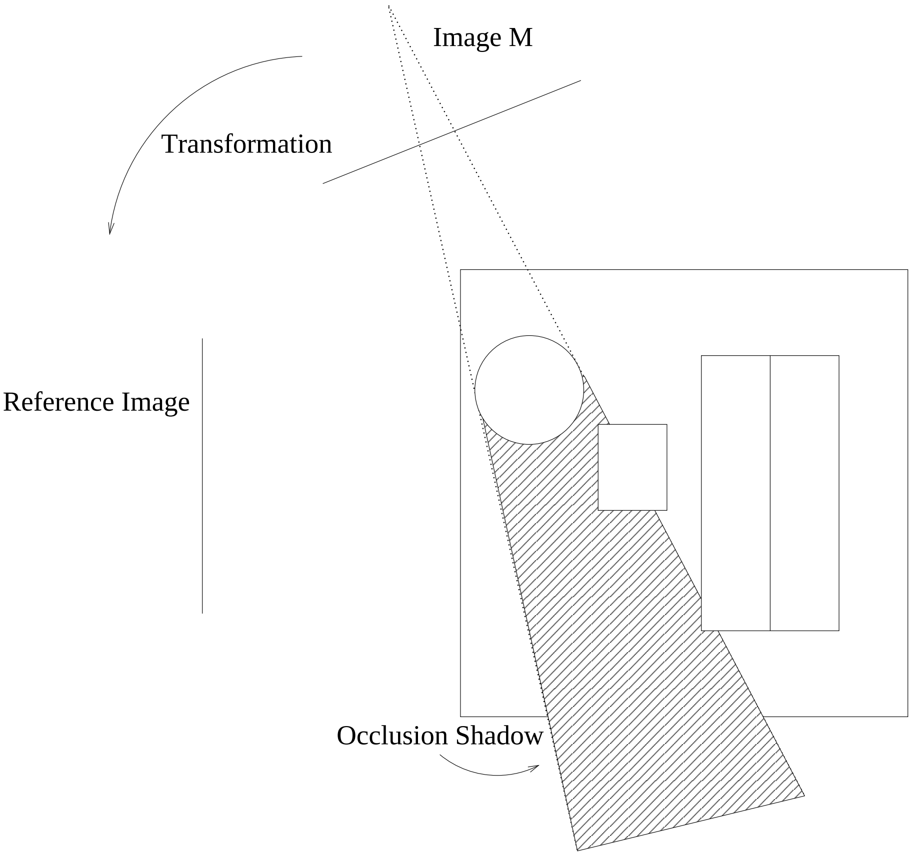


Figure 6.3: Occlusion shadows in image warping. The transformation function can never reconstruct the area within the occlusion shadow.

6.1.2 Background Model

Several algorithms exist for combining the (possibly warped) images into a model of the background ranging from simple averaging of pixel values to complex statistical models of each pixel (Harville et al. (2001)). A good background model must create a tight fit around the possible color values of the background pixel (caused by noise in the image, illumination changes, and so on) in order to keep the amount of misclassification to a minimum.

6.1.3 Segmentation

Finally each new image is compared to the model of the background, and the foreground is segmented from the background using a threshold on the difference between the model and the image. A threshold value which produces the minimum number of misclassified pixels must be determined. If it is set too low image noise and illumination variations will cause background pixels to be classified as foreground, and if it is set too high camouflage effects where foreground pixels are classified as background pixels will occur.

6.2 Approach

The reconstruction system uses a movable camera, and therefore an image warping step is needed. In addition to the image sequence a 3D model of the scene and camera parameters for each image are available (See chapter 2 for more information).

6.2.1 Image Warping

The approach used for image warping is to back-project pixels p_{ref} of the reference image into the 3D model of the scene to get the corresponding 3D points P_{model}. These 3D points are then projected into each of the other views in order to determine which pixel p_N correspond to this 3D point (and hence, pixel in the reference image), hereby creating a transformation between pixels in the reference image and pixels in any of the other views. Figure 6.4 illustrates the approach.

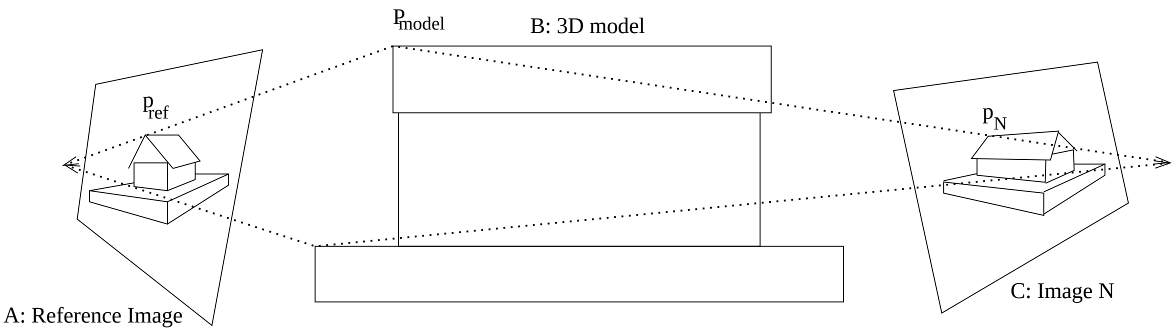


Figure 6.4: The 3 steps of the image warping process. A: Pixels in the reference image. B: Corresponding 3D points in the scene model. C: Corresponding pixels in another views.

Back-projection consists of finding the nearest point in the 3D model which intersects the eye ray which goes from the optical center of the camera through a given pixel, but as the purpose of the image warping step is to only create an image of the background from the viewpoint of the reference image and not of the complete scene this intersection point must lie behind the virtual object as illustrated in figure 6.5. The 3D scene is clipped by a plane parallel to the image plane of the reference image and positioned directly behind the virtual object, and the clipped 3D scene is then used for back-projection.

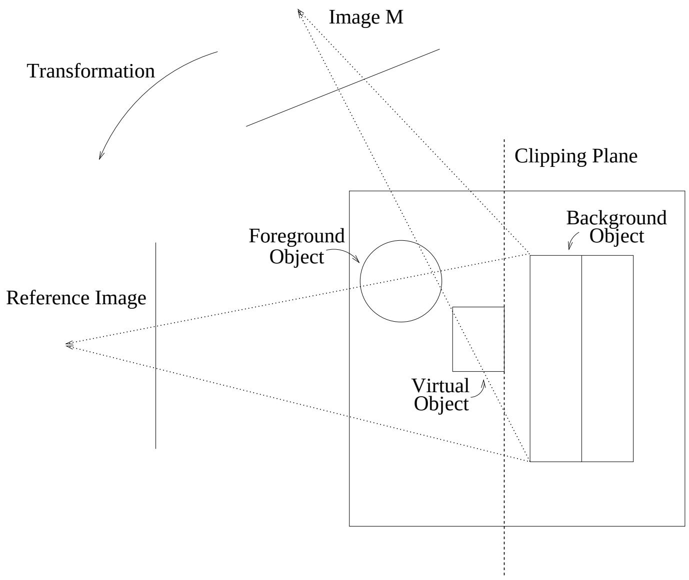


Figure 6.5: Only the part of the scene which lies behind the virtual object should be used for back-projection.

This approach effectively treats the virtual object as if it was flat and as if any 3D point can be classified as background or foreground simply by its position relative to this clipping plane. Figure 6.6 shows that this is generally not true, and it is therefore a source of error.

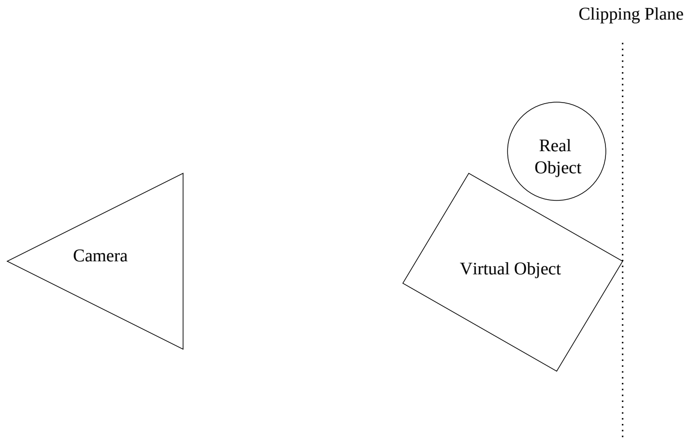


Figure 6.6: The clipping plane problem. Even though the real object is in front of the clipping plane it is occluded by the virtual object and should therefore be part of the background.

A bruteforce intersection test between the eye ray and all triangles of the 3D scene can be very slow as the 3D models consists of more than 100.000 triangles, but several optimizations are possible. A simple optimization is to use the depth sorting capabilities of the Z-buffer on a 3D graphics card by rendering a color coded version of the clipped 3D scene from the viewpoint of the reference camera. In this color coded image the color of a given pixel directly tells which triangle is nearest to the eye ray. After the triangle has been found it is only needed to calculate the intersection point between the triangle and eye ray instead of between the eye ray and all triangles. This algorithm is described in more detail in (Foley, van Dam, Feiner, and Hughes, 1995, p. 668).

Occlusion shadows are handled in a similar way as the back-projection: The complete scene is rendered using the same color coding but from the point of view of each of the other views. If the color (and hence triangle visible) at the pixel p_N which the 3D point P_{model} projects into is not the same as the color of the triangle found by back-projection, then this pixel is in an occlusion shadow in this view (see 6.4). Unfortunately this algorithm breaks down when the projected area of a triangle becomes smaller than one pixel as several triangles will “fight” about the color of a pixel. This can be solved by using a connectivity structure and also classify a triangle as visible if the pixel is of the color of any of its neighboring triangles. Unfortunately this approach can lead to false positives, where a neighboring triangle is visible even though the triangle is in an occlusion shadow. Figure 6.7 illustrates the problem. These false positives can be removed using backface culling on the center triangle (marking the triangle as being in an occlusion shadow if the normal of the triangle faces away from the camera).

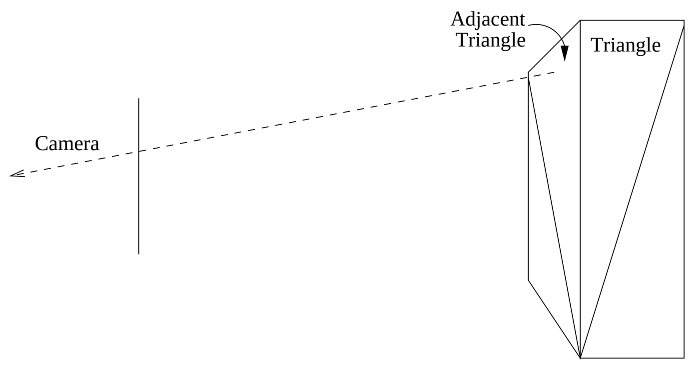


Figure 6.7: Including adjacent triangles can lead to false positives as an adjacent triangle can be visible even though the center triangle is not.

As a pixel p_{ref} in the reference image most likely does not exactly transform into a single pixel p_N in any of the other images sub-sampling of the image values in the other views using bilinear interpolation is used to determine the color. When sub-sampling is used to give a more accurate estimate of the color value of another viewpoint it is important to only combine color values of pixels which correspond to 3D points which are close to each other, otherwise visual artifacts can occur. Figure 6.8 illustrates the problem. Pixel a and b with the corresponding 3D points A and B are close to each other, but not to C or D. Seen from another viewpoint C and D would not lie on the same line of sight as A and B and should therefore not be used for sub-sampling.

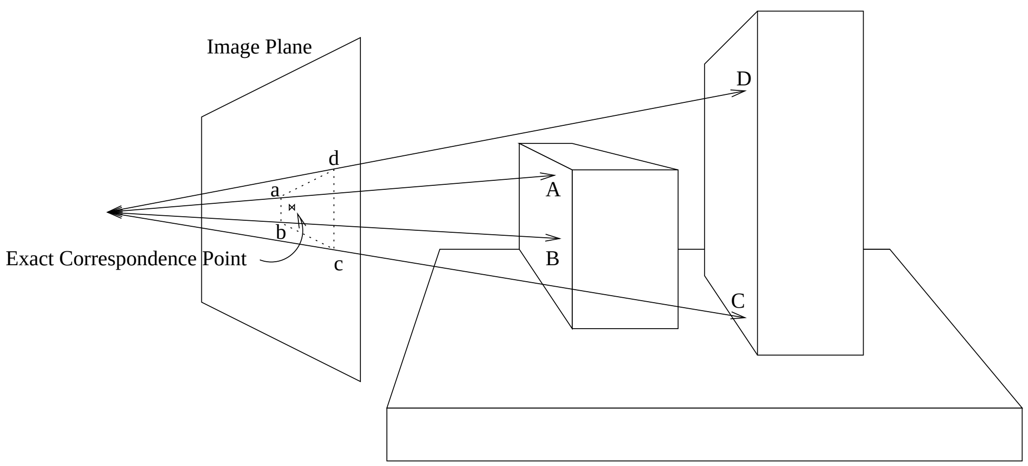


Figure 6.8: Bilinear filtering should only combine pixels which correspond to 3D points which are close to each other.

We define two 3D points to be close to each other if they lie on adjacent triangles which can quickly be determined from the color coded rendering and the connectivity structure.

6.2.2 Background Model

The following background model algorithms have been implemented:

- Average color per pixel
- Median color per pixel

Both algorithms produce a single estimate of the background value per pixel and are therefore not able to model local view dependent phenomena as e.g. highlights. If highly textured scenes are to be expected (combined with an inaccurate 3D model) then a more advanced background model which can model several background values per pixel may be needed.

The average background model takes all pixels into consideration, whereas the median model is less sensitive to outliers as long as the amount of outliers are smaller than the amount of inliers.

6.2.3 Segmentation

The last step of the background subtraction process is segmentation on the difference between the background model and the input image.

The segmentation process can be performed as a global threshold, or incorporating local information from the background model.

Examples of local information are:

- Color variance
- Number of views contributing to background model

If the confidence on the background model is very low (point only visible in a few views) then depth information from the 3D model can be used to classify a pixel into foreground or background. This is problematic though as the depth map is known to contain errors.

6.2.4 Optimization

The entire background subtraction process does not need to be performed over the whole image, but only on the pixels which the virtual object projects into. Depending on the size of the virtual object this can have a high impact on the performance of the algorithm.

6.3 Results

In order to test the background subtraction algorithm a sequence of 6 synthetic images containing colored boxes has been constructed. The sequence can be seen in figure 6.9.

The task is to insert a virtual object between the red box and the gray box in image 4 using the information from the other 5 views and the 3D model of the scene. A rendering of the virtual object from the same viewpoint as image 4 can be seen in figure 6.10.

6.3.1 Image Warping

First a clipping plane parallel to the image plane is inserted directly behind the virtual object. Figure 6.11 shows a rendering of the 6 views with the clipping plane inserted.

After the clipping plane has been inserted the pixels of view 4 in figure 6.11 are backprojected and the pixels of the remaining 5 views are warped to the reference image. The result of the

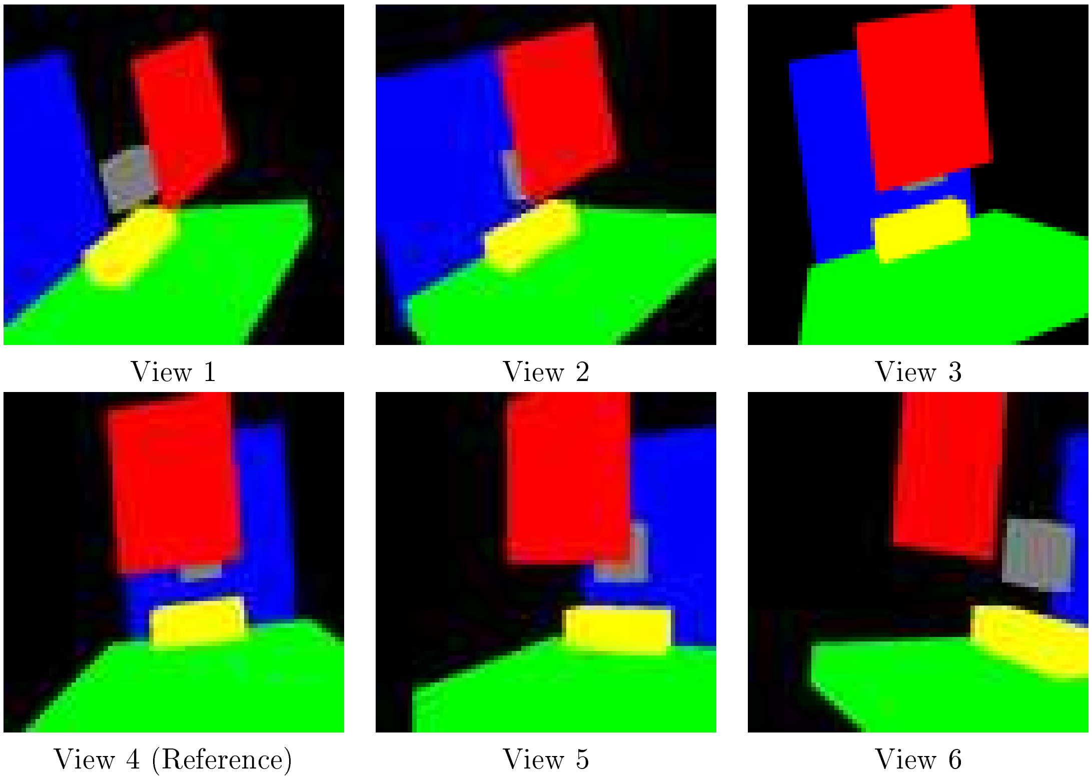


Figure 6.9: The synthetic test sequence. Image 4 is the reference image.

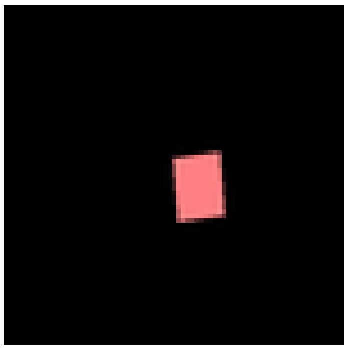


Figure 6.10: The virtual object to be inserted in the test sequence rendered from view 4.

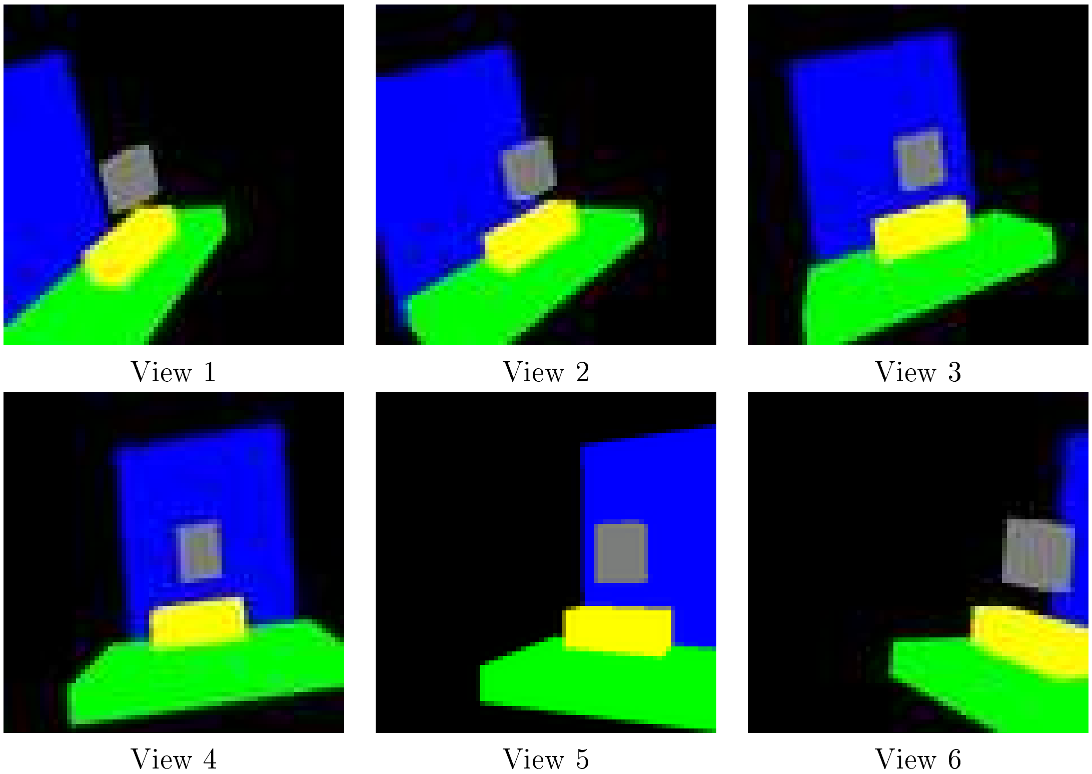


Figure 6.11: The synthetic test sequence with a clipping plane inserted directly behind the virtual object.

image warping process for the 5 views can be seen in figure 6.12. Areas in occlusion shadows are shown as white.

6.3.2 Background Model

The warped images can now be merged into a background model. Figure 6.13 illustrates the two background models. The bottom row contains the models with visibility encoded in the alpha channel (The more views a pixel is visible in, the closer the color is to the one in the top row, the less views it is visible in, the more white it is).

6.3.3 Segmentation

As a perfect 3D model and simple geometry has been used, and no change in lighting conditions have occurred the two background models generate the same model. Figure 6.14 shows a difference image between the reference image and the median background model, a histogram of the difference image and the foreground mask extrated from segmenting the difference image using a global threshold value of 1/4 of the range.

Clipping the virtual object of figure 6.10 with the foreground mask of figure 6.14 gives the final augmented image. This image can be seen in figure 6.15.

6.3.4 Camouflage

In order to illustrate the problems with camouflage the gray box in the background is replaced with a red box of the exact same color as the foreground box. The 6 images of the new sequence can be seen in figure 6.16.

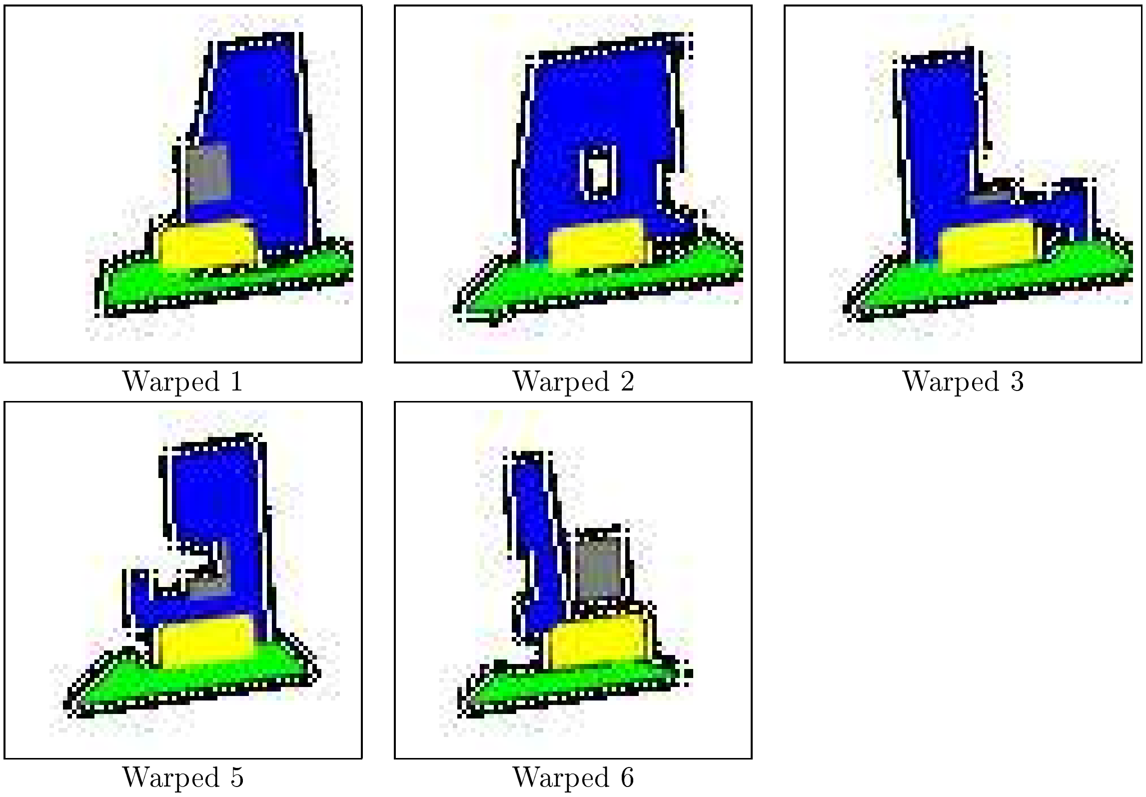


Figure 6.12: The warped sequence. The white parts are areas within occlusion shadows.

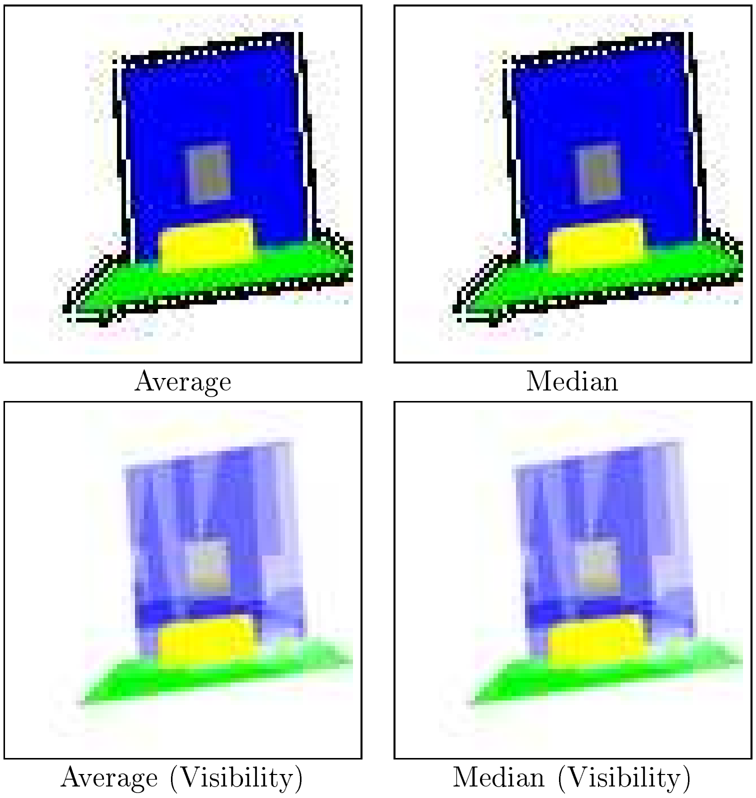
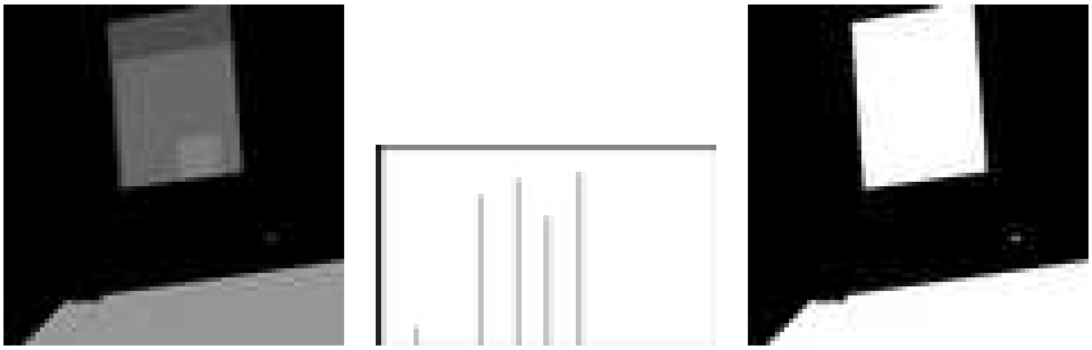


Figure 6.13: The two background models. The bottom row contains the two models with visibility encoded in the alpha channel.



Difference Image | Histogram | Foreground Mask

Figure 6.14: Difference image between reference image and median background model, histogram of difference image and foreground mask from using a global threshold of $1/4$ of the range. Notice the dot in the lower right corner from the area which is in an occlusion shadow in all 5 views.

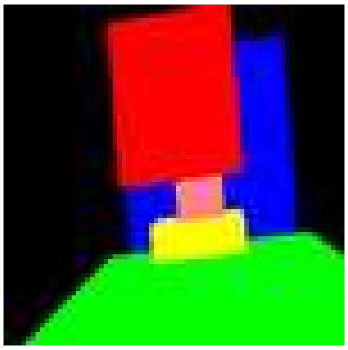
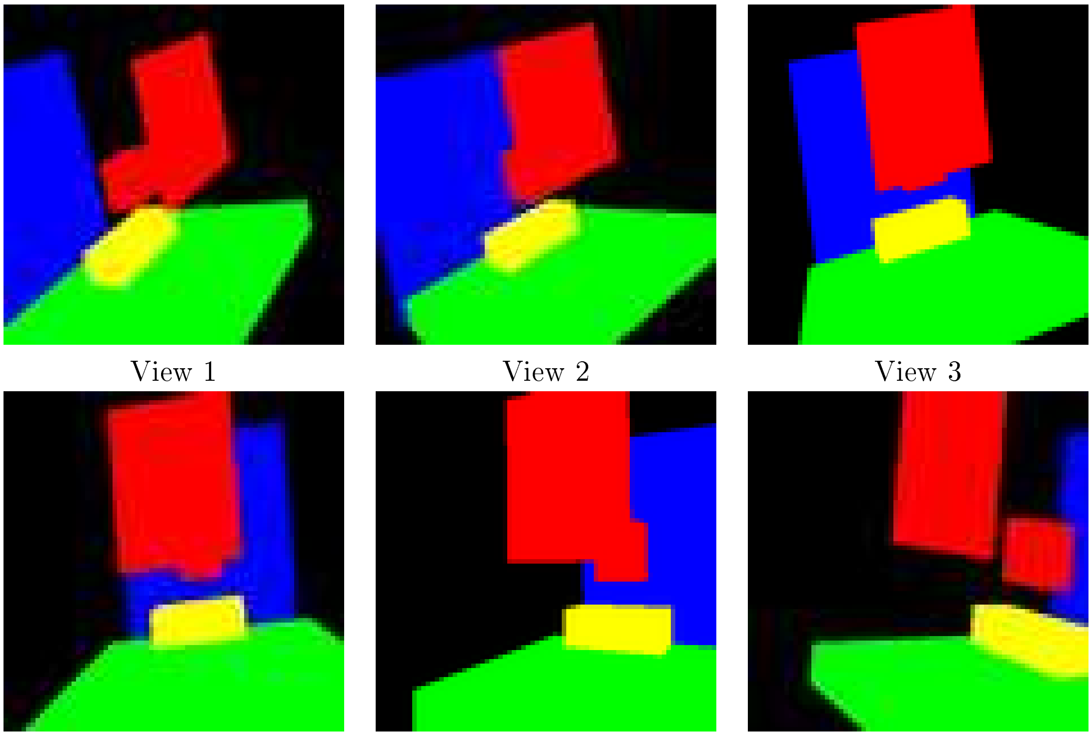


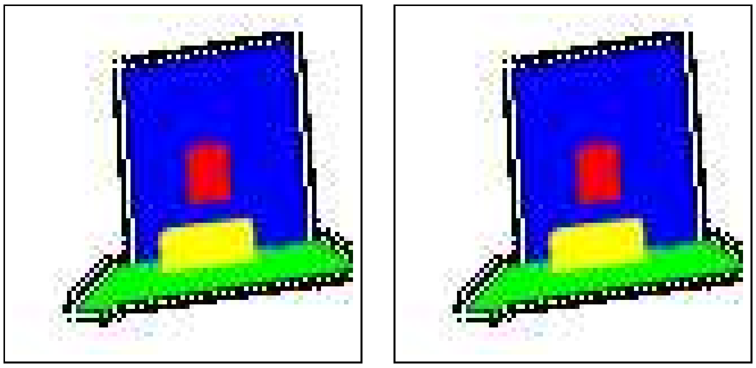
Figure 6.15: Final augmented image produced by clipping the virtual object of figure 6.10 with the foreground mask of figure 6.14. No visible artifacts exist in the image.



View 4 (Reference) | View 5 | View 6

Figure 6.16: The camouflage test sequence. Image 4 is the reference image.

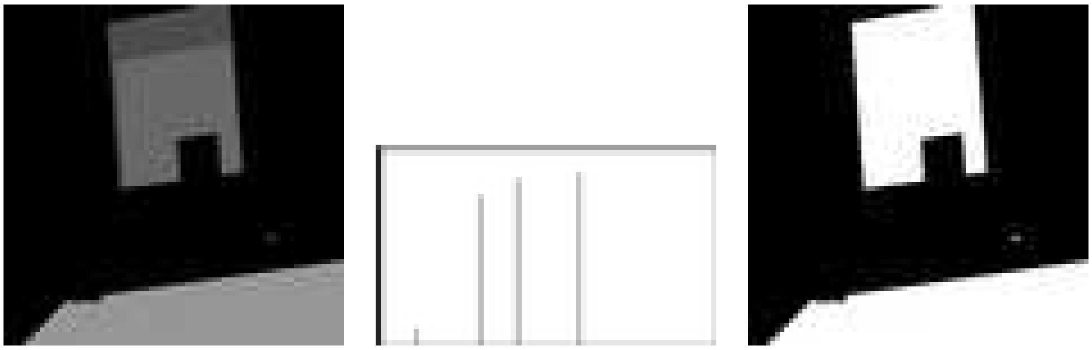
The result of performing image warping and background subtraction on this sequence can be seen in figure 6.17. Again there is no difference in the two models.



Average Median

Figure 6.17: The two background models for the camouflage sequence.

Figure 6.18 shows a difference image between the reference image and the median background model, a histogram of the difference image and the foreground mask extrated from segmenting the difference image using a global threshold value of 1/4 of the range for the camouflage sequence. The camouflage effect is clearly visible in the center of the difference image.



Difference Image Histogram Foreground Mask

Figure 6.18: Difference image between reference image and median background model, histogram of difference image and foreground mask from using a global threshold of 1/4 of the range for the camouflage sequence. Notice the hole in the center caused by camouflage effect.

Clipping the virtual object of figure 6.10 with the foreground mask of figure 6.18 gives the final augmented image. This image can be seen in figure 6.19.

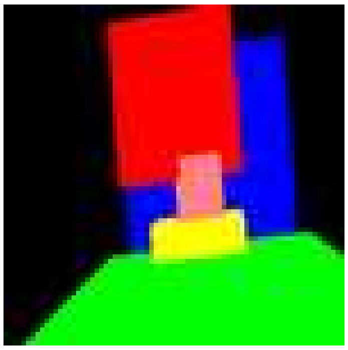


Figure 6.19: Final augmented image of camouflage sequence produced by clipping the virtual object of figure 6.10 with the foreground mask of figure 6.18. Artifacts caused by camouflage effects are clearly visible in the center of the image.

6.4 Summary

The background subtraction algorithm has been presented and adapted to the project. The use of a clipping plane to divide between foreground and background leads to problems if virtual objects are placed close to real objects. The background subtraction process creates a good estimate of the foreground mask if no camouflage effects occur, but further refinement of the silhouette is required to handle camouflage effects.

CHAPTER 7

Visual Hull

7.1 Overview

This chapter introduces the approach used for eliminating the problems added to the silhouette by the background subtraction due to camouflage effects. The methods used in this process are specially designed for this project as no other method was found that eliminated problems caused by camouflage for this type of application.

The task of the visual hull process in this project is to refine the edges of the silhouettes and remove holes in the silhouettes caused by camouflage effect. The silhouette (occlusion mask) describes the foreground objects which is defined as the 3D geometry in front of the virtual objects and is generated using background subtraction (see chapter 6).

The camouflage effect is introduced when there are identical colors in the foreground object and background model along a ray cast from the view for which the background subtraction is performed. As the silhouette, generated using background subtraction, depends on the colors of the background model it possible to have a more or less complete silhouettes of the occluding objects in different views. The idea is to add missing structure to less complete silhouettes by combining the information from silhouettes of all the views.

Using multiple silhouettes describing the same object from different angles results in any camouflage holes present in the silhouettes (holes caused by camouflage effect) occuring in different areas of the silhouettes (see figure 7.1). It is then possible to combine information of multiple silhouettes into one description of the object without/less holes caused by camouflage using visual hull with post processing.

The remaining holes in the visual hull representation are either a real hole or camouflage hole that has not been filled by the other silhouettes. Each hole/region is the determined to be either real hole, or camouflage hole and then filled if necessary.

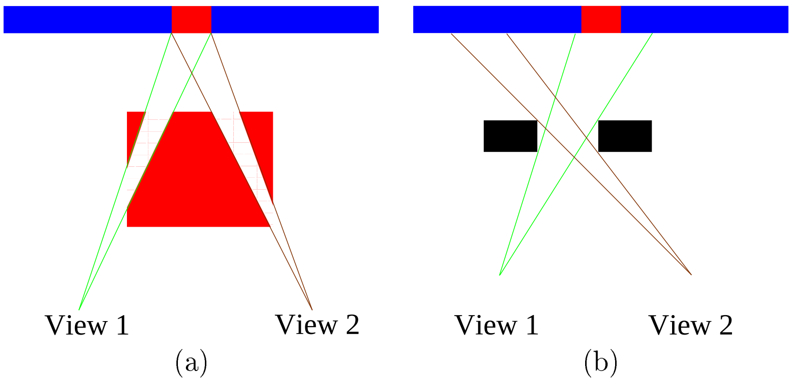


Figure 7.1: Figure (a) illustrates how holes caused by camouflage develop throughout the silhouettes. In view one the hole is in the left side of the silhouette whereas it is placed in the right side of the silhouette in view two. Figure (b) illustrates how real holes develop throughout the silhouettes. Both in view one and two the hole is in the part of (middle) the silhouette

7.2 Method

A visual hull representation of an 3D object is constructed using the visible silhouette information of the object in the reference images. It uses the silhouettes from a series of reference images to carve away regions in 3D space where the object is not present. Figure 7.2 illustrates this carving away of regions in 3D space.

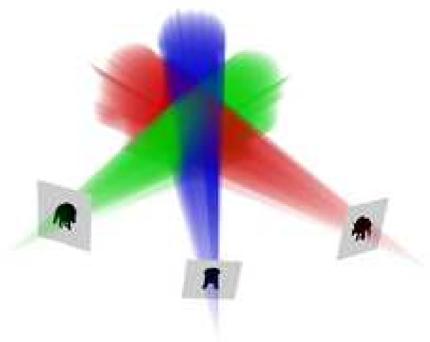


Figure 7.2: The intersection of silhouette cones defines an approximated geometric representation of an object called the visual hull (Wojciech et al., 2000, Figure 1)

Figure 7.3 shows a flowchart of the visual hull with post processing.

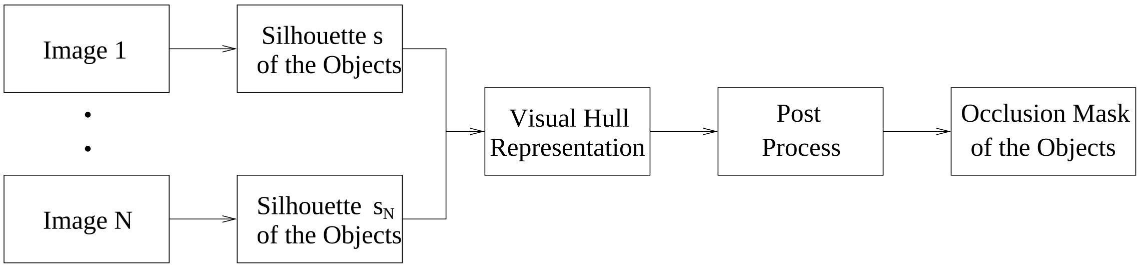


Figure 7.3: Flowchart of the visual hull process with post processing.

Take a 3D object and capture it as a set of reference views R (image{1..N} in figure 7.3). Each reference view r has the silhouette s_r which pixels covers the object. For view r create a cone-line volume vh_r defined by all the rays starting at the views optical center C_r and passing through the pixels p_r that covers the object. The object must be contained in the volume $vh_R = \cap_{r \in R}$. As the size of R goes to infinity, and includes all possible views, vh_R converges to a shape known as the visual hull vh_∞ of the original geometry, but this is only true for object that does not contain concave surface regions and the silhouettes of the object are exact.

Computing Image-Based Visual Hull

The algorithm used for computing the image-based visual hull is based on Wojciech et al. (2000). This description of the algorithm is to give an overview of the different steps involved. The algorithm has two distinct characteristics: it is computed in the image space of the reference images and the resulting representation is viewpoint dependent. The details on the implementation is described in section 7.3.1.

Computing the image-based visual hull involves three steps. These steps are illustrated in figure 7.4. The following steps are:

For every pixel in the desired image do:

1) A 3D ray is cast from the optical center of the desired view (resulting view) passing through a pixel in the image plane. The ray is projected into a reference image

2) Along the projected ray the intersections with the silhouette in the reference images are registered. These intersections result in a list of pixel intervals along the ray which corresponds to the cone's cross-section

3) Each interval is then lifted back into the 3D using a projective mapping, and then intersected with the result of the ray-cone intersections from the other reference images

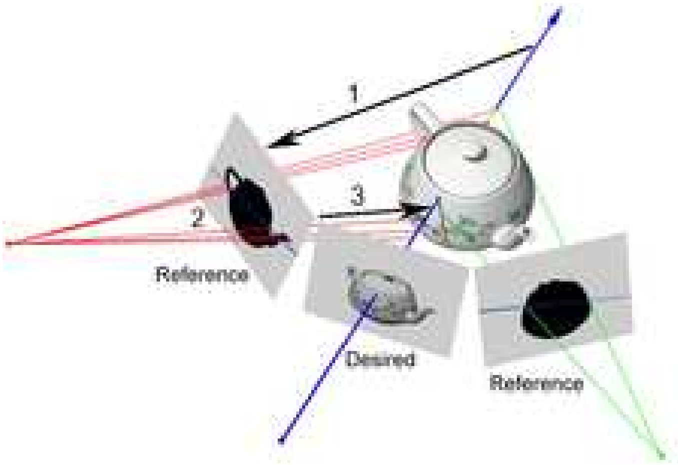


Figure 7.4: Computing the *image-based visual hull* involves three steps. First, the desired ray is projected onto the reference image. Next, the intervals where the projected ray intersects the silhouette are determined. Finally, the intervals are lifted back onto the the desired ray where they can be intersected with intervals from other reference images.

An algorithm for computing these image-based ray intersections is as follows:

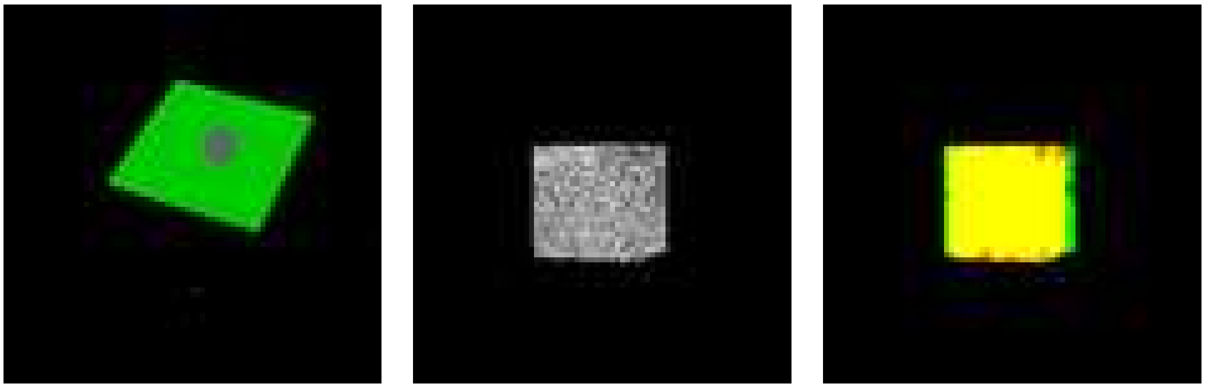
```
For each reference image r in R
    compute silhouette edges of r
For each pixel in desired image d do
    p.intervals = {0, inf }
For each reference image r in R
    For each scanline s in d
        For each pixel p in s
            ray3D ray3 = compute3Dray(p, p.caminfo)
            linesegment2D l2 = project3Dray(ray3, r.caminfo)
            intervals int2D = calcIntervals(l2, r.SilhoutteEdge)
            intervals int3D = liftIntevals(int2D, ray3, r.caminfo)
            p.intervals = p.intercals ISECT int3D
```

Examples of the Image-Based Visual Hull

The algorithm shown above is tested on two different cases: The first case is without camouflage effect and based on hand segmented silhouettes of the 3D structure, and the second case is with camouflage effect. The result of the second case is explained from the result of the first case

The first case without camouflage effects consists of six views. Five of the views are used as reference views (the blue spheres in figure 7.5 (a)) and the desired view for which the rays are cast (the red sphere in figure 7.5 (a)). The synthetic scene used for testing the algorithm is simple and contains a box placed on a flat surface.

Figure 7.5 (a) illustrates the scene used for testing, whereas (b) and (c) shows the result in the desired view, and figure 7.6 shows all five silhouettes of the reference views used for computing the image-based visual hull and ground truth of the desired view.



(a) (b) (c)

Figure 7.5: Figure (a) illustrates the synthetics scene from an elevated position. The sphere with the red color is the view from which the rays are cast (desired view). The spheres with the blue color are views used for carving away regions in 3D space where the object is not present. Figure (b) illustrates the result of the visual hull algorithm (desired view). Figure (C) illustrates the differences between the result of the visual hull algorithm and a hand segmented silhouette of the red view (ground truth).

The result in figure 7.5 (c) is to be interpreted in the following way: The yellow area is where there is consistency between the visual hull representation and ground truth. The red area is where ground truth indicates 3D structure and the visual hull does not. The green area is where the visual hull indicates 3D structure and ground truth does not.

The algorithm causes the description of the 3D object (the gray box) in the desired view to be wider than than the ground truth. The reason is the lack of views to carve away regions in 3D

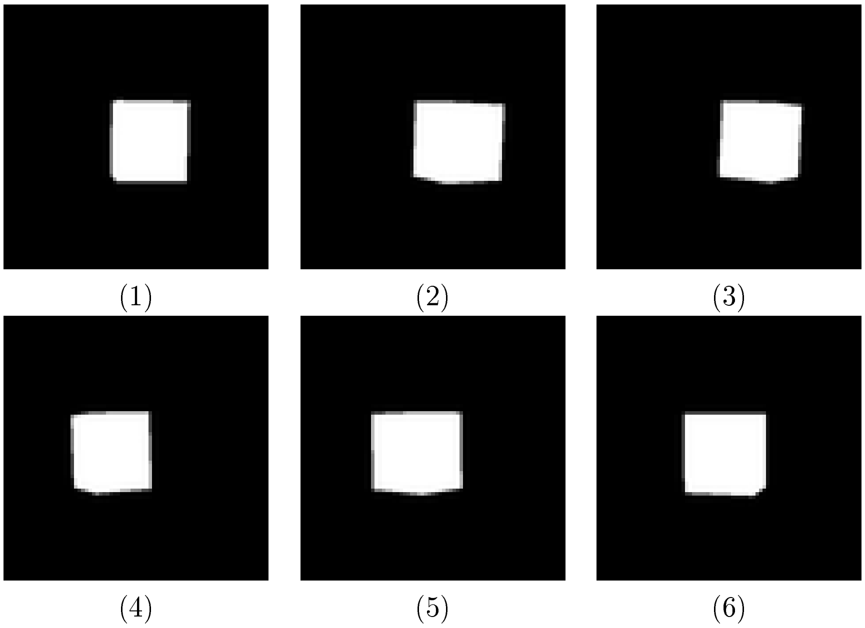


Figure 7.6: Figure (1) to (6) shows the ground truth silhouettes of the views in figure 7.5 (a) segmented by hand. Figure (1) is the view (blue sphere) to the left of the desired view (red sphere). Figure (2) is the next view (blue sphere) to the left. The rest are counted clockwise until figure (6), that represent the desired view (red sphere).

space where the object is not present. This phenomenon is illustrated in figure 7.7. The area inside the white lines is the region in 3D space that is not carved away by the reference views. Figure 7.7 (b), which is an enlargement of figure 7.7 (a), illustrates that it is possible to cast a ray that is inside the white area and at the same time outside the gray box. The rays cast in this area causes the errors in the visual hull representation of the gray box shown as the green color in figure 7.5 (c).

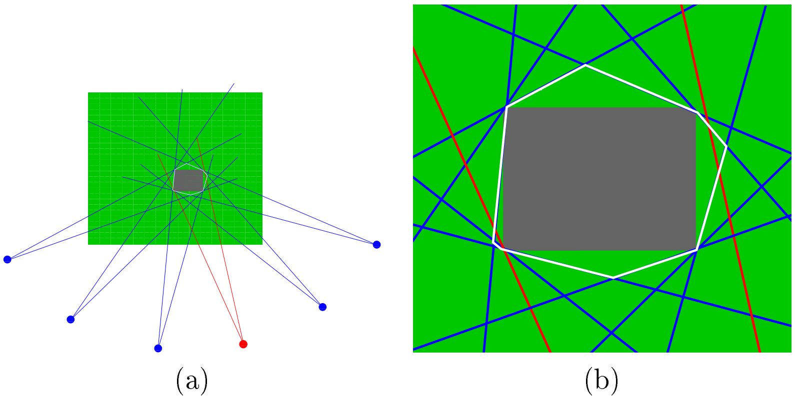


Figure 7.7: Figure (a) illustrates the synthetics scene from an elevated position. The blue lines illustrates where each reference view carves away 3D space. The result of this carving is the white area that based on the current views gives an representation of the 3D object. The gray area is the actual 3D object. Figure (b) illustrates an enlargement of figure (a). It illustrates the possibility of casting a ray outside the gray area and at the same time inside the white area. These cases results in errors illustrated in figure 7.5 (c) as the green area.

The image-based algorithm has some disadvantages: It closes all holes inside the object. This is both real holes and holes caused by camouflage (test case two). An other disadvantages is that it requires a high number of views and does not use the silhouette of the desired view to carve away 3D space where the object is not present.

7.3 New Approach

In order to overcome the problems caused by camouflage effect a new approach has been developed.

This approach uses the silhouette of all views involved and the result of this new approach is a common visual hull representation which is generated by computing a visual hull representation from all views against all other views and then combined.

The algorithm for all views against all other views has some advantages: The common visual hull representation got all the information of all the silhouettes involved with the constraint that it has to be represented in other views. Using the approach of casting rays from all views against all other views makes the algorithm more robust towards camouflage effects. The reason is that the background subtraction uses different parts of the background to generate the silhouettes, which might results in a silhouette with less or changed pattern of the camouflage effect.

Disadvantages of the new approach is that it is slower as it calculates the visual hull representation from all views instead of only from the desired view. It only sends rays from pixels inside the silhouettes and not all pixels of the image as the original visual hull algorithm though.

The common visual hull representation might still have some holes that are not eliminated by this process. The unresolved holes are either caused by camouflage or are real holes in the object.

The second refinement determines if a hole is real or caused by camouflage. Again the silhouettes of all the views are used, but this time is it the refined silhouettes from the first step that are used. The approach used for determine what causes the hole is based on the pattern of holes in the different views, and how they interact with each other. The approach is to see how the hole in question evolves if it is observed from other angles. It is the same approach humans use to determine if there is a hole in an object or not (without touching the object and using a higher level of knowledge).

An example of this interaction is shown in figure 7.8. The hole where view one casts a ray through the center (red solid line) is investigated for causes. If the appearance of the hole changes when observed from an other angle (view) then it is possible to say something about the cause of the hole.

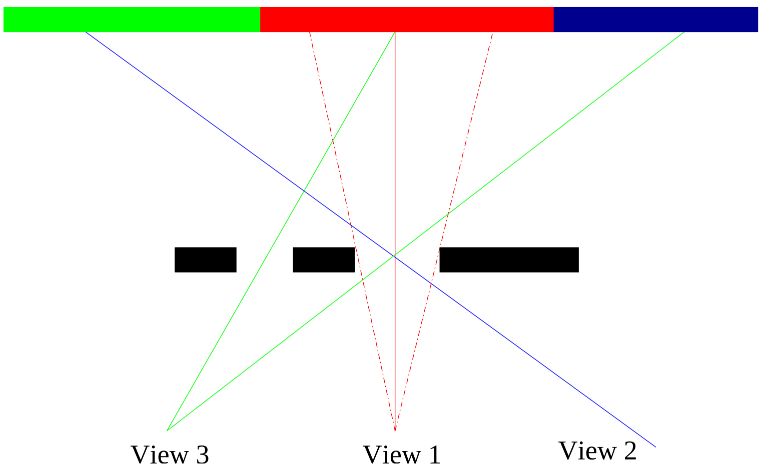


Figure 7.8: The figure illustrates the interaction between a hole in the different view, and the appearance of the hole based on what can be "seen" through the holes, in this case the colors of the background. If the color of the hole changes with the angle from which the hole is observed means it is a real hole

Table 7.1 list the two steps of the refinement of the silhouettes,

Step 1: Compute the visual hull representation using the new approach algorithm

1. Convert the silhouettes generated using background subtraction into lines describing the contour
2. Determine the pixels for which the rays are cast
3. Compute the visual hull and visual hull surface representation

   The following procedure is performed for each pixel determined in step 1.2 for each view

   (a) Cast an infinite ray through the determined pixel

   (b) Compute the 2D intersection between the epipolar line (projection of the cast ray) and the lines describing the contour of the silhouette. This is done for all the views in the sequence

   (c) Lift the intersections back to the cast ray

   (d) Limit the cast ray based on the intersections found in step 1. 3c so as little as possible is added to the visual hull representation

   (e) Project the limited ray into each of the views in the sequence

Step 2: Post processing of the visual hull representation by determine if holes in the visual hull representation are real or camouflage holes

1. Locate the boundary of object based on the visual hull surface representation using convex hull
2. Locate the different closed regions inside the boundary found in step 2.1
3. Initial determination of the type of regions (real or camouflage hole)
4. Compute the color histogram of each region
5. Determine if a region is caused by camouflage or a real hole in the object

   (a) Cast a ray through the center of mass of the region in question

   (b) Compute 3D intersection between the cast ray and rays cast from all the other regions of all the other views

   (c) Compare color histograms between regions with a 3D intersection at the same 3D point

   (d) Determine if the region is caused by camouflage based on the 3D intersection points along the ray and the similarity of the colors inside the regions

Table 7.1: Overview of the two-step refinement of the silhouettes.

7.3.1 The Silhouettes used for the Visual Hull Representation

The silhouettes of the objects are generated using background subtraction (see section 6). The problem is that the silhouettes generated using background subtraction are not robust toward the camouflage effect. This results in a silhouette of the foreground object that includes as few pixels belonging to the background as possible but it might also introduce extra holes to the silhouettes.

A silhouette is forced to have an equal number of entry and exit points along a straight line through the silhouette. This is done by using the edge of the image to clip the silhouette. This approach is reasonable as there is no absolute knowledge about the shape of the silhouette outside of the image.

The silhouettes are converted into a line description of the contour as the method uses 2D intersections between the epipolar lines and the contour of the object in order to find entry- and exit point, see section 7.2.

The conversion into lines is done using chain codes (section 8.1.1 in (Gonzalez and Woods, 1993)) that are cut into as few straight lines as possible to decrease computation time.

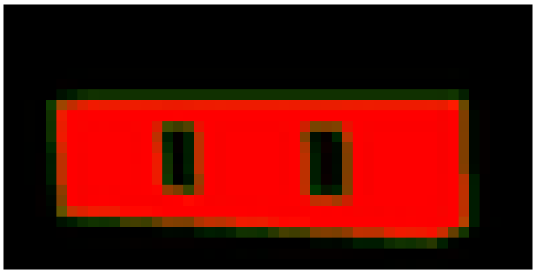
7.3.2 Pixels Inside the Silhouette

In contrast to the image-based algorithm the rays are only cast inside the silhouettes, whereas they are cast in the complete image for the image-based algorithm. To increase the robustness toward noise in the image and estimation of camera parameters the area for which the rays are cast is eroded. It is not possible to use a regular kernel (section 8.4 in (Gonzalez and Woods, 1993)) to erode the silhouettes with, as thin structures might be erased. It is important to keep all of the contour of the object as it otherwise will be excluded from the visual hull representation.

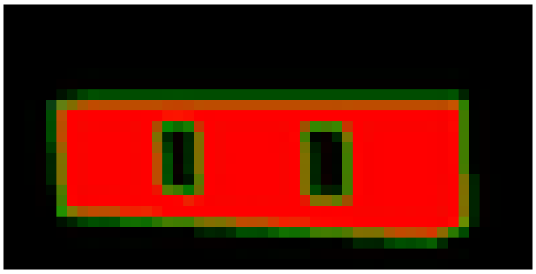
The algorithm used for eroding the contour without removing any contour is a two pass algorithm and is listed in table 7.2

Example of this Erosion Algorithm

Figure 7.9 illustrates an example of the erosion algorithm. Especially pay attention to the upper left corner of the silhouette. The branch is preserved and connected to the silhouette after the second erosion.



(a)



(b)

Figure 7.9: Figure (a) illustrates the result of a single erosion of a silhouette. The green area illustrates the part of the silhouette that has been removed. Figure (b) illustrates the result of a double erosion (two erosion after each other) of a silhouette. Note that the little branch in the upper left corner of the silhouette is preserved after the second erosion.

7.3.3 Compute Visual Hull Representation

Computing the visual hull representation is based on the image-based algorithm described in section 7.2.

First pass: divides the pixels into three categories: 1) Pixels that are totally surrounded by other structure like a regular 3×3 erosion kernel (section 8.4 in (Gonzalez and Woods, 1993)). The pixels in this group are included in the eroded structure. 2) Pixels that are partly surrounded by other structure. Pixels in this group are set for further test in the second pass. 3) Pixels that are not surrounded by any structure are not included (the rest). The result of the first pass is an temporary image with a different value for each group

Second pass: tests if the pixels in second group of the first pass have to be included in the eroded structure or not. The pattern of the kernel used for the second pass is as follows: *Cell_1* is the pixel North of pixel in question. *Cell_2* is the pixel to the East. *Cell_3* is the pixel to the West, and *Cell_4* is the pixels to the South. The four pixels are from the temporary image where as the pixel *Old-value* is from the original image and South of the pixel in question

The final value *Sum* used in the second pass is the sum of pixels surrounding the pixel in the temporary image.

The following six test are conducted in the listed order:

1. Cell_1 && Cell_2 && Cell_3 && Old-value != Cell_4 => include
2. Cell_1 && Cell_2 && Cell_3 && Old-value => include
3. Sum > 0 => not include
4. !Cell_1 && !Cell_2 && !Cell_4 => include
5. Cell_1 && Cell_4 && !Cell_3 => include
6. !Cell_1 && Cell_2 && !Cell_4 => include

   Not include

Table 7.2: Algorithm for eroding the contour of the silhouette for which the rays are cast.

Cast an Infinite Ray

The infinite ray is cast from the optical center through a pixel $\mathbf{m}$ in the eroded structure. A 3D point in homogeneous coordinates on the ray is computed as follows,

$$\mathbf{M} = \left[\mathbf{t} + \lambda \mathbf{R}\mathbf{K}^{-1}\mathbf{m}, 1\right]^{\top} \tag{7.1}$$

where λ is a scaling factor that indicates how far from the optical center the 3D point $\mathbf{M}$ is located on the ray. $\mathbf{t}$, $\mathbf{R}$, $\mathbf{K}$ are the parameters of the camera (see appendix A).

The infinite ray cast from one view is projected into all the other views using the fundamental matrix (two view geometry). The fundamental matrix describes the correspondence between a point and its epipolar line in an other view (see appendix A).

Compute the 2D Intersections

The intersection between the different silhouettes are moved from 3D space to 2D space in order to make it faster and avoid finding intersection in 3D space. The problem is that two rays does not necessary intersect in 3D space due to the resolution of the images from which the rays are cast and imperfect camera matrices. When moving the intersection from 3D space to 2D space is done using two view geometry.

The intersection between the projection of the cast ray (epipolar line of point $\mathbf{m}$) and the silhouette in form of the line description of the contour is a 2D line intersection algorithm.

The algorithm used for computing the 2D line intersection is described in Bourke (1989).

The list of intersections along the epipolar line is first sorted according to increasing x-coordinate and then y-coordinate. As the silhouette is limited by the image frame it is possible to determine if the intersection point is an entry- or exit-point to the silhouette. The first intersection point in the sorted list is an entry point, the next is an exit point and so on. The intersection points are combined in into lines that describe the silhouette along the current epipolar line.

Lift the Intersections Back to The Cast Ray

The lines found by combing the entry- and exit-points are lifted back on the cast ray. The points on the ray are represented as λ values in equation 7.1.

The λ values are computed as a projection of the 3D point $\mathbf{M}$ in equation 7.1, into a point $\mathbf{m}_2$ in image 2 as follows,

$$\begin{aligned} \mathbf{m}_2 &= \mathcal{P}_2\mathcal{P}_1^{-1}\mathbf{m}_1 \\ \mathbf{m}_2 &= \mathbf{K}_2\left[\mathbf{R}_2^{\top}(\mathbf{t}_1 - \mathbf{t}_2) + \lambda \mathbf{R}_2^{\top}\mathbf{R}_1\mathbf{K}_1^{-1}\mathbf{m}_1\right] \\ \mathbf{m}_2 &= \underbrace{\mathbf{K}_2\mathbf{R}_2^{\top}(\mathbf{t}_1 - \mathbf{t}_2)}_{\mathbf{e}_2} + \lambda \underbrace{\mathbf{K}_2\mathbf{R}_2^{\top}\mathbf{R}_1\mathbf{K}_1^{-1}\mathbf{m}_1}_{\mathbf{F}_{12}} \end{aligned} \tag{7.2}$$

To find a solution, using SVD (section 6.7 in Strang (1998)), for λ two variables Φ and Ψ are added to the equation,

$$\Phi\mathbf{m}_2 = \Psi\left(\mathbf{K}_2\mathbf{R}_2^{\top}(\mathbf{t}_1 - \mathbf{t}_2)\right) + \lambda\left(\mathbf{K}_2\mathbf{R}_2^{\top}\mathbf{R}_1\mathbf{K}_1^{-1}\mathbf{m}_1\right) \tag{7.3}$$

Each special solution to equation 7.4 gives a valid solution to λ,

$$\mathbf{A}\mathbf{x} = \begin{bmatrix} \mathbf{e}_2^\top & \mathbf{F}_{12}^\top & \mathbf{m}_2^\top \end{bmatrix} \begin{bmatrix} \Phi \\ \lambda \\ \Psi \end{bmatrix} = \mathbf{0} \tag{7.4}$$

The matrix $\mathbf{A}$ is solved using SVD and the result of the SVD is given in,

$$\mathbf{A} = \begin{bmatrix} \mathbf{e}_2^\top & \mathbf{F}_{12}^\top & \mathbf{m}_2^\top \end{bmatrix} = \begin{bmatrix} \mathbf{U} & \Sigma & \mathbf{V}^\top \end{bmatrix} \tag{7.5}$$

The solution is found in the null-space of matrix $\mathbf{A}$,

$$\begin{bmatrix} \Phi \\ \lambda \\ \Psi \end{bmatrix} = \begin{bmatrix} \mathbf{V}^\top [min\ \sigma\ of\ \Sigma] \end{bmatrix} \tag{7.6}$$

Where the smallest value of σ in Σ gives the row in $\mathbf{V}^\top$ which is the null-space of $\mathbf{A}$, where the λ value is scaled by Φ.

When a λ value is computed for each the entry- and exit-point in the previous found lines are they tested against four special cases. This is done for ordering the λ values along the cast infinite ray. The four special cases are shown in figure 7.10.

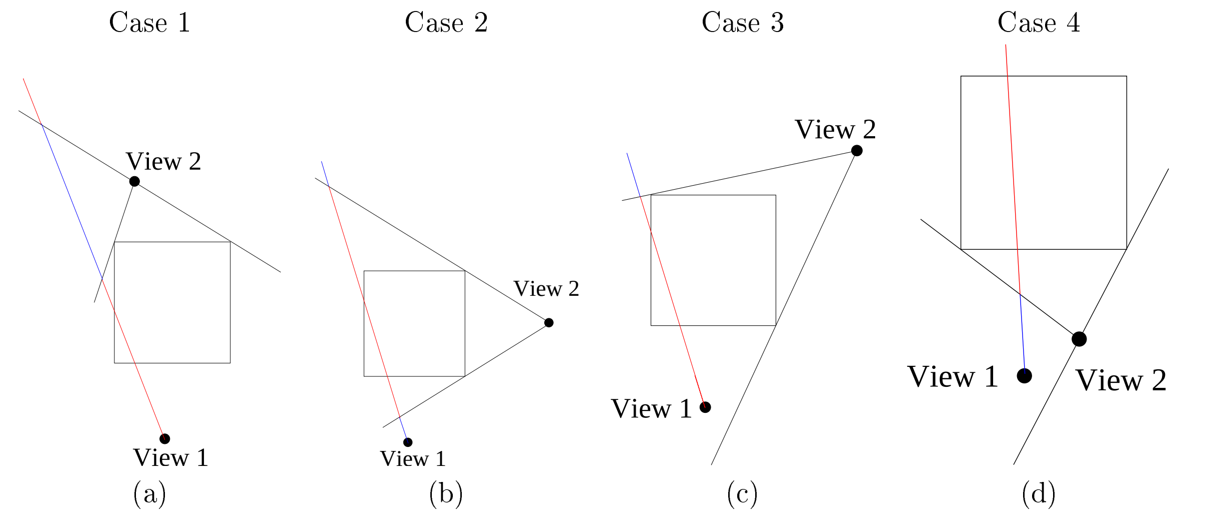


Figure 7.10: The figure illustrates the four special cases of intersection along a infinite ray. The red part of the infinite line cast from View 1 is inside the silhouette if projected into View 2, and the blue part is outside.

Each line is tested against the following four cases and depending on the which of the cases are the λ values of the line converted into 3D points on the ray.

Special case 1: is when both λ values are positive and the point at infinity is projected on the line in View 2. The projection of the point at infinity is defined as: $\mathbf{m}_2 = \mathbf{K}_2\mathbf{R}_2^\top\mathbf{R}_1\mathbf{K}_1^{-1}\mathbf{m}_1$. The line is lifted back to the ray between the 3D points with λ_1 = zero (optical center of the view 1) and λ_2 = the smallest found λ value (the 3D point closest to the optical center of the view 1)

Special case 2: is when both λ values are positive and not special case 1. The line is lifted back to the ray between the 3D points with λ_1 = the smallest found λ value (the 3D point closest to the optical center of the view 1) and λ_2 = the largest found λ value (the 3D point furthest away from the optical center of the view 1)

Special case 3: is when one of the λ values is positive and the other is negative and the 3D point between the optical center and 3D point with at the positive λ is projected on the line. The line is lifted back to the ray between the 3D points with λ_1 = zero (optical center of the view 1) and λ_2 = the positive λ value (the 3D point in front of view 1)

Special case 4: is when one of the λ values is positive and the other is negative and not special case 3. The line is lifted back to the ray between the 3D points with λ_1 = the positive λ value (the 3D point in front of view 1) and $\lambda_2 = \infty$

The final case is where both λ values are negative is the line discarded as the intersection are behind the view (camera).

Limit the Cast Ray

The 2D intersections that are lifted back to the cast infinite ray are used for locating the part along the cast ray where most views agree on 3D structure. The intersections are divided into two categories: Intersection inside the image frame and intersections on the image border (silhouette cut by the edge of the image). First the intersections inside the images are used for locating the biggest overlap of lines lifted back on the ray.

The idea is to find the λ intervals which gives the biggest overlap. First the list of intersection points is sorted after increasing λ value. Next the sorted list is converted into a histogram where the value of the bins equals the number of overlapping lines lifted back on the ray. This is done in the following way: For every entry point the number is increase by one, and for every exit point is the number decreased by one. The highest bin in the histogram gives the λ values (3D point along the cast ray) for biggest overlap along the ray. The problem is that missing structure in one view caused by camouflage results in less overlapping lines along the ray even though it is present in the scene. To overcome this problem it is not only the highest bins there are selected but instead it is peeks in the histogram that are selected with a certain number of views agreeing on 3D structure. If a selected bin is an entry point is the nearest exit point selected and wise versa. The two λ values defines the part of the cast ray that is noted as the limited ray. It is possible to have more than one bin with the highest number. This result in a multiple number of limited rays.

This is illustrated in figure 7.11

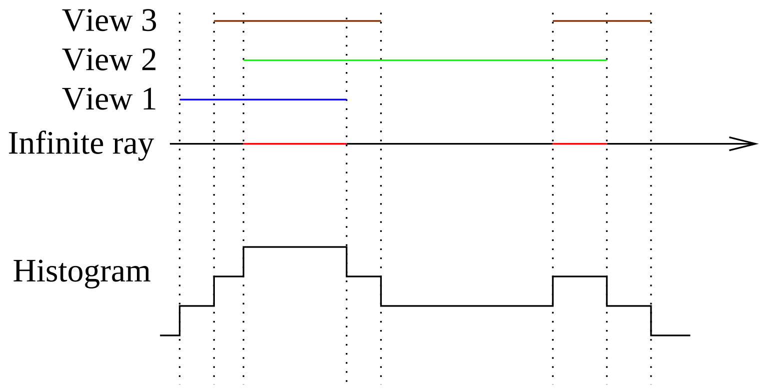


Figure 7.11: The figure illustrates how the lines are lifted back to the infinite ray from three views and converted in a histogram of intersections. The two red parts of the infinite ray are limited rays

In the case where there are no intersections inside the images, e.g. when the projection of the object is bigger than the image frame in all views, are the intersections on the image boarder used in the same way as previously described.

Project the Limited Ray Into the Views

The now limited ray is projected into all of the views. As λ values might be zero or ∞ is it necessary to performed a test before projecting the limited ray.

The reason for this test is to prevent a possible overflow when projecting the start- and end-point of the limited ray into the other views. This all depends on the relative location of the views.

The maximum λ value that does not result in overflow is calculated. The four most extreme images coordinates (the corners) are lifted back to the limited ray giving four λ values. The biggest λ value defines the biggest λ value able to project without overflow and the smallest value is used to prevent underflow.

7.3.4 Robustness of a Visual Hull Representation

The visual hull representation is tested on silhouettes with noise added. Figure 7.12 shows the result of the test.

As it can be seen in figure 7.12 the algorithm is robust for noise up to one pixel. Noise of one pixels means that the silhouettes of each view used for the test is shifted one pixel in a randomly picked direction. It is the complete image of the silhouette that is shifted to simulate errors in the estimation of the camera parameters. There are 8 directions for which the silhouette might be shifted plus no shift. Figure 7.13 illustrates a top view of the scene used for the test.

The shift in x- and y-direction for each silhouette are listed in table 7.3.4

Amount of noise	silh. of view 1	silh. of view 2	silh. of view 3	silh. of view 4	silh. of view 5	silh. of view 6
One pixel	x = 0 y = 1	x = 1 y = 1	x = -1 y= 0	x = -1 y =1	x =1 y = 0	x = -1 y = 0
Two pixels	x = -1 y = -2	x = 1 y = -2	x = -1 y = 1	x = 1 y = 1	x = 1 y = -2	x = -2 y = -2

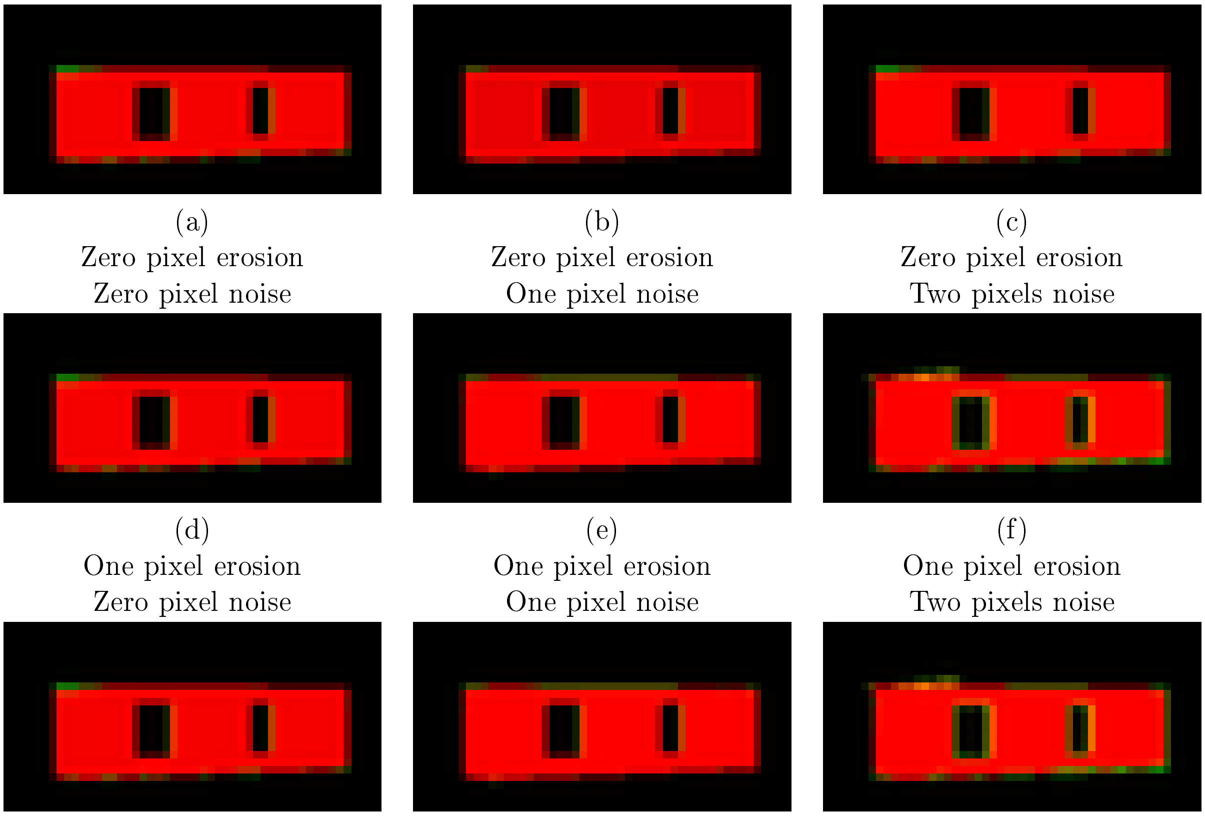
7.4 Post Processing

The purpose of the post processing is to determine if the holes inside the visual hull representation are caused by camouflage or are real holes in the object. In case a hole is caused by camouflage it is filled (removed from the silhouette) otherwise it is left as a hole in the silhouette. As previous mentioned the idea is to combine the pattern of the holes in the silhouettes and test if there are a color similarity between the holes, in the different silhouettes, that corresponds to the same potential hole in the object.

Below is a more detailed description on the different steps involved in the test of a region/hole is caused by camouflage or is a real hole in the object.

7.4.1 Locate the Boundary of the Visual Hull Surface Representation

The visual hull surface is computed in section 7.3.1 and consists of all the starting points (3D points with the smallest λ) of the lines used for building up the common visual hull representation using the new approach. The silhouette generated using background subtraction is merged with



(a) Zero pixel erosion Zero pixel noise

(b) Zero pixel erosion One pixel noise

(c) Zero pixel erosion Two pixels noise

(d) One pixel erosion Zero pixel noise

(e) One pixel erosion One pixel noise

(f) One pixel erosion Two pixels noise

(g) Two pixels erosion Zero pixel noise

(h) Two pixels erosion One pixel noise

(i) Two pixels erosion Two pixels noise

Figure 7.12: Figure (a) to (i) illustrates the differences between the original silhouette and the computed visual hull representation under different setups. The columns indicated to amount of random noise add to the images containing the silhouettes. The rows indicate how many pixels the structure, for which the rays are cast, are eroded. The red color is where both silhouette and visual hull representation are present. The yellow color is where there are only visual hull representation.

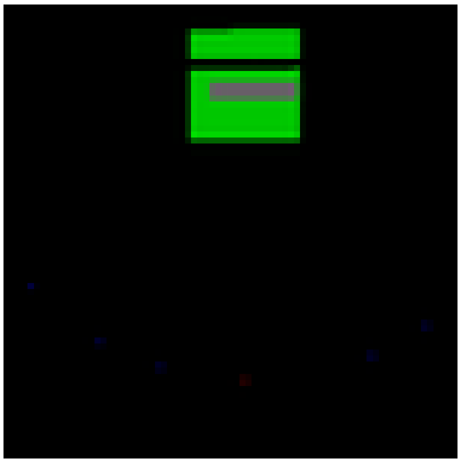
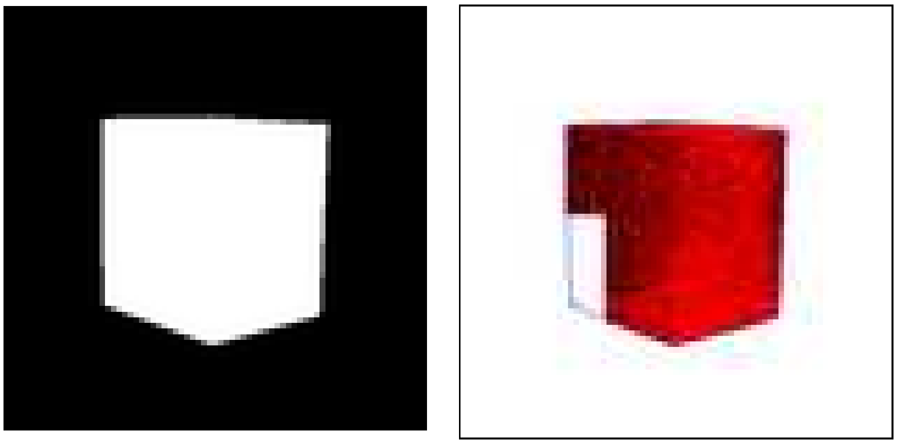


Figure 7.13: The figure shows a top view of the scene used for robustness test. The red circle illustrate the first view and the rest are located clockwise from the first view

the visual hull surface representation of the same view. Merging the two silhouettes gives a more robust result as it combines the result of the background subtraction with the refined silhouette. The outliers in this representation if any are removed/minimized by removing all isolated pixels.

The merged silhouette consist of both closed and open regions. In order to combine the pattern of the holes in the silhouettes of the different views they have to be closed (see figure 7.14). The open regions are closed by a convex hull around the merged silhouette. All closed regions are now considered a potential real hole in the object.



(a) (b)

Figure 7.14: The figure (a) shows the silhouette of the object without camouflage effect added. The figure illustrates that it is possible to use the surface visual hull representation with a convex hull to locate the *real* boundary of the object. The red color represent the silhouette with camouflage used in the visual hull algorithm, and the black color represent the surface visual hull representation, and finally the blue color represent the boundary of the object found using convex hull on the surface visual hull representation

Figure 7.15 illustrates the scene and the silhouettes used for generating the results in figure 7.14

7.4.2 Locate the Different Holes Inside the Boundary

The outer boundary of the silhouette is now determined using convex hull, and the next step is to locate all the different holes inside this boundary. This is done by grouping all four-connected pixels the are not part of the silhouette and inside the found boundary (section 2.4 in (Gonzalez and Woods, 1993)). Each group represent a hole in the silhouette.

7.4.3 Initial Determination of the Type of Holes

To increase the robustness and speed of determining the type of holes present in the silhouette it crucial to decrease the number of holes in the silhouettes. This is possible by dividing the holes into four categories:

Noise is when the number of pixels in the region defining the hole is below a certain threshold. The hole is removed by filling up the region in the silhouette

Medium hole is when the number of pixels in a hole is above the definition of *Noise* and smaller than a certain threshold. The hole is mixture (blending) of the image and the virtual object. Gives the effect of a fence from a distance where it is hard to tell what is behind due to the small holes

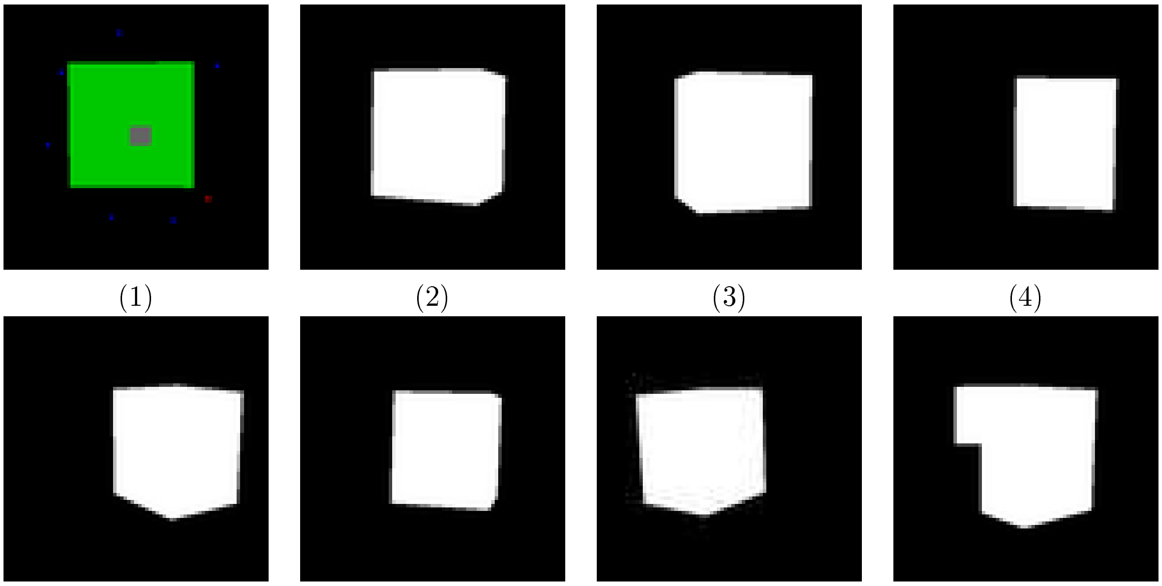

(1) (2) (3) (4)
(5) (6) (7) (8)

Figure 7.15: Figure (2) to (8) shows the ground truth silhouettes of the views in figure (1) segmented by hand. Figure (2) is the view (blue sphere) to the left of the view marked with a red sphere. Figure (3) is the next view (blue sphere) to the left. The rest are counted clockwise until figure (8), which represents the view with added camouflage (red sphere).

Real hole is when the hole is present in the depth map as a part of the background meaning that this part is reconstructed by the reconstruction system. The hole in the silhouette remains unchanged

Unknown hole are the remaining holes that have to be tested for being a camouflage hole or a real hole in the object. The test is described below

7.4.4 Compute the Color Histogram of Each Region

The color histogram is used as a similarity measurement when determining if a hole is caused by camouflage or it is a real hole. The purpose is to test if a hole in the silhouettes that corresponds to the same hole in the object changes when observed from different angles.

There are different ways to measure a color distribution in a region. It can be based on pure color values in a selected color space or some geometric representation can be added to the histogram. One geometric representation might be the distance, meaning that the histogram indicates something about how the colors are combined at a certain distance (e.g 3 pixel). The problem of adding geometric presentations into the histogram is that it becomes sensitive to zoom.

The measurement used in this project is based on a color distribution in RGB and the measure is the Bhattacharyya coefficient. As a measure of divergence between the two normalized populations Bhattacharyya use the square of the angle between the two position vectors (Aherne, Thacker, and Rockett, 1997). The two populations p and q represent the color histograms of the two holes to be compared, k is the number of elements. The measure is:

$$\cos\theta = \sum_{i=1}^{k} \sqrt{p_i q_i} \tag{7.7}$$

If the two populations are identical this results in,

$$\cos\theta = \sum_{i=1}^{k} p_i = 1 \tag{7.8}$$

corresponding to $\theta = 0$ meaning the vectors are co-linear.

7.4.5 Determine if a Region is Caused by Camouflage or a Real Hole in the Object

The *unknown holes* from the initial determination are not yet determined. The idea is to use the pattern for which a hole develops when extracted from different view points. Figure 7.16 illustrates the differences in how a real hull and a hole caused by camouflage develops throughout the views.

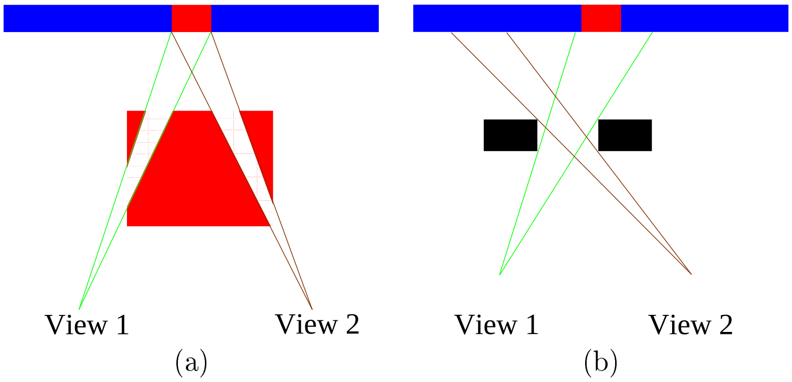


Figure 7.16: Figure (a) illustrates how holes caused by camouflage develop throughout the silhouettes. In view 1 the hole is in the left side of the silhouette whereas it is placed in the right side of the silhouette in view 2. Figure (b) illustrates how real holes develop throughout the silhouettes. Both in view one and two the hole is in the same part of (middle) the silhouette

The holes caused by camouflage are located differently in the silhouette extracted from different view points, whereas the real holes are located in the same part of the silhouettes.

Humans use the information on how the hole behave from different angles to determine if the hole is real or not. The problem is that a human tracks the hole while moving to a new location. In the present system there are only a number of independent silhouettes available, so it is impossible to track the hole while moving to a new location. The task is to find matching holes in the different silhouettes and combining the information given by the corresponding holes.

Characteristics for a real hole

- Rays cast from different views through matching holes in the silhouettes intersects inside the 3D object. see figure 7.17 (a)
- Different parts of the background are observed through matching holes leading to different color histograms (except for repeatable patterns)

Characteristics for a hole caused by camouflage

- Different parts of the background are observed through matching holes but leading to same color histogram

- The same part of the background is observed through unique holes but leading to the same color histogram

The algorithm used for determining the type of hole is listed in table 7.3 and figure 7.17 is used for support to the algorithm.

1. Compute the distance from the camera to the 3D structure surrounding the hole. The maximum and minimum values are found in the depth-map of the current view
2. Cast a ray through the hole to be determined (desired ray).
3. Cast rays through the *unknown* or *real* holes in the silhouettes of the other views (reference ray)
4. Compute 3D intersections between the desired ray and reference rays
5. Locate the 3D intersection inside a sphere located on the desired ray between the maximum and minimum distance found in step 1. The radius of the sphere is maximum distance from the center of mass of the hole to the border of the hole, or difference between minimum and maximum found in step 1 depending on which is the biggest. see figure 7.17 (b)
6. If intersection inside the sphere described in the previous step then compare the color histograms of the holes involved
   - If color histograms are different (using the measure in section 7.4.4) is it a real the hole in the object
7. Project the center of the sphere into the other views and compare the color histograms of a circular region in the image of the actual view point according to the projection of the sphere
   - If the color histograms are different (using the measure in section 7.4.4) is it a real hole in the object. (only used the views for which the center of the sphere is visible)

Table 7.3: Overview of the algorithm to determine the cause of the hole.

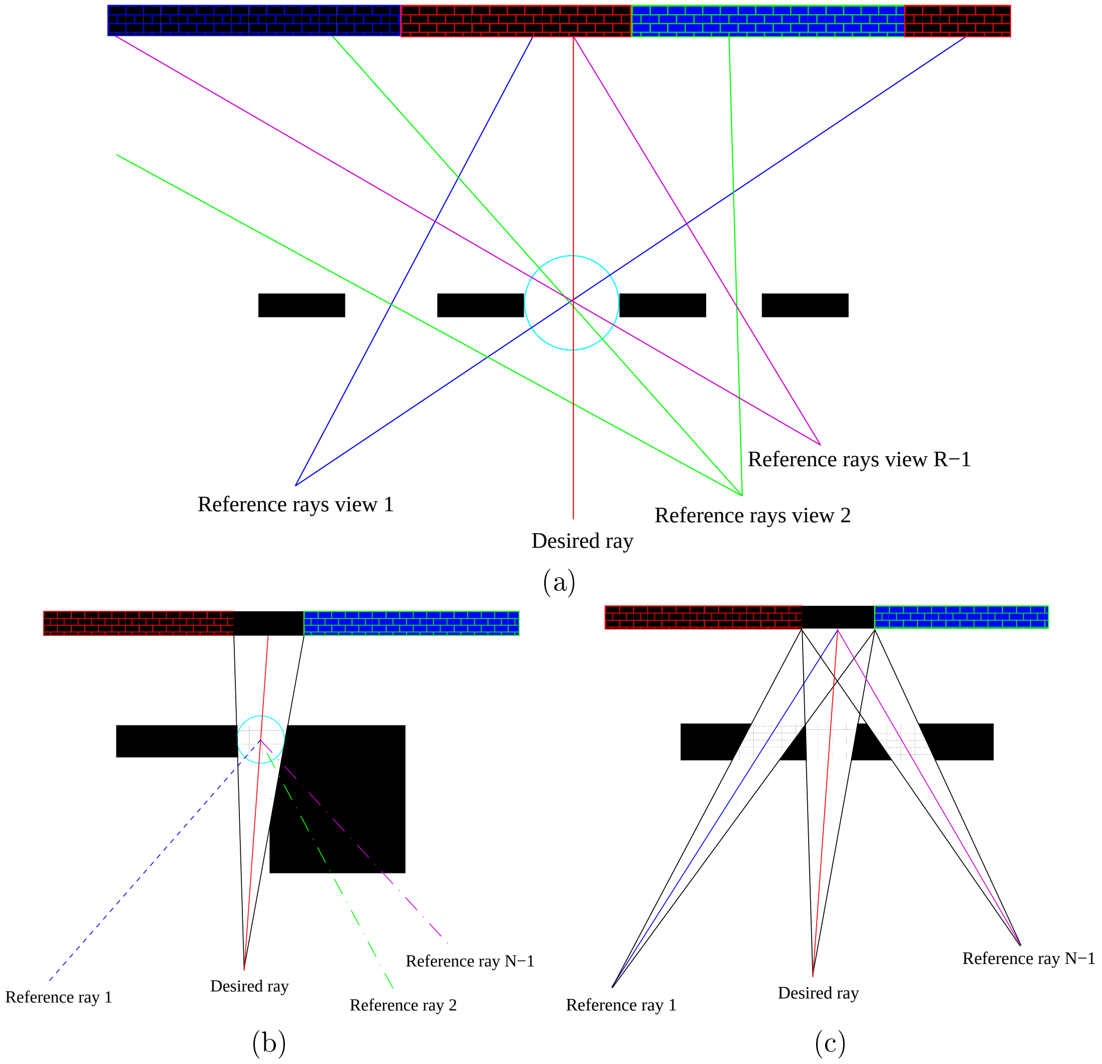


Figure 7.17: Figure (a) illustrates the case with real holes. The intersections inside the sphere result in different color histograms for the different reference rays meaning that the hole for which the desired ray is cast is real. Figure (b) illustrates a camouflage case where only the center of the sphere is visible for reference ray one. As reference ray one is not cast through a hull in the silhouette, but back-projected from the center of the sphere, and the colors of reference rays are similar This means that the hole in question (desired ray) is caused by camouflage. Figure (c) illustrates a camouflage case where there only are intersection on the background meaning the hole is cased by camouflage.

7.5 Summary

The two step refinement of the silhouettes generated using background subtraction is able to determine the cause of a hole in the silhouette, but there are still cases that the algorithm breakes down e.g. a fence in front of a brick wall.

The property of the refinement process is that it is only able to add to the occlusion mask, so it is better to select a threshold in the background subtraction process which results in a silhouette that does not contain all structure of the occluding object than one with background pixels classified as foreground.

CHAPTER 8

Visualization

8.1 Overview

The task of the visualization subsystem is to combine 3D renderings of virtual objects with the calculated occlusion masks and information about the real scene to generate the augmented output image. A flowchart of the process can be seen in figure 8.1. The various steps of the flowchart is described in the following sections.

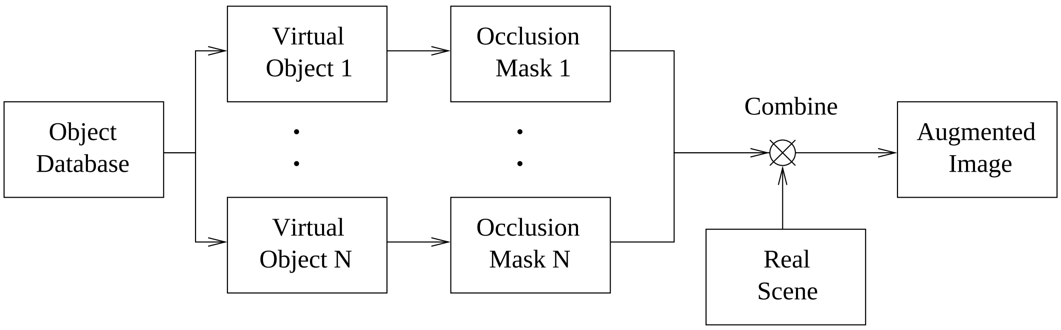


Figure 8.1: Flowchart of the visualization process.

8.2 Real Scene

The image of the real scene for the visualization can be aquired in two ways:

- Raw input image
- 3D rendering

The raw input image aquired from the camera can directly be used as the image of the real scene or a rendering of the textured 3D model from the estimated camera position can be used. Figure 8.2 illustrates the two approaches.

(A) (B)

Figure 8.2: Visualization using the raw input image (A) and a rendering of the 3D model from the same point of view (B).

The two approaches have different advantages (+) and disadvantages (-).
Properties of using the raw input image are:

+ High image quality.
+ Fast because the input images are readily available.
+ Suitable for use on semi-transparant augmented reality glasses (real scene not rendered).
- Jitter between virtual objects and real scene in case of inaccurate camera parameter estimation.
- Cannot show novel views, only overlay virtual objects.

Properties of using a rendering the 3D model are:

- Lower image quality because of inaccurate 3D model / texture.
- Slower as complex 3D model must be rendered (100k+ textured triangles).
+ No Jitter as virtual objects and real scene is rendered with same camera parameters.
+ Flexible. The 3D model can be rendered from novel views and special effects (change of lighting / texture) can be performed.

Based on the listed properties the raw input image is used as image of the real scene.

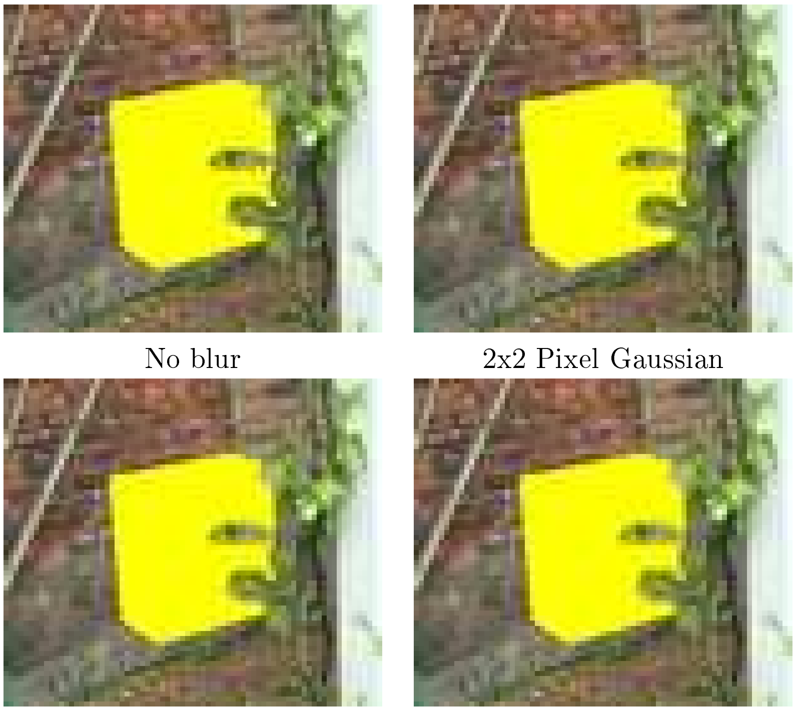
8.3 Virtual Objects

Virtual objects are rendered using the estimated camera parameters. In order to get a credible augmented output image, rendering must be performed using similar lighting conditions as the real scene.

8.4 Occlusion Masks

The occlusion masks are binary masks describing which parts of the virtual objects are visible and which are not. The virtual objects can be clipped to the occlusion masks effectively by 3D rendering hardware if the hardware supports a stencil buffer and stencil tests during rendering.

The occlusion masks most likely will contain errors (misclassified pixels), which can adversely affect the quality of the augmented images, especially if the silhouette edges change over time. In order to get a more pleasing visual output alpha-blending can be used to smooth the occlusion edges. Figure 8.3 illustrates the visual effect of performing gaussian blur with varying kernel size on the occlusion mask. The occlusion mask produced by the 2x2 radius kernel has a significantly more pleasing visual output than the raw occlusion mask, whereas the bigger kernel sizes leads to very blurry edges.



No blur — 2x2 Pixel Gaussian — 4x4 Pixel Gaussian — 6x6 Pixel Gaussian

Figure 8.3: Visualization using an occlusion mask with no blurring, and gaussian blur using a kernel with a radius of 2x2, 4x4 or 6x6 pixels

Gaussian blur can be effectively approximated in hardware (Evans (2001)), and alpha blending is relatively cheap on consumer hardware. Alpha blending can be performed by rendering the virtual object to a texture with an alpha channel, and then render the blurred occlusion mask to the alpha channel and finally render a screen aligned billboard using the texture at the position of the virtual object with alpha blending enabled. The reason for this extra step is that rendering alpha blended objects is ordering dependent.

8.5 Combining

The clipped virtual objects must be sorted in back-to-front Z-order before combining them with the image of the real scene, as virtual objects further away otherwise could occlude nearby virtual objects.

The complete algorithm is itemized below.

1. Copy input image to frame buffer
2. Sort virtual objects in back-to-front order
3. For each virtual object do:

    3.1 Render virtual object to texture

3.2 Calculate occlusion mask

3.3 Blur occlusion mask

3.4 Copy mask to alpha channel of texure

3.5 Render virtual object as screen aligned billboard

8.6 Summary

An effective approach for visualization of the combined result of input data, virtual objects and occlusion masks has been presented.

No attempt on rendering virtual objects using similiar lighting parameters as the real scene has been done as the reconstruction system does not estimate lighting parameters.

CHAPTER 9

System Test

The following chapter documents a test of the developed system on real data.

9.1 Overview

A prototype has been developed in C++ using OpenGL and The Visualization Toolkit (Inc. (2001)) on a Linux platform.

A test sequence has been recorded in the botanical garden in Leuven, Belgium. The sequence has been recorded using a Sony DCR-TRV900E DV camcorder running in progressive scan at a resolution of 720x576 pixels. Every 20th frame of the video sequence has been extracted (37 images) and processed by the reconstruction system developed by VISICS.

Every third image of the botanical sequence together with the reconstructed 3D model using depth map 10 can be seen in appendix B.

The depth map from image 10 has been selected as 3D model as it has a good view of the area behind the white pillar in the foreground.

The task of the augmented reality system is to insert a yellow virtual box behind the white pillar in the foreground of the scene.

The system test is divided into each of the sub-tasks of the developed system, namely:

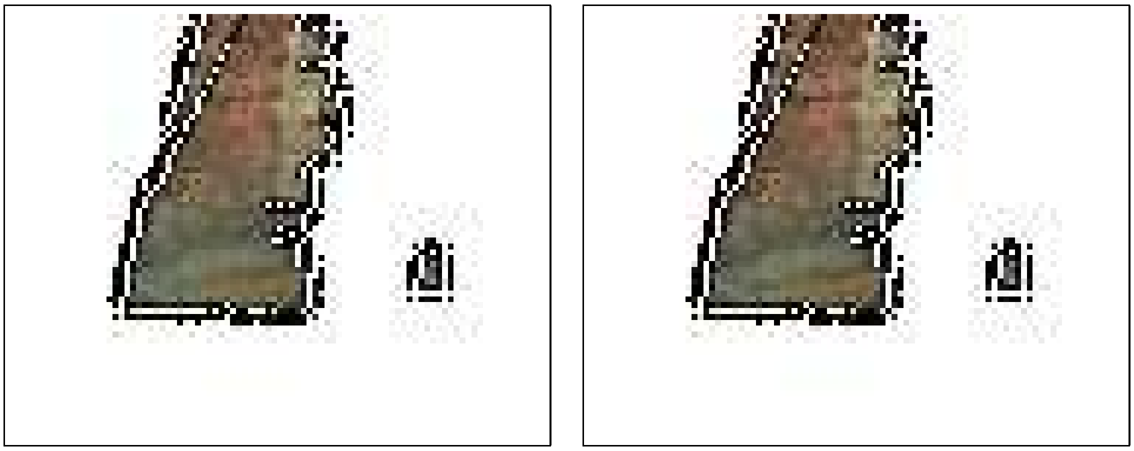
- Background subtraction
- Visual hull
  - New approach
  - Post processing
- Visualization

Each sub-test is only shown for image 15 in this chapter. Results for every third view can be found in appendix C.

9.2 Background Subtraction

9.2.1 Background Model

Image warping using the depth map from image 10 as 3D model has been performed. The two background models of image 15 can be seen in figure 9.1. Just as the test results for synthetic data (see section 6.3) there is very little difference between the two background models, and therefore the rest of the background subtraction tests are only performed on the median background model.



Average Median

Figure 9.1: Average and median background model of image 15.

Notice that the background models only contains information about a small part of the image. This is caused by the occlusion shadows on both sides of the pillar in the depth map of view 10, which is used as 3D model. See appendix B.2 for images of the 3D model.

9.2.2 Image Warping

In order to illustrate the precision of the image warping algorithm the median background model has been overlayed on top of the input image from the same viewpoint. This can be seen in figure 9.2 together with the raw input image.

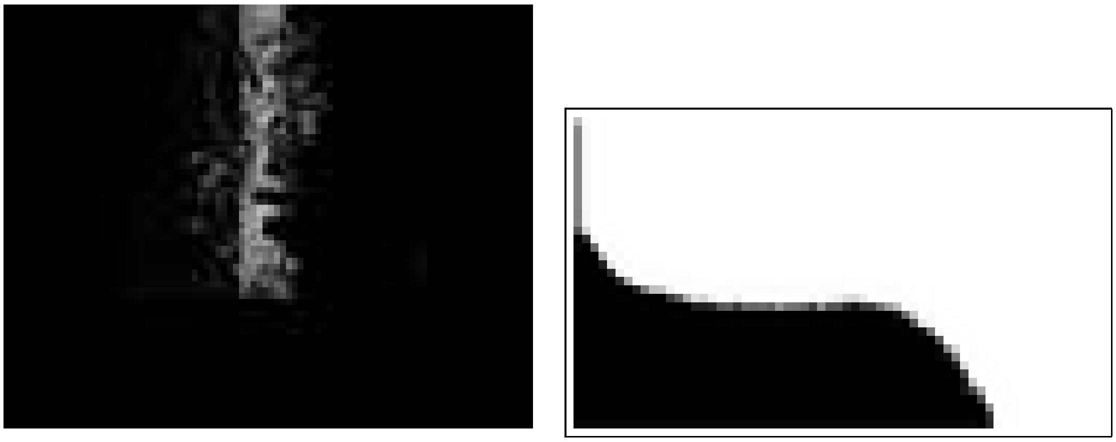
Input Image Input Image with Median Background Model

Figure 9.2: Input image 15 and input image 15 with median background model overlayed on top.

As the accurate fit of the background model on top of the input image shows the estimated camera parameters and structure of the background from the reconstruction system only contains minor errors.

9.2.3 Segmentation

Figure 9.3 illustrates the difference image between the background model and the input image.



Difference Image Histogram

Figure 9.3: Difference image between image 15 and median background model and histogram of difference image.

The pillar and leaves of the foreground are clearly visible in the difference image, but as the histogram shows the optimal segmentation threshold is not trivially selected. Instead several segmentation thresholds has been tested. Figure 9.4 illustrates the segmented occlusion masks using a threshold of 1/10, 1/8, 1/6 and 1/5.

The refinement process can only add foreground pixels, but not remove any. Therefore the amount of background pixels classified as foreground should be kept to a minimum. Based on this a threshold of 1/5 has been selected.

9.3 Visual Hull

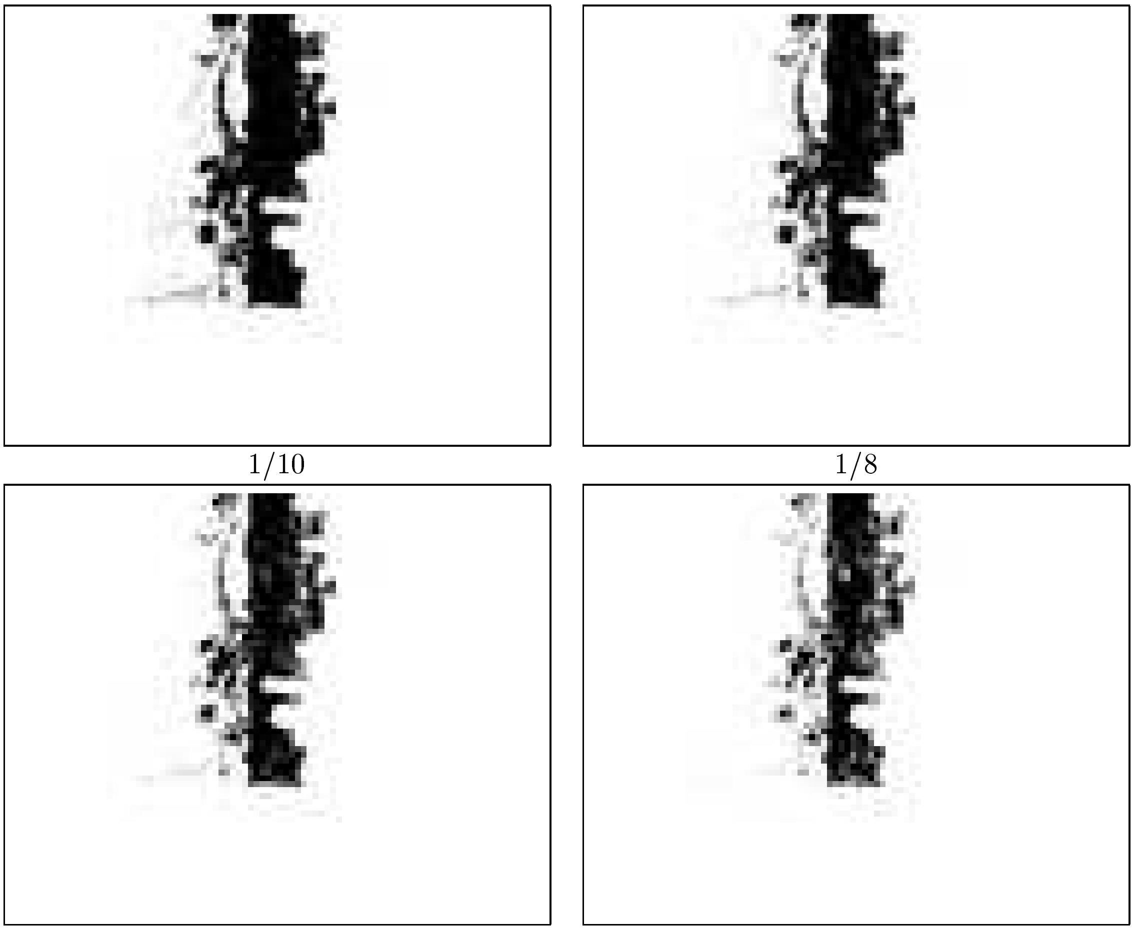
The refinement steps is not tested on the reconstructed scene but the functionality of the methods used are briefly related to the initial occlusion mask.

9.3.1 New Approach

To illustrate the effect of the first refinement the occlusion mask of view 5 is overlayed with the image of view 5. The result is shown in Figure 9.5 where the red line illustrates the epipolar line passing through the pillar where structure is missing. The epipolar line is the result of a ray is cast through the pillar from another other view. As there are no intersection of the epipolar line inside the pillar from the current view, and given there are intersections along the cast ray from other views that describes the 3D structure of the pillar, it is possible to add the structure in the current occlusion mask.

9.3.2 Post Processing

The post processing is illustrated in Figure 9.6 showing the occlusion mask from view 20. The purpose of this step is to determine the cause of a hole in the occlusion mask. A few closed



1/10 1/8

1/6 1/5

Figure 9.4: Segmentation using thresholds of $1/10$, $1/8$, $1/6$ and $1/5$.

Figure 9.5: An example of structure added to the occlusion mask using visual hull

regions (holes in the occlusion mask) are marked with red and blue colors. The regions marked with blue are considered noise and removed without any test. The regions marked with red are a potential real hole. This region is tested as described in section 7.4. The results determine if the regions are removed from the occlusion mask or not..

Figure 9.6: Examples of holes in the occlusion mask that are either removed direction based on the size (blue regions), and holes (red regions) where a test is used for determining the cause of the hole

9.4 Visualization

The visual result of using the calculated occlusion masks from the background subtraction process is compared to using the raw depth maps in figure 9.7. The four images shows the final augmented images using the 3D model of view 10, using the depth map of the current view, and using the calculated occlusion masks to calculate visibility of the virtual object. The occlusion masks are blurred using a gaussian blur with a kernel with a radius of 2x2 pixels as described in chapter 8.

The two holes in the white pillar on the images using the calulated occlusion masks is caused by missing background information for those areas. The occlusion mask with a threshold of 1/5 lacks a lot of structure as the refinement process has not been performed. Of the four augmented images the occlusion mask using a threshold values of 1/10 contains the visually most pleasing result.

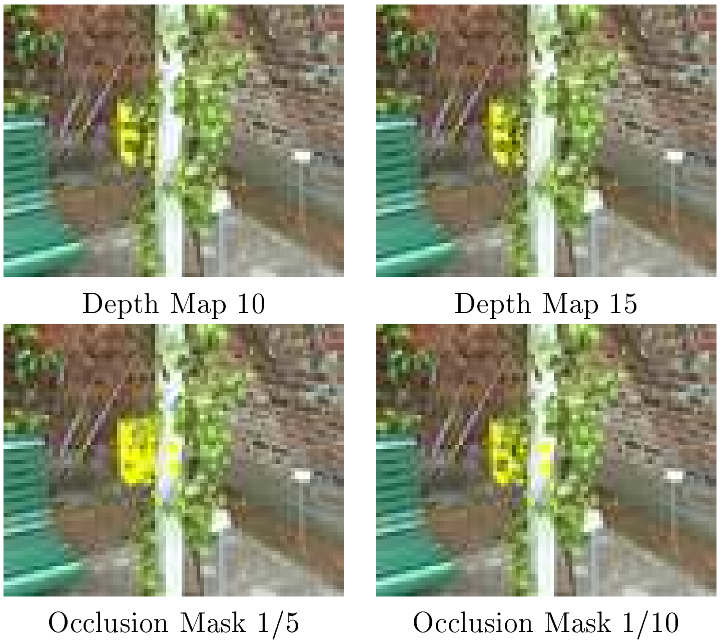


Figure 9.7: Final augmented image using the depth map from image 10, 15 and the occlusion mask using a threshold of 1/5 and 1/10 to handle occlusions. The occlusion masks have been blurred with a 2x2 pixel gaussian blur.

9.5 Summary

The system test is performed on the result of a reconstruction from VISICS. The initial occlusion mask is computed for each view in the sequence and is input for the refinement steps. The refinement steps are only tested on synthetics scenes as the initial occlusion mask first was available late in the project. The reason for this late occlusion mask was due to problems with another reconstruction (scene) from VISICS used for an early test. The reconstruction was later discarded due to problems with the data. The visualization step is tested during the implementation phase.

CHAPTER 10

Conclusion

This chapter contains the conclusion and discussion of the project.

10.1 Conclusion

The purpose of the project was to determine how to resolve occlusions in augmented reality.

The motivation for constructing such a system was the reconstruction system made by the Department of Vision for Industry Communications and Services (VISICS) at Katholieke Universiteit Leuven, Belgium (Pollefeys, 2000), which is capable of estimating camera parameters and construct a textured 3D surface model of a static scene from a sequence of un-calibrated images.

The project is an extension to the reconstruction system, and the aim was to determine the silhouette (occlusion mask) of real objects occluding virtual objects more accurately than the reconstruction system at hand. This lead to the specific problem of the project.

> How can a robust extension be constructed, which by use of an estimated 3D surface model and multiple images acquired from a hand-held camera, resolves occlusions in augmented reality ?

10.1.1 System Design

The extension design was divided into four steps,

- Color based segmentation of foreground objects using background subtraction
- Refinement of edges of the occlusion masks using visual hull
- Removal of camouflage holes inside occlusion masks
- Visualization of augmented scene

Color based segmentation of foreground object using background subtraction

The *purpose* of performing color based segmentation was to extract masks of foreground objects defined as objects in front of the virtual object to be inserted in the scene.

The *properties* of background subtraction was to extract the difference between input images and a model of the background, which resulted in a mask of what was in front of the clipping plane defined by the location of the virtual object in the 3D scene.

Background subtraction using a movable camera was made possible using an image warping process developed by the project group.

In contrast to masks extracted using the depth maps of the reconstruction system, the masks generated using background subtraction had the following advantages,

- No problems around depth discontinuities
- Robust against missing reconstruction of the foreground object as long as 3D geometry behind the clipping plane was reconstructed.

The *drawbacks* of using background subtraction for segmentation of foreground objects were.

- Camouflage effects caused holes in the masks when the color of foreground objects matched the color of the background model.
- Use of a simple clipping plane which caused problems placing a real object between the virtual object and the clipping plane (see Figure 6.6)

The problem of the camouflage effects were corrected in the next step, whereas the problem of the simple clipping plane was not handled in the extension.

Refinement of edges of the occlusion masks using visual hull

The *purpose* of performing a refinement of the edges of the occlusion masks generated in the previous step was to locate 3D structure found in multiple views, which were discarded in the current view due to camouflage.

The *properties* of visual hull was to locate 3D structure based on 2D silhouettes of the structure from different views (two view geometry). If a number of views (not necessarily all) could agree on 3D structure along a ray, the 3D structure was considered a part of the 3D objects in the scene and added to the occlusion mask.

The property of the common visual hull representation used in the project was to combine all the 3D structure found from different views into one 3D representation of objects in front of the clipping plane in the scene.

The advantages of using a visual hull representation for refinement of the edges were,

- Pixels added to masks if multiple views agreed on 3D structure (common 3D structure)

The *drawbacks* of visual hull for refinement of the edges were,

- Pixels inside holes caused by camouflage effects were not always added

The problem with holes caused by camouflage not always being filled-up was handled in the next step.

Removal of camouflage holes inside occlusion masks

The *purpose* of this step was to remove holes in the masks of foreground objects caused by camouflage.

The *properties* of this step was to combine the pattern of holes in the silhouettes and test if there were a color similarity between holes which corresponded to the same potential hole in the object.

Holes were divided into four type of holes: *Noise*, *Medium holes*, *Real holes*, and *Unknown holes*. The *Unknown holes* were tested for camouflage effects whereas the rest were determined directly.

The advantages of this step was,

- All holes were classified as either a real hole, or a hole caused by camouflage and removed from the silhouette, or something in between.

The *drawbacks* of this step was,

- Not robust for backgrounds with repeatable or no texture

Visualization of augmented scene

The *purpose* of the visualization step was to combine the virtual objects with images of the real scene through the occlusion masks, in a visually credible way.

The *properties* of the visualization step was to cut away the parts of the virtual objects which were occluded by objects located in front of the virtual objects. The clipped virtual objects were then mixed images of the real scene to produce the final augmented images.

Alpha blending capabilities of 3D hardware was used to enhance the visual quality of the augmented scene.

The *drawbacks* of the visualization step was that lighting parameters of the real scene was not estimated in order to improve the visual quality of the final augmented images.

10.2 The Obtained Results

The results obtained are based on the conducted system test and the component tests. The tests was conducted on a prototype, which had been developed in C++ using OpenGL and The Visualization Toolkit (Inc. (2001)) on a Linux platform.

The result of the background subtraction was a good initial occlusion mask, which was robust toward depth discontinuities and missing 3D geometry in front of the virtual object.

The refinement was not tested on real data but tested for robustness on synthetic scenes. The refinement was tested on component level and was robust toward inaccurate estimated camera matrices up to one pixel noise in the image plane.

10.3 Discussion

In the following possible future improvements of the system are discussed.

10.3.1 3D model

During the project period several problems with the 3D model from the reconstruction system was found. Possible suggestions for improvements include:

- Volumetric 3D model
- Decimation of 3D model

Using only a single depth map leads to problems with occlusion shadows in everything but the simplest scenes. Implementing the volumetric model approach described in section 2.6.2 could solve these problems. The problem about memory consumption could be solved by slicing up the voxel space into smaller sub-spaces which could be processed independently.

The current simple 2D mesh techniques evenly distributes vertices around the scene independently of the complexity of the real structure being reconstructed. Performance improvements could be obtained by the use of decimation algorithms.

10.3.2 Background Subtraction

A number of things could be done to improve the background subtraction process:

- Background model
- Segmentation threshold
- Clipping plane

More advanced background models could be used which are able to model more than one color and also takes variance and depth from the reconstruction system into account could be used (Harville et al. (2001)). Furthermore spatial or temporal coherence using the image warping could be used to minimize misclassification.

Feedback from the occlusion mask refinement process could be used to adjust segmentation thresholds adaptively.

The clipping plane problem could be handled by segmenting the scene into three segments instead of two: In front of virtual object, behind virtual object, and in between. Elements in the third segment could then be handled as a special case and processed further using the visual hull algorithm to determine its exact 3D position.

10.3.3 Visual Hull

In order to improve the visual hull representation additional techniques might be investigated to add structure in extension to the structure found in the representation already. The problem is that there has to be a peak in the overlapping line pieces for adding structure.

Another thing is selection of usable views. Some views are better that others and a more robust representation can be achieved when using the right views in the sequence.

10.3.4 Visualization

The visual quality of the final augmented images could be improved by taking lighting parameters in to consideration. The 3D model of the scene could be used to render shadows from virtual objects on the real scene or the opposite way.

Bibliography

F. J. Aherne, N. A. Thacker, and P. I. Rockett. The Bhattacharyya Metric as an Absolute similarity Measure for Frequency Coded Data. *Kybernetika*, 32(4):001–007, 1997.

J. Berton. Pixar's renderman ®.
WWW: http://www.pixar.com/renderman/artist_tools/tools/renderman.html, 2002.

P. Bourke. Deformable models i.
WWW: http://www.astronomy.swin.edu.au/pbourke/geometry/lineline2d/, April 1989.

K. Cornelis. Snakes and gradient vector flow. ESAT-PSI, K. U. Leuven, August 2000.

K. Cornelis. Resolving Occlusions in Augmented Reality using Background Subtraction. Computer Vision and Graphics project proposal, K.U. Leuven, 2001.

R. Courant and D. Hilbert. *Methods of Mathematical Physics, Volume 1.* wiley, 1989. ISBN 0-471-50447-5.

A. Evans. Four tricks for fast blurring in software and hardware.
WWW: http://www.gamasutra.com/features/20010209/evans_01.htm, February 2001.

O. Faugeras and Q.-T. Luong. *The Geometry of Multiple Images.* the MIT Press, 2001. ISBN 0-262-06220-0.

J. Foley, A. van Dam, S. Feiner, and J. Hughes. *Computer Graphics: Principles and Practice.* Addison Wesley, 1995. ISBN 020-184-840-6.

R. C. Gonzalez and R. E. Woods. *Digital Image Processing.* Addison-Wesley Publishing Company, 3rd edition, 1993. ISBN 0-201-50803-6.

M. Harville, G. Gordon, and J. Woodfill. Foreground Segmentation Using Adaptive Mixture Models in Color and Depth. In *2001 IEEE Workshop on Detection and Recognition of Events in Video*, pages 3–11, July 2001. ISBN 0-7695-1293-3.

K. Inc. The visualization toolkit.
WWW: http://public.kitware.com/VTK/, 2001.

M. Pollefeys. Tutorial on 3D Modeling from Images. Technical report, Industrial Image Processing group (VISICS/PSI/ESAT), K.U. Leuven, 2000.

G. Strang. *Introduction to Linear Algebra.* Wellesley-Cambridge Press, 2nd edition, 1998. ISBN 0-9614088-5-5.

H. H. Thodberg. Deformable models i.
WWW: http://www.imm.dtu.dk/courses/02503/, February 2002. Course in Advanced Image Analysis.

E. Trucco and A. Verri. *Introductory Techniques for 3-D Computer Vision*. Prentice Hall, 1998. ISBN 0-13-261108-2.

E. W. Weisstein. Eric weisstein's world of mathematics.
WWW: http://mathworld.wolfram.com/, 2001.

J. Weng, P. Cohen, and M. Herniou. Camera Calibration with Distortion Model and Accuracy Evaluation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(10):965–980, October 1992.

M. Wojciech, C. Buehler, R. Raskar, L. McMillan, and S. J. Gortler. Image-based visual hull. *SIGGRAPH*, 2000.

C. Xu and J. L. Prince. Snakes, Shapes, and Gradient Vector Flow. *IEEE Transactions on Image Processing*, pages 359–369, March 1998.

APPENDIX A

Projective Geometry

A.1 Introduction

The purpose of computer vision is to create a model of the world based on 2D images. The problem is that it is not possible to determine the 3D position of points in a scene from a single image alone. The solution is to use multiple images and geometric constraints.

The Euclidean geometry is often used in computer vision to describe 3D geometry (our world). It makes measurements in terms of lengths, angles, parallelism, and orthogonality as they are preserved by a change of coordinates which is a *rigid motion* (rotation and translation), *a rigid transformation*.

Projective geometry is an extension of Euclidean geometry, which describes a larger class of transformations than just rotations and translations, including the perspective projection performed by a camera. It makes it possible to describe infinity. An example of this are railroad tracks that are parallel in 3D world space, but in the image they are no longer parallel. They appear to converge as they disappear in the horizon, toward a vanishing point. Beside parallelism is not preserved by projection, neither are distances or angles.

Through calibration of the cameras, it is possible to change from Projective geometry to Euclidean geometry. The result of the calibration is the projective camera matrix.

This appendix is based on (Faugeras and Luong, 2001, chap 1 and 5), (Pollefeys, 2000, chap 3), and (Trucco and Verri, 1998, chap 2)

A.2 Camera Model

The purpose of the camera model is to make a transformation from the 3D world space to the 2D image plane. This is done using a pinhole camera whose components are as follows:

- A plane $\mathcal{R}$, called the *retinal plane* or *image plane*.
- A point C which does not belong to $\mathcal{R}$: The *optical center*.
- The *focal length* f, the distance between the point C and the plane $\mathcal{R}$.

Figure A.1 shows the general pinhole camera model. The *optical axis* is the line going through C and perpendicular to the retinal plane $\mathcal{R}$. It pierces the plane at the *principal point* c. The 2D orthonormal system of coordinates, which lies in the retinal plane, centered at c, is denoted as $\mathcal{F}_c = (c, x_c, y_c)$, and the 3D orthonormal system of the coordinates is called the *camera coordinate system* and is centered at the optical center C with the two axes of coordinates parallel to the retinal ones and the third is parallel to the optical axis $\mathcal{F}_C = (C, X_C, Y_C, Z_C)$. In the coordinate systems, shown in Figure A.1, the relationship between the coordinates of the 3D point M and its projective image m is as following:

$$x_c = f\frac{X_C}{Z_C}, \quad y_c = f\frac{Y_C}{Z_C} \tag{A.1}$$

The projection equations A.1 are not linear, but using homogeneous coordinates for m (x_c, y_c, w) and M (X_C, Y_C, Z_C, W) for $W \neq 0$, they become linear:

$$\begin{bmatrix} Z_C x_c w \\ Z_C y_c w \\ Z_C w \end{bmatrix} = \underbrace{\begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}}_{\mathcal{P}_C} \begin{bmatrix} X_C W \\ Y_C W \\ Z_C W \\ W \end{bmatrix} \tag{A.2}$$

A point in homogeneous coordinates is represented by a vector of $\mathbb{R}^4$, e.g. $M = (\mathrm{X, Y, Z, W})$. The same point in Euclidean coordinates is equal to $m = (\frac{\mathrm{X}}{\mathrm{W}}, \frac{\mathrm{Y}}{\mathrm{W}}, \frac{\mathrm{Z}}{\mathrm{W}})$ represented by a vector of $\mathbb{R}^3$, where $w = \frac{\mathrm{Z}}{\mathrm{W}} = f$.

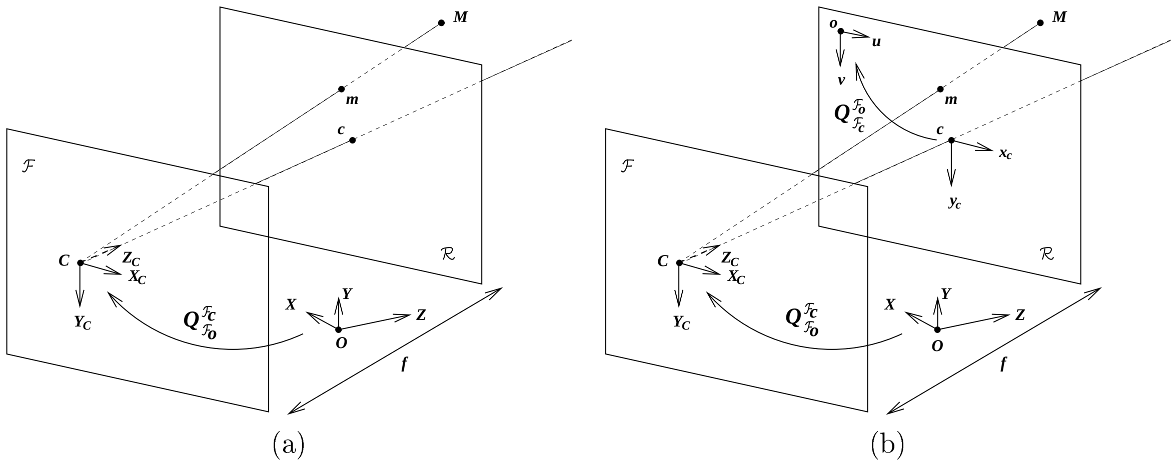


Figure A.1: Figure (a) illustrates the rotation and transformation of the camera ($\mathbf{Q}^{\mathcal{F}_C}_{\mathcal{F}_O}$), that places the camera at the desired position in world space ($\mathcal{F}_O$). Figure (b) illustrates the general perspective or pinhole camera model. The projection ($\mathbf{Q}^{\mathcal{F}_o}_{\mathcal{F}_c}$) from camera space ($\mathcal{F}_c$) to the image space ($\mathcal{F}_o$) is added to Figure (a). Based on (Faugeras and Luong, 2001, p.179).

A.3 Projective Camera Matrix

The general perspective projection matrix expresses the relationship between 3D points and 2D pixels as a morphism which maps points from $\mathbb{R}^4$ to $\mathbb{R}^3$. This morphism can be described by a 3 x 4 matrix $\mathcal{P}$, called the *perspective projection matrix* of the camera:

$$m = \begin{bmatrix} x \\ y \\ w \end{bmatrix} \backsimeq \mathcal{P} \begin{bmatrix} X \\ Y \\ Z \\ W \end{bmatrix} = \mathcal{P}M \tag{A.3}$$

The properties of the projection matrix $\mathcal{P}$:

- The matrix is a 3×4 matrix of rank 3, defined up to a scale factor (noted with $\backsimeq$).
- The null-space is of dimension 1, corresponding to a unique point of $\mathbb{R}^4$, the optical center C.

Considering the general Euclidean world coordinate system (shown in Figure A.1) $\mathcal{F}_O = (O, X, Y, Z)$ which is related to the normalized camera coordinate system $\mathcal{F}_C = (C, X_C, Y_C, Z_C)$ by a rigid motion by the rotation matrix $\mathbf{R}$ and the translation vector $\mathbf{t}$. This change of coordinate system is considered as a change of projective coordinate system and corresponds to the *extrinsic* camera parameters.

The retinal coordinate system $\mathcal{F}_o = (o, u, v)$ in the retinal plane is related to the coordinate system $\mathcal{F}_c = (c, x_c, y_c)$ by a change of scale of magnitude k_u and k_v along the u- and v-axes, respectively, followed by a translation $[u_0,\ v_0]$. This change including the focal length f corresponds to the *Intrinsic* camera parameters.

A.3.1 Extrinsic Camera Parameters

The rigid motion between $\mathcal{F}_O = (O, X, Y, Z)$ and $\mathcal{F}_C = (C, X_C, Y_C, Z_C)$ is represented by the 4x4 matrix

$$\mathbf{Q}_{\mathcal{F}_O}^{\mathcal{F}_C} = \begin{bmatrix} \mathbf{R}^T & -\mathbf{R}^T\mathbf{t} \\ \mathbf{0}_3^T & 1 \end{bmatrix} \tag{A.4}$$

Where the rotation matrix $\mathbf{R}$ represented by a 3x3 matrix describes the rotation of camera $\mathcal{F}_C$ compared to Euclidean world coordinate system $\mathcal{F}_O$, and the translation vector $\mathbf{t}$ represented by a 3x1 vector describes the location of the camera in the Euclidean world coordinate system.

A.3.2 Intrinsic Camera Parameters

The coordinate system $\mathcal{F}_o$ is the coordinate system used to address the pixels in the image which usually is centered at the upper left corner of the image which usually differs from is the point c. In general, the principal point $c = [u_0, v_0]^T$ is relatively close to the image center.

Another issue is the shape of the pixels. They are usually not square and have aspect ratio, $(\frac{k_u}{k_v})$ depending on the actual size of the photosensitive cells of the camera. In cases of the retinal axes are not orthogonal (skew) this has to be taken into consideration. Figure A.2 show the relationship between the two coordinate systems $\mathcal{F}_c$ and $\mathcal{F}_o$.

The two coordinate systems are aligned along the first axis. This gives the following relation $\mathbf{x}_c = k_u\mathbf{u}$ and $k_v\mathbf{v} = \mathbf{x}_c \cos\theta + \mathbf{y}_c \sin\theta$. Therefore, $\mathbf{y}_c = -k_u \cot\theta \mathbf{u} + \frac{k_v}{\sin\theta}\mathbf{v}$

The change of coordinate system between $\mathcal{F}_c$ and $\mathcal{F}_o$ is represented by a 3x3 matrix

$$\mathbf{Q}_{\mathcal{F}_c}^{\mathcal{F}_o} = \begin{bmatrix} k_u & -k_u \cot\theta & u_0 \\ 0 & \frac{k_v}{\sin\theta} & v_0 \\ 0 & 0 & 1 \end{bmatrix} \tag{A.5}$$

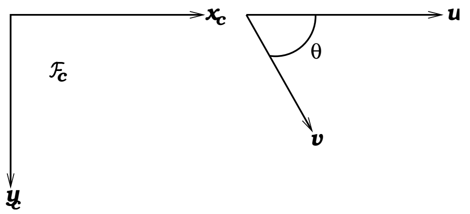


Figure A.2: The relation between the coordinate systems $\mathcal{F}_c$ and $\mathcal{F}_o$, based on (Faugeras and Luong, 2001, p.209).

A.3.3 Projective Camera Matrix

The projective camera matrix expresses the relation between the image coordinates $\mathcal{F}_o$ and the Euclidean world coordinate system $\mathcal{F}_O$. It can be described as a rigid motion, followed by the projection expressed in the normalized camera coordinate system, and finally a change of retinal coordinates:

$$\mathbf{m} \simeq \mathbf{Q}_{\mathcal{F}_c}^{\mathcal{F}_o} \mathcal{P}_{\mathbf{C}} \mathbf{Q}_{\mathcal{F}_O}^{\mathcal{F}_C} \mathbf{M} = \mathcal{P}\mathbf{M} \tag{A.6}$$

The projective camera matrix is represented by the 3x4 matrix

$$\mathcal{P} = \begin{bmatrix} k_u & -k_u \cot\theta & u_0 \\ 0 & \frac{k_v}{\sin\theta} & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{R}^T & -\mathbf{R}^T\mathbf{t} \\ \mathbf{0}_3^T & 1 \end{bmatrix} = \mathbf{K} \begin{bmatrix} \mathbf{R}^T & -\mathbf{R}^T\mathbf{t} \\ \mathbf{0}_3^T & 1 \end{bmatrix} \tag{A.7}$$

A.3.4 Lens Distortion

The projective camera matrix $\mathcal{P}$ describes the camera model without radial distortion added by the lens. There are mainly two different types of distortion (Weng, Cohen, and Herniou, 1992): The first one is caused by imperfect lens shape and coursing positional error only. The second one is caused by improper lens and camera assembly and generate both radial and tangential error in the image plane. The two types of distortion are illustrated in figure A.3 (a).

In the case where the principal point is the center of distortion, there is no tangential distortion. This section only deals with radial distortion.

Radial distortion causes an inward or outward displacement of an image point from its ideal position. A negative radial displacement of the image points is referred to as barrel distortion. It causes outer points to crowd increasingly together and the scale to decrease. A positive radial displacement of the image points is referred to as pincushion. It causes outer points to spread and the scale to increase. Figure A.3 (b) illustrates the effect of radial distortion.

Radial Distortion

Radial distortion can be modeled as a Taylor power series. Using Zhang's notation,

$$\begin{aligned} \tilde{x} &= x + x \sum_{n=1}^{\infty} \kappa_n (x^2 + y^2)^n \\ \tilde{y} &= y + y \sum_{n=1}^{\infty} \kappa_n (x^2 + y^2)^n \end{aligned} \tag{A.8}$$

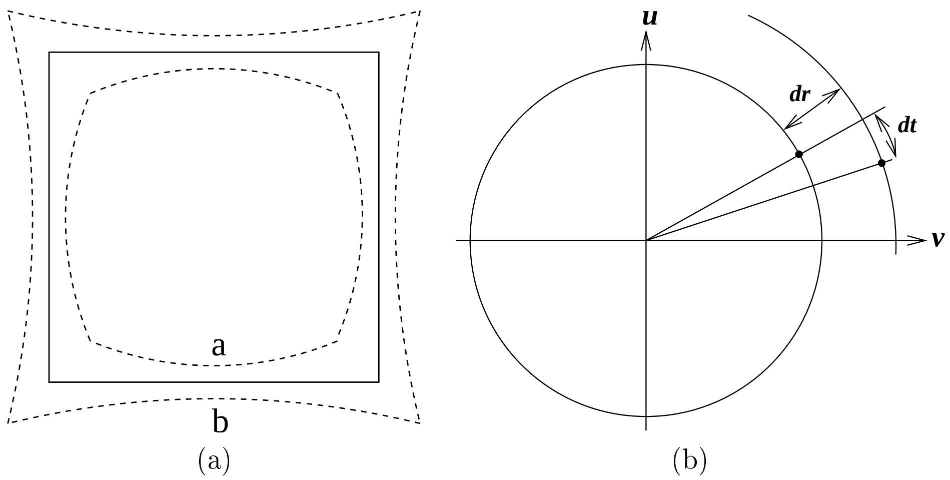


Figure A.3: Figure (a) illustrates the two different type of lens distortion. Radial and tangential distortions which is illustrated with *dr* and *dt*, respectively. Figure (b) illustrates the effect of radial distortion. The solid lines are the undistorted images, and the dashed lines are the distorted images: where (a) appears when the first coefficient κ_1 is negative, and (b) if κ_1 is positive

where $(x,\ y)$ represent the ideal distortion-free image coordinates, and $(\tilde{x},\ \tilde{y})$ are the radially-distorted image coordinates. κ_n is the nth coefficient of radial distortion, and $r^2 = x^2 + y^2$ is the squared distance from the image centrum to the image point .

Radial Distortion Model used at VISICS

The system at VISICS uses three coefficients of radial distortion to model the radial distortion in the images. They remove the distortion in the camera frame instead of the previous mentioned image frame. The difference is, that in the case of working in the camera frame; the image points are centered around the optical axis, where as in the image frame it is the upper left corner. Beside the different centers, there are also a difference in the operating range, and the range for with κ's are found. In the camera frame, the range of the coordinates are between -1 and +1, where as in the image frame it is the resolution of the image that defines the range. Their model is as follows,

$$\begin{aligned} \tilde{x} &= x(1 + \kappa_1 r^2 + \kappa_2 r^4 + \kappa_3 r^6) \\ \tilde{y} &= y(1 + \kappa_1 r^2 + \kappa_2 r^4 + \kappa_3 r^6) \end{aligned} \tag{A.9}$$

They use the following steps to un-distort the images: First they lift the image back into the camera frame using $\mathbf{K}^{-1}$. Then they un-distorts all the coordinates using equation A.10. Finally the coordinates are projected back into the image frame using $\mathbf{K}$.

A.4 The Epipolar Geometry

The epipolar geometry describes the basic constraints which arises from the existence of two viewpoints. These two viewpoints can be obtained either by observing the scene with two different cameras at the same time (stereo-vision) or with a moving camera at two different times (structure from motion).

Consider two images taken by two pinhole cameras (or one, using structure from motion) with distinct optical centers, as shown in figure A.4.

The epipolar geometry takes basic in the *epipolar plane* which contains the points *M, C, and C'* in 3D world space (the plane Π in figure A.4). Properties of the epipolar geometry are

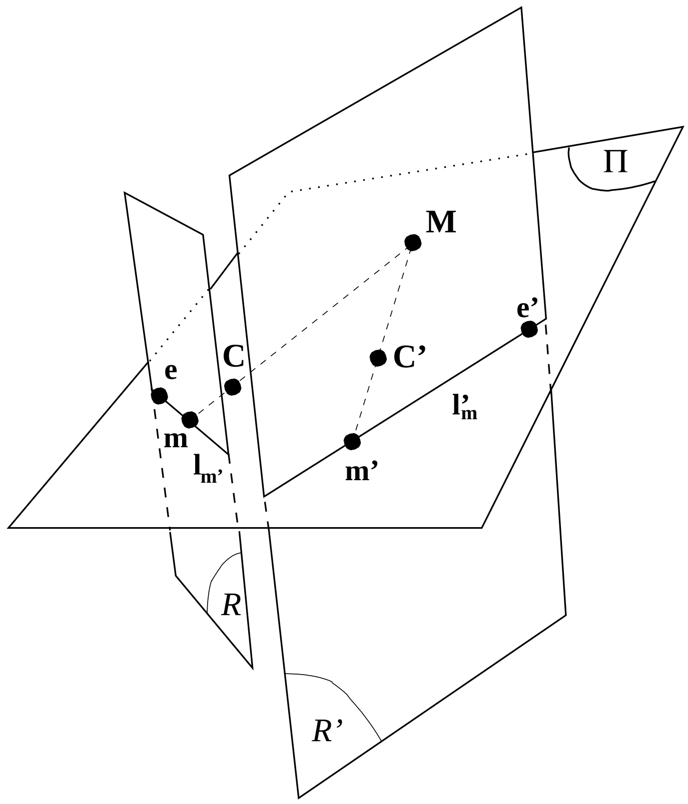


Figure A.4: The epipolar geometry in the general projective case, (Faugeras and Luong, 2001, p.260).

- The epipolar line of an image is the intersection of epipolar plane and the retinal plane of the camera.
- The epipole of an image is the projection of the optical center of an other camera into the retinal plane of the camera.
- The fundamental matrix is the mapping between points and epipolar lines.

A.4.1 The Epipolar line

The epipolar line of an image is the intersection of epipolar plane and the retinal plane of the camera. To elaborate, given an infinite ray from the optical center of the first camera (C) passing through the 3D point M: the projection of this infinite line into the retinal plane of the second camera results in the epipolar line (l'_m) in the second image.

All the 3D point on the infinite ray (containing the points C and M) projects into the same image point m in first camera, but different points in the second image. This property makes it easier to find the same 3D point in two images (first and second camera, respectively), as given the point m in the first image the point m' in the second image lies along the epipolar line l'_m in the second image. Similarly, given point m' in the second image the point m in the fist image lies along the epipolar line $l_{m'}$ in the first image. Reducing it to a 1D search.

A.4.2 The Epipole

Let C and C' be the two optical centers of the first and second camera, respectively. The line between the two cameras optical centers (C,C') intersects the two retinal planes $\mathcal{R}$ and $\mathcal{R}'$ at the points e and e', respectively. They are called *epipoles* and are computed as following:

$$e \backsimeq \mathcal{P}C' \quad \text{,and} \quad e' \backsimeq \mathcal{P}'C \tag{A.10}$$

The epipole is the point where all epipolar lines of the image passes through, as all the infinite ray (C,M') passes through the optical center (C), and is not limited to be inside the image.

A.5 Fundamental Matrix

The fundamental matrix, or F-matrix, is a 3x3 matrix that describes the correspondence between a point and its epipolar line, like $\mathbf{l'}_m \backsimeq \mathbf{Fm}$ in figure A.4. To elaborate, the fundamental matrix gives the parameters of the epipolar line linearly (without the explicit need of projecting two 3D point along the ray into the image and then extracting parameters of the line between these two points).The fundamental matrix operates using pixel coordinates, and a pair of views for which the F-matrix is known is said to be weekly calibrated.

The parameters of the epipolar line is based on two points in the image along the epipolar line. In order to the get the line between two points linearly, the following approach is used:

It is possible to get the parameters of a line between two point by taking the cross-product of the two points as follows:

$$\mathbf{l} \backsimeq v_1 \times v_2 \tag{A.11}$$

The two points along the epipolar line can be arbitrarily selected. As all epipolar lines goes through the epipole (e' in figure A.4), it is picked at the first point (v_1).

$$v_1 = e' \backsimeq \mathcal{P}'C \tag{A.12}$$

The second point (v_2) is a projection of a 3D point along the ray (C,M), m' in figure A.4.

The 3D point M is found from the first camera in homogeneous coordinates as follows,

$$\lambda M = \mathcal{P}^{\dagger} m \tag{A.13}$$

The inverse camera projection matrix of $\mathcal{P}$ (see equation A.7) is any 4×3 matrix, such as,

$$\mathcal{P}\mathcal{P}^{\dagger} \backsimeq I_3 \tag{A.14}$$

The following inverse projection matrix $\mathcal{P}^{\dagger}$ satisfies the constrains given in equation A.14,

$$\mathcal{P}^{\dagger} \backsimeq \begin{bmatrix} \mathbf{RK}^{-1} \\ 0_3^T \end{bmatrix} \tag{A.15}$$

The projection of the 3D point M into the second camera,

$$m' = \mathcal{P}'\lambda M \tag{A.16}$$

The equation for the 3D point M, with $\lambda = 1$, inserted into equation A.16,

$$v_2 = m' = \mathcal{P}'\mathcal{P}^{\dagger}m \tag{A.17}$$

The epipolar line in the second image of the 2D point m in the first image is found as follows,

$$\mathbf{l'}_m \backsimeq v_1 \times v_2 = \underbrace{\left[\mathcal{P}'\,C\right]_{\times}\mathcal{P}'\mathcal{P}^{+}}_{\mathbf{F}}\,\mathbf{m} \tag{A.18}$$

In order to compute the cross-product linear it can be written as a matrix cross-product $v \times u = [v]_{\times}u$ as follows,

$$v \times u = \begin{bmatrix} 0 & -v_3 & v_2 \\ v_3 & 0 & -v_1 \\ -v_2 & v_1 & 0 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} = \begin{bmatrix} v_2u_3 - v_3u_2 \\ v_3u_1 - v_1u_3 \\ v_1u_2 - v_2u_1 \end{bmatrix} \tag{A.19}$$

The fundamental matrix that describes the correspondence between a point in the fist image and its epipolar line in the second image is computed as,

$$\mathbf{F} \backsimeq \left[e'\right]_{\times}\mathcal{P}'\mathcal{P}^{\dagger} = \begin{bmatrix} 0 & -e'_3 & e'_2 \\ e'_3 & 0 & -e'_1 \\ -e'_2 & e'_1 & 0 \end{bmatrix} \mathbf{K} \begin{bmatrix} \mathbf{R}^T & -\mathbf{R}^T\mathbf{t} \\ \mathbf{0}_3^T & 1 \end{bmatrix} \begin{bmatrix} \mathbf{RK}^{-1} \\ 0_3^T \end{bmatrix} \tag{A.20}$$

The fundamental matrix is computed as follows,

$$\mathbf{F} \backsimeq \begin{bmatrix} 0 & -e'_3 & e'_2 \\ e'_3 & 0 & -e'_1 \\ -e'_2 & e'_1 & 0 \end{bmatrix} \mathbf{K}'\mathbf{R}'^T\mathbf{RK}^{-1} \tag{A.21}$$

A.6 Summary

The projective camera matrix expresses the relation between the Euclidean world coordinate system and image coordinates. The properties of the general perspective projection matrix:

- The matrix expresses the relationship between 3D points and 2D pixels as a morphism which maps points from $\mathbb{R}^4$ to $\mathbb{R}^3$.
- The matrix is a 3 x 4 matrix of rank 3, defined up to a scale factor.

- The null-space is of dimension 1, corresponding to the to a unique point of $\mathbb{R}^4$, the optical center C.

The epipolar geometry is the basic constraints which arises from the existence of two viewpoints. The key geometric concepts of the epipolar geometry are:

The epipole is the projection of the optical center C of the second camera onto the retinal plane of the first camera. All the epipolar lines of the retinal plan passes through the epipole of the retinal plane.

The epipolar line is the projective line that for each point m in the first image, the corresponding point m' in the second images lies on the epipolar line l'_m (see figure A.4).

The Fundamental matrix , or F-matrix, is a 3x3 matrix that describes the correspondence between a point and its epipolar line. A pair of views for which the F-matrix is known is said to be weakly calibrated. The properties are:

- The matrix is a 3 x 3 matrix of rank 2, defined up to a scale factor.
- The matrix is build up of the geometric information relating two uncalibrated cameras
- The matrix operates using pixel coordinates, and can be recovered from image measurements alone.

APPENDIX B

Botanical Sequence

The following appendix contains a sequence of 37 images of a single sweep from left to right of a corner of the botanical garden in Leuven, Belgium. The sequence is recorded using a Sony DCR-TRV900E DV camcorder in progressive scan mode. The resolution is 720x576 pixels.

B.1 Input Sequence

Figure B.1 shows every third image of the botanical sequence.

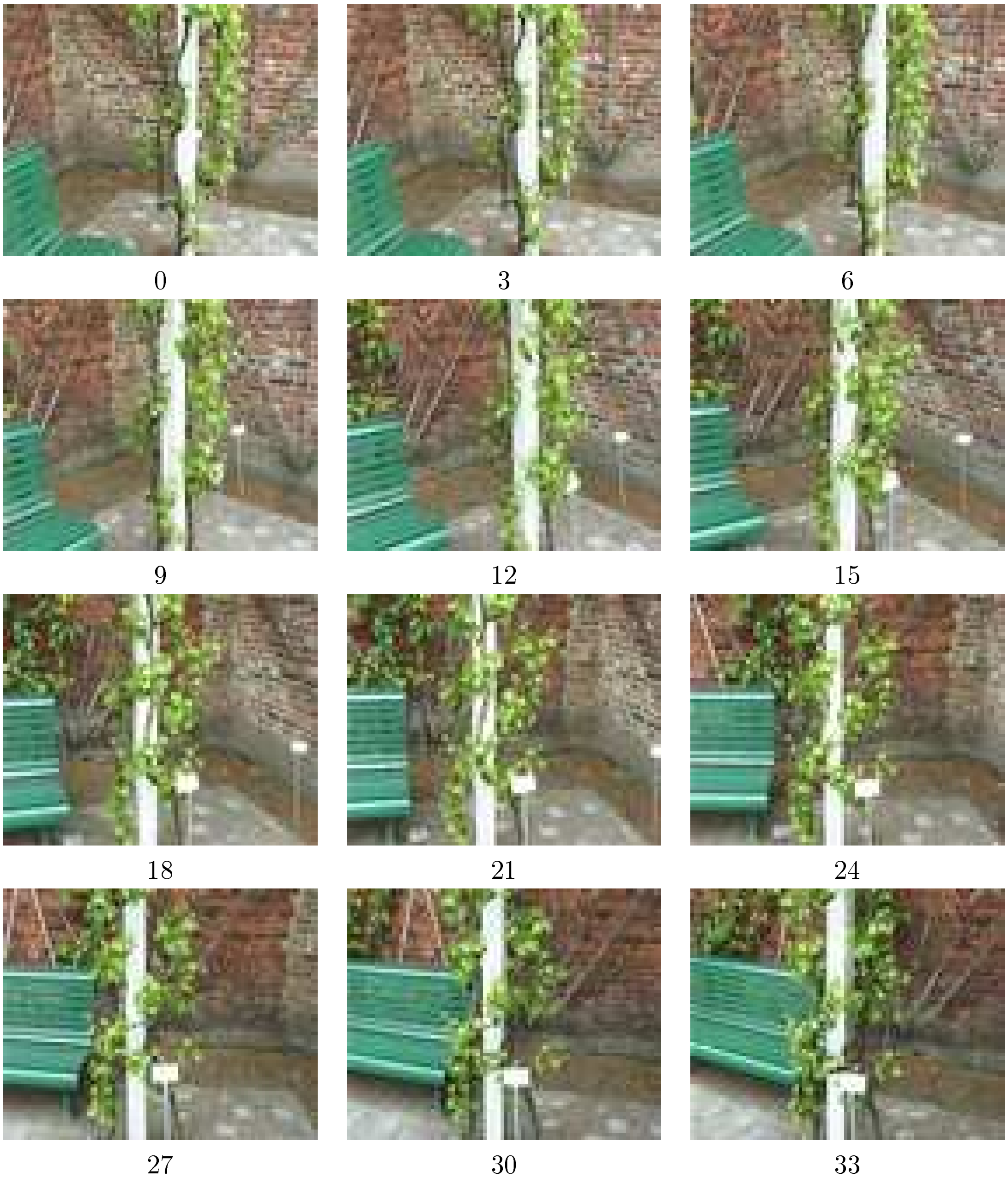


Figure B.1: Every third image of the botanical sequence.

B.2 Reconstruction

Figure B.2 shows every third image of a rendering using the estimated camera parameters and the calculated depth map of image 10. Input image 10 is used as texture. The 3D model consists of 186.206 triangles.

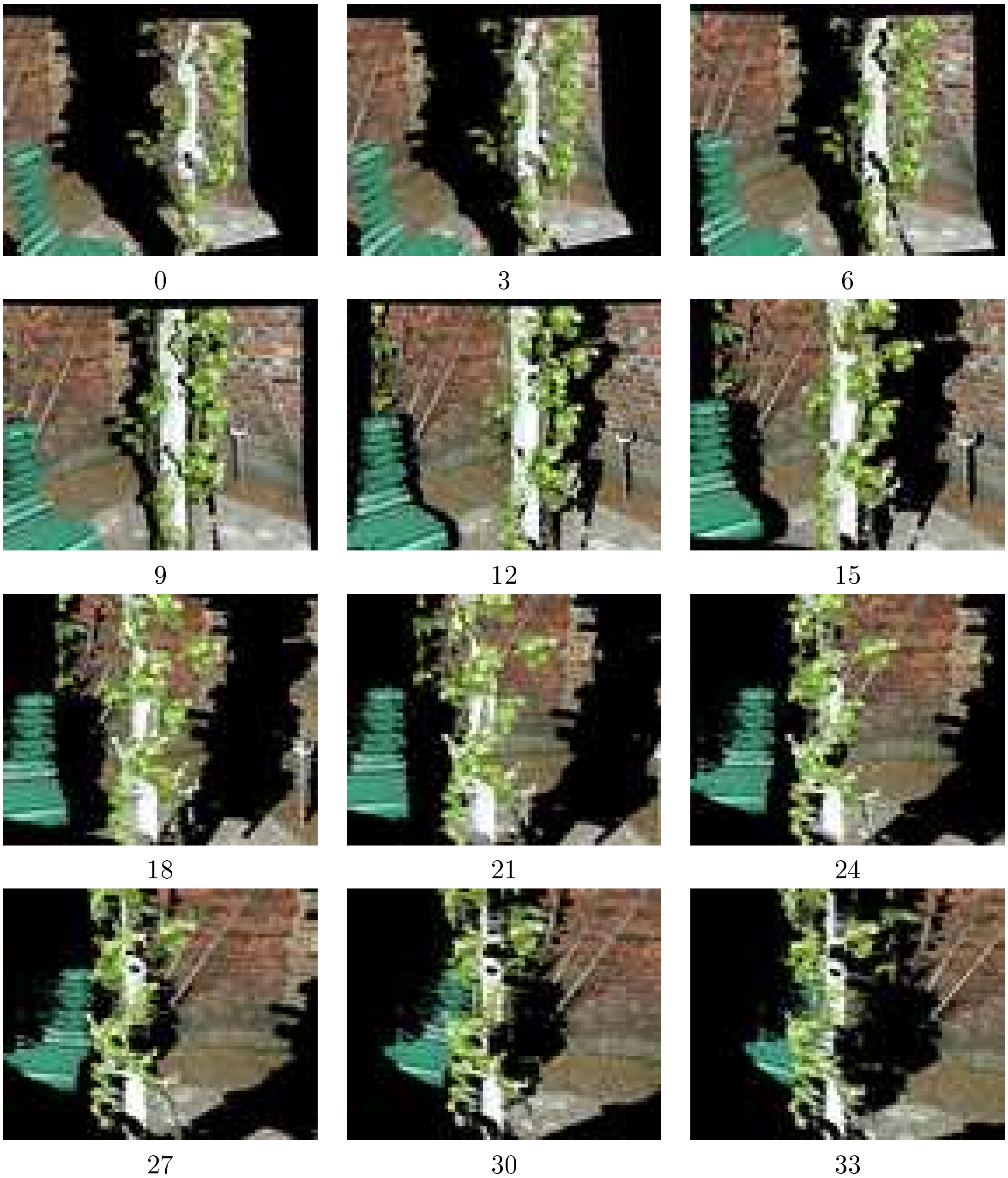


Figure B.2: Every third image of a rendering of the botanical sequence using the depth map of frame 10 and input image 10 as texture.

APPENDIX C

Additional Test Results

The following appendix contains additional test results from the system test for every third image in the botanical sequence (see appendix B). The system test is described in detail in chapter 9.

C.1 Median Background Model

Figure C.1 shows the median background model of every third image of the botanical sequence.

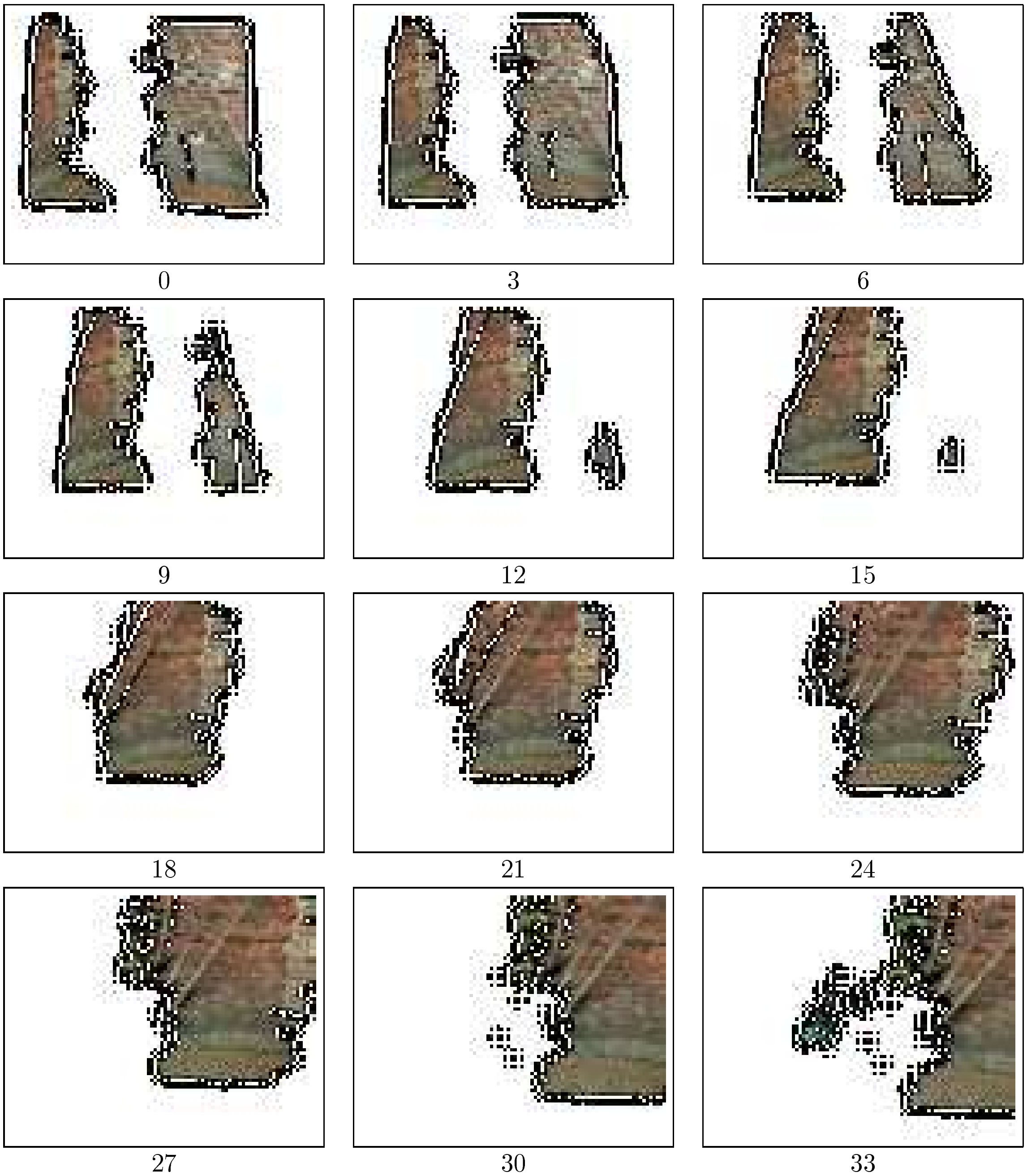


Figure C.1: Median background model of every third image of the botanical sequence.

C.2 Occlusion Masks

Figure C.2 shows the occlusion mask of every third image of the botanical sequence. The occlusion mask is calculated using from the difference image between the median background model and the input image using a threshold value of 1/5.

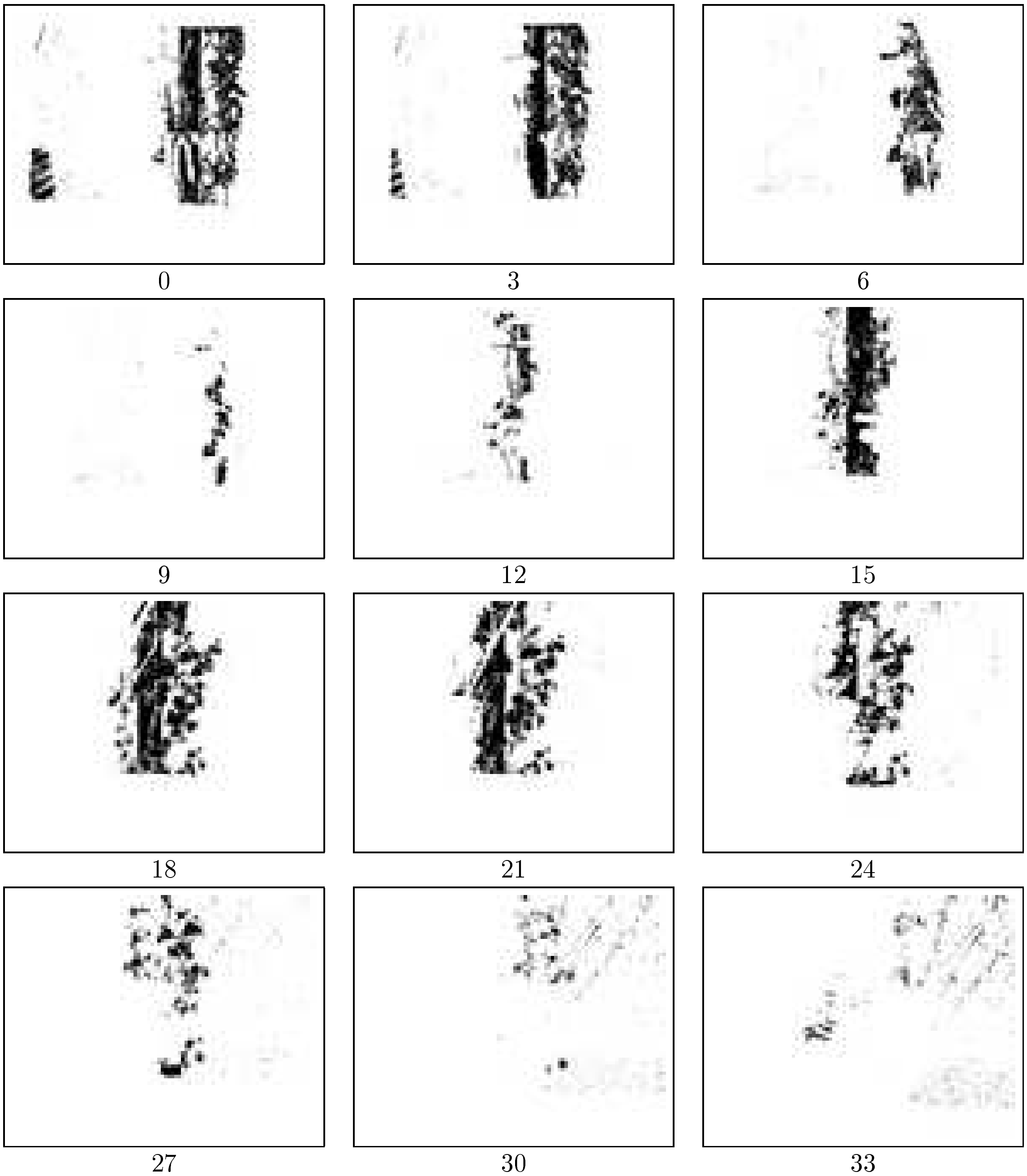


Figure C.2: Occlusion masks of every third image of the botanical sequence using the median background model and a threshold value of 1/5.

C.3 Augmented Images

C.3.1 Depth Map From View 10

Figure C.3 shows every third image of the botanical sequence with a yellow virtual box augmented in the scene using the depth map from view 10 to solve the occlusion problem.

Figure C.3: Every third image of the botanical sequence with a yellow virtual augmented into the scene using the depth map from view 10 to solve the occlusion problem.

C.3.2 Depthmap From Each View

Figure C.4 shows every third image of the botanical sequence with a yellow virtual box augmented in the scene using the depth map from each view to solve the occlusion problem.

Figure C.4: Every third image of the botanical sequence with a yellow virtual augmented into the scene using the depth map from each view to solve the occlusion problem.

C.3.3 Occlusion Mask From Background Subtraction

Figure C.5 shows every third image of the botanical sequence with a yellow virtual box augmented in the scene using the occlusion maps from section C.2 to solve the occlusion problem.

Figure C.5: Every third image of the botanical sequence with a yellow virtual augmented into the scene using the occlusion masks from section C.2 to solve the occlusion problem.