Master Thesis

Pose estimation for face recognition using stereo cameras

Author(s):
Breitenstein, Michael D.

Publication Date:
2006

Permanent Link:
https://doi.org/10.3929/ethz-a-005375328

Rights / License:
In Copyright - Non-Commercial Use Permitted
Master Thesis

Pose Estimation for Face Recognition using Stereo Cameras

Michael D. Breitenstein
Department of Computer Science, ETH Zurich

Sept 2005 - March 2006

Prof. Dr. M. Gross
Computer Graphics Lab.
ETH Zürich

Dr. H. Pfister
Mitsubishi Electric Research Lab., Cambridge Ma.
M.S. Thesis Project for Michael Breitenstein

Pose Estimation for Face Recognition using Stereo Cameras

Introduction

The Computer Graphics Laboratory of ETH Zurich in collaboration with Mitsubishi Electric Research Laboratories (MERL) in Cambridge, MA, have acquired 3D face data from a large cross-section of the population. By applying 3D face models to robust automatic face recognition we are addressing the most critical factors limiting performance: illumination and pose variation. The purpose of this thesis is to implement a 3D pose estimation algorithm that can be integrated into the MERL face recognition system.

Assignment

The main components of the thesis are:

- Development of a two-camera system with real-time depth-from-stereo algorithm using available software (e.g., OpenCV).
- Application of the Viola / Jones face and feature detectors to the stereo images.
- Investigation of the failure modes (e.g., extreme pose) for Viola / Jones feature detection.
- Development of a new algorithm to estimate extreme face pose (up to profile views) based on the depth map and a morphable 3D face model.
- Development of an algorithm to rotate the depth map and face texture into normal (frontal) position by either using image-based rendering or morphable face models.
- (optional) Investigation of methods to deal with occluded face areas, for example, by using face symmetry.
- (optional) System integration of the MERL 2D face recognition framework.
- (optional) Measurement of the face recognition performance of the overall system using ROC curves for a suitably large number of test subjects.

The thesis should result in a complete system, going from stereo cameras to image and information display on the screen, allowing for future development, algorithm testing, and demos.

Remarks

A written report and an oral presentation conclude the work. The semester thesis is overseen by Prof. Markus Gross and is supervised by Dr. Hanspeter Pfister, MERL, and Tim Weyrich, Institute of Computational Science.

Hand-out: September 15, 2005
End-date: March 14, 2006
Abstract

One of the most important criteria of face recognition systems for scenarios in an uncontrolled environment and with uncooperative behavior of persons is pose robustness. The most current and successful systems capable to deal with large pose variation use 3D Morphable Models, which require a time-consuming iterative optimization procedure and hence are not suitable for real-world applications.

In this thesis, we propose a generative approach for face recognition which is robust to extreme pose variation of $180^\circ$ azimuth angle by synthesizing a virtual frontal view. The core step of our system is a fast two-stage pose estimation procedure. We developed an algorithm which performs a brute-force pose space search based on shape matching of a general face model. The algorithm detects a face and estimates its position, size and rough pose by using coarse depth information acquired by passive stereo. This provides a good initial guess which can be refined using Iterative Closest Points.

The rough pose estimation algorithm is implemented exploiting the computational power of state-of-the-art graphics hardware and its potential for massive parallelism. Computations are parallelized using vertex and fragment shaders by off-screen rendering to framebuffer-attachable objects. We evaluate the performance of our algorithm on depth maps acquired in a semi-controlled indoor environment. The achieved results are within a maximal pose estimation error of $15^\circ$. 
Zusammenfassung

Eines der wichtigsten Kriterien für Gesichtserkennungssysteme, um sie in unkontrollierbaren Umgebungen und Szenarien einsetzen zu können, ist ihre Robustheit in Bezug auf eine Drehbewegung des Gesichts. Die derzeit am ehesten geeigneten Systeme verwenden 3D Morphable Models, welche jedoch den Nachteil haben, dass sie eine iterative Optimierung erfordern und deshalb für reelle Anwendungen nur bedingt geeignet sind.

In dieser Arbeit schlagen wir einen Ansatz vor, der robust ist für eine Gesichtsdrehung von bis zu 180° Azimuth, indem virtuelle, frontale Aufnahmen des Gesichts erzeugt werden. Das Kernstück unseres Systems ist ein zweiteiliger Algorithmus, der die Rotation des Gesichts abschätzt. Wir haben dazu eine Methode entwickelt, die ein generisches Modell benutzt und mit Eingabedaten vergleicht, um die optimale Lösung zu finden. Der Algorithmus detektiert dabei das Gesicht und berechnet seine Lage, Größe und ungefähre Drehung unter Ausnutzung von Tiefeninformationen, welche mittels eines passiven Stereo-Verfahrens gewonnen werden. Die dadurch erzeugte, grobe Schätzung kann durch das Iterative Closest Points-Verfahren verfeinert werden.

Acknowledgments

This thesis is the result of a stage at Mitsubishi Electric Research Laboratories in Cambridge, Ma. I’d like to cordially thank to all the people who made this great experience possible.

First, to Prof. M. Gross at ETH for his flexibility and patience. Without his help, this internship and thesis would not have been possible.

My host at MERL, Dr. H. Pfister who showed me the meaning of positive thinking and spent a lot of his valuable time for discussions.

The collaborators in my project, Dr. M. Jones and Dr. P. Beardsley at MERL as well as Prof. S. Rusinkiewicz at Princeton University, for their helpful advice and prospective contributions. Furthermore, Eric Chan and Louis-Philippe Morency from MIT for many discussions about graphics and shader programming, and pose estimation.

Michael Waschbüsch, Urs Breu and Marcel Germann from ETH and Dr. H. Pfister, MERL for correcting my thesis. Furthermore Tim Weyrich, ETH for administrative support concerning the internship and thesis.

My friends from the lab Bernd Bickel, Alexandre Alahi and Chien-i Tu for many invaluable discussions and making my time enjoyable.

Last but not least, my friends and especially Tina, who supported me in time of doubt and never stopped to believe in me.
# Contents

Abstract iii  
Zusammenfassung v  
Acknowledgments vii  

1 Introduction 1  
1.1 Motivation 1  
1.2 Approach 2  
1.3 Contributions 4  
1.4 Overview 4  

2 Background and Related Work 5  
2.1 Face Recognition 5  
2.1.1 A Taxonomy 5  
2.1.2 Eigenspaces 7  
2.1.3 3D Morphable Models 7  
2.2 Pose Estimation 9  
2.2.1 Range Data Registration 10  

3 Face Recognition 13  
3.1 Boosted Local Features 13  
3.2 Failure modes 15  
3.3 Adding Depth Information for Pose Robustness 18  

4 Depth from Stereo Vision 21  
4.1 Epipolar Geometry 22  
4.1.1 Camera System Calibration 22  
4.1.2 Stereo Correspondence 23  
4.1.3 Undistortion 24  
4.2 Depth Estimation 25  
4.3 Validation of Depth Maps 26
5 Pose Estimation
  5.1 Brute Force Pose Space Search ........................................ 32
  5.2 Sum of Squared Differences Error Term ............................... 35
  5.3 Boundary Weighted Error Term ......................................... 38
  5.4 Full Head Model .......................................................... 40
  5.5 Curvature Based Region Weighting ..................................... 41
  5.6 Iterative Closest Points .................................................. 44

6 Implementation with Programmable Graphics Hardware .......... 47
  6.1 Programmable Shaders and Framebuffer-attachable Objects .... 47
  6.2 Encoding Depth Values to Color Textures ............................ 48

7 Results and Discussion ...................................................... 51

8 Conclusions and Future Work ............................................. 67

A Face Recognition Library .................................................. 69

B Stereo Vision Library ....................................................... 71

C Fragment Shader for Error Computation ............................... 75

Bibliography ................................................................. 79
## List of Figures

1.1 Schema of the System ........................................... 3  
2.1 3D Morphable Models for Face Recognition ..................... 9  
2.2 View based Eigenspaces ...................................... 10  
3.1 Rectangle Filters ............................................. 14  
3.2 MERL Face Scanning Dome ................................... 16  
3.3 Failure Modes .................................................. 17  
3.4 Raw Input Image .............................................. 19  
3.5 Raw Rotated Point Cloud ..................................... 20  
4.1 Bumblebee Stereo Camera ..................................... 22  
4.2 Stereo Input Images .......................................... 23  
4.3 Epipolar Model ................................................ 24  
4.4 Undistorted Stereo Input Images ............................... 25  
4.5 Lines Of Sight ................................................ 27  
4.6 Stereo and Raw Disparity Image ................................ 27  
4.7 Validated Disparity Image and Resulting Point Cloud ............ 29  
5.1 Average Face Model .......................................... 33  
5.2 Input Depth Maps and Reference Color Images ................. 34  
5.3 Fitting Result of Correct Fit with Sum of Squared Differences .... 36  
5.4 Error Graph for Correct Fit with Sum of Squared Differences .... 36  
5.5 Fitting Result of Incorrect Fit with Sum of Squared Differences .... 37  
5.6 Error Graph for Incorrect Fit with Sum of Squared Differences .... 37  
5.7 Binary Silhouette and Euclidean distance transform .......... 39  
5.8 Fitting Result with additional Boundary Weighted Error Term ... 40  
5.9 Full Head Model ................................................ 41  
5.10 Fitting Result with Full Head Model .......................... 42  
5.11 Result Graph with Full Head Model ........................... 42  
5.12 Region Textures ............................................... 43  
5.13 Fitting Result with Region Weighting .......................... 45  
5.14 Result Graph with Region Weighting .......................... 45
LIST OF FIGURES

6.1 Framebuffer-attachable Objects ............................... 48
6.2 Depth Value Color Textures .................................. 49
6.3 Error Value Color Texture .................................... 50
6.4 Error Value Rendering ........................................ 50

7.1 Fitting Result for $-90^\circ$ Pose Rotation ....................... 53
7.2 Result Graph for $-90^\circ$ Pose Rotation ....................... 53
7.3 Fitting Result for $-75^\circ$ Pose Rotation ....................... 54
7.4 Result Graph for $-75^\circ$ Pose Rotation ....................... 54
7.5 Fitting Result for $-60^\circ$ Pose Rotation ....................... 55
7.6 Result Graph for $-60^\circ$ Pose Rotation ....................... 55
7.7 Fitting Result for $-45^\circ$ Pose Rotation ....................... 56
7.8 Result Graph for $-45^\circ$ Pose Rotation ....................... 56
7.9 Fitting Result for $-30^\circ$ Pose Rotation ....................... 57
7.10 Result Graph for $-30^\circ$ Pose Rotation ....................... 57
7.11 Fitting Result for $-15^\circ$ Pose Rotation ....................... 58
7.12 Result Graph for $-15^\circ$ Pose Rotation ....................... 58
7.13 Fitting Result for $0^\circ$ Pose Rotation ......................... 59
7.14 Result Graph for $0^\circ$ Pose Rotation ......................... 59
7.15 Fitting Result for $15^\circ$ Pose Rotation ......................... 60
7.16 Result Graph for $15^\circ$ Pose Rotation ......................... 60
7.17 Fitting Result for $30^\circ$ Pose Rotation ......................... 61
7.18 Result Graph for $30^\circ$ Pose Rotation ......................... 61
7.19 Fitting Result for $45^\circ$ Pose Rotation ......................... 62
7.20 Result Graph for $45^\circ$ Pose Rotation ......................... 62
7.21 Fitting Result for $60^\circ$ Pose Rotation ......................... 63
7.22 Result Graph for $60^\circ$ Pose Rotation ......................... 63
7.23 Fitting Result for $75^\circ$ Pose Rotation ......................... 64
7.24 Result Graph for $75^\circ$ Pose Rotation ......................... 64
7.25 Fitting Result for $90^\circ$ Pose Rotation ......................... 65
7.26 Result Graph for $90^\circ$ Pose Rotation ......................... 65

A.1 DiamondClassify Hierarchy ................................. 70
Chapter 1

Introduction

1.1 Motivation

The task of automatic face recognition has received considerable attention from the computer vision community over the past decades. One of the driving forces behind this research in the overlapping area of computer vision, machine learning and computer graphics is the wide range of commercial and law enforcement applications related to it. Moreover, the human capability of recognizing faces under variable viewing conditions such as light and pose variation, occlusions, and the presence or absence of facial features like glasses or beards is remarkable, and keeps on attracting the attention of many researchers from different fields such as computer science, neurobiology and psychology.

In the last few years, a certain level of maturity of the algorithms has been reached and a number of methods have been developed, substantially driven by an increasing commercial interest. Most of the successful systems of these days are designed to solve specific tasks for applications in surveillance or human-computer interaction. In large, independent US government tests (e.g. in FERET [Phillips 00] and FRVT [Phillips 03]), the leading algorithms have been compared and their performance as well as their limitations demonstrated.

Current systems are typically limited to a very specific task and a controlled environment; they lack of generality which is needed for real world applications. This includes primarily robustness against variance in size, position, pose and illumination of the face appearance in the acquired image. While many appearance and feature-based approaches have been developed which are capable to solve some of these issues, most of them typically fail when pose and illumination variation occur (see chapter 2).

One recent trend to overcome such limitations is to exploit the 3D information of human faces, which can be obtained directly from range scanners [Chang 03], esti-
mated from one single image [Blanz 03], or from multiple images [Georghiades 00]. The 3D information can be used either by using 3D directly as a pose and illumination independent signature [Blanz 03], or in a generative approach by generating synthetic images under desired viewing conditions [Lee 05], [Georghiades 00], [Blanz 05]. In both cases, a 3D Morphable Model is learned. Although major improvements concerning pose robustness have been shown, this technique requires an iterative nonlinear optimization procedure and careful manual initialization of facial landmarks. Hence, it is time-consuming and not adequate for automatic real-time applications.

The recent development and availability of accurate and low-cost range image acquisition devices opens new possibilities. In addition to intensity images, depth information can be acquired and used to address the problems of position, scale and pose variation for object detection and recognition, without obtaining and learning a full 3D model. This thesis investigates the possibilities to exploit this new potential and proposes a system for a typical real-world scenario like a public gate where people are not expected to behave cooperatively and hence large variation in appearance has to be considered.

1.2 Approach

We investigate the exploitation of coarse depth information for face recognition and propose a system which is fast and robust for a pose range from frontal to profile views. The schema of the complete system is visible in figure 1.1. In this thesis, we concentrate on the crucial step of pose estimation which is a topic of ongoing research itself. We developed and implemented a new algorithm for detecting a face and estimating its pose without initial guess. These stages of main concern in this thesis are shaded dark in figure 1.1. Besides, we investigated state-of-the-art algorithms for solving the slightly shaded steps in figure 1.1. They are provided by existing libraries, but are not yet integrated in our framework. Moreover, we suggest methods to realize the implementation of the unshaded stages in figure 1.1.

Currently, most systems using depth information acquire depth by active stereo techniques (e.g. Structured Light) which drastically restricts the range of application for real-world scenario. Therefore, we acquire a 2.5 D point cloud by passive stereo using a stereo camera and validation methods. A face is detected and its pose estimated in a two-stage procedure. The exact problem we are solving can be stated as follows: the target is to find an algorithm capable of estimating the pose of a specific object from a known object class in a 2.5D point cloud which is possibly afflicted with noise, holes and occlusions, to an accuracy such that a face recognition algorithm reliably works on a generated input image of the rotated object based on this estimation. Therefore, a range data registration algorithm is developed which performs coarse
1.2 Approach

Figure 1.1: Schema of the system
The complete schema of our system and the algorithms for the realization of the single stages. We concentrate in this work mainly on the shaded steps.

alignment of a generic face model for rough pose estimation by a brute force pose space search. This initial guess from our algorithm can be used for the refinement of the pose estimation by the Iterative Closest Points algorithm (if necessary). To realize the algorithm, we extensively exploit the potential and computational power of state-of-the-art graphics hardware which offers massive parallelism. Therefore, programmable vertex and fragment shaders as well as framebuffer-attachable objects are used for offscreen rendering and to achieve the required performance.

The resulting knowledge about size, position and exact face pose of the face can be used for the rotation of the detected face. A synthetic frontal face image can be produced by either directly rendering the rotated point cloud or applying image-based rendering techniques. If necessary, those parts of the face where no information was available in the original view can be completed under the assumption of symmetry, which has been shown to be feasible in psychology. Further image correction like illumination normalization could be performed to adapt to specific conditions in a typical outdoor environment.

Finally, a feature-based 2D face recognition algorithm consisting of boosted local features can be used, which proved to be a reliable and fast method for frontal-pose situations [Jones 03].
1.3 Contributions

The idea to synthesize virtual views for the improvement of face recognition algorithms has recently been suggested (see section 2.1.3). However, former methods generally are not capable of dealing with extreme pose variation or are not applicable to real-world scenarios because of limitations such as time-consuming optimization procedures or complex depth acquisition systems. Key points of this work are:

**Acquisition of depth information**: Most former attempts use active stereo methods like Structured Light or rotating laser scanning systems. We acquire depth directly from passive stereo, without additional constraints or limitations. This has been becoming possible recently because of improvements of the camera systems and stereo algorithms, and the increased computational power of computer hardware.

**Pose estimation**: Most attempts for pose estimation can be classified in feature-based or appearance-based approaches. Methods from the first class generally have the problem that pose-invariant features valid for pose variation are hard to find, whereas methods from the second class require precise alignment and are usually not robust to illumination variance. In contrast to such approaches, our attempt is purely based on the exploitation of depth information and hence not affected by these problems.

**Face recognition**: A real-time 2D face recognition system already exists at MERL, which is very reliable for frontal pose faces. We investigate its failure modes to motivate our idea for synthetically producing frontal images of the inputs and hence providing a possibility to enhance such existing algorithms.

**Implementation**: We show how computationally expensive and hence not yet possible algorithms can be implemented on the GPU\(^1\), exploiting the potential for massive parallelism and new mechanisms like framebuffer-attachable objects and programmable shaders for off-screen rendering.

11

11Graphics Processing Unit, in contrast to the Central Processing Unit (CPU)

1.4 Overview

In chapter 2, we give a short historical overview about face recognition and compare the most popular algorithms to motivate our approach by highlighting their fundamental problems. Since this thesis mainly deals with pose estimation, recent work in range data registration is included. The necessary theoretical concepts for the different parts of this thesis are introduced in short sections of the corresponding chapters. The limitations of a state-of-the-art 2D face recognition algorithm are investigated in chapter 3. Chapter 4 describes the depth acquisition procedure, and the development of the pose estimation method is presented in chapter 5. Chapter 6 outlines the implementation of the algorithm with programmable graphics hardware. The final results are shown and analyzed in chapter 7. We conclude by mentioning open issues and possibilities for further improvement of the system in chapter 8.
Chapter 2

Background and Related Work

During this project, the focus evolved from face recognition in general to one of the most essential parts of our system, pose estimation. In this chapter, a short overview over the work in these ongoing research fields is given, and the problems of the most recent and promising methods are shown.

2.1 Face Recognition

A general statement of the problem of machine recognition of faces can be formulated as follows [Zhao 03]: given still or video images of a scene, identify or verify one or more persons in the scene using a stored database of faces. Available collateral information such as race, age, gender, facial expression, or speech may be used in narrowing the search and enhancing recognition. The solution to the problem involves:

1. segmentation of faces from cluttered scenes (detection)
2. feature extraction from the face regions
3. identification or verification of the person (recognition)

In identification problems, the input to the system is an unknown face, and the system reports back the determined identity from a database of known individuals, whereas in verification problems, the system needs to confirm or reject the claimed identity of the input face.

2.1.1 A Taxonomy

To give a short overview over the development in face recognition, we first introduce the taxonomy proposed in [Zhao 03] and then outline different methods in a short
Chapter 2. Background and Related Work

historical survey. For the particular task of recognition from still images (in contrast to video-based approaches), three main categories can be distinguished: 1

**Appearance-based or holistic matching methods** The visual context of the complete face region is used as input. A low-dimensional feature vector is extracted and compared to stored examples. Typical examples are the Eigenspace method [Turk 91], [Moghaddam 97], Fisherfaces [Belhumeur 97] and ICA-based representations [Bartlett 98].

**Feature-based or structural matching methods** *Local features* such as eyes, nose or mouth are extracted, and their locations and local statistics are used as descriptors to build a feature vector which is fed into a structural classifier. A typical example is the Elastic Bunch Graph Matching system [Wiskott 97], which uses wavelets to encode local appearance.

**Hybrid methods** Motivated by the human perception system and supported by different studies in psychology, both local and global descriptors together are used for recognition. Prominent examples are the Modular Eigenspace approach [Pentland 94] and the Flexible Appearance Model [Lanitis 95].

Computer vision research in face recognition goes back to the 1960s, where systems relied on the geometry of manually extracted points like corners of eye, nose and mouth as well as their spatial relationships such as angles and length ratios [Bledsoe 66]. The first fully automatic system was developed in the 1970s [Kanade 73]. However, researchers were often disappointed by the low recognition rates of systems based on this *feature-based paradigm*. In the 1980s, experiments with raw 2D images were made, making use of the appearance (or texture) of facial images. Among others, several systems with layered neural networks were developed based on this new paradigm, profiting from advances in connectionist models [O’Toole 88], [Flemming 90]. In the 1990s, the ground-breaking work with the Karhunen-Loève Transform of faces led to the Principal Component Analysis based Eigenspace method which has been one of the major driving forces behind face representation, detection, and recognition [Kirby 90], [Turk 91] and is still a commonly used standard technique for dimensionality reduction and feature extraction. In a comparative study [Brunelli 93], *appearance-based* techniques proved to be significantly superior, which mostly settled the debate on features vs. appearance. Feature-based methods proved to be less sensitive to variations in illumination and viewpoint and to inaccuracy in face localization and alignment, but were often not accurate and assumed geometric or textural models.

In this work we concentrate on still image approaches as opposed to video-based methods which incorporate information from inter-image differences in addition to intra-image variance.
In the current decade, generative approaches have been developed for pose invariant recognition, whereas virtual views are synthesized under desired viewing conditions. The most prominent methods are using 3D Morphable Models introduced in [Blanz 99] (see 2.1.3). Furthermore, the feature-based approach was rediscovered and a very successful detector based on a boosted cascade of weak learners presented in [Viola 01], [Viola 03] was extended to a frontal-pose recognition algorithm [Jones 03] (see chapter 3).

2.1.2 Eigenspaces

The standard method of the appearance-based approach for 2D face recognition is the Eigenspace method.\(^2\) Statistical redundancies within natural images are reduced and the output is decorrelated with the Principal Component Analysis (PCA) [Kirby 90]. Sample vectors \( \mathbf{x} \) can be expressed as linear combinations of the orthogonal basis

\[
\Phi_i : \mathbf{x} = \sum_{i=1}^{n} a_i \Phi_i \approx \sum_{i=1}^{m} a_i \Phi_i
\]

(typically \( m \ll n \)) by solving the eigenproblem

\[
C \Phi = \Phi \Delta
\]

where \( C \) is the covariance metric for input \( \mathbf{x} \).

An advantage of using such representations is their reduced sensitivity to noise. However, a very important disadvantage is the necessary alignment and cropping of the images for the learning and testing procedure. Additionally, the method is not robust to size, position, appearance, facial expression and illumination variance.

[Pentland 94] showed how the problem of pose change can be solved with view-dependent eigenspaces where the feature vector is compared to several databases, each representing a certain pose (see section 2.2). The disadvantage is the need for many different views per person in the database.

2.1.3 3D Morphable Models

The majority of newer methods for pose invariant recognition are generative approaches, where virtual views of a learned 3D model are synthesized for desired viewing conditions. The most prominent and successful methods use 3D Morphable Models which represent face-specific information extracted from a dataset of 3D scans. The Morphable Model (introduced in [Blanz 99]) is a vector space of 3D shapes and

\(^2\)Analogously, the term Eigenface method is used for the decomposition and representation of faces.
textures that is spanned by a dataset of examples and captures its variations. The shape and texture vectors

\[ S = \sum_{i=1}^{m} \alpha_i S_i \quad T = \sum_{i=1}^{m} b_i T_i \]  

(2.3)

are defined such that any linear combination of examples is a realistic face, given that \( S \) and \( T \) are within a few standard deviations from their average. Each vector \( S_i \) stores the 3D shape of a high-resolution 3D mesh, and textures \( T_i \) contain their RGB values. After establishing dense point-to-point correspondence of all scans with a reference scan, PCA is performed to estimate the probability distribution of the faces around the averages \( \bar{s} \) and \( \bar{t} \), and the basis vectors \( S_i \) and \( T_i \) are replaced by orthogonal eigenvectors \( s_i \), \( t_i \) in

\[ S = \bar{s} + \sum_{i=1}^{m-1} \alpha_i s_i \quad T = \bar{t} + \sum_{i=1}^{m-1} \beta_i t_i \]  

(2.4)

During the optimization procedure, a fitting algorithm minimizes the difference between the rendered projection of the model \( M(\alpha, \beta) \) to an input image \( I \) with respect to the model parameters \( \alpha_i \), \( \beta_i \). Furthermore, the head orientation, position and illumination is estimated.

As illustrated in figure 2.1, the 3D Morphable Model can be incorporated in two different approaches for non-frontal face recognition from 2D still images: it can serve as a preprocessing step by estimating the 3D shape of novel faces from the non-frontal input images, and generating frontal views of the reconstructed faces at a standard illumination using computer graphics [Lee 05], [Blanz 05]. The recognition step is performed by a conventional frontal pose 2D face recognition algorithm. In the second method, face recognition is performed directly based on the estimated model coefficients by comparing them to the parameters of the saved gallery images [Blanz 03].

Both methods are reliable (as shown e.g. in the Face Recognition Vendor Test FRVT 2002 [Phillips 03]) but require careful manual initialization and are slow because an iterative nonlinear optimization procedure is needed for fitting the model in an analysis-by-synthesis loop. [Blanz 05] reported a convergence time of about 4.5 minutes on an Intel Pentium 4 2 GHz processor. [Fransens 05] achieved a performance of 35 seconds on a Intel Pentium 4 2.6 GHz processor using a similar method and a pair of calibrated stereo cameras.

The major challenge in face recognition hence remains to develop a system that performs pose invariant recognition without additional constraints about size or position of the object. Additionally, the implementation should be fast enough for real-world scenarios, and hence algorithms without learning of a 3D model are necessary.
2.2 Pose Estimation

3D human head pose is an important cue for scene interpretation. To determine the pose of a head in an image, one must first determine the position of the head in the image. Therefore, the pose estimation task is a chicken-and-egg problem: we must know where the face is to determine its pose, but in general it’s necessary to know the head pose to find the face. A brute force approach is to construct a face detection module for each possible pose, and scan exhaustively for each position, scale and pose. Among other disadvantages, this approach is computationally expensive and needs a lot of training data labelled with head pose angles.

Different methods have been proposed for pose estimation from a single 2D view, which use either feature-based or appearance-based approaches (see 2.1.1). Among them, the most popular methods are color and coarse template matching [Birchfeld 96], pattern classifiers [Ng 02], or graph matching techniques [Lades 93]. A often used approach is based on view-based eigenspaces of intensity images using PCA [Pentland 94]. In [Morency 03], this approach is extended to incorporate depth.

Figure 2.1: 3D Morphable Models for Face Recognition
On the left side, the face recognition system based on estimating the 3D shape from a non-frontal input image and generating a transformed frontal view. On the right, the system for recognition based on the estimated model parameters. [Blanz 05]
Chapter 2. Background and Related Work

Figure 2.2: View based Eigenspaces
The first three intensity eigenvectors (rows) partially displayed for 7 horizontal views (columns) [Morency 03].

information in the eigenspaces. When presented with intensity or depth images of a subject in an unknown pose, these systems find the view with minimal reconstruction error. However, the registration of the training images has to be extremely precise, and faces in the test images have to be detected and cropped accurately. That is normally not possible, especially when the illumination conditions vary.

Methods that exploit a 3D model are generally more accurate, since 3D representations model the appearance of objects more closely. These approaches maintain the 3D structure of the subject in a state vector which is updated as images are observed. The updates require that correspondences between features in the model and features in the image are known. Yet, it is not an easy task to compute these correspondences accurately.

2.2.1 Range Data Registration

In this work, we develop and use methods for pose estimation based on range data registration. Given as input two shapes, in our case the model and the data, each in its own coordinate system, the goal of registration is to find a rigid transform that optimally positions (or registers) that data with respect to the model. This problem, which is a fundamental problem in shape acquisition and modeling, is particularly hard when no information is available about the initial position of the model and data shapes, the inputs contain noise, or the shapes overlap only over parts of their extent.

Most range data registration approaches fall into two general classes. First, voting methods exhaustively search for the small number of parameters needed to specify the optimal transform, making use of the fact that rigid transform is low-dimensional. Although voting methods are guaranteed to find the optimal alignment between data and model shapes, and are independent of the initial pose of the input shapes, they are computationally extensive and hence not often used.
The most successful registration algorithms belong to the second class and use an iterative approach that ideally converges to the best multiple data set registration. However, many algorithms only converge to a good solution if the initial relative pose estimate is sufficiently close to the optimal registration. So, a very important issue is to find a good initial relative pose estimate. To solve this problem, we propose a new method based on shape matching (see chapter 5).

Among the registration algorithms, a popular approach for aligning two point clouds is the *Iterative Closest Point (ICP)* algorithm [Besl 92], [Chen 91], which tries to solve the underlying correspondence problem (see section 5.6). It has been improved by using *geometric descriptors* [Gelfand 05], which is a quantity computed for each point of the model and the data, based on the shape of the local neighborhood around the point. Points whose descriptors are similar potentially correspond. *High-dimensional descriptors* such as spin images [Johnson 98] and shape contents [Belongie 02] provide a fairly detailed description of the shape around each point in a transformation-independent manner. Thus, only a few points in the model shape will have similar descriptors and they can therefore be directly used to solve the correspondence problem or to recognize objects. On the other hand, *Low-dimensional descriptors* usually compute only a few values per point. Examples include curvature such as shape index [Koenderink 90] or a volume integral descriptor [Gelfand 05]. They are typically easier to compute and compare, but many points of the model may have the same descriptor values and hence the assignment ambiguous.
Chapter 2. Background and Related Work
Chapter 3

Face Recognition

Face recognition for frontal pose images has been a research area for several decades and many successful algorithms have been presented. Among the most successful ones, a feature-based method using boosted local features has been developed [Jones 03] and implemented at MERL [FaceRecognition].

This work can be regarded as an extension and enhancement of the above-mentioned system, as its target is mainly to expand the operable pose range from several degrees to a variance of frontal view to profile view.

In this chapter, we present the algorithm which can be used in the final recognition stage of our system and show the results of a case study to investigate its failure modes. It clearly points out the need for more flexible systems which are applicable to real-world scenarios where frontal pose situations cannot be assumed. Subsequently, we outline our idea to overcome these limitations. Informations about the used library can be found in appendix A.

3.1 Boosted Local Features

The general problem of face recognition assumes to have a gallery of face images with known identities and consists of two related problems: in recognition, the system has to decide whether a presented probe face belongs to a face in the gallery, and in verification it has to determine whether a presented probe face belongs to an alleged identity. Both problems can be solved with a similarity function \( F \), which takes two cropped and rectified face images as input and outputs a similarity measure. For a verification system, the output is thresholded.

Jones and Viola [Jones 03] define this face similarity function \( F \) as a linear combination of Rectangle Features \( f_i \), each consisting of a rectangle filter \( \phi_i \), which acts on
Figure 3.1: Rectangle Filters
The five basic types of trained and used rectangle filters. [Jones 03]

Both input images $I_1$ and $I_2$.

$$F(I_1, I_2) = \sum_{i=1}^{N} f_i(I_1, I_2)$$  \hspace{1cm} (3.1)

$$f_i(I_1, I_2) = \begin{cases} 
\alpha & \text{if } |\phi_i(I_1) - \phi_i(I_2)| > t_i \\
\beta & \text{otherwise}
\end{cases} \hspace{1cm} (3.2)$$

with $t_i \in \mathbb{R}$ being a feature threshold.

A set of linear rectangle filters is constructed, which are reminiscent of Haar basis functions (so called because they are computed similarly to the coefficients in Haar wavelet transforms). In [Viola 01], [Viola 03], a subset has been used to successfully construct a face detector. The filters are computed by summing the intensities of all pixels in the dark regions and subtracting the sum of the intensities of all pixels in the light regions. If the number of dark and light pixels is unequal, a multiplier guarantees that they have the same weight. A set of chosen basic filters is shown in figure 3.1. The complete set ranges over all scales, aspect ratios and locations within the search window.

The computation of rectangle filters can be accelerated by first computing an intermediate image representation called integral image [Viola 01]. With this representation, the computation of a rectangle filter can be performed in constant time, i.e. is independent of the number of pixels.

Each feature measures a particular property at a given location, scale and aspect ratio. The filters therefore pick up characteristic regions like lines, edges, or shaded regions and the threshold determines which variations are acceptable.
3.2 Failure modes

A statistical learning algorithm picks the best set of features, thresholds and weights such that the difference between intrapersonal and extrapersonal variation is maximized and same faces are reliably distinguished from different ones. Therefore, an improved version of AdaBoost was developed, which includes confidence-rated predictions.

AdaBoost proceeds in rounds. On each round the algorithm assigns weights $D_i$ to the training examples $x_i = \{I_a, I_b\}$, and a weak classifier $\{\phi_j, t_j\}$ is chosen such that the error

$$
\epsilon_j = \sum_{i: y_i = +1 \land b(x_i) \leq t} D_i + \sum_{i: y_i = -1 \land b(x_i) > t} D_i
$$

is minimized. The first term is the sum of the weights $D_i$ for examples which are false negatives of $f(x_i)$ and the second term for false positives, respectively. Furthermore, a resampling procedure is introduced such that the classifier is not biased by the huge amount of negative matching face pairs compared to the positive examples in the gallery.

In [Jones 03] the algorithm has been tested with two different datasets, each containing a vast number of frontal pose faces, and has shown to be comparable to the best published recognition results.

3.2 Failure modes

In a case study, we tested the robustness of our state-of-the-art 2D face recognition system described above with respect to pose variation. The recognition algorithm was applied to an input image set consisting of 16 different views of 27 persons. The images were taken in the face-scanning dome developed at MERL, which originally was built to capture the 3D geometry as well as the texture of faces (see figure 3.2). We used the pictures of the 16 cameras taken with an illumination of 75 LED light sources to simulate realistic illumination conditions.

The pose images of one person are shown in figure 3.3. The angles of rotation for elevation $\theta$ and azimuth $\phi$ are indicated in table 3.1. The rotation is specified from a person’s point of view, i.e. $\theta$ is positive when the person is looking upwards and $\phi$ when the head is turned to the left, respectively.

We computed the recognition rates as the percentage of positive matched identities compared to the number of total input images. The face recognition algorithm performs well on pictures 6 and 7 and still gets acceptable results on pictures 15 and 16 at recognition rates of above 95% and around 80% (see figure 3.3 for exact values). When more face rotation occurs, the rate drops drastically.
Figure 3.2: Scanning Dome
The face-scanning dome consists of 16 digital cameras, 150 LED light sources, and a commercial 3D face-scanning system.

Table 3.1: Angles for Images from the Dome
The angles $(\theta, \phi)$ (in degree) for every pose of the images in fig. 3.3. A positive $\theta$ indicates that the person is looking upwards, positive $\phi$ turning the head to the left.
Figure 3.3: Failure Modes
One image set taken from the 16 different viewpoints indicated in table 3.1. The percentage of positive matches compared to the number of all inputs is stated.
Chapter 3. Face Recognition

The azimuth of $18^\circ$ in pictures 6 and 7 does not seem to cause any troubles, but neither more rotation ($55^\circ$ in pictures 5 and 8) nor combined rotation above a certain level (pictures 2, 3, 10 and 11) is tolerated. However, it is remarkable that the performance is still high on pictures 15 and 16 where a rotation in the elevation of $33^\circ$ and $59^\circ$ is present. Since the faces in those pictures where the algorithm is successful show a considerable amount of symmetry (both eyes are clearly visible in the input images 6, 7, 15 and 16), we can conclude that the symmetric features (see figure 3.1) play an important role in the trained recognition system.

3.3 Adding Depth Information for Pose Robustness

For humans, it is generally not problematic to recognize a pose rotated face. However, it has been proven to be hard to find rotation-invariant features for artificial systems, since different facial features might be of importance for different poses. Other methods like appearance-based approaches suffer from other drawbacks like the need for very accurate registration, as discussed in 2.1.

Our approach is to add an essential component of the human visual system to the artificial system in order to achieve pose robustness: the acquisition and exploitation of depth information. When presented only with texture or color information, the spatial information has to be deduced, and hence visual perception gets much harder. For example, it is easier for us to recognize objects when 3D information is available, since the shape is an important cue. Consequently, our target is to build a recognition system which is capable to acquire and use depth information. For a sketch of the system, see figure 1.1.

However, we don’t use depth information directly to perform recognition. It is rather used to detect the object and determine its spatial location and attributes, such as position, size and pose. Our target is then to synthesize a virtual frontal view such that the frontal pose face recognition algorithm described in section 3.1 can be used.

Therefore, a pose estimation algorithm is necessary which operates without any knowledge about the initial pose of the face. Since this has been an unsolved issue in the computer vision community, we concentrate on this crucial step of our system for the rest of this thesis.

Once the pose is known, we could use rendering techniques to generate the virtual view. A certain area of the resulting face will naturally be inexistent, since there is no information about these face areas in the original image and not a full 3D model

\footnote{It is a controversial issue if humans actually use depth information for face recognition.}
3.3 Adding Depth Information for Pose Robustness

Figure 3.4: Raw Input Image
An input image to our system, whereas the visible head is heavily rotated.

...is acquired but only a 2.5 dimensional image. A 2.5D point cloud is a simplified three-dimensional surface representation that contains at most one depth value for every point in a x/y-plane. We use a depth camera to acquire depth information since geometry acquisition systems which produce full 3D models are generally complex, expensive and not yet applicable to real-world applications and scenarios. Figure 3.5 illustrates the idea by showing the original and the manually rotated point cloud resulting from the stereo input images, one of them shown in figure 3.4. Additionally to the holes which originate from missing depth information (see figure 3.5, left), the part of the face which was not visible in the original image is occluded in the rotated point cloud in figure 3.5, right.

If necessary, the originally occluded part of the face could be completed with information from the visible side of the face under the assumption that faces are nearly symmetrical. Furthermore, a significant enhancement could be achieved using additional cameras to get texture information from different viewpoints. Image-based rendering techniques can then be used to generate the rotated image, whereby holes and occluded parts are avoided.

\(^2\)For 3D model acquisition products see for example [Cyberware].
Figure 3.5: Raw Rotated Point Cloud

On the left side, a raw 2.5D point cloud is rendered. On the right side, the point cloud is manually rotated. Holes resulting from the depth acquisition are visible, and almost half of the face is inexistent because of missing depth information.
Chapter 4

Depth from Stereo Vision

The purpose of stereo vision is to acquire depth information through range measurements based on images obtained from cameras with a certain offset. To get the necessary depth information to estimate the rotation of faces, we use a stereo vision module called Bumblebee which was developed at the Laboratory for Computational Intelligence at the University of British Columbia and is being marketed by Point Grey Research, Inc. [Bumblebee].

We use a model which consists of two color Sony CCD sensors with a lens focal length of 6 mm (50° horizontal field of view) and a resolution of 640x480 pixels at a frame rate of 30 fps each (see figure 4.1). Our data is taken under non-uniform, indoor illumination conditions, as can be seen in figure 4.2. The only artificial help is a commercial photographer lamp for top lighting. The whole scenery consists of various objects and is complex because of its depth range and the illumination conditions (multiple light sources and direct illumination).

The focus of this project was not to develop a camera stereo system but analyze how applicable a recent system and its algorithms are. The Point Grey Triclops Software Development Kit offers a variety of methods for stereo processing and validation. However, the selection and usage is not trivial, since an adequate combination of algorithms has to be chosen and the appropriate parameters found. We present the used algorithms and shortly outline the theory of stereo vision as a basis for following discussions. In appendix B, the library methods and parameters chosen for the stereo and validation algorithms can be found.

Principle of Stereo Processing

In stereo processing, three main steps have to be performed:

1. Establishing the correspondence between image features in the two images

---

1For the exact technical specifications see [Bumblebee].
Figure 4.1: Bumblebee Stereo Camera
The size of the stereo camera system is approximately 160x40x50 mm and consists of two Sony CCD color sensors.

2. Calculating the relative displacement between feature coordinates in each image

3. Determining the 3D location of the features relative to the cameras using the knowledge of the camera geometry

Considering two images, taken from slightly different views of horizontally displaced cameras, we find a certain feature point at a pixel \((x, y)\) from the reference image \(I_{\text{left}}\) at a different position \((x', y')\) in the other image \(I_{\text{right}}\). This difference between the coordinates of the same features in the left and right images is called disparity. Because the cameras are horizontally aligned, only the horizontal displacement is relevant. The correspondence is then given by

\[
x' = x + s \cdot d(x, y)
\]  

(4.1)

where \(s = \pm 1\) is chosen such that the disparity \(d\) is always positive.

If the disparity \(d(A)\) of a feature \(A\) is different from the disparity \(d(B)\) of another feature \(B\), their distance to the camera system is different. For example, \(d(A) > d(B)\) indicates that point A is closer in the scene than point B, as illustrated in figure 4.2.

4.1 Epipolar Geometry

4.1.1 Camera System Calibration

Calibration of a stereo camera system requires a calculation of the intrinsic and extrinsic parameters to correct for lens distortion and camera misalignment. Based on the calibration, the relationship between 3D points in the scene and their different camera images can be found.
4.1 Epipolar Geometry

Figure 4.2: Stereo Input Images
In the left and right raw input image, two feature points A and B are marked. Since the horizontal displacement and hence the disparity for feature A is bigger than for feature B, feature A must be closer to the viewer than feature B.

The following equation maps the real-world point $X$ in homogeneous coordinates to its projection $x$ using the common pinhole camera model:

$$
\begin{align*}
\lambda \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} &= \begin{bmatrix} f s_x & f s_y & o_x \\ 0 & f s_y & o_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} R & T \\ 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \\
&= K \Pi \begin{bmatrix} R & T \\ 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}
\end{align*}
$$

where $\Pi$ is the canonical projection matrix. $K$ is the internal calibration matrix and consists of the intrinsic parameters of the camera: the focal length $f$, the relative aspect $s_x$ and $s_y$ of each pixel and the skew in the shape of the pixel $s_\theta$, i.e. its deviation from an axis aligned rectangle which usually is zero. $o_x$ and $o_y$ specify the coordinates of the principal point which is the intersection of the camera’s optical axis with the image plane. The external calibration matrix $E$ consisting of the extrinsic camera parameters defines the positions of the cameras. $R$ is a $3 \times 3$ rotation matrix and $T$ is a vector in $\mathbb{R}^3$.

The Bumblebee system is already precalibrated and constructed such that a recalibration normally is not necessary.

4.1.2 Stereo Correspondence
Considering the geometry of two calibrated cameras viewing a scene (see figure 4.3), the cameras are related to a rigid body motion $(R, T)$ and centered at $o_1$ and $o_2$. The intersections of the baseline $(o_1, o_2)$ with each image plane are called epipoles $e_1$ and
Figure 4.3: The epipolar model
The intersections of the baseline \((o_1, o_2)\) with two image planes are called epipoles \(e_1, e_2\). Two projections \(x_1, x_2\) of a point \(p\) are lying on each other’s epipolar lines \(l_1, l_2\).

\(e_2\). The lines \(l_1\) and \(l_2\) are the epipolar lines, which are the intersection of the plane \((o_1, o_2, p)\) with the two image planes.

The epipolar geometry is defined by the fundamental matrix \(F\) which yields for a point \(x\) in the first view the corresponding epipolar line \(l’\) in the second view.

\[ l’ = F^T x \] (4.3)

From that, the following condition can be derived for corresponding points:

\[ x'^T F x = 0 \] (4.4)

Therefore, the pinhole camera model implies that the correspondent of a point in one image lies on its epipolar line in the other image, and corresponding points can be found by scanning every point’s epipolar lines.

4.1.3 Undistortion
The pinhole camera model does not take into account any nonlinear lens deformation, the most important being radial distortion. This deviation from the linear model projects lines as curves, which is most evident at the borders of the image. As the quality of the depth estimation depends on correspondence search along them, this has to be taken into account.

The radial distortion is dependent on the distance \(r\) from the optical axis. The distortion function \(L(r)\) is usually modelled by an even polynomial

\[ L(r) = 1 + ar^2 + br^4 + cr^6 \] (4.5)
4.2 Depth Estimation

Figure 4.4: Undistorted Stereo Input Images
The result of the undistortion of the input image pair from figure 4.2. Lines such as the edges of the whiteboards are now straight.

The projection of a point $X$ in Eq. 4.2 is therefore extended and the point non-linearly distorted.

By back-projecting the images onto a plane parallel to the baseline, the images from the camera are radially undistorted. Thus, the epipolar lines are horizontally aligned with the image rows, and the correspondence search is simplified since points in one image plane map to the horizontal scan line with the same $y$ coordinate on the other image plane.

The result of this process of the input image pair from figure 4.2 is presented in figure 4.4.

4.2 Depth Estimation

A correlation-based stereo algorithm is used, which establishes correspondence between images by computing the sum of absolute differences. The reason to use this score compared to others (e.g. sum of squared differences, graph cut method) was the better optimization potential for speed compared to the minimal impact in quality.\(^2\)

Therefore, a mask around each pixel in the reference image is compared with a mask around each pixel within a certain minimum and maximum distance in the matching image along the epipolar line. For each pixel, the disparity value $d$ with the minimum

\(^2\)According to D. Murray, Point Grey Research. The experiments have not been published.
error value is chosen according to:

$$\arg \min_{d=d_{\text{min}}} \sum_{i=-m}^{m} \sum_{j=-m}^{m} \| I_{\text{ref}}(x+i)(y+j) - I_{\text{match}}(x+i+d)(y+j) \|$$  \hspace{1cm} (4.6)

where \(d_{\text{min}}\) and \(d_{\text{max}}\) are the minimum and maximum disparities, \(m\) is the mask size, and \(I_{\text{ref}}\) and \(I_{\text{match}}\) the images.

The choice of the parameters \(d_{\text{min}}, d_{\text{max}}\) and \(m\) is crucial, since they strongly influence the performance of the algorithm and thus the quality of the disparity estimate (see appendix B for the chosen parameters). The disparity range \([d_{\text{min}}, d_{\text{max}}]\) determines in which relative distance the best match for a certain pixel is searched. Reducing the disparity range will allow the system to run faster and will reduce the chance of a mismatch, but will reduce the identified depth range within the image. The correlation mask \(m\) controls the coarseness of features compared between images. Larger masks will produce depth maps that are denser and smoother, but may have problems to identify the position of depth discontinuities.

The distance from the stereo camera and the depth is computed using the disparities and the geometry of the camera sensors. For our stereo vision system, with aligned optical axes and therefore focus at infinity, the depth \(Z(d)\) is a function of the disparity \(d\), the focal length \(f\) of the lenses, and the baseline \(B\) between the single camera sensors.

$$Z(d) = \frac{fB}{d}$$  \hspace{1cm} (4.7)

Therefore, for a given pixel \((x, y)\) in the reference image and a disparity result \(d\) from stereo matching, a 2D position can be determined by triangulation at the intersection of the lines-of-sight, which represents the resulting depth \(Z\) (see figure 4.5). An example of a disparity map resulting from a stereo input can be seen in figure 4.6, where \(Z\) values are encoded into color values.

### 4.3 Validation of Depth Maps

To compute reliable dense depth maps, a stereo algorithm has to preserve depth discontinuities and avoid gross errors.\(^3\) Correlation-based algorithms rely on the fact that the same texture can be found at corresponding points in a stereo image pair and are in general very sensitive to noise. Noise due to stereo mismatches is a serious problem. Scenes containing specular surfaces, repetitive patterns, and time-varying light

\(^3\)We call a depth map dense, if there is a disparity estimate at each pixel position.
4.3 Validation of Depth Maps

Figure 4.5: Lines of Sight
For a given pixel $x_i$ in the reference image and a disparity result $d_j$ from stereo matching, a depth value $z$ can be determined as the intersection of the lines of sight of the centers of the two pixels.

Figure 4.6: Stereo and Raw Disparity Image
In the left image, the disparity information is encoded in the red and green channel. In the resulting disparity map on the right side, a lot of noise is visible and hence postprocessing is necessary.
sources can cause errors. These mismatches are reduced by validation algorithms or by increasing the number of cameras in a multi-baseline system. We use the following validation methods from [Murray 00] and [Fua 93], provided by the library. The appropriate parameters for each algorithm can be found in appendix B.

**Surface Validation**

This method removes spikes, which are characteristic of mismatches in correlation-based stereo vision, but preserves thin structures that are part of a coherent structure. Spikes are difficult to remove with standard filtering techniques, because they appear to be a valid signal instead of noise. The test segments the disparity image into connected regions and invalidates surfaces below a certain size.

**Uniqueness Validation**

The value of the score for a certain pixel is normalized by the sum of all the scores for this pixel. If the result is above a certain threshold, the match is considered to be insufficiently unique and classified as invalid. This error generally occurs in occluded areas.

**Texture Validation**

The test guarantees that there is sufficient variation in the correlation image patches. Without enough information from contrast or texture, the matching results may be ambiguous. Therefore, the edge strength inside the stereo mask is thresholded.\(^4\)

**Back-Forth Validation**

In this test, the image plays a symmetric role such that the correlation is performed twice by reversing the roles of the two images. As valid are considered only those matches for which the same depth is measured at corresponding points when matching Image 1 to Image 2 and vice versa. This validation method is likely to fail in presence of an occlusion. Although very effective, it lowers the frame rate drastically.

In the postprocessed disparity map in the left image of figure 4.7, top row, a significant improvement over the raw disparity map in figure 4.6 is recognizable. In the right image, the resulting point cloud is rendered whereas the points as well as the computed surfaces are colored. The holes result from stereo mismatches and invalidations, and occur because of lighting problems such as specular highlights and ambiguities due to repetitive patterns.

\(^4\)Better results were achieved without using this validation method, see appendix B.
4.3 Validation of Depth Maps

Figure 4.7: Validated Disparity Image and Resulting Point Cloud
In the first row, the postprocessed disparity map with limited disparity range (left) and the rendered resulting point cloud (right). On the bottom, the uncolored surface, slightly rotated to show the spatial structure and quality (right).

As described in section 3.3, our target is to robustly estimate the pose of the face from such point clouds resulting from depth maps. To illustrate this, the uncolored computed surface is rendered in figure 4.7, bottom. To clarify the spatial structure quality, it is rotated in the right image.
Chapter 5

Pose Estimation

As described in section 2.2, the fundamental task of estimating the face pose in an image appears to be a chicken-and-egg problem. We have to know the location of the face in an image to estimate its pose, but we want to perform pose estimation for face detection and recognition itself.

Because of the difficulties in finding features for face detection which are robust for the pose range from profile to frontal view, a feature-based approach was discarded (see section 3.3). On the other hand, first experiments with an appearance-based algorithm constructing view-based eigenspaces\(^1\) proved the need for extremely accurate registration and cropping of faces within an image, and therefore the necessity of face detection (described in section 2.1.2). Another approach is to use a skin-color detector. The limitation of this method is generally its dependency on illumination conditions and problems of generalization (e.g. for different skin types).

Our suggestion to approach the chicken-and-egg problem is to exploit depth information instead of relying only on appearance. Since the general shape of the object we are looking for is known, we can use it as a cue. This method is loosely related to early approaches where an ellipse is fitted to intensity images preprocessed by an edge detection algorithm.

The exact problem we are trying to solve can be stated as follows: the target is to find an algorithm capable of estimating the pose of a specific object from a known object class in a 2.5D point cloud which is possibly afflicted with noise, holes and occlusions, to an accuracy such that a face recognition algorithm reliably works on a generated input image of the rotated object based on this estimation.

We are estimating the pose of a specific object by the registration of a general model to the input depth map. As described in section 2.2.1, a successful and very popular method for shape registration is the Iterative Closest Points algorithm (ICP),

\(^1\)Using the open source library Intel OpenCV [OpenCV ]
but a good initial guess is required for its convergence. We therefore suggest a two stage-algorithm for pose estimation, performing a rough estimation by a brute force pose space search and then using ICP based on this initial guess to refine the estimation.

In this chapter, we describe the development of our rough pose estimation algorithm and present different stages and enhancements. An error function consisting of two error terms is developed which ensures that the model is matched to the depth map with correct size and position. The rotation of the model causing the least error is then chosen to be the optimal pose estimation. Furthermore, we demonstrate how another model and different weightings of the error terms improve the characteristics of the error function and hence the fitting performance. Additionally, ICP is shortly presented in section 5.6, and we show that the convergence conditions are satisfied by our algorithm.

5.1 Brute Force Pose Space Search

Assuming an input depth map $D$ from our stereo camera system and a general face model $M$, we perform a brute force pose space search to find the position, size and pose of a face in the input (see algorithm 1). $M$ is rendered with a rotation $(\theta, \phi)$, translation $(x, u)$ and scale $s$, and the error function $\epsilon$ compares the depth values of $M$ and $D$. The algorithm iterates over parameters $(\theta, \phi, x, y, s)$ with step sizes $(\delta_\theta, \delta_\phi, \delta_x, \delta_y, \delta_s)$, selecting the optimal rotation $(\theta_o, \phi_o)$ which minimizes $\epsilon$.

Such extensive and thus computationally expensive search procedures were not realizable until recently. Because of the recent progress of programmable graphics hardware, they can now be implemented efficiently by exploiting the massive parallel computational power of the GPU and off-screen rendering techniques (see chapter 6).
5.1 Brute Force Pose Space Search

Figure 5.1: Average Face Model
This average face model is computed based on face scans in the MERL scanning dome.

As a model $M$, we use the average face mask from [Lee 03] which is downsampled to 3'500 vertices. It is computed based on a linear combination of Eigenheads obtained by PCA (see section 2.1.3) of laser-scanned 3D faces (97 male and 41 female adults). The $z$-values of the model are normalized to $[0, 1]$. In figure 5.1, a few rotations of the rendered average face model are shown.

We acquired input images in the scenery described in chapter 4 and computed input depth maps with a ground truth. The subjects were sitting on a turnable chair and visually fixating marked points around them of a pose range from $-90^\circ$ to $+90^\circ$ and a step size of $15^\circ$. Since we need only a rough pose estimation, this setup should be accurate enough. For now, we omit dealing with elevation $\theta$.

Some of the resulting input depth maps together with the reference color images are shown in figure 5.2 for rotation steps of $30^\circ$ azimuth. The depth values are visualized as color information; cold colors indicate more distance to an object than hot colors. The same facial feature point (e.g. the nose) can have a different color in diverse depth maps since the depth ranges differ due to noise, and the same absolute depth value is encoded in different colors. In the visualizations of the depth maps in figure 5.2, different face poses are distinguishable.

The acquired depth maps are not manually cleaned, and hence noise is present (clearly recognizable in figure 5.2). It is caused by the stereo vision algorithm which showed to be very sensible to contrast of the object compared to the background, and to the characteristics of the background itself. Especially noise around the head silhouette caused by specular light reflections in the background as well as dark or skin-colored background is problematic since the depth of the background is then incorrectly estimated to the $z$-region of the object. Furthermore, holes are visible which mainly occur either at flat regions (like cheek or forehead) or at dark, uniform regions (like hair). In such regions, variance or texture information is missing, and hence ambiguous depth estimations are eliminated by the validation methods.
Figure 5.2: Input Depth Maps and Reference Color Images
Situations with pose rotations $\phi$ of $-90^\circ, -60^\circ, -30^\circ, 0^\circ, 30^\circ, 60^\circ, 90^\circ$. 
5.2 Sum of Squared Differences Error Term

The error function \( \epsilon \) from algorithm 1 is chosen to be the sum of squared differences of the model \( M \) and the input depth map \( D \):

\[
\epsilon_{ssd}(D, M) = \frac{1}{N_{ssd}} \sum_{x,y} \epsilon_{ssd}(\phi, x, y)
\]

(5.1)

\[
\epsilon_{ssd}(\theta, x, y) = \begin{cases} 
|z(D(x, y)) - z(M(\phi, x, y))|^2 & \text{if } z(M(\phi, x, y)) \neq 0 \\
0 & \text{otherwise}
\end{cases}
\]

(5.2)

\[
N_{ssd} = |\{(x, y) \mid z(M(\phi, x, y)) \neq 0\}|
\]

(5.3)

From the input depth map \( D \), a window of a certain size and position according to the current iteration step is chosen, its z-values normalized to \([0, 1]\), orthogonally projected and rendered with a resolution of 128x128 pixels. At every pixel covering the model \( M \), the squared difference of the z-value of \( M \) and the corresponding z-value of \( D \) is computed and summed up. The sum is normalized by the number of summands.

The normalization of the depth values is necessary because the model only covers a part of the whole head and hence the depth ranges of corresponding regions are not comparable. Furthermore, the restriction of the error computation to positions where the model exists guarantees a punishment if \( M \) is aligned outside the head in \( D \), but not vice versa. This is necessary because \( D \) covers the whole head but \( M \) only the face.

The algorithm is tested on several input sets of 13 pose situations each. In the subsequent sections, results are illustrated like in figures 5.3 and 5.4: For each azimuth pose rotation \( \phi \) from \(-90^\circ\) to \(+90^\circ\) and a step size of \(15^\circ\) the result of the fitting procedure is displayed in figure 5.3 where the input depth maps are superposed by the fitted average face models with those sizes and positions causing the least error for each tested rotation. The appropriate error graph is plotted in figure 5.4. The minimal fitting error values for each tested rotation is indicated, whose minimum hence specifies the estimated rotation.

We demonstrate the behavior of the algorithm showing the results of two extreme input situations (frontal and profile view, illustrated in image 1 and 4 of figure 5.2).

In figures 5.3 and 5.4, the fitting results and minimal error values for the frontal situation are shown. The global minimum error in figure 5.4 is at position 7, which corresponds to the fit in image 7 of figure 5.3.\(^2\) This pose estimation is correct because

\(^2\)The images are numbered row wise from left to right.
Figure 5.3: Fitting Result of Correct Fit with Sum of Squared Differences
For each rotation step, the best fit w.r.t. size and position is shown for a frontal situation (0° rotation).

Figure 5.4: Error Graph for Correct Fit with Sum of Squared Differences
For each rotation step, the error of the optimal fit w.r.t. size and position is plotted. The global minimum is at position 7, which corresponds to the fit in image 7 of figure 5.3, and is the correct pose estimation.
5.2 Sum of Squared Differences Error Term

**Figure 5.5:** Fitting Result of Incorrect Fit with Sum of Squared Differences
For each rotation step, the best fit w.r.t. size and position is shown for a profile situation (−90° rotation).

**Figure 5.6:** Error Graph for Incorrect Fit with Sum of Squared Differences
For each rotation step, the error of the optimal fit w.r.t. size and position is plotted. The global minimum is at position 13 which a wrong pose estimation. The situation corresponds to a profile view of −90° rotation.
position 7 corresponds to a face rotation $\phi$ of $0^\circ$ and hence to a frontal situation. However, the error function is not optimal because of local minima, i.e. there is no direct relation between pose error (distance from the correct pose) and error value.

In contrast, the fitting result for the profile situation illustrated in figures 5.5 and 5.6 is an incorrect pose estimation. The minimal error value appears to be at position 13, corresponding to a rotation of $+90^\circ$. The error at the position of the ground truth $-90^\circ$ is high because the model is fitted to the relatively flat, uniform area of the cheek at a wrong scale.

Two problems can be recognized: the fitting to a wrong position and the fact that a wrong model size causes the minimal error. One reason is the unimportance of distinctive facial regions like eyes and nose for the fitting to a profile depth map since they cover only a small amount of the whole input area. Besides, the projection of the model with this rotation appears to be rather flat, hence the tendency to fit to uniform surfaces like the cheek.

The fundamental problem is the intrinsic contrary nature of the different pose situations. In a frontal view of a face, the distinctive regions around the eyes, nose, mouth and chin cover a good portion of the head and are located in its center, whereas these regions cover a much smaller area and are located at the boundary of the silhouette in a profile view.

A further reason for the incorrect shape registration is the noise in the depth map, especially around the silhouette shape (see figure 5.5). The area of important and intuitively distinctive cues in a profile view like chin and nose is small compared to the whole input shape, and hence noise occurring around such regions has negative influence.

### 5.3 Boundary Weighted Error Term

To improve the choice of the correct size, we try to attract especially the silhouette. Thus it gets more important for the fitting procedure such that the size of the mask is chosen appropriately. Furthermore, the probability to cover important regions is increased by enlarging the size of the model.

A second error term $\epsilon_{xor}$ is introduced to impose higher penalty for the mismatch near the boundary of the input silhouette. Motivated by [Lee 03], a pixel wise exclusive-or error term is chosen which is weighted inversely proportional to the silhouette boundary distance:
5.3 Boundary Weighted Error Term

<table>
<thead>
<tr>
<th>Figure 5.7: Binary Silhouette and Euclidean distance transform</th>
</tr>
</thead>
<tbody>
<tr>
<td>In the left image, the binary silhouette image computed from a depth map of a $-45^\circ$ situation is shown. On the right side, the Euclidean distance transforms $d_{edt}(x, y)$ are visualized, where the distance to the boundary is encoded with a color between black and green.</td>
</tr>
</tbody>
</table>

\[
\epsilon_{xor}(D, M) = \sum_{x,y} c(x, y) 
\]

\[
c(x, y) = \begin{cases} 
  w(x, y) & \text{if } z_M(x, y) \oplus z_D(x, y) \neq 0 \\
  0 & \text{otherwise} 
\end{cases} 
\]

\[
w(x, y) = \begin{cases} 
  1 - \frac{d_{edt}(x, y)}{d_{max}} & \text{if } (x, y) \in \text{silhouette}(D) \\
  0 & \text{otherwise} 
\end{cases} 
\]

Thereby, the error $w(x, y) \in [0, 1]$ is added by the xor-operation $\oplus$ for each pixel where the model $M$ does not cover the silhouette of the face in the input depth map $D$. The error value decreases with increasing distance to the silhouette boundary of $D$, and is set to zero outside of the silhouette. The distance $d(x, y)$ is computed by the normalized Euclidean distance transform $d_{edt}(x, y)$ of the binary input image $S$ and its inverse $\bar{S}$, according to the algorithm of Saito and Toriwaki [Saito 94]. Figure 5.7 visualizes the binary silhouette image and the resulting Euclidean distance transform.

The overall error

\[
\epsilon(D, M) = \epsilon_{ssd}(D, M) + \lambda \cdot \epsilon_{xor}(D, M) 
\]

now consists of the boundary weighted error term introduced above and the sum of squared difference from section 5.2. The balance weight $\lambda$ is experimentally chosen.
such that the magnitude of both error terms is the same.

To illustrate the effect of the additional error term to the fitting procedure, the result for the problematic profile situation from figure 5.5 is shown in figure 5.8. The target is achieved and the size of the model correctly chosen. However, the fitting procedure selects the wrong position because of the unimportance of distinctive facial regions and noise.

![Fitting Result with additional Boundary Weighted Error Term](image)

**Figure 5.8: Fitting Result with additional Boundary Weighted Error Term**
The size of the model is correctly chosen by the additional error term but the wrong position is selected.

### 5.4 Full Head Model

The replacement of the average face model with a full head model solved the intrinsic problem of fitting one model to both profile and frontal face views. The new model is acquired by a laser scan and can be freely downloaded from Cyberware [Cyberware]. It consists of roughly 31'000 vertices. A few rotations are illustrated in figure 5.9.³

³We did experiments with different full head models; the one of a bare head without neck shown in figure 5.9 worked the best.
5.5 Curvature Based Region Weighting

The figures 5.10 and 5.11 demonstrate the correct fitting of the model to the profile input (see image 1 of figure 5.10). The error value in figure 5.11 is minimal at position 1 which corresponds to the fit in image 1. Since the estimation matches the ground truth for this situation, the pose estimation is correct.

The advantage of this model compared to the average face model used before is mainly the similar size and shape compared to the whole input. Therefore, the model is fitted to the whole input instead of the small part covering only the face. The boundary weighted error term for this model is very effective since the whole head shape can be covered in every input situation from frontal to profile view. Moreover, the negative influence of local noise and holes is decreased because a larger area is considered for the error computation.

Although the pose is correctly estimated, the characteristics of the error function still would be more optimal if the global minimum was more distinctive compared to the local minima. If the error value would even increase with growing rotation error, a gradient descent approach could be implemented to increase the performance of our algorithm.

5.5 Curvature Based Region Weighting

To enhance the robustness of the error function, the influence of head regions which are distinctive for the error computation is increased. We multiply the squared difference error term of eq. 5.2 by a weight $w_r(x, y) \in [0, 1]$ such that the error value is dependent on the pixel location. The equations 5.1 to 5.3 are hence extended to equations 5.8 to 5.10:

$$
\epsilon'_{ssd}(D, M) = \frac{1}{N'_{ssd}} \sum_{x,y} \epsilon^r_{ssd}(\phi, x, y) 
$$  

(5.8)
Chapter 5. Pose Estimation

**Figure 5.10: Fitting Result with Full Head Model**
With the full head model, both the size and position are correctly estimated for the profile situation.

**Figure 5.11: Result Graph with Full Head Model**
The global minimum is at position 1 which corresponds to a profile view of $-90^\circ$. This corresponds to image 1 in figure 5.10 and is a correct estimation.
5.5 Curvature Based Region Weighting

Figure 5.12: Region Textures
Top left, the texture generated by ABF++. The direction of the normals are encoded by colors. The other images show the manually generated region weighting textures. The best results were achieved with the texture bottom right.

\[
e_{ssd}'(\phi, x, y) = \begin{cases} 
  w_r(x, y) \cdot |z(D(x, y)) - z(M(\phi, x, y))|^2 & \text{if } z_M(\phi, x, y) \neq 0 \\
  0 & \text{otherwise}
\end{cases} 
\]  
\hspace{1cm} (5.9)

\[
N_{ssd}' = |\{(x, y) \mid z_M(\phi, x, y) \neq 0\}| + \sum |w_r(x, y)| 
\]  
\hspace{1cm} (5.10)

The curvature of the facial surface at distinctive geometric regions like nose, eye-hole, mouth or chin is higher compared to other regions like cheek or forehead. Hence, such regions are more significant for shape registration.

We can benefit from this observation by weighting depth differences according to the geometric curvature at the appropriate positions. The curvature can therefore be approximated by methods from discrete geometry (e.g. by Gaussian and mean curvatures).

We circumvent this computation and estimate the weights by manually defining facial regions. Hence, we create a texture map for the model and encode weights to its color channels. Furthermore, a uv-parameterisation is added to the 3D mesh of the model. We use the algorithm ABF++ [Sheffer 05] which is implemented in Graphite
and generates conformal parameterizations by low edge stretching. The basic texture for the full head model is shown in the top left image of figure 5.12. The colors represent the normal directions.

We performed experiments with different region textures which differ in the partition into facial regions as well as the number of different weights. For the texture in figure 5.12 top right, two different values were assigned and ears included in addition to the mentioned intuitive distinctive facial features. The texture left bottom encodes three different weights with higher values around eyes and nose. The inclusion of the ears did not result in an improvement because the reconstruction of the ears from the depth maps is often of low quality due to insufficient contrast and occlusion by the hair. We observed furthermore that the allocation of more than two weight values did not show any significant improvement. The texture shown in figure 5.12 bottom right produced the best results despite its simplicity. The area of higher weight matches approximately the area covered by the average face model.

The fitting results of our algorithm using the described error weighting are shown in figure 5.13, and figure 5.14 demonstrates the achieved improvement on the problematic profile situation. The differences between the global minimum error value and local minima are distinctive compared to the results without this weighting (see figure 5.11). Hence, the robustness of the pose estimation is enhanced. Further results for all 13 acquired pose situations are discussed in chapter 7.

5.6 Iterative Closest Points

If a refinement of the pose estimation achieved by the presented algorithm is necessary, a conventional registration method like Iterative Closest Points (ICP) can be used. As mentioned in section 2.2.1, the drawback of this approach is the necessity of a good initial guess. However, since position, size and rotation are roughly estimated by our algorithm, we can use ICP for the refinement.

The goal of a fine registration algorithm like ICP is to find an exact rigid body transform $\tau = (R, \vec{t})$ consisting of a rotation matrix $R$ and a translation vector $\vec{t}$, such that an input point cloud $Q = \{\vec{q}_1, \vec{q}_2, \ldots, \vec{q}_m\}$ best matches a model point cloud $P = \{\vec{p}_1, \vec{p}_2, \ldots, \vec{p}_n\}$. Given an initial guess of the relative position, ICP chooses a set of $k$ point pairs $\{\vec{p}_i, \vec{q}_i\}$ and approximates the distance between the point clouds by the sum of distances between the point pairs of this set, which is assumed to be a good approximation.

In each round of this iterative procedure, a rigid body transform is computed which
Figure 5.13: Fitting Result with Region Weighting
The fitting results for all pose rotations for the profile input situation are visualized. The model with the correct rotation is fitted perfectly, as can be seen in image 1.

Figure 5.14: Result Graph with Region Weighting
The robustness of the pose estimation is enhanced, i.e. the differences between the global minimum error value and local minima are increased and more distinctive.
minimizes the error

$$\epsilon(\tau) = \sum_{i=1}^{k} d(\tau(\vec{q}_i), \vec{p}_i) = \sum_{i=1}^{k} d(R\vec{q}_i + \vec{t}, \vec{p}_i)$$ \hspace{1cm} (5.11)

The distance $d$ can be the point-to-point distance

$$d = ||\vec{q} - \vec{p}||^2$$ \hspace{1cm} (5.12)

which measures the Euclidean distance between corresponding points [Besl 92], or the point-to-plane distance

$$d = (\vec{n} \cdot (\vec{q} - \vec{p}))^2$$ \hspace{1cm} (5.13)

of [Chen 91] which is the distance between a point and a planar approximation of the surface at the corresponding point. The algorithm converges faster when using the second error metric, if the initial positions are close and noise is low, but might oscillate otherwise ([Gelfand 03]).

After each iteration, the error is computed and the rigid body transform $\tau$ refined. When the initial positions of the model and data point clouds are close, the correspondences and the transform are usually found. However, it has been shown in [Rusinkiewicz 01] that the convergence rate of ICP heavily depends on the choice of the corresponding point pairs, and the distance function that is being minimized.

We use the library \textit{trimesh2} [Trimesh2 ] as a fast and robust implementation of ICP. Our experiments aligning different face shapes show that it is able to align two range images in around ten milliseconds. Trimesh2 is expected to work robustly when two input faces differ in size not more than about 10%, the relative rotation is not more than about 10° to 20°, and the translation is within half the length of the nose.\footnote{Based on personal communication with the author of the library, S. Rusinkiewicz.} These conditions are satisfied by the initial guess from our rough pose algorithm, as verified in first experiments and shown in chapter 7. However, the two algorithms are not yet combined in a common framework and thoroughly tested. If the estimation from our algorithm is accurate enough for our purpose, the refinement could even be omitted. This can only be definitely ascertained when the whole system proposed in section 1.2 and illustrated in figure 1.1 is completed.
Chapter 6

Implementation with Programmable Graphics Hardware

6.1 Programmable Shaders and Framebuffer-attachable Objects

For our algorithm 1 for rough pose estimation presented in section 5.1, fast per-pixel computations based on depth values are necessary. Therefore, the OpenGL graphics pipeline is used. In its newest version 2.0, two important new features have been introduced. Firstly, programmable vertex and fragment shaders allow to bypass the fixed functionality of the corresponding stages in the pipeline of a modern graphics processor using the OpenGL Shading Language (GLSL). Secondly, the Framebuffer Object Extension provides off-screen rendering by directly writing to and reading from textures.

The OpenGL framebuffer is a collection of logical buffers like color, depth, stencil or accumulation buffer. To render to destinations which are not provided by the window system, rendering objects are introduced which are attachable to the framebuffer and hence called framebuffer-attachable objects (FBOs). They consist of render buffer images or texture images and can be used either as source or destination for fragment operations to replace the logical buffers provided by the window system. For an illustration of FBOs see figure 6.1.

We use programmable shaders to directly read and write values to textures attached as FBOs. Therefore, we can use the graphics pipeline for our own purpose.
Chapter 6. Implementation with Programmable Graphics Hardware

Figure 6.1: Framebuffer-attachable Objects
Texture and render buffer objects are attachable as rendering targets.

6.2 Encoding Depth Values to Color Textures

Different steps involving off-screen rendering to texture are necessary for the implementation of our algorithm 1 in section 5.1. Therefore, a texture specifying the necessary data type has to be created and attached as a FBO indicating its purpose, e.g. depth or color component. In a fragment shader, values and weights from such textures are read, and the errors computed and written to such textures. However, no knowledge about neighboring pixels is available within a shader since only local computations are supported by the parallel graphics pipeline architecture.

The Framebuffer Object Extension specification provides the possibility to specify a texture as a target to save depth values (see figure 6.1, dotted arrow). We performed experiments using this feature by creating textures of the internal format type GL_DEPTH_COMPONENT and attaching it as a framebuffer-attachable object of type GL_DEPTH_ATTACHMENT. However, the result was not as expected since depth values were not correctly written. The reason is probably an incorrect implementation of the specification by the driver, because such special features are usually not used in standard applications and hence not tested. Instead, we succeeded in specifying textures of type GL_RGBA32F_ARB which store 32bit float values as a RBGA color texture and are attached to the framebuffer as GL_COLOR_ATTACHMENT0_EXT. Depth values, weights and error values are encoded in channels of the color texture by corresponding fragment shaders.

---

1We used a nVidia GeForce 7800 GTX graphics card, 256 MB ram, driver version 8.1.9.5, September 11, 2005.
6.2 Encoding Depth Values to Color Textures

In a first step, the head model is rendered to different textures according to different pose rotations. For each pixel, the depth value of the model is written to the green channel, and the weight from the region weighting texture (introduced in section 5.5) is read and written to the red color channel. The resulting texture is illustrated in figure 6.2, top row. Modern graphics cards support saving and simultaneous maintenance up to 16 textures on its memory. The input textures are saved to positions GL_TEXTURE1 to GL_TEXTURE13. The values for the curvature based region weighting are saved to GL_TEXTURE14.

Secondly, the depth value from the stereo vision algorithm (see chapter 4) is written to the green channel of another texture (as shown in figure 6.2, middle row), addressed by GL_TEXTURE0.

Furthermore, the weight resulting from the Euclidean distance transform for the boundary weighting (introduced in section 5.3) is saved to the red channel of a separate color texture (see figure 6.2, bottom row), addressed by GL_TEXTURE15.

Based on these textures, the error of the model of one pose and the input depth map is computed for a certain position and size (see inner loop of algorithm 1). Therefore, the depth range of the input depth map within that specific window is computed to normalize the camera input. Then, the error values are computed and written to a result texture by the fragment shader in listing C.1 in appendix C. The four possible states of the resulting 32 bit color texture are illustrated in figure 6.3. In each row, the color channels for one pixel are shown. As described in chapter 5, error terms are written depending on the depth values of the camera and the model at the corresponding position.

In figure 6.4, such a resulting error color texture is visualized, where the different errors and the region weighting according to figure 6.3 are visible. For each iteration step of the algorithm 1 the fitting error is computed based on such an error texture.
Chapter 6. Implementation with Programmable Graphics Hardware

**Figure 6.3: Error Value Color Texture**
The four different states of the resulting 32 bit color texture saving the computed error values are illustrated, depending on the input values. In each row, the color channels of one pixel of the corresponding texture is illustrated.

```
if \((z_{\text{model}} = 0) \land (z_{\text{camera}} = 0)\)

<table>
<thead>
<tr>
<th>R</th>
<th>G</th>
<th>B</th>
<th>A</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

if \((z_{\text{model}} = 0) \land (z_{\text{camera}} \neq 0)\)

<table>
<thead>
<tr>
<th>R</th>
<th>G</th>
<th>B</th>
<th>A</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>xor error term</td>
<td>1</td>
</tr>
</tbody>
</table>

if \((z_{\text{model}} \neq 0) \land (z_{\text{camera}} = 0)\)

<table>
<thead>
<tr>
<th>region weight</th>
<th>ssd error term</th>
<th>xor error term</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>R</td>
<td>G</td>
<td>B</td>
<td>A</td>
</tr>
</tbody>
</table>

if \((z_{\text{model}} \neq 0) \land (z_{\text{camera}} \neq 0)\)

<table>
<thead>
<tr>
<th>region weight</th>
<th>ssd error term</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>R</td>
<td>G</td>
<td>B</td>
<td>A</td>
</tr>
</tbody>
</table>
```

**Figure 6.4: Error Value Rendering**
The error texture from one iteration of our algorithm. The sum of squared difference, boundary-weighted xor values and region weights are encoded in color channels.
Chapter 7

Results and Discussion

In this chapter, we present and discuss the final results of our rough pose estimation algorithm developed in chapter 5, where two input situations were demonstrated (see figures 5.3 and 5.5). They were chosen due to their contrary characteristics and hence raise different problems for a shape matching procedure (discussed in section 5.2). Here, the performance of the algorithm is discussed considering the results for all 13 input pose situations.

In figures 7.1 to 7.26, the fitting results and resulting error graphs from which the pose estimation is inferred are presented for all pose situations. An overall pose variation from $-90^\circ$ to $90^\circ$ azimuth rotation and a step size of $15^\circ$ is considered. Some of the resulting error graphs show a very nice behavior, e.g. figures 7.2, 7.20, 7.24 and 7.26. The global minimum error is clearly smaller than other local minima and hence distinctive. Furthermore, the error value increases for growing pose rotation error, especially in figures 7.20 and 7.22. However, the error function decreases again sometimes when a model of contrary pose rotation to the ground truth is fitted, as in figures 7.6, 7.8 and 7.12. Especially the error curve in fig. 7.12 shows perfect characteristics except for the low error values for the rotations of $-90^\circ$ and $-75^\circ$. Otherwise, a gradient descent approach could be implemented for narrowing the pose space and hence accelerating the brute-force pose space search.

We did experiments with input depth maps resulting from 6 different persons, including both genders and different ethnicity (Caucasian, Asian and Persian). We observed that our algorithm generalizes well for variations between persons, but depends strongly on the quality of the input depth map. Hence, the performance of the algorithm is conditioned by the depth estimation step. Depending on the background and illumination, missing contrast and texture within the acquired images result in holes and noise and hence an imprecise pose estimation. We could produce different inputs from every person for which our algorithm both worked or failed, only depending on the exact position of the person with respect to the background (see figure 5.2 for an
Table 7.1: Error of Pose Estimation
For each pose situation indicated in the top row, the error of the pose estimation results from figures 7.1 to 7.26 compared to the annotated ground truth is presented in the bottom row.

Table 7.1: Error of Pose Estimation
For each pose situation indicated in the top row, the error of the pose estimation results from figures 7.1 to 7.26 compared to the annotated ground truth is presented in the bottom row.

<table>
<thead>
<tr>
<th>Pose Situation</th>
<th>Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>0°</td>
<td>15°</td>
</tr>
<tr>
<td>15°</td>
<td>15°</td>
</tr>
<tr>
<td>15°</td>
<td>15°</td>
</tr>
<tr>
<td>15°</td>
<td>15°</td>
</tr>
<tr>
<td>15°</td>
<td>15°</td>
</tr>
<tr>
<td>15°</td>
<td>15°</td>
</tr>
<tr>
<td>15°</td>
<td>15°</td>
</tr>
<tr>
<td>15°</td>
<td>15°</td>
</tr>
<tr>
<td>15°</td>
<td>15°</td>
</tr>
<tr>
<td>15°</td>
<td>15°</td>
</tr>
<tr>
<td>15°</td>
<td>15°</td>
</tr>
<tr>
<td>15°</td>
<td>15°</td>
</tr>
<tr>
<td>15°</td>
<td>15°</td>
</tr>
<tr>
<td>15°</td>
<td>15°</td>
</tr>
</tbody>
</table>

Because of this sensibility, we considered an exhaustive statistical analysis including different input situations useless, since we could produce arbitrary pose estimation results by providing the inputs of the corresponding quality. In chapter 8, we describe how we plan to overcome these limitations and to statistically proof the reliability of our algorithm.

We omitted to deal with rotation in elevation additionally to azimuth variation because of the limiting factors for a reliable analysis, as discussed above.

The depth estimation for the demonstrated results in figures 7.1 to 7.26 is of average quality, with precise contours but existing noise and holes. As summarized in table 7.1, the pose estimation is correct for all pose situations if a maximum variance of 15° is tolerated.

By thresholding the optimal error value for an input, we can perform face detection and infer the size and position of a face. Therefore, the algorithm provides an initial guess for ICP (see section 5.6). Moreover, by adjusting the disparity range (see equation 4.6 in section 4.2) and the search parameters of our algorithm, we can determine the size range of a face to be detected. This is for example useful for an application scenario where multiple people are standing at a gate in a line.

Our current straightforward GPU implementation evaluates the error function more than 600 times per second. The performance of the algorithm depends on the range of possible face sizes and hence on the application scenario. The current system iterates over 64 image positions and 3 possible face sizes, and hence needs about 4 seconds for the pose estimation. This can be drastically reduced by optimizing the implementation for a real-time system.
Figure 7.1: Fitting Result for $-90^\circ$ Pose Rotation

Figure 7.2: Result Graph for $-90^\circ$ Pose Rotation
Chapter 7. Results and Discussion

Figure 7.3: Fitting Result for $-75^\circ$ Pose Rotation

Figure 7.4: Result Graph for $-75^\circ$ Pose Rotation
Figure 7.5: Fitting Result for $-60^\circ$ Pose Rotation

Figure 7.6: Result Graph for $-60^\circ$ Pose Rotation
Figure 7.7: Fitting Result for $-45^\circ$ Pose Rotation

Figure 7.8: Result Graph for $-45^\circ$ Pose Rotation
Figure 7.9: Fitting Result for $-30^\circ$ Pose Rotation

Figure 7.10: Result Graph for $-30^\circ$ Pose Rotation
Chapter 7. Results and Discussion

Figure 7.11: Fitting Result for $-15^\circ$ Pose Rotation

Figure 7.12: Result Graph for $-15^\circ$ Pose Rotation
Figure 7.13: Fitting Result for $0^\circ$ Pose Rotation

Figure 7.14: Result Graph for $0^\circ$ Pose Rotation
Chapter 7. Results and Discussion

Figure 7.15: Fitting Result for 15° Pose Rotation

Figure 7.16: Result Graph for 15° Pose Rotation
Figure 7.17: Fitting Result for 30° Pose Rotation

Figure 7.18: Result Graph for 30° Pose Rotation
Figure 7.19: Fitting Result for 45° Pose Rotation

Figure 7.20: Result Graph for 45° Pose Rotation
Figure 7.21: Fitting Result for 60° Pose Rotation

Figure 7.22: Result Graph for 60° Pose Rotation
Chapter 7. Results and Discussion

**Figure 7.23:** Fitting Result for $75^\circ$ Pose Rotation

**Figure 7.24:** Result Graph for $75^\circ$ Pose Rotation
**Figure 7.25:** Fitting Result for 90° Pose Rotation

**Figure 7.26:** Result Graph for 90° Pose Rotation
Chapter 8

Conclusions and Future Work

In this thesis, we proposed a system for pose-robust face recognition by synthesizing a virtual frontal view. We developed an algorithm for pose estimation exploiting depth information acquired by stereo vision and described its implementation on programmable graphics hardware. We demonstrated the results on a data set acquired in a semi-controlled indoor environment and discussed the limitations.

The next step in the project initiated at MERL by this thesis is an extensive reliability analysis. We will render the 250 face models acquired in the MERL scanning dome and generate rotated pose inputs. Hence, we omit the interference of the pose estimation results by the depth map quality and can test our algorithm on data with reliably annotated pose. After having investigated the exact accuracy of the rough pose estimation we can decide if a refinement by Iterative Closest Points is necessary. Furthermore, we will optimize the implementation with respect to performance. The goal is a real-time system which robustly detects a face and outputs its pose in a continuous video-stream.

Based on this reliable and real-time pose estimation system, the missing stages for the complete face recognition system can be implemented. Firstly, the acquired point cloud is rendered with the estimated pose rotation to generate the normalized pose image. Secondly, the image has to be corrected. Based on the correspondence of a position on the fitted model and an image pixel, occluded parts in the generated image can be filled. Therefore, the corresponding position of a missing image pixel on the model is computed, its symmetric position on the other face side determined based on the precomputed model symmetry plane, and the color of the corresponding image pixel from the complete face side in the input image assigned. Furthermore, the image can be relighted based on the method for illumination normalization developed at MERL [Lee 05], which approximates the lighting directions. Finally, the system will be integrated to the MERL face recognition framework and its existing 2D algorithm used for the recognition step.
The depth estimation procedure could be enhanced using alternative correspondence algorithms. Moreover, multi-baseline stereo approaches would avoid the ambiguity of stereo systems because it is unlikely that an erroneous minimum in one correspondence input image pair coincides with another one. Another possibility is to exploit inter-frame information from video-based approaches to enhance the depth estimation.

If the depth quality is good enough and hence producing a nice error function, we expect that a gradient descent approach can be used for the enhancement of the pose estimation performance.

Furthermore, additional cameras could be used for image-based rendering techniques. Hence, a pose-normalized virtual image can be generated without reducing the image quality because of holes. This can be combined with the multi-baseline approach mentioned above.
Appendix A

Face Recognition Library

For the case study mentioned in chapter 3, the library DiamondClassify developed at Merl was used [DiamondClassify]. It implements the detection and recognition algorithms published in [Viola 01], [Viola 03] and [Jones 03]. Both are based on the same rectangle-based features, trained and selected by AdaBoost as described in section 3.1.

The hierarchy of the system is shown in figure A.1, where the relatedness and dependency of the detection and recognition code is visible. For simplicity’s sake and because of its analogy we used the detection API for our case study.

In listing A.1, the method call is indicated. If a face is detected by the specified detector based on the features found in an image of the given list, the specified recognizer evaluates the previously trained similarity function based on the found features. The image is then cropped and the found features saved in a file with the indicated ending. We used the trained detector and recognizer supported by the library.

Listing A.1: DiamondClassify Method Call

```
FaceApiScanRectifyLists
   -l allpics.txt
   -d Classifiers/faces_new.detector
   -r Classifiers/Combined9ptsNew-alldata.recognizer
   -s _mbtest
```
Figure A.1: DiamondClassify Hierarchy
Appendix B

Stereo Vision Library

As mentioned in chapter 4, we used the Triclops Software Development Kit provided by Point Grey Research with the Bumblebee camera to realize our stereo system. In our experiments, we determined which parameters and validation methods are optimal. These parameters are strongly dependent on the setup, e.g. distance of the person to the camera, illumination or background characteristics. Most parameters are chosen regarding a tradeoff between a more accurate depth estimation resulting in small, disconnected areas with noise and holes, and connected areas where correct discontinuities are deleted but noise is omitted.

The following overview explains the most important parameters and the effects of the choice. In chapter 4, the algorithms and methods are explained.

**Stereo Mask** The size of the stereo correlation mask for the comparison of two pixel regions (see section 4.2, formula 4.4). A larger mask produces denser and smoother depth maps but ignores depth discontinuities.

**Disparity Range** The minimum and maximum difference between the coordinates of the same feature in the left and right image (see section 4.2, formula 4.4). Reducing the disparity range accelerates the algorithm, but reduces the identifiable depth range. Furthermore, a larger range boosts the chance for a mismatch.

We tried to enlarge the minimum disparity value as far as possible to avoid having background objects in the resulting depth map. The maximum value on the other hand is decreased as far as possible such that the nearest part of the face still is estimated correctly.

**Edge Mask** The size for the edge detection mask. A larger mask reduces the performance but enhances an accurate edge detection.

**Surface Validation Difference** The maximum disparity difference between two adjacent pixels that will still allow the two pixels to be considered part of the same surface (see section 4.3).
Surface Validation Size The minimum number of pixels a surface has to cover to be considered valid (see section 4.3). Since the surface of noise usually is small, a small value enhances noise. On the other hand, a large value invalidates small areas of target objects.

Uniqueness Validation Threshold If the correspondence value is below the threshold, it is not enough unique and hence invalidated. (see section 4.3). If the threshold is chosen too small, too many estimations are considered invalid.

Texture Validation Threshold The threshold which determines the minimum allowed amount of texture (see section 4.3). We achieved better results in our experiments without using this validation method.

In table B.1, the used parameters as well as their working ranges for our experiments are presented.

Listing B.1 presents how the parameters are set. In the first part, the stereo context is initialized and the image resolution set. Then, stereo and validation parameters are set by the appropriate methods.

<table>
<thead>
<tr>
<th>parameter</th>
<th>value used</th>
<th>working range</th>
</tr>
</thead>
<tbody>
<tr>
<td>stereo mask</td>
<td>15</td>
<td>11 – 23</td>
</tr>
<tr>
<td>disparity range</td>
<td>103 – 146</td>
<td>(explained above)</td>
</tr>
<tr>
<td>edge mask</td>
<td>7</td>
<td>7 – 9</td>
</tr>
<tr>
<td>surface validation threshold</td>
<td>1.0</td>
<td>1.0 – 3.0</td>
</tr>
<tr>
<td>surface validation size</td>
<td>200</td>
<td>200 – 400</td>
</tr>
<tr>
<td>uniqueness validation threshold</td>
<td>1.44</td>
<td>1.0 – 3.0</td>
</tr>
<tr>
<td>texture validation threshold</td>
<td>(not used)</td>
<td>(not used)</td>
</tr>
</tbody>
</table>

Table B.1: Stereo Parameter Values
The used values as well as the range of values which worked in our experiments.
Listing B.1: Stereo Vision Parameters

//===============================================
// --- INITIALIZATION ---
//===============================================
digiclopsCreateContext(*pDigiclops);
digiclopsInitialize(*pDigiclops, 0);
digiclopsSetImageTypes(*pDigiclops, STEREO_IMAGE | RIGHT_IMAGE);

// get the camera module configuration
digiclopsGetTrilopsContextFromCamera(*pDigiclops, pTrilops);

// use 'HALF' resolution when faster throughput is needed
digiclopsSetImageResolution(*pDigiclops, DIGICLOPS_FULL);
triclopsSetResolution(*pTrilops, 480, 640);

// do stereo in 2-camera mode
triclopsSetCameraConfiguration(*pTrilops, TriCfg_2CAM_HORIZONTAL);

//===============================================
// --- STEREO PARAMETERS ---
//===============================================
triclopsSetStereoMask(*pTrilops, 15);
triclopsSetDisparity(*pTrilops, 103, 146);

//===============================================
// --- VALIDATION PARAMETERS ---
//===============================================
// required for texture and uniqueness validation
triclopsSetEdgeCorrelation(*pTrilops, 1);
triclopsSetEdgeMask(*pTrilops, 7);
triclopsSetSubpixelInterpolation(*pTrilops, 1);

// switch on surface validation
triclopsSetSurfaceValidation(*pTrilops, 1);
triclopsSetSurfaceValidationDifference(*pTrilops, 1.00);
triclopsSetSurfaceValidationSize(*pTrilops, 200);

// switch on uniqueness validation
triclopsSetUniquenessValidation(*pTrilops, 1);
triclopsSetUniquenessValidationThreshold(*pTrilops, 1.44f);

// switch off texture validation
triclopsSetTextureValidation(*pTrilops, 0);
triclopsSetTextureValidationThreshold(*pTrilops, 0.4f);

// switch on back–forth validation
triclopsSetBackForthValidation(*pTrilops, 1);
Appendix C

Fragment Shader for Error Computation

The fragment shader in listing C.1 is used for the error computation to implement the pose estimation algorithm developed in section 6.2. It is implemented in the programming language GLSL, the C-style OpenGL Shading Language.

In chapter 6, we described the encoding of the values to the textures used in the shader, as well as the preparation steps. In figure 6.3, the different states of the error texture channels are visualized, according to the corresponding parts in the code (see // −−− error computation and color channel writing −−−):

In the first case, there is no model depth value since the model does not cover this position. If there is a camera depth value at this point, the boundary-weighted xor error is weighted to the corresponding channel. We discussed in section 5.2 why we don’t assign a squared difference error in this case.

In the second case, both the camera input depth and model depth values are normalized, and the squared difference error computed. If a camera depth value was not available, the boundary-weighted xor error is written to the appropriate channel.

The resulting error texture is evaluated outside the shader using the region weights in the red channel both as weights for the squared difference errors in the green channel, as well as for the normalization of the sum of squared differences. The xor errors in the blue channels are already multiplied by the inverted boundary weights in the shader.
Listing C.1: Fragment Shader

// input textures and variables
uniform sampler2D cTex; // camera texture
uniform sampler2D aTex; // model texture
uniform sampler2D wTex; // region weighting texture
uniform float zmin; // minimum camera depth value
uniform float zmax; // maximum camera depth value
uniform float zminAvg; // minimum model depth value
uniform float zmaxAvg; // maximum model depth value

void main(){
    // variables
    float zC; // depth from camera input
    float zA; // depth from model
    float score; // final error value
    float range; // camera input depth range
    float diff; // camera depth minus minimum
    float zCScaled; // normalized camera depth
    float rangeAvg; // model depth range
    float diffAvg; // model depth minus minimum
    float zAScaled; // normalized model depth
    float w; // normalized Euclidean distance transform
    float wNorm; // weight for boundary-weighted xor
    float wR; // weight for region weighting

    // texture lookups
    zA = (texture2D(aTex, gl_TexCoord[1].xy)).y;
    zC = (texture2D(cTex, gl_TexCoord[0].xy)).y;
    w = (texture2D(wTex, gl_TexCoord[0].xy)).x;
    wR = (texture2D(aTex, gl_TexCoord[1].xy)).x;

    // invert weight for boundary weighted xor
    // i.e. most weight for silhouette pixels
    wNorm = 1.0 - w;

    // error computation and color channel writing
    // if no z-value from model, no ssd error
    if(zA == 0.0){
        // boundary weighted xor error
        if(zC != 0.0){
            // contents of error calculation
        }
    }
}
gl_FragColor = vec4(0.0, 0.0, wNorm, 1.0);

} else{
    gl_FragColor = vec4(0.0, 0.0, 0.0, 1.0);
}

} else{

    //—— normalization of camera input ——
    range = zmin - zmax;
    // circumvent division by 0
    if(range == 0.0) range = 1.0;
    // provoke division by 0
    else if(range < 0.0) range = 0.0;
    if(zC > 0.0){
        // circumvent negative values
        diff = zC-zmax;
        if(diff == 0.0){
            diff = 0.000001;
        }
    }
    else diff = 0.0;
    zCScaled = diff/range;

    //—— normalization of model input ——
    rangeAvg = zminAvg - zmaxAvg;
    if(rangeAvg == 0.0) rangeAvg = 1.0;
    else if(rangeAvg < 0.0) rangeAvg = 0.0;
    if(zA > 0.0){
        diffAvg = zA - zmaxAvg;
        if(diffAvg == 0.0){
            diffAvg = 0.000001;
        }
    }
    else diffAvg = 0.0;
    zAScaled = diffAvg/rangeAvg;

    // squared difference error
    score = (zCScaled-zAScaled) * (zCScaled-zAScaled);

    // boundary weighted xor error
    if(zC == 0.0){
        gl_FragColor = vec4(wR, score, wNorm, 1.0);
    } else{
        gl_FragColor = vec4(wR, score, 0.0, 1.0);
Appendix C. Fragment Shader for Error Computation

```
Bibliography


[Ng 02] J. Ng & S. Gong. *Composite support vector machines for
detection of faces across views and pose estimation*. Image and

[OpenCV ] Intel Research: OpenCV.

approach to recognition memory for spatially transformed faces*.

[Pentland 94] A. Pentland, B. Moghaddam & T. Starner. *View-Based and
Modular Eigenspaces for Face Recognition*. Computer Vision
and Pattern Recognition, pp. 84-91, 1994.

Evaluation Methodology for Face-Recognition Algorithms*.
IEEE Transactions on Pattern Analysis and Machine


Distance Transformation of an n-Dimensional Digitized Picture
with Applications*. Pattern Recognition, vol. 27, no. 11, pp.

[Sheffer 05] Alla Sheffer, Bruno Levy, Maxim Mogilnitsky & Alexander
Bogomyakov. *ABF++: Fast and Robust Angle Based


Cascade of Simple Features*. Computer Vision and Pattern
