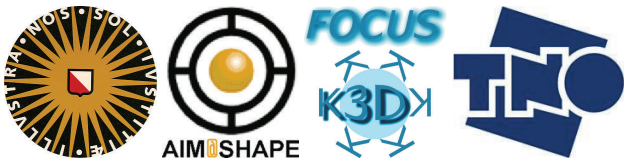
Reconstruction and Analysis of Shapes from 3D Scans



Cover design using MeshLab, a tool supported by 3D-CoForm

Printed in The Netherlands by Proefschriftmaken.nl

ISBN 978-90-8891-109-5

Reconstruction and Analysis of Shapes from 3D Scans

Reconstructie en analyse
van vormen uit 3D scans
(met een samenvatting in het Nederlands)

PROEFSCHRIFT

ter verkrijging van de graad
van doctor aan de Universiteit Utrecht
op gezag van de rector magnificus, prof.dr. J.C. Stoof,
ingevolge het besluit van het college voor promoties
in het openbaar te verdedigen
op maandag 7 september 2009 des middags te 4.15 uur

door

Frank Bart ter Haar

geboren op 1 januari 1980
te Vlaardingen

Promotor: Prof.dr. R.C. Veltkamp

This work was financially supported by the FP6 IST Network of Excellence 506766 AIM@SHAPE - Advanced and Innovative Models And Tools for the development of Semantic-based systems for Handling, Acquiring, and Processing knowledge Embedded in multidimensional digital objects. This work was also supported by FOCUS-K3D FP7-ICT-2007-214993, to Foster the Comprehension and Use of Knowledge intensive 3D media.

Contents

Chapter 1

Introduction

In this thesis, we measure 3D shapes with the use of 3D laser technology, a recent technology that combines physics, mathematics, and computer science to acquire the surface geometry of 3D shapes in the computer. We use this surface geometry to fully reconstruct real world shapes as computer models, and to analyze the characteristics and proportions of human body models for biometric purposes. Tasks that would require a tremendous effort if not done automatically by a computer.

Ever since people walked the earth they were fascinated by measurements and proportions, whether it was on hunting larger animals, gathering enough food, or constructing properly sized shelters. In ancient Egypt, mathematicians already had a grasp of proportions and were able to write down and calculate with fractions. Their computations proved to be a useful tool for their administrative issues, such as the distribution of rations and for the planning of constructions [51]. As the principal unit of measurement they used the royal cubit, which approximated the length of a man's forearm from elbow to the tip of middle finger. With cubit-rods as rulers and ropes of one hundred cubits they could measure differently sized objects and proportion their construction sites. Their measuring tools enabled the ancient Egyptians to conduct basic geometric computations, such as calculating a rectangle's area and estimating the area of a circle by subtracting one-ninth from its diameter and squaring the result [Papyrus Rhind 49/50].

In ancient Greece, mathematicians set the basis for geometry, the mathematical study of points, lines, curves and surfaces. Pythagoras, the father of numbers, discovered the theory of mathematical proportions and Euclid, the father of geometry, wrote the principles of today's Euclidean geometry [44]. Theories that were consistently applied in Greek and Roman architecture as the Roman architect Vitruvius wrote in his book *De architectura*:

> "Geometry affords much aid to the architect: to it he owes the use of the right line and circle, the level and the square" [47].

In his work, Vitruvius carefully describes the esthetically pleasing proportions of the classical architectural orders Doric, Ionic, and Corinthian (Fig. 1.1). He claims that no building can be well designed without proper proportions and symmetry, and that both are as necessary to the beauty of a building as to that of a well formed human figure. Inspired by the detailed description of the human body's proportions, Leonardo Da Vinci drew centuries later the *Vitruvian Man* for the book *De Divina Proportione* on

mathematical and artistic proportion. The human body inscribed in the circle and the square, shows the divine proportions of the human figure which has four fingers in a palm, six palms in a cubit, four cubits in the body's height, which in turn equals the length of the outspread arms.

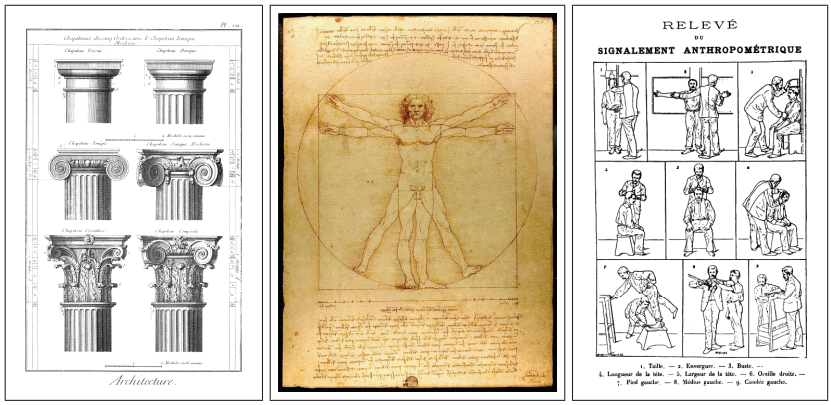


Figure 1.1: Measurements in the classical architectural orders, the divine proportions of the human body, and a standardized identification system.

The fact that there is no such thing as a universal set of proportions for the human body, the field of anthropometry arose. Anthropometry, in which the individual variations of persons are investigated, was put to use for person identification in 1883. By measuring bony structures that remain practically constant during adult life, this system was able to distinguish individuals [83]. In this standardized person identification system, nine body measurements were performed, including the person's height, the length of the stretched arm, the length of the torso from head to seat when seated, the width of the head, and the lengths of the right ear, left foot, left middle finger, and left cubit (Fig. 1.1).

In biometrics, all methods to uniquely identify and verify a person are studied. Handwriting, signatures, and body measurements are among the first methods to identify a person or his work. Nowadays, to claim our identity and to protect our (digital) information we can even use two-dimensional face images, finger prints, ear prints, hand prints, voice recordings, gait recordings, retina scans, iris scans, and three-dimensional laser range scans of the body or face. Each of these biometric techniques captures a physical property that cannot be forgotten or mislaid like a password [2].

In this thesis we focus on 3D laser scans for its wide range of applications. A 3D laser range scan can be best compared to a 2D photograph which captures the depth of objects in a scene instead of their color information. This recent measuring instrument projects a laser dot on an object's surface, captures the reflected light with a sensor and measures the object's distance using the generalization of Pythagoras' theorem. With thousands of depth measurements in a second, the geometry of the object's surface is automatically captured and represented as points, lines, curves and surfaces on the computer. Where a 2D photograph loses the depth information in a scene and thus the true proportions of objects, a 3D range scan does not. This makes the 3D range scan a valuable measuring tool for proportions of the human face for biometric identification systems, for proportions of the human body for anthropometric studies, and for the digital reconstruction

of historical buildings and statues. This thesis deals with the difficulties of the automatic acquisition, reconstruction and analysis of shapes with the use of 3D laser range scanners.

The application of 3D laser range scanning and the acquired 3D shapes are certainly not limited to biometrics, anthropometry, and cultural heritage, but are also regularly used in the entertainment industry, geographical information systems, and robotics. *Biometrics* is a recent application domain for 3D range scanning. By constructing a database of 3D face scans of personnel, trespassers can be rejected from entering the building by automatically scanning and comparing their 3D face scans to those in the authorized personnel database. Scans of the human body are used in *anthropometry*. The statistical analysis on body proportions in a population allows for custom fit products, such as prostheses, costumes, gas masks, ergonomic office chairs and car seats. Instead of the traditional plaster cast to properly fit a prosthesis to the body surface, a 3D range scan could suffice. This also applies to *cultural heritage*, where a plaster cast might damage ancient artifacts. The acquisition and reconstruction of such artifacts with 3D scanners enables the preservation of their current shape. In the *entertainment industry*, 3D modeling software is used to develop characters, objects and scenes. Instead of directly modeling new animation characters on the computer from scratch, they can be modeled with clay, scanned, and reconstructed to acquire its computer model. *Geographical information systems* are shifting from 2D image data to 3D worlds for advanced urban planning and disaster simulation. In *robotics*, an automated robot can be equipped with a 3D laser range scanner to detect its surroundings and plan its path. For each of these applications, the reconstructed 3D shapes must satisfy requirements such as accuracy, completeness and level of detail. In the field of computation geometry, researchers develop algorithms to fulfill such geometric requirements on shapes.

1.1 Acquisition, reconstruction and analysis of 3D shapes

The acquisition and reconstruction of 3D shapes is the process of making a surface geometry description of a 3D object based on acquired data points. To construct a 3D model out of a real world object, the object's surface can be digitized with the use of a laser range scanner. Such a scanner measures the distance from the scanner to the object. The most common laser range scanners can only scan one side of an object at a time, and only those parts of the object's surface that are not occluded and are within the scanner's range. To capture more data of the object's surface, the object has to be scanned from many different sides and for several poses. Because each scan is generated with respect to the scanner's coordinate system, they need to be transformed into a common coordinate system in which they are aligned to each other. A common approach to obtain such an alignment of range scans is to interactively align pairs of overlapping meshes in such a way that the proportions of the model correspond to those of the object. The partially overlapping scans can be merged into a single geometric surface model and improved with geometric operations, such as smoothing, hole filling, and remeshing. The additional challenge is that the acquired surface data can have wrong points, missing points, or not enough points to accurately describe the critical surface regions. For the proper interaction, analysis, and printing of 3D computer models, special require-

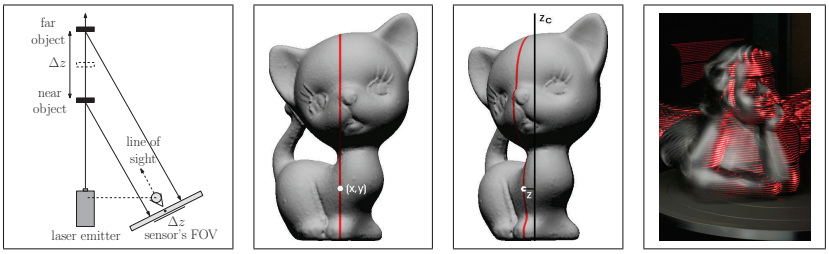


Figure 1.2: The optical triangulation system in a laser range scanner (a). The projected laser line from (b) the laser's point of view and (c) the sensor's point of view. Photograph of the scanning process with long exposure time (d).

ments on geometry, topology and physics are desired for which algorithms are being developed.

1.1.1 3D range scanning

To construct the 3D geometry of a real world object, the object's surface can be digitized with the use of 3D range scanners. These scanners measure the distance from the scanner to the object's surface. One of the most popular three-dimensional scanning device is the *laser range scanner* [38, 14, 62]. During the acquisition these scanners project either a laser dot or laser stripe on the object's surface. A sensor, often a CCD camera, looks at the scene and detects the dot or stripe at the peak of reflected laser light. The location of these peaks in the sensor's field of view (FOV) depend on the distance between the laser emitter and the projected dot or stripe (Fig. 1.2). The laser emitter, the projected laser dot, and the sensor form a triangle. In this optical triangulation system the direction of the laser light, the sensor's line of sight, and the sensor's position with respect to the laser emitter are known. With this knowledge and the detected laser dot with respect to a calibrated depth in the sensors field of view, the depth of a laser dot can be determined accurately. The surface normal at an extracted 3D surface point can be estimated using the direction of the laser light and the sensor's line of sight.

Various types of laser range scanners have been developed. The most common laser range scanners scan one side of an object at a time (*plane scan*) producing a 2D image storing per pixel the distance from the corresponding laser dot to the object's surface. These images are called depth images or range scans (Fig. 1.3). Another laser range scanner producing depth images is the rotation scanner. These scanners rotate either the object or itself around the y-axis while scanning vertical stripes (*cylindrical scan*). Generated depth images store the depth per height and rotated angle (Fig. 1.3).

Other frequently used 3D range scanners include the structured light and the time-of-flight scanners, the first for its acquisition speed and the latter for its ability to scan over long distances. Both scanners acquire plane scans. A *structured light scanner* uses a projector to project a pattern of light on the object [85, 122]. Similar to the laser range scanner the structured light scanner uses a sensor to retrieve the projected pattern (Fig. 1.4). The distortion of the color transition pattern is used to compute 3D information of an entire view of the object. The location of a color transition in the sensor's field of

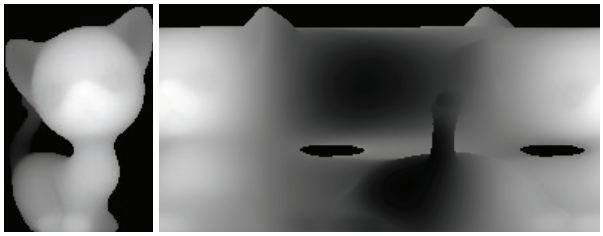


Figure 1.3: A depth image or range scan acquired from a plane scan (left) and a rotation scan (right).

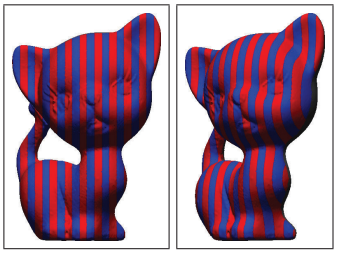


Figure 1.4: The projected structured light from the laser's point of view (left) and the sensor's point of view (right).

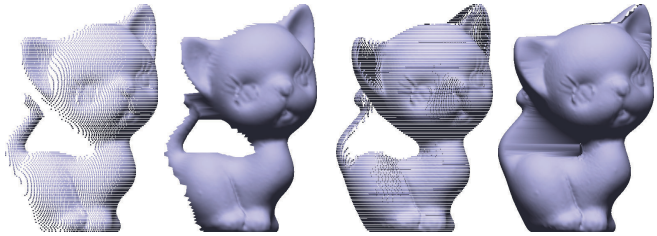
view depends on the distance between the projector and the object's surface. So, when an object has large depth transitions, parts of the colored stripes may appear to run out of phase. This introduces an ambiguity in finding the correct order of stripes, which may result in retrieving incorrect 3D information. The use of a multiple color pattern helps in retrieving the correct order of stripes [122]. Structured light systems are very fast, because multiple stripes (i.e. color transitions) are extracted at once. A time-of-flight scanner sends out very short pulses of light and measures the time for the light to be sensed by a sensor [116]. Knowing the speed of light and the time-of-flight, the distance to the object can be calculated.

Scanners that extract 3D surface data in an unstructured manner cannot produce a depth image. An example of such a scanner is the hand-held scanner in combination with a tracking device. These scanners enable the user to extract 3D surface data from all around the object, while the tracking device helps to place each 3D point in the common coordinate system. The result is a rather complete 3D point cloud with estimated normals.

Combinations of the fast structured light scanners and fast stereo vision techniques are investigated [117] for real-time and high resolution scan applications. The scientific challenge in 3D scanning is to optimize the scanner's acquisition time and usability within its acquisition range.

1.1.2 Surface mesh creation

The advantage of a depth image or range scan is the ease with which the intrinsically stored 3D point cloud is converted into a triangular surface mesh (Fig. 1.5). This is simply done by connecting the 3D points that represent adjacent pixels in the depth image.



(a) plane scan (b) rotation scan

Figure 1.5: 3D point cloud and its 3D surface mesh from a single plane scan (a) and rotation scan (b).

In case of a *plane scan*, a 3D surface mesh from one side of the object is acquired. To extract the entire surface of an object, the object has to be scanned from many different sides and for several poses. In case of a *rotation scan*, a more complete 3D surface mesh can be obtained in a similar matter. A major drawback of rotation scanners is that the object should fit the cylinder described by the rotating scanner or table. Again multiple scans are required to capture the object's top and bottom and concavities in the object's surface. When the connectivity of a 3D point cloud is unknown the entire surface has to be built from scratch using only the acquired 3D point data and their estimated normals.

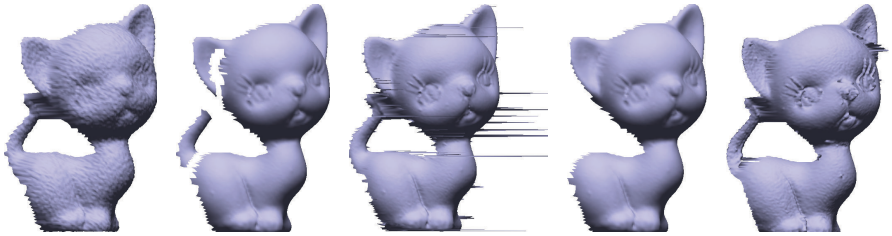
In case a 3D laser scanner produces a depth image, the surface mesh creation is a trivial problem. The triangulation of a 3D point cloud, on the other hand, requires geometric algorithms that find the optimal connectivity among the imprecise scan data.

1.1.3 Surface mesh quality

The quality of a surface mesh extracted from a laser range scan depends on the properties of both the laser range scanner and the object itself. The two basic scanner properties that influence the acquired surface quality are the resolution and the accuracy. The *resolution* is the smallest distance between two points that the scanner measures. The *accuracy* is how close a measured value is to the true value. The resolution is related to the size of a laser dot or the width of a laser line, the distance to the object, and the sensor's FOV resolution. The accuracy depends on the ability to correctly measure the coordinates from the sensor's FOV. For the optical triangulation, the object's surface must reflect the projected laser light towards the sensor.

When the resolution is too low, the laser scanner fails to capture the tiny surface details. Limitations in the scanner's accuracy causes speckle noise on the surface mesh. The object's properties, such as color, material, and shape have their effect on the optical triangulation system. When the object's surface is too dark, too transparent, too specular, or protrusions of the object block the reflected laser light (self occlusion), then the sensor is unable to detect the laser light. In that case, optical triangulation is no longer possible and the surface misses data. Specular reflection to wrong locations in the sensor's FOV results in incorrect measurements causing outliers. To some extend, the surface acquisition can be improved by reducing the environmental lighting, decreasing the laser strength, and making the object's surface more diffuse.

Because each scanner is designed to scan objects at a calibrated distance and for an



(a) noise (b) occlusion (c) outliers (d) cut-off (e) accurate scan

Figure 1.6: The range scan quality suffers from limitations in laser acquisition.

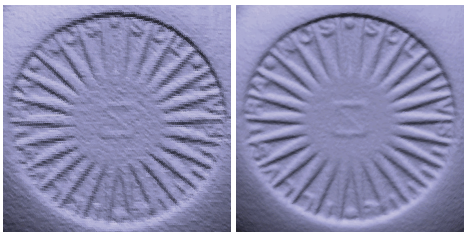


Figure 1.7: The reconstructed surface of the Utrecht University logo from a single range scan (left) and eleven range scans (right).

optimal resolution, they usually have an effective sensor area in which they operate. Outside this area, data is often discarded or simply not measured. This cut-off distance is another cause of missing data. Altogether, these limitation in laser range scanning and object properties result in surface meshes with noise, outliers and missing data (Fig. 1.6). Simple solutions are available to improve on the surface quality. Many outliers can be eliminated by removing slender triangles and removing the vertices that become disconnected. The level of noise can be reduced by smoothing the surface. However, care should be taken not to loose tiny details. A better but more time consuming solution to reduce the level of noise is to acquire many nearly identical (low) resolution range scans of a single view, to combine the data and extract a single super-resolution surface [58]. Such a super resolution range scan levels out the noise and enables the acquisition of tiny details (Fig. 1.7). To cover the whole object's surface and to fill in missing data, an object has to be scanned from different sides and angles and for several poses. This might not be possible due to fixed or heavy objects or stationary laser systems.

To improve on the final surface mesh quality, more advanced 3D laser systems are being developed with higher resolution, higher accuracy, and user-feedback on missing data.

1.1.4 Surface mesh alignment

Each range scan is generated with respect to the scanner's coordinate system. To recover the object's shape, they need to be transformed into a common coordinate system in which they are aligned to each other (Fig. 1.8). A common way to obtain such an alignment of range scans is to interactively align pairs of overlapping meshes. This coarse alignment is then refined by a geometric algorithm that brings all the overlapping

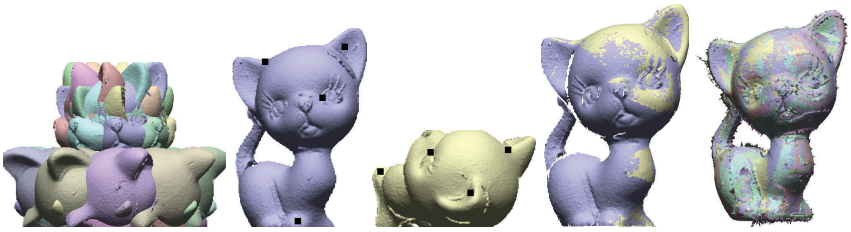


Figure 1.8: The meshes in the scanner's coordinate system (left), can be interactively aligned by clicking corresponding points on the surface, to recover the object's shape (right).

surface meshes closer to each other, which is an intractable task for a user.

While this refinement is always performed automatically, the *coarse alignment* is performed either interactively or automatically. An *interactive pairwise* alignment of meshes is performed by a user, who decides which pairs of meshes have parts of their surface in common. He or she then either selects a few corresponding points on the common surface of two meshes, or manually rotates and translates one mesh toward the other, to bring them into alignment. An *automatic pairwise* approach will try to find a set of corresponding points on two meshes automatically. When enough correspondences are found, the two meshes are brought into alignment. A problem with the automatic pairwise approach occurs when incorrect correspondences are selected to align meshes, with an unsuccessful alignment as a result. An *automatic multiview* approach will try to solve this with the use of a global consistency check for pairs of meshes with high correspondence.

The coarse alignment is hard to automate because each scan covers only a part of the object, contains noise, and is usually captured from an unknown point of view. The challenge in surface mesh alignment is the development of automatic alignment algorithms that effectively and efficiently align large sets of meshes for fast object reconstruction and user-feedback during the acquisition phase.

1.1.5 Surface mesh integration

When an accurate fine alignment of meshes is obtained, a merge method can be applied to integrate the partially overlapping meshes and construct a single surface mesh (Fig. 1.9). Many algorithms have been developed based on either the aligned meshes (structured data) obtained from the range scans, or based on the point cloud defined by the aligned range scans (unstructured data). Algorithms based on the former can reuse the triangles and focus on stitching and blending the overlapping surface. The methods that build a new surface mesh based on the unstructured point cloud data require much more effort. An important property for an integration algorithm is the invariance to the surface mesh quality.

1.1.6 Surface mesh completion

It is hard and sometimes even impossible to acquire the entire surface of an object with a laser range scanner. Remaining holes in the surface mesh can be filled using hole filling

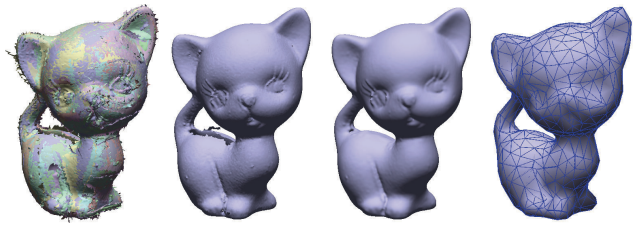


Figure 1.9: The meshes in their final fine alignment are merged into one surface, the holes in the surface are filled, and the surface simplified for fast rendering.

techniques based on the triangulation of boundary components [9], volumetric repair [39, 55] and surface interpolation [93]. Surface reconstruction techniques that ensure the output of a watertight surface mesh have been proposed as well (e.g. [4, 40]). A watertight surface is a surface without holes, so that imaginary water inside cannot get out. In many applications, such as CAD modeling and 3D printing, watertightness is required. For faster rendering of 3D models in virtual environments the final surface can be simplified [45] by reducing the number of points and triangles (Fig. 1.9).

1.1.7 Surface mesh analysis

As laser range systems and geometric algorithms to process the acquired scan data are becoming more advanced, the acquired and reconstructed 3D shapes are reaching high levels of detail and accuracy. Instead of measuring the real object with traditional instruments, the object can now be scanned and measured rapidly and even reproduced by the computer itself (Fig. 1.10). In anthropometry, 3D human body scans allow for the analysis of human proportions in a population. With the acquired body surface the height and center of mass can be determined automatically, but also more complicated measurements such as the distance over the surface (geodesic distance) from one location to another can be computed. Such body measurements can be used to determine the average body shape and the range of most common body shapes in a populations. These statistics are useful in industrial design, where the development of ergonomic and custom fit products plays an important role.

In biometrics, such geodesic and Euclidean distances between distinct locations (landmarks) can even be used to identify individuals. By constructing a database of 3D face scans of personnel as we do in this thesis, a computer can automatically match a new face scan to a person in the database with the use of these landmarks. When none of the face scans sufficiently matches the new scan, then the new scan is unknown to the system. This strategy can be applied to automatically reject trespassers from entering a secured building.

Since the acquisition and reconstruction of 3D shapes from laser scan data becomes easier, the number of 3D shapes in databases increases as well as the need to effectively query for a particular shape. The analysis of (local) surface properties helps to retrieve 3D shapes similar to a query shape or query description, which helps the user to reuse and combine shapes.

Figure 1.10: Colored geodesic measurements on the author's reconstructed body surface effectively show the boundaries of the model. When the surface is made watertight, a 3D laser printer can turn the model into a real world object.

1.2 Contributions of this thesis

In this thesis we use 3D laser range scans for the acquisition, reconstruction, and analysis of 3D shapes. In Chapter 2, we study the acquisition and reconstruction pipeline from a real object to a 3D surface model. Based on ground truth models we compare the accuracy of various mesh acquisition, mesh alignment, and mesh merge systems and the influence on one another. Also algorithms to fill the remaining holes in the merged 3D surface model are tested. No quantitative comparison of these system's accuracy has been done before, which is of practical importance for the reconstruction of accurate 3D surface models.

The main bottleneck in the acquisition and reconstruction pipeline is the coarse alignment of the unordered surface meshes, which requires intensive user-interaction. Many algorithms have been proposed that align two *overlapping* meshes, but the main difficulty is to find those meshes that actually have overlap and then to align all meshes simultaneously in a globally correct way. Automating this process in an efficient manner is essential for fast object reconstruction. We present in Chapter 3 a new multiview alignment algorithm that performs both the coarse and fine alignment of unordered sets of range scans both effectively and efficiently.

In Chapter 4, we use 3D laser range scans for biometric purposes. By scanning one's face surface with a laser range scanner, we acquire the 3D surface geometry of the face which differs from one person to another. With geometric algorithms we can analyze and compare such 3D face scans, which are the core elements in 3D face recognition systems. Such a system stores 3D face scans with known identity in a database and compares a new face scan to all of them to find the most similar ones. The best match can be used to identify the new face scan or declare it unknown if the best match is not similar enough. A 3D face identification system overcomes several limitations of existing 2D face identification systems, but with the same user-friendliness. We present a complete 3D face recognition pipeline, that automatically detects, segments, and improves the face surface and does the face matching with profile and contour curves. To select effective combinations of curves as a compact representation of the face for the recognition, a 3D curve matching framework is designed. This framework extracts curves with different properties and evaluates these curves for their ability to identify persons.

In Chapter 5, we present an algorithm to automatically fit a 3D morphable face model to 3D scan data of faces for their recognition. A 3D morphable face model is a statistical model built from example faces and linearly interpolates between these faces to construct new ones. With the use of model coefficients, the model deforms along the principal axes of data variance, which changes for instance the size of the face, the position of the eyes, or the length of the nose statistically correct. By fitting this model to the scan data, we create a new face instance without holes nor noise, which are often a problem for 3D face recognition algorithms. We evaluate the accuracy of the automatically fitted face instances and show that our algorithm is more accurate than existing methods. By dividing the face model into multiple face components, we further improve on the model fitting accuracy. By assigning anthropometric landmarks (such as the nose tip and eye corners) to the morphable face model once, they are automatically morphed towards their statistically reliable locations in different face scans. Furthermore, the way the morphable model is deformed to fit the scan data provides geometric clues of the face, which are captured by the model's coefficients. For the recognition of 3D faces, we use the contour curves from Chapter 4, the automatically detected landmarks, and the acquired model coefficients and show the superior performance of the latter with recognition rates up to one hundred percent on the UND dataset.

In Chapter 6, we present a new algorithm to automatically enhance the 3D morphable face model with new face data. For a 3D face recognition system based on model coefficients it is important that the properties of many realistic faces are captured in the model. In case a face cannot be modeled, the automatically acquired model coefficients are less reliable, which may hinder the face identification. The bootstrapping algorithm that we present automatically detects if a new face scan cannot be sufficiently modeled, establishes full point correspondence between the face scan and the model, and enhances the model with this new face data. New in this is the use of multiple face components to fit the model more accurately to scan data, a correspondence repairing algorithm, and an automatic check to avert redundant face properties.

In Chapter 7, we perform expression invariant face recognition by incorporating expression-specific deformation models in our face modeling approach. In a global to local fitting scheme, the identity and expression coefficients of this model are adjusted

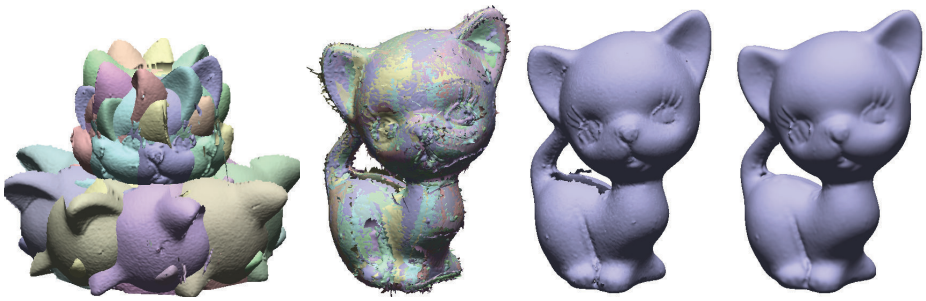
such that the produced face instance accurately fits the 3D scan data of the face. Quantitative evaluation shows that the expression deformation as well as a set of predefined face components improve on the fitting results. 3D face matching experiments on the publicly available UND, GAVAB, BU-3DFE, FRGC v.2 datasets show high recognition rates of respectively 99%, 98%, 100%, and 97% with the use of the identity coefficients. Results show that not only the coefficients that belong to the optimized model fit perform well, but that the coefficients of four locally optimized model fits can produce similar recognition rates. Finding the optimal model fit is hard and loosening this requirement could make a system more robust.

1.3 Relevant publications

The publications on which this thesis is based are listed below.
F.B. ter Haar and R.C. Veltkamp. 3D Face Model Fitting for Expression Invariant Recognition. *In preparation*.
F.B. ter Haar and R.C. Veltkamp. Automatic Bootstrapping of a Morphable Face Model using Multiple Components. In *3-D Digital Imaging and Modeling (3DIM)*, 2009.
F.B. ter Haar and R.C. Veltkamp. 3D Face Model Fitting for Recognition. In *European Conference on Computer Vision (ECCV)*, pages 652-664, 2008.
F.B. ter Haar and R.C. Veltkamp. A 3D Face Matching Framework for Facial Curves. *Graphical Models*, 71(2):77-91, 2009.
F.B. ter Haar and R.C. Veltkamp. A 3D Face Matching Framework. In *Shape Modeling and Applications (SMI)*, pages 103-110, 2008.
F.B. ter Haar and R.C. Veltkamp. Automatic Multiview Quadruple Alignment of Unordered Range Scans. In *Shape Modeling and Applications (SMI)*, pages 137-146, 2007.
T. Whitmarsh, R.C. Veltkamp, M. Spagnuolo, S. Marini, and F.B. ter Haar. Landmark Detection on 3D Face Scans by Facial Model Registration. In *1st Int. Workshop on Shape and Semantics*, pages 71-76, 2006.
F.B. ter Haar, P. Cignoni, P. Min, and R.C. Veltkamp. A Comparison of Systems and Tools for 3D Scanning. In *3D Digital Imaging and Modeling: Applications of Heritage, Industry, Medicine and Land*, 2005.
F.B. ter Haar and R.C. Veltkamp. Alignment, Merging and Hole Filling Experiments with 3D Range Scans. Technical Report UU-CS-2005-047, Utrecht University, 2005.
R.C. Veltkamp and F.B. ter Haar. SHREC2008: 3D Shape Retrieval Contest. In *Shape Modeling and Applications (SMI)*, pages 215-263, 2008.
F.B. ter Haar, M. Daoudi, R.C. Veltkamp and F.B. ter Haar. SHape REtrieval Contest 2008: 3D Face Scans. In *Shape Modeling and Applications (SMI)*, pages 225-226, 2008.
R.C. Veltkamp and F.B. ter Haar. SHREC2007: 3D Shape Retrieval Contest. Technical Report UU-CS-2007-015, Utrecht University, 2007.
R.C. Veltkamp, R. Ruijsenaars, M. Spagnuolo, R. van Zwol, and F.B. ter Haar. SHREC2006: 3D Shape Retrieval Contest. Technical Report UU-CS-2006-030, Utrecht University, 2006.

Chapter 2

Acquisition and reconstruction quality



Many systems and methods have been developed to aid the user in the acquisition and reconstruction pipeline from a real world object to a 3D computer model. This chapter presents a comparison of systems for the acquisition, alignment, merging and hole filling of 3D scan data. No quantitative comparison of such systems has been done before, which is of practical importance for the reconstruction of accurate 3D surface models. In our comparisons, we measure the difference of a system's output to reference models. Both actual scans (from physical objects) and virtual scans (from 3D models) are used in the evaluation. Our results quantify that acquisition, alignment, and merge systems differ in accuracy even when they are based on similar algorithms or heuristics. We show that (1) a more precise alignment of meshes will result in more accurately merged models, (2) that the best performing merge method for virtual scans turns out to perform worst for actual scans, (3) that the optimal resolution of volumetric merge methods is bounded by the level of noise in the range scans, and (4) that hole filling remains a challenging problem.

2.1 Introduction

The process of reconstructing a 3D computer model out of a set of range scans has been a well studied field of research for several decades. During these years various techniques

and software tools were developed to aid the reconstruction of a 3D model, based on two different types of reconstruction sequences.

The first type is to turn the range scans directly into meshes, which is often done automatically by the scanning software, and then to perform the alignment and merging of these meshes to obtain the 3D model. The second type is to align the range scans first and then to reconstruct the surface from the unorganized set of 3D points [110]. We focus on the first type of reconstruction sequence and assume acquisition of surface meshes rather then unorganized point clouds.

The alignment of meshes consists of a coarse and a fine alignment step. During the *coarse alignment*, a transformation for each of the meshes is found to place them in a common coordinate system in which they are coarsely aligned to each other. During the *fine alignment*, the relative positions of the coarsely aligned meshes are optimized automatically. The coarse and fine alignment both distinguish a *pairwise* and a *multiview* approach. The pairwise approach finds a transformation for one pair of meshes only, while the multiview approach is characterized by finding transformations for all meshes simultaneously.

An *interactive pairwise* alignment of meshes is performed by a user, who decides which pairs of meshes have parts of their surface in common. He or she then either selects a few corresponding points on the common surface of two meshes, or manually rotates and translates one mesh towards the other, to bring them into alignment.

An *automatic pairwise* approach will try to find a set of corresponding points on two meshes automatically. When enough correspondences are found, the two meshes are brought into alignment. Several techniques have been developed to perform the pairwise alignment of meshes automatically, including: the exhaustive search for corresponding points [28, 30] and the use of surface signatures such as spin-images [54], point signatures [31], bitangent curves [119], spherical attribute images [48], and volumetric grids of local surface [72].

Methods to perform the coarse alignment according to the *automatic multiview* approach includes work of Huber and Hebert [50], Novatnack and Nishino [77], and Mian et al. [70]. These methods find the pair-wise alignment of all meshes according to the *automatic pairwise* approach. Then they construct either a correspondence graph or correspondence tree in which a node represents a mesh and an edge represents the pair-wise transformation between them. Edges with high confidence and that pass a global consistency check of the structure are added to end up with a connectivity that represents the coarse multiview alignment.

The most popular method for the *fine alignment* of coarsely aligned meshes is the Iterative Closest Point (ICP) algorithm, which was introduced by Chen and Medioni [29], and Besl and McKay [16]. It starts with an initial guess for the relative rigid-body transformation of two meshes obtained from the coarse alignment. Then the algorithm iteratively refines this transformation by repeatedly selecting pairs of closest point correspondences on the meshes while minimizing an error metric. Because the method operates on one pair of meshes only, we will refer to it as the *pairwise ICP* algorithm. Many variants of the pairwise ICP algorithm have been introduced [88]. The alignment of several pairs of meshes can be performed by applying the pairwise ICP algorithm sequentially to all pairs of overlapping meshes, which may result in the accumulation of alignment errors. To avoid this several techniques have been developed to finely align

multiple pairs of meshes at once rather than single pairs [12, 76, 82, 96]. The basic goal of these *multiview ICP* methods is to spread the alignment error evenly across the available mesh pairs.

The *merging* of a set of aligned range scans remains a challenging problem. Many techniques have been developed based on either the aligned meshes obtained from the range scans, or based on the point cloud defined by the aligned range scans. The two main approaches for the merging of aligned meshes are surface zippering [107], and the volumetric merge based on a discrete distance field [37]. Surface zippering is an algorithm that detects surface overlap, removes the redundant surface patches, and zippers the surface boundaries. Volumetric merge methods construct a 3D volumetric grid in which each surface mesh is discretized as a weighted distance field. By blending the weighted distance fields together a new zero level set is acquired in the volumetric grid. From the zero level set, the merged surface mesh can be extracted with the Marching Cubes algorithm [64]. The volumetric approach has several variants (summarized in [84]). For the construction of a surface out of point clouds (e.g. aligned range scans) popular techniques include the moving least-squares (MLS) surface approach [7], the use of radial basis functions [25], ball-pivoting [13], and stochastic surface reconstruction [89].

Remaining holes in the surface mesh can be filled using hole filling techniques based on the triangulation of boundary components [9], volumetric repair [39, 55] and surface interpolation [93]. Surface reconstruction techniques that ensure the output of a watertight surface mesh have been proposed as well (e.g. [4, 40]).

Previous comparisons focused on pairwise variants of the ICP algorithm [88], and on multiview ICP variants [36, 41]. Also, several overviews on the 3D acquisition and reconstruction process have been published [14, 24, 124]. In our previous work [99], we created an initial setup to compare a few systems using (non-optimized) default settings.

2.1.1 Organization of this chapter

In this chapter, we evaluate methods that are used in the acquisition and reconstruction pipeline from object to 3D model (Fig. 2.1). To do so we use a set of four physical objects and three existing 3D models. The physical objects are the *cylinder*, *box*, *pierrot*, and *memento* shown in Fig. 2.2. The existing models are the *armadillo*, *dragon*, and *knot* shown in Fig. 2.3. These objects were selected for their different properties in shape, appearance and manufacturing: The *cylinder* and *box* are metal objects from the car industry both created with high precision. The *pierrot* and *memento* are small objects (60mm and 110mm high) with smooth parts and small details. The *armadillo* and *dragon* are reconstructed models from 60 to 70 range scans using VripPack downloaded from the Stanford 3D Scanning Repository [95]. The *knot* is a model constructed using 3D Studio MAX with many occlusions that would make it difficult to scan with a range scanner if it were a physical object.

From the physical objects we acquire range scans using laser range scanners and for the existing 3D models we simulate this scanning process. To evaluate the results at different stages of the pipeline we use reference models. For the existing 3D models, we simply use the original 3D model as a reference. For the physical objects we use

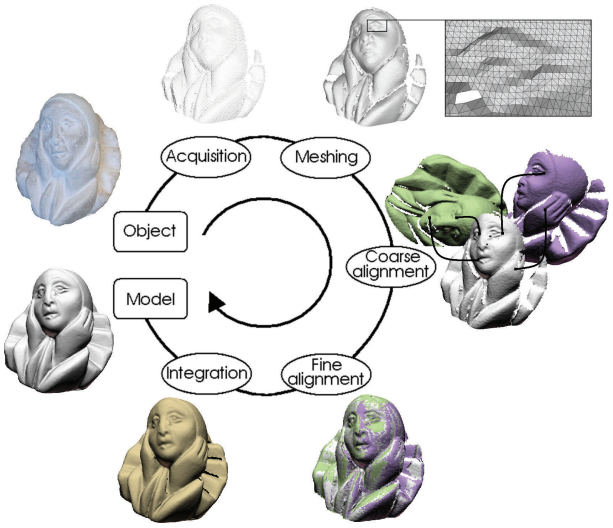


Figure 2.1: The acquisition and reconstruction pipeline. The results of the coarse alignment, fine alignment and merging are evaluated in the comparison.

Figure 2.2: Real objects *cylinder*, *box*, *pierrot*, and *memento* used for the evaluation of reconstruction systems.

Figure 2.3: Existing models *armadillo*, *dragon*, and *knot* used for the evaluation of reconstruction systems.

accurately constructed or reconstructed reference models as explained in the following sections.

The evaluation is performed using the Root Mean Square (RMS) distance, computed with Metro [32], which is a mesh comparison tool able to compute the approximated RMS distance between two aligned 3D models. Metro does this by selecting a set of samples $(p_0, \ldots, p_n)$ on the first model (M_1) and determining the minimal Euclidean distance (e_{min}) of each sample to surface samples on the second model (M_2).

$$e_{min}(p, M_2) = min_{p' \in M_2} d(p, p')$$

These samples may include vertex, edge and/or face samples. The distances of these samples are used to compute the RMS distance, which we employ to determine the accuracy of our models with respect to the reference models.

$$d_{rms}(M_1, M_2) = \sqrt{\frac{1}{n} \sum_{i=1}^{n} e_{min}(p_i, M_2)^2}$$

The evaluation of acquisition and alignment systems is based on the acquired data (vertices) and after the merging additional samples on the triangles and edges are considered as well. We also perform qualitative evaluation to support the quantitative evaluation.

In the following sections we evaluate the accuracy of two systems for the acquisition, three systems for the alignment, and five algorithms for the merging of range scans. Finally, six algorithms are investigated to fill holes in the integrated surface mesh.

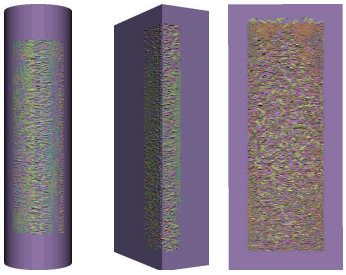
2.2 Acquisition accuracy

This section describes the evaluation of two laser range acquisition systems based on optical triangulation. The evaluation is performed using scans from two objects from the car industry, which were produced with high accuracy.

2.2.1 Experimental setup

We have access to two laser range systems for the data acquisition, namely the *Minolta VI-900* and the *Roland LPX-250*. The former sweeps a laser line over the surface, while the latter moves the laser emitter to acquire the 3D position of each projected laser dot. The two laser range scanners were used to scan the objects *box* and *cylinder*, which are metal objects from the car industry both created with high precision. Both objects were spray-painted white to improve on their reflective properties. To compare the accuracy of the *Minolta* and *Roland* laser range scanners, we compare their acquired range data to reference models. We measured the *box* and *cylinder* using a sliding gauge with a sub-millimeter precision. According to these measurements 3D reference models were created.

The *box* was scanned from an edge view and a plane view, and the *cylinder* from one side only. Each view was scanned with the *Roland* scanner using the two highest resolutions 0.4×0.4 mm and 0.2×0.2 mm. For the *Minolta* scanner the resolution depends on



(a) cylinder (b) box edge (c) box plane

Figure 2.4: The regions of interest of the nine range scans (in different colors) aligned to their reference. Vertices of the range scans are used for evaluation.

Object	Scanner	Resolution	RMS distance			
			scan 1	scan 2	scan 3	average
cyl	Minolta	0.22×0.22	0.023	0.023	0.023	0.023
edge	Minolta	0.18×0.18	0.026	0.026	0.026	0.026
plane	Minolta	0.18×0.18	0.038	0.035	0.034	0.036
cyl	Roland	0.2×0.2	0.067	0.066	0.066	0.066
edge	Roland	0.2×0.2	0.054	0.055	0.054	0.054
plane	Roland	0.2×0.2	0.057	0.057	0.057	0.057
cyl	Roland	0.4×0.4	0.070	0.068	0.069	0.069
edge	Roland	0.4×0.4	0.056	0.055	0.057	0.056
plane	Roland	0.4×0.4	0.057	0.055	0.056	0.056

Table 2.1: RMS distance (in mm) of the range data to their reference models.

the object size in the sensors field of view, obtaining the box with a resolution of approximately 0.18×0.18 mm and 0.22×0.22 mm for the cylinder. Each view was scanned three times to compute the average accuracy. All scans were aligned to their reference model with the ICP algorithm using the RMS distance and the accurate point-to-plane distance of all vertices to the reference model. Regions of interest were selected and aligned to the reference models for a second time to obtain the best achievable alignment for the range data in these regions (Fig. 2.4). The final RMS distance after convergence of the ICP algorithm is used to evaluate the average accuracy of the two laser range scanners (Table 2.1).

2.2.2 Results

Results show that the accuracy of the Roland scanner is similar for its two highest resolutions 0.4×0.4 mm and 0.2×0.2 mm. With an average accuracy up to 0.069 mm the highest resolution of the Roland scanner tends to produce noisy surfaces. Therefore, we prefer to use the Roland scanner at a 0.4×0.4 mm resolution. The Minolta scanner is able to produce high resolution surfaces up to 0.18×0.18 mm with twice the accuracy of the Roland scanner.

Other properties in favor for the Minolta scanner are its acquisition speed, portability longer acquisition range, and the ability to capture the texture. The Roland scanner is limited to a rotation table, but allows both plane scanning and rotation scanning and generates orthogonal plane scans.

(a) pierrot (b) memento (c) armadillo (d) dragon (e) knot

Figure 2.5: Reference models used for the evaluation of reconstruction systems.

2.3 Alignment accuracy

This section describes the evaluation of three fine alignment systems based on similar heuristics, that is a multiview ICP algorithm. The evaluation is performed using range data from two physical objects with highly accurate reference models and synthetic range data from three existing models.

2.3.1 Preprocessing

We use the scans of the *pierrot*, *memento*, *armadillo*, *dragon* and *knot*. To evaluate different alignments using reference models, we need to produce reference models for the object *pierrot* and *memento*. For the other objects we can simply use the original 3D model. To create the highly accurate reference models, the physical objects were scanned using the *Minolta Vivid 900* laser range scanner. The *pierrot* was scanned from eight views for three different poses creating 24 range scans with a resolution of 0.18×0.18 mm in the process. For the more complex *memento*, we generated twelve range scans with a resolution of 0.24×0.24 mm for three different poses. After the interactive coarse alignment and automatic fine alignment of these scans using MeshAlign, the meshes were merged using the volumetric merge method MeshMerge with 0.2 mm^3 sized voxels. Afterwards, the reconstructed surface mesh was cleaned and filled interactively to obtain a watertight reference model. The highly accurate reference models used in this section are shown in Fig. 2.5. The reference models and their scans are publicly available in the AIM@SHAPE shape repository [1].

For all objects we acquired four range scans from five different poses. For the physical objects we used the *Roland LPX-250* laser range scanner at a resolution of 0.4×0.4 mm to acquire this set of 20 range scans. This scanning process is simulated for the models *armadillo*, *dragon*, and *knot* using the same resolution after resizing these models to a height similar to the physical objects (100 mm). We did not simulate scanner noise when we created the virtual scans. Because we acquired less range data with a less accurate laser range system, we can assume that the generated reference models suffice for the comparison of systems based on similar heuristics.

To remove most of the incorrect faces, our meshes were cleaned similar to [54]: faces with a normal almost perpendicular to the scan direction are likely to be wrong, so when the angle between a face's normal and the scan direction is larger than a threshold t_α

(e.g. 80°) the face is removed. Then faces with an edge longer than a threshold t_e (e.g. $4\times res$=1.6) are removed, and finally the disconnected vertices and the small patches with less than t_p (e.g. 100) faces are removed as well.

The systems for the fine alignment considered in this section are all based on a variant of the *multiview ICP* algorithm and require an initial coarse alignment of all meshes. For the five sets of range scans we created this coarse alignment in an interactive manner using MeshAlign [53]. We started with the mesh of the object's front view and aligned overlapping meshes sequentially, using four manually selected correspondence points from the overlapping surfaces. As a result, all meshes were transformed to the coordinate system of the first mesh. For most mesh pairs it was easy to select four corresponding feature points, but some mesh pairs lacked features. The *memento*, for instance, had several views for which its range scans were without features due to the smoothness of the surface. In such cases, we selected approximately corresponding points on specific surface areas (like the arms of the *memento*). For each object we performed this coarse alignment only once, which resulted in a set of coarsely aligned meshes for each object.

2.3.2 Experimental setup

For the fine alignment of meshes we use the results of the interactively aligned meshes. We compare the accuracy of three different alignment systems:

MeshAlign (v.2) is a system developed by ISTI-CNR [53] to perform the coarse alignment and fine alignment of meshes. A multiview ICP algorithm as described by Pulli [82] is used for the fine alignment.

RapidForm (2004 PP2) is a commercial system developed by INUS-Technology [52] and able to perform both the coarse alignment, fine alignment and merging of a set of meshes, as well as many other 3D modeling operations such as hole filling.

Scanalyze (v1.0.3) is a software distribution developed by Stanford's Computer Graphics Laboratory [95] for the coarse and fine alignment of meshes. For the fine alignment, this system can use one of the variants of the pairwise ICP algorithms described in [88]. It automatically fine aligns all meshes by optimizing the parameters of the multiview ICP algorithm [82] while it iteratively aligns neighboring meshes. For this system we will only use its multiview ICP algorithm, because we want to align all meshes simultaneously.

Each system has a number of parameters that need proper settings. To obtain accurate alignments the ICP algorithm should reach convergence. Parameter settings for the alignment and merging systems were experimentally determined, as described in [101].

After each fine alignment we store the vertices of the aligned range scans as a 3D point cloud. To evaluate the position of each 3D point relative to the reference model, we align the point cloud to the reference model. To do so we apply the ICP algorithm using the RMS distance and the accurate point-to-plane distance of all vertices to the reference model. The final RMS distance after convergence of the ICP algorithm is used to evaluate the average accuracy of 3D point cloud, i.e. the aligned range data.

Object	Interactive	MeshAlign	RapidForm	Scanalyze
armadillo	0.4105	0.0052	*0.0029*	0.0065
dragon	0.4359	0.0047	*0.0022*	0.0047
knot	0.9991	0.0012	*0.0006*	0.0026
pierrot	0.9079	0.2205	*0.2033*	0.2427
memento	1.0603	0.2145	*0.1887*	0.2180

Table 2.2: RMS distance (in mm) of the alignments to their references. RapidForm produces the most accurate alignments.

2.3.3 Results

Alignment results of the *armadillo*, *dragon*, *knot*, *pierrot* and *memento* are shown in Table 2.2. Because the virtual scans did not suffer from scan inaccuracies, the RMS distances for the *armadillo*, *dragon* and *knot* are much lower than the RMS distance of the *pierrot* and the *memento*. If we compare the results of the fine alignments (MeshAlign, RapidForm, and Scanalyze) with the coarse interactive alignment we see a considerable improvement. The three fine alignment systems all obtain highly accurate results, with RapidForm slightly outperforming the other systems. Depending on the level of noise, the systems may differ more in accuracy. However, no harsh conclusions can be drawn from these results because all alignments are very accurate and more exhaustive parameter tuning may improve the accuracy of a particular system even further. What is more important is the effect of these small differences in accuracy in the evaluation of the merging systems.

2.4 Merging accuracy

In this section, we evaluate three volumetric merge methods, one surface zippering method, and one octree based MLS surface reconstruction method. The three volumetric merge methods are different implementations of the popular range scan integration approach described by Curless and Levoy [37]. The other two merge methods are different algorithms that target the surface reconstruction problem too.

2.4.1 Experimental setup

To evaluate different merge methods, we use the final fine alignments of the *armadillo*, *dragon*, *knot*, *pierrot* and *memento* that we acquired in the previous section. To evaluate the influence of the fine alignment on the merge results, we use each of the three fine alignments obtained with MeshAlign, RapidForm, and Scanalyze as input for each merge method. The output of a merge method is compared to the reference model and its accuracy reported using the RMS distance. The five merge methods are part of the following commercial and non-commercial systems, for which we carefully selected the parameters as described in [101].

MeshMerge (v1.01) [53] is a volumetric merge method that builds a carefully weighted distance field for each mesh, and blends all the distance fields together in a seamless way in a single volumetric representation similar to the approach proposed by Curless

and Levoy [37]. The final surface is reconstructed through the standard Marching Cubes algorithm [64].

RapidForm (2004 PP2) [52] provides two kinds of merge methods, surface zippering and volumetric merging. The surface zippering is the implementation of a technique developed by Turk and Levoy [107], that removes redundant surfaces, zippers adjacent meshes, and optimizes the triangles in the zippered areas. The volumetric merging allocates the geometry information of the range scans to a volumetric grid and applies a Marching Cubes algorithm.

VripPack (v0.2) [95] is an implementation of the volumetric approach described by Curless and Levoy [37]. The nodes of the volumetric grid store the weighted signed distances of the grid nodes to the nearest range scans along the sensor's line of sight. The final surface is extracted from the volumetric grid using the Marching Cubes algorithm.

Octree Merger (OM) (v1.03) [43, 53] is an octree based MLS surface reconstruction method that creates a surface using the projection operator of point set surfaces defined in [7]. The input to this tool are the point sets defined by the vertices of the aligned range scans, with normals computed independently on each range map, and the output is the reconstructed surface.

For the three volumetric merge methods, the resolution of the grid should be related to the spatial resolution of the scanning device. We use both 0.4 mm^3 (equals the scan resolution) and 0.2 mm^3 (limit in memory) sized voxels in the experiments. Altogether, we have thee final fine alignments for each of the five objects. These fifteen different alignments are merged in eight different ways and compared to their reference models using Metro.

After the merging process only the largest connected surface component of the *pierrot* and the *memento* is retained (so noisy patches that got separated from the main surface mesh were removed). For the other objects this is not necessary since they are without noise. Before measuring the RMS distance, the output and the reference model need to be aligned, because of small rotations and translations during the scan alignment and merging process. We also perform a visual inspection of the merged models, to investigate the performance of the merging systems in two specific situations.

2.4.2 Results

The RMS distances between the merged models and their reference models are shown in Table 2.3 and visualized in Fig. 2.6. For each object, the merging was performed for each of the three fine alignments and the best merging technique for a particular alignment is highlighted in italic. The most accurate combinations of alignment and merging for a particular object are marked in bold (before rounding). These results show that the RMS distance of merge results based on noisy scan data is much higher than for the noiseless synthetic scan data, which shows the importance of testing geometric algorithms to real data. Furthermore, we see that the slightly more accurate alignments of RapidForm result in merged models with higher accuracy as shown by their lower RMS distances in Fig. 2.6. For the actual (noisy) range scans, each of the volumetric merge method performs better with a lower resolution grid of $0.4 \times 0.4 \times 0.4$ mm, but for the synthetic scans with little noise the high resolution grid of $0.2 \times 0.2 \times 0.2$ mm is preferred. Clearly the volumetric merge resolution and accuracy is bounded by the

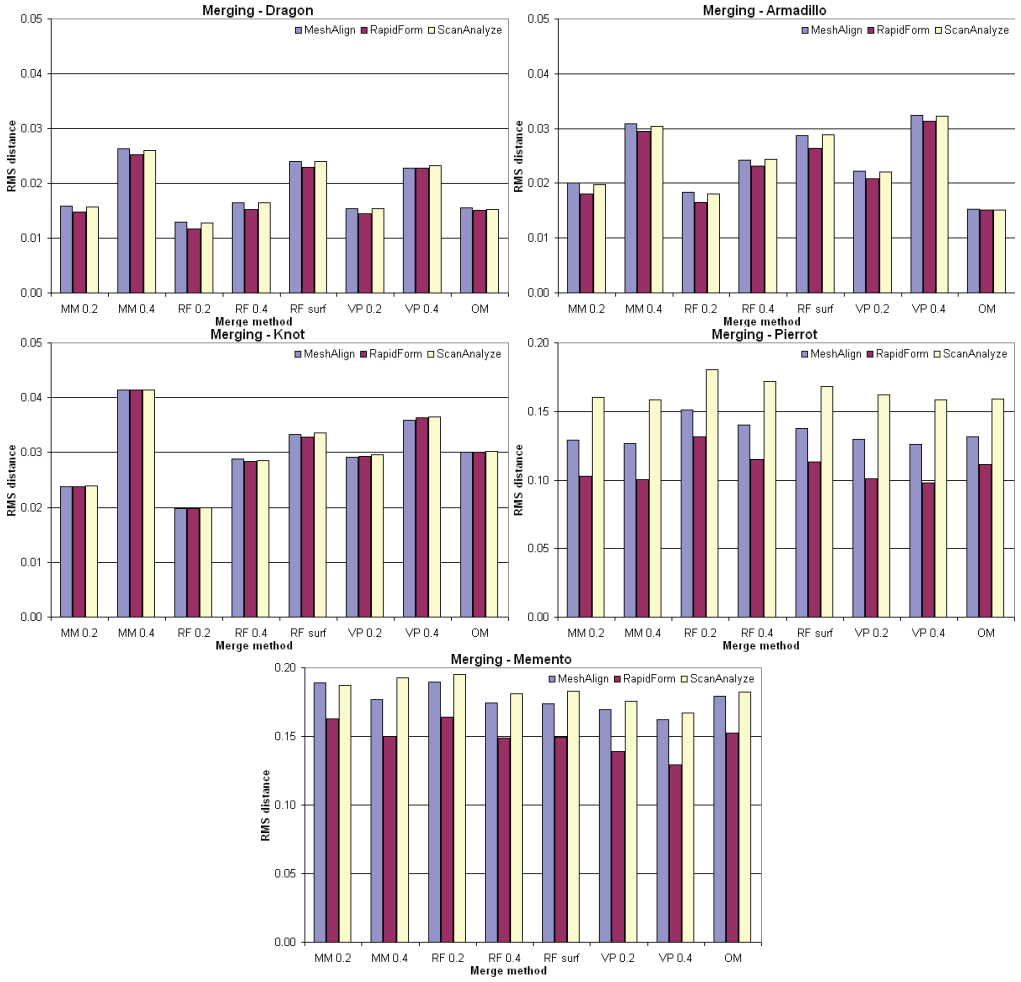


Figure 2.6: Bar charts of the RMS distances between the merged surfaces and their reference model. The accuracy of the merged model clearly depends on the accuracy of the alignment, level of noise, and merge method.

Model	MeshMerge		RapidForm			VripPack		OM
alignment	*Vol 0.2*	*Vol 0.4*	*Vol 0.2*	*Vol 0.4*	*Surf.Zip.*	*Vol 0.2*	*Vol 0.4*	
armadillo								
- MeshAlign	0.020	0.031	0.018	0.024	0.029	0.022	0.032	*0.015*
- RapidForm	0.018	0.029	0.017	0.023	0.026	0.021	0.031	**0.015**
- Scanalyze	0.020	0.030	0.018	0.024	0.029	0.022	0.032	*0.015*
dragon								
- MeshAlign	0.016	0.026	*0.013*	0.016	0.024	0.015	0.023	0.016
- RapidForm	0.015	0.025	**0.012**	0.015	0.023	0.014	0.023	0.015
- Scanalyze	0.016	0.026	*0.013*	0.017	0.024	0.015	0.023	0.015
knot								
- MeshAlign	0.024	0.041	*0.020*	0.029	0.033	0.029	0.036	0.030
- RapidForm	0.024	0.041	**0.020**	0.028	0.033	0.029	0.036	0.030
- Scanalyze	0.024	0.041	*0.020*	0.029	0.034	0.030	0.037	0.030
pierrot								
- MeshAlign	0.129	0.127	0.151	0.140	0.138	0.129	*0.126*	0.132
- RapidForm	0.103	0.100	0.132	0.115	0.113	0.101	**0.098**	0.111
- Scanalyze	0.160	*0.158*	0.181	0.172	0.168	0.162	*0.158*	0.159
memento								
- MeshAlign	0.189	0.177	0.190	0.174	0.174	0.169	*0.162*	0.179
- RapidForm	0.163	0.150	0.164	0.149	0.150	0.139	**0.129**	0.153
- Scanalyze	0.187	0.193	0.195	0.181	0.183	0.175	*0.167*	0.182

Table 2.3: RMS distances (in mm) between the merged models to their references based on three different fine alignment systems. The best merging system for a particular alignment is shown in italic. The best combinations for the alignment and merging of the object's meshes are shown in bold.

level of noise in the range scans. When we compare the individual performance of the different merge methods we see that:

1. Although volumetric merge methods use similar heuristics and the same grid resolution, their accuracy is different.

2. The volumetric merge method of RapidForm with 0.2 mm^3 sized voxels performs *best* in case of the noiseless *armadillo*, *dragon*, and *knot* scans.

3. The volumetric merge method of MeshMerge with 0.4 mm^3 sized voxels performs *worst* in case of the noiseless *armadillo*, *dragon*, and *knot* scans.

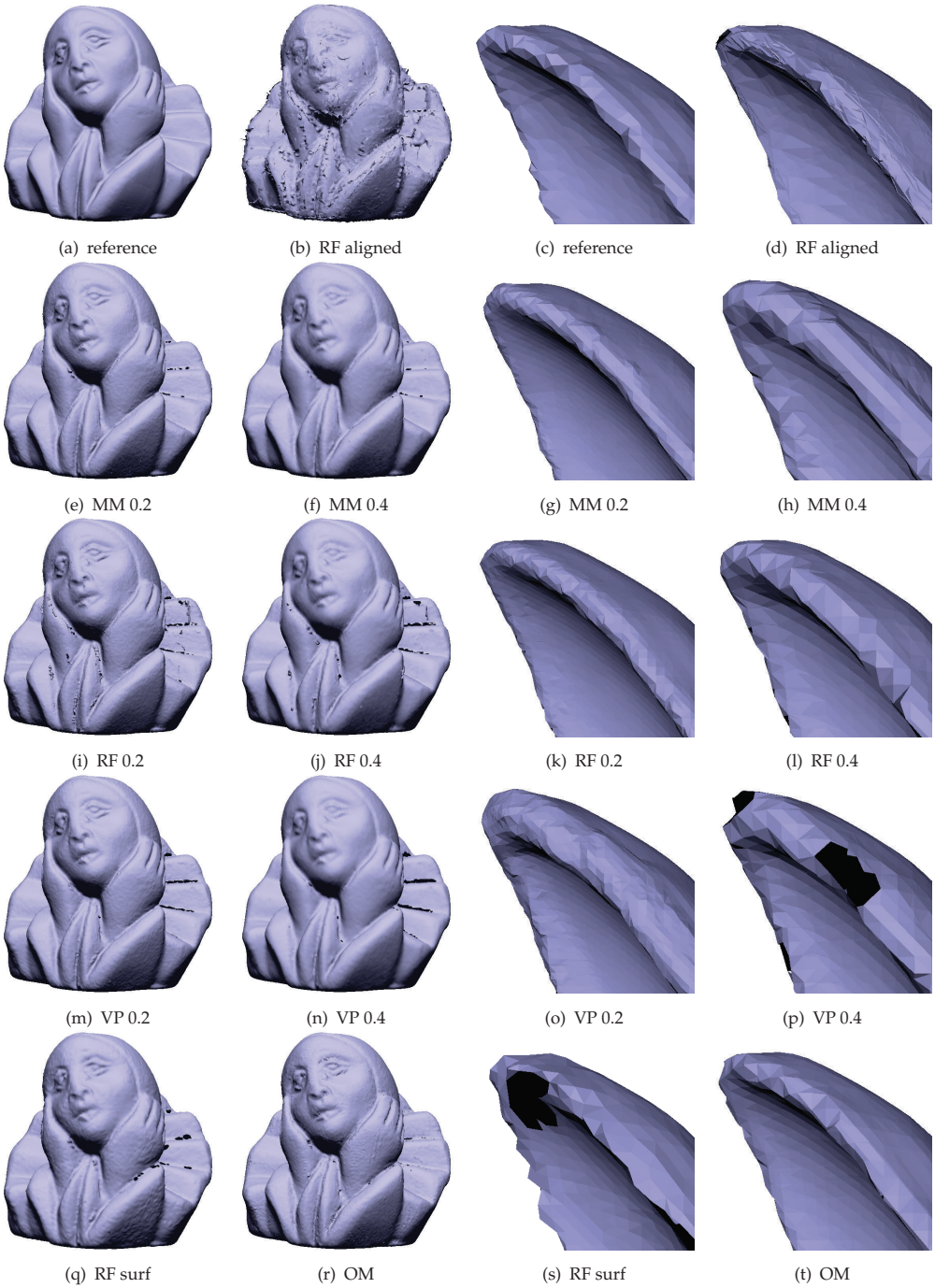
4. The volumetric merge method of VripPack with 0.4 mm^3 sized voxels performs *best* in case of the noisy *pierrot* and *memento* scans.

5. The volumetric merge method of RapidForm with 0.2 mm^3 sized voxels performs *worst* in case of the noisy *pierrot* and *memento* scans.

6. The MLS surface reconstruction of OM performs very well for the *armadillo* and *dragon* models.

7. RapidForm's surface zippering shows consistently high RMS distances.

To investigate how high curvature features and noise are handled by the merging systems, we visually compared the merge results of the *armadillo*'s ear and the *pierrot*.



(a) reference (b) RF aligned (c) reference (d) RF aligned

(e) MM 0.2 (f) MM 0.4 (g) MM 0.2 (h) MM 0.4

(i) RF 0.2 (j) RF 0.4 (k) RF 0.2 (l) RF 0.4

(m) VP 0.2 (n) VP 0.4 (o) VP 0.2 (p) VP 0.4

(q) RF surf (r) OM (s) RF surf (t) OM

Figure 2.7: Merge results of the noisy pierrot and the armadillo's ear based on the alignment using RapidForm (RF). Both MeshMerge (MM) and VripPack (VP) show very accurate results for the noisy pierrot alignment, while RapidForm (RF) and OM show less accurate results. For the armadillo's ear, the high resolution volumetric methods ($0.2mm^3$) and OM are the preferred systems. Similar results are shown in the bar charts.

The *armadillo*'s ear was selected for its high curvature, and the *pierrot* for its noise. Renderings of the results are shown in Fig. 2.7. These results are based on the alignments performed by RapidForm, because these alignment have a slightly higher accuracy.

In Fig. 2.7, we see that the volumetric merge techniques of MeshMerge (MM 0.2/0.4) and VripPack (VP 0.2/0.4) result in the best resemblance to the *pierrot* reference, with slightly more noise when using a higher resolution grid. OM and RapidForm (RF 0.2/0.4/surf) show more noise in their merged models, with RapidForm's high resolution grid in particular. The results for the right ear of the *armadillo* show reconstruction failures in case of low resolution grids and surface zippering. Most successful in this case are MeshMerge and RapidForm, both with a high resolution grid (MM/RF 0.2) and OM. These results confirm our conclusions based on the quantitative results.

2.5 Hole filling accuracy

It is hard and sometimes even impossible to observe the entire surface of an object with a laser range scanner. The result is that we miss range data in some areas. One cause of missing data might be the limited number of range scans from different viewing directions, another cause comes from the basic principle of laser range scanning itself. The acquisition of parts of the surface may fail due to absorption (by a dark colored surface), refraction (by a transparent surface) and reflection (by a specular surface) of laser light, or due to occlusion (the sensor is not able to sense the laser dot). So even if it is physically possible to scan the entire surface of an object, it is most likely that holes appear in the final merged model. In this section four systems are used to fill holes in the merged surface mesh, of which one system provides three different hole filling methods. In case a hole filling technique leaves a complicated hole untouched, the RMS distance does not change. So instead of reporting the RMS distance of a final model, we only qualitatively evaluate the results. A possible way to report quantitative results is to relate the RMS distance to the remaining boundary length.

2.5.1 Experimental setup

In this section we attempt to fill the holes in the final merged models with the use of several hole filling systems MeshMerge, RapidForm, VripPack, and VolFill. MeshMerge and VripPack were designed to merge meshes, but additionally include a method to fill holes at the same time. VolFill operates on the volumetric distance field grid of VripPack that was constructed during the merging of meshes, which is a limitation. For a fair comparison, we apply the systems MeshMerge and RapidForm on the merged model obtained with VripPack, and apply the hole filling systems VolFill and VripPack itself on the volumetric grid from which the merged model is extracted (Fig. 2.8). In the experiment we fill the holes of the *memento* model, which was reconstructed with VripPack using both the $0.4 \times 0.4 \times 0.4$ mm and $0.2 \times 0.2 \times 0.2$ mm resolution voxel grid. We use three particular views to investigate the performance of the hole filling techniques (Fig. 2.9). The following hole filling techniques are used.

MeshMerge (v1.01) [53] is able to fill holes in the volumetric distance field using a number of refilling steps. When the merged model is used as input for MeshMerge, a volu-

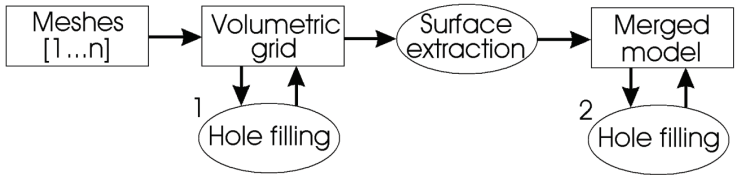


Figure 2.8: The hole filling process of both VripPack and VolFill (1) requires the volumetric distance field grid constructed by VripPack, while the other hole filling systems use the merged model extracted from this grid (2).

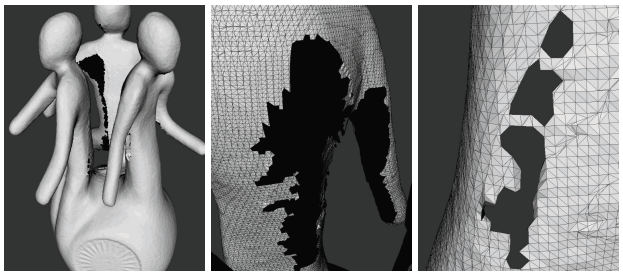


Figure 2.9: Three selected views to investigate the performance of the hole filling techniques: a global view (left), a huge and complicated hole (middle), and a few small and less complicated holes (right).

metric distance field grid is constructed first. To get a good indication of this hole filling technique we apply ten refilling steps.

RapidForm (2004 PP2) [52] automatically fills holes using either a flat based, smooth based, or curvature based method. In case of the flat based method, the boundary is simply triangulated if possible. The smooth based method constructs a polygonal structure, and both the hole and its surrounding region are remeshed. In case of the curvature based hole filling, the polygonal structure is curved to match the surrounding area.

VripPack (v0.2) [95] provides a space carving operation that determines empty and unseen areas in its volumetric grid. During the hole filling process the distance field is expanded at the holes along the border of the unseen and empty areas until the holes are filled [37]. Finally, the surface mesh is extracted from the volumetric grid.

VolFill (v1.0) [95] fills holes by the expansion (volumetric diffusion) of VripPack's volumetric distance field grid at the border of the unseen and empty areas as described in [39]. The volumetric diffusion in this system is the blurring of the volumetric distance field, which is performed a number of iterations. A hole is filled when the diffusion of the distance field connects the boundaries of a hole. After a number of blurring iterations the final surface mesh is extracted from the volumetric grid using the Marching Cubes algorithm. Six blurring iterations are applied, which corresponds to the ten refilling steps of MeshMerge.

2.5.2 Results

In this experiment we have filled holes of the *memento* model, which was reconstructed with VripPack using a high ($0.2 \times 0.2 \times 0.2$ mm) and a low resolution ($0.4 \times 0.4 \times 0.4$

mm) voxel grid. Results of the hole filling techniques were inspected for three particular views (Fig. 2.9), and for both the high and low resolution grid. Figures 2.10, 2.12, and 2.14 show the results of the hole filling systems with respect to the holes in the low resolution model. Figures 2.11, 2.13, and 2.15 show these results for the high resolution model.

Fig. 2.10a shows the holes that remain after the merging process of VripPack. These holes occur due to missing data during the acquisition stage. MeshMerge (2.10b) is able to fill the large hole, but shows a rather extruding surface. The same holds for VolFill (2.10f) with even more extrusions. The surface based hole filling techniques of RapidForm (2.10cde) are able to fill in the missing data almost perfectly, with a slightly better result for its curvature filling. VripPack (2.10g) on the other hand shows incorrect expansion of the volumetric grid into regions that are supposed to remain empty, which was also pointed out in [39]. For the holes in the high resolution model (2.11a), the results are very much the same. Only now, a smaller part of the holes is filled with the use of either MeshMerge or VolFill, because the same number of applied iterations generates a smaller amount of new (high resolution) surface. VripPack fails for the high resolution data due to the amount of required memory.

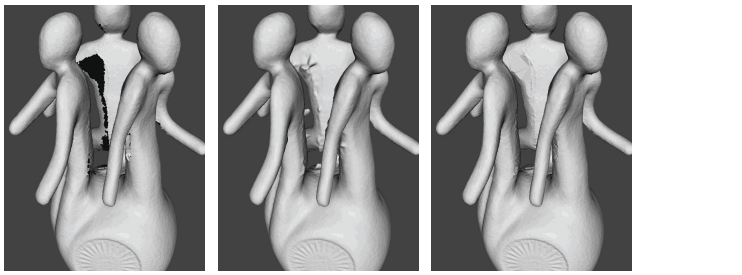
The large and complicated hole in the low resolution model (Fig. 2.12) shows again good results for the curvature filling (2.12e) and flat based filling (2.12c) with RapidForm. Reasonable results are obtained with MeshMerge (2.12b) and the smooth based variant of RapidForm (2.12d). The other systems expand the surface in an incorrect manner (2.12fg). For the high resolution model (Fig. 2.13) almost none of the complicated holes were filled, only RapidForm's curvature based filling was able to fill one large hole.

The third set of inspected holes are much smaller and less complicated than the previous ones (see Fig. 2.14 and 2.15). Almost all hole filling systems were able to fill in these holes. If we look at the result of MeshMerge applied to the high resolution model (2.15b), we notice the creation of a tube-like surface. This system expands the surface according to the orientation of faces around the hole.

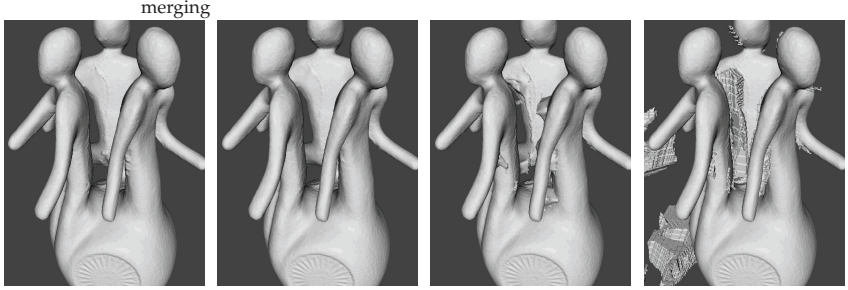
2.6 Discussion

Our quantitative evaluation was based on range scans of seven test objects. Scans of two object were used to evaluate two laser range systems. The range scans of the other five objects were: (1) coarsely aligned with an interactive method, (2) finely aligned with three systems (MeshAlign, Scanalyze, and RapidForm), and (3) merged with MeshMerge, RapidForm, VripPack, OM. For the volumetric merge methods, we used two different grid resolutions ($0.4 \times 0.4 \times 0.4$ mm and $0.2 \times 0.2 \times 0.2$ mm). To evaluate the accuracy of these alignment and merging systems, we compared their output model to reference models with Metro. Metro computes the RMS distance between two models, which we used as a measure for the system's accuracy. In the end we tested several hole filling techniques and qualitatively evaluated the results.

The quantitative evaluation of the systems for the fine alignment of range scans show highly accurate results and only slight differences between them. The evaluation of the merged models based on these fine alignments shows nonetheless that even a

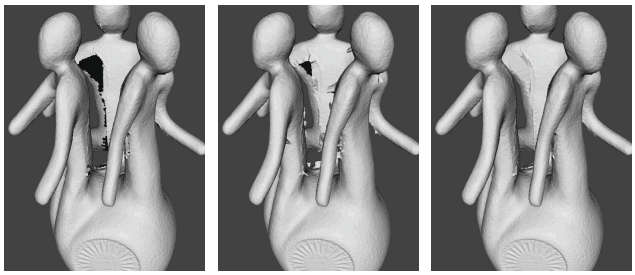


(a) Holes after VripPack merging (b) MeshMerge (c) RapidForm flat

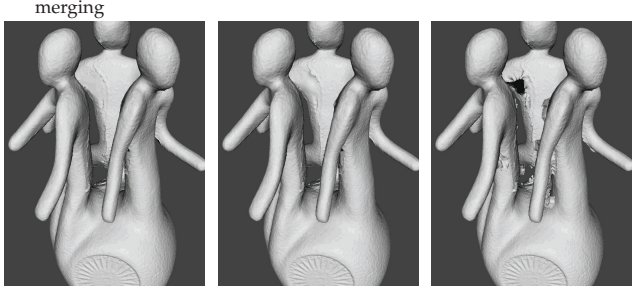


(d) RapidForm smooth (e) RapidForm curvature (f) VolFill (g) VripPack

Figure 2.10: Results of the hole filling methods for the merged memento. VripPack with a 0.4 mm^3 sized voxel grid was used for the merging of meshes.

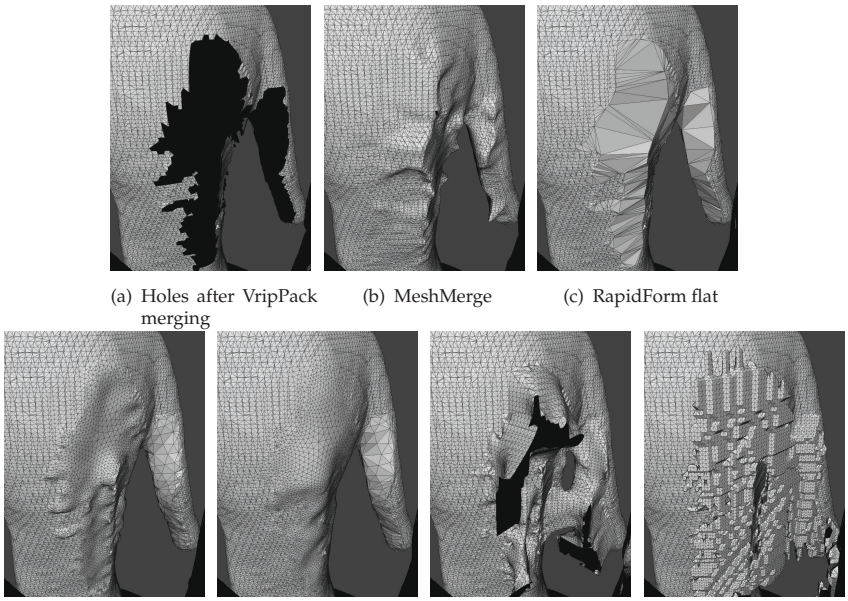


(a) Holes after VripPack merging (b) MeshMerge (c) RapidForm flat



(d) RapidForm smooth (e) RapidForm curvature (f) VolFill

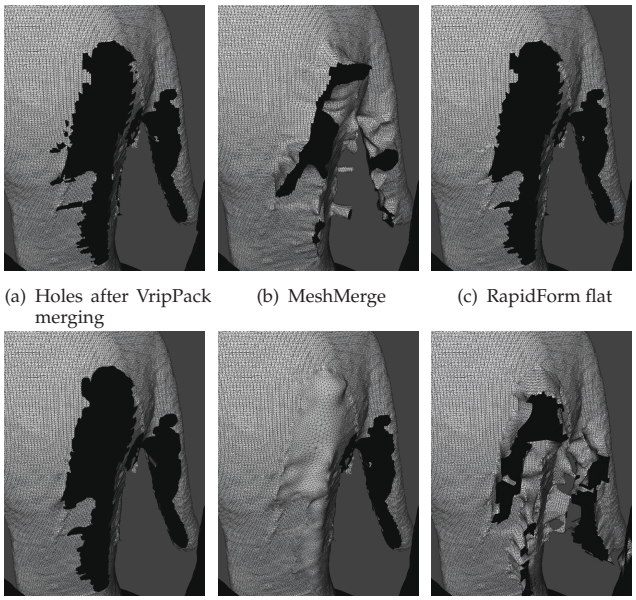
Figure 2.11: Results of the hole filling methods for the merged memento. VripPack with a 0.2 mm^3 sized voxel grid was used for the merging of meshes. VripPack failed.



(a) Holes after VripPack merging (b) MeshMerge (c) RapidForm flat

(d) RapidForm smooth (e) RapidForm curvature (f) VolFill (g) VripPack

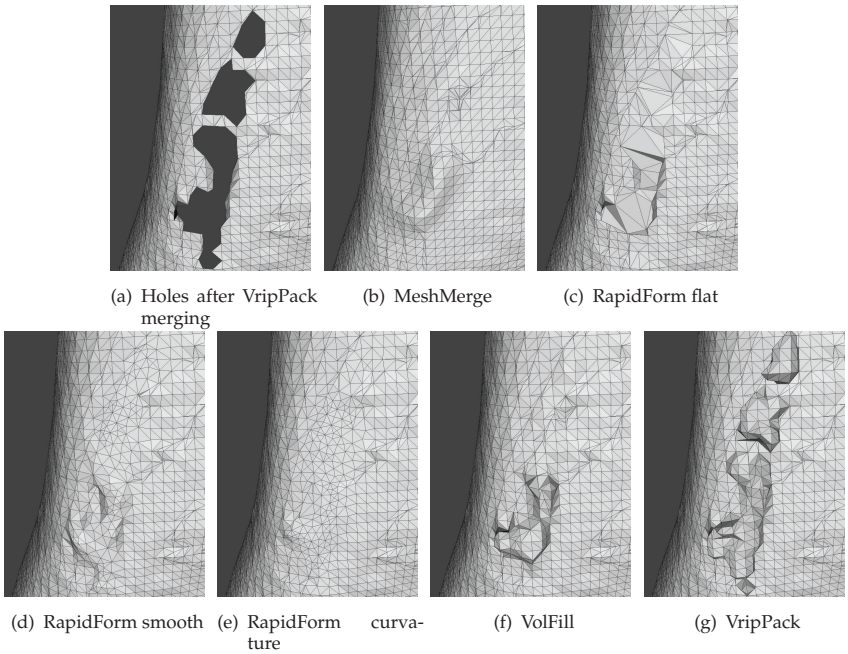
Figure 2.12: Results of the hole filling methods for the merged memento. VripPack with a 0.4 mm^3 sized voxel grid was used for the merging of meshes.



(a) Holes after VripPack merging (b) MeshMerge (c) RapidForm flat

(d) RapidForm smooth (e) RapidForm curvature (f) VolFill

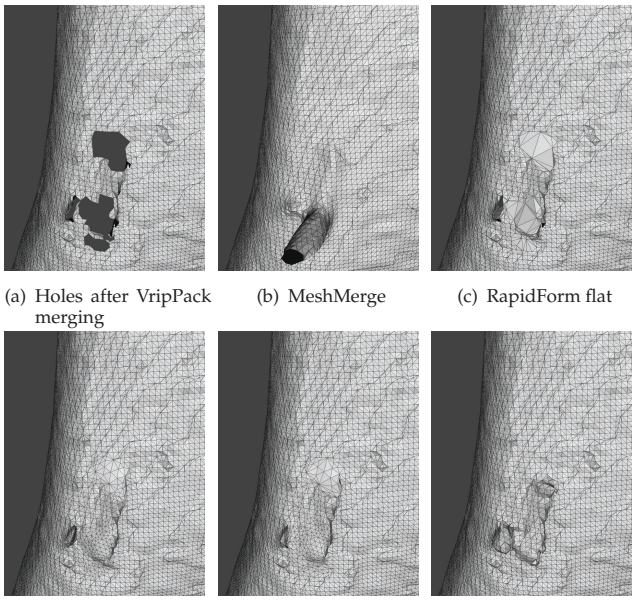
Figure 2.13: Results of the hole filling methods for the merged memento. VripPack with a 0.2 mm^3 sized voxel grid was used for the merging of meshes. VripPack failed.



(a) Holes after VripPack merging (b) MeshMerge (c) RapidForm flat

(d) RapidForm smooth (e) RapidForm curvature (f) VolFill (g) VripPack

Figure 2.14: Results of the hole filling methods for the merged memento. VripPack with a 0.4 mm^3 sized voxel grid was used for the merging of meshes.



(a) Holes after VripPack merging (b) MeshMerge (c) RapidForm flat

(d) RapidForm smooth (e) RapidForm curvature (f) VolFill

Figure 2.15: Results of the hole filling methods for the merged memento. VripPack with a 0.2 mm^3 sized voxel grid was used for the merging of meshes. VripPack failed.

marginal improvement of this alignment results in an improved accuracy of the merged model. For volumetric merge methods, we found that a higher resolution of the voxel grid will not always result in an improved accuracy of the merged model. This improvement is bounded by the level of noise in the range scans. These conclusions correspond to our prior assumptions concerning the alignment and merging of 3D range scans. Nevertheless, it is important to have these assumptions quantified using an extended evaluation.

Visual evaluation of the merge results shows that different variants of the volumetric merge approach based on a distance field can have a very different output for sharp features, even if the merged meshes do not contain noise. Surface zippering is one of the earlier methods for merging range scans that is known to fail in regions with high curvatures. More recently developed methods, such as volumetric methods and MLS surfaces, have surpassed it in accuracy: this was confirmed by our results.

The hole filling experiment showed how difficult it is to fill different types of holes correctly. Small and not very complicated holes were easy to fill for almost all of the hole filling systems, but more complicated holes in the *memento* showed major differences in the precision of systems. Although the use of surface curvature or simple flat triangles performed reasonably well, these techniques failed to fill the large and complex holes. In fact, none of the described systems filled all of the holes in the *memento* model in a satisfactory manner.

2.7 Concluding remarks

In this chapter, we evaluated systems for the acquisition and reconstruction of objects based on laser range scans. The alignment, merging, and hole filling techniques use reasonable heuristics, such as the ICP algorithm for the alignment. No quantitative comparison of such systems has been done before. Even though a comparison of software systems depending on many different parameters is difficult to control, it is nevertheless of practical importance to compare the accuracy of such complex systems given recommended parameter settings.

Heuristic surface reconstruction methods seem to work well within this entire process, but there is no guarantee on their accuracy. Since each step in the acquisition and reconstruction pipeline clearly affects each following step, a promising research direction is to develop algorithms that are guaranteed to have a certain accuracy, depending on input parameters such as the scanning resolution, scanning accuracy, and surface curvature.

Since the filling of complex holes in a surface mesh is a hard problem, a challenging direction is to automate the coarse alignment of range scans during the scanning process so that the user knows which areas of the object are under-sampled or missing. As a first step in that direction we present in the next chapter an efficient algorithm to perform the automatic coarse alignment of range scans.

Chapter 3

Automatic alignment

In this chapter, we present a new multiview alignment algorithm that performs both the coarse and fine alignment of unordered sets of range scans. Our algorithm selects quadruples of range scans, which have feature points of their 2D projections in common. These quadruples are then verified using an Iterative Closest Point (ICP) algorithm. The accepted quadruples form incomplete models of an object that can be aligned using isometries of the Principal Component Analysis (PCA) in combination with an ICP algorithm. The output of our method is a set of finely aligned meshes. Our method was applied to range scans of different clusters of objects varying in the type of acquisition system, the number of range scans, the scan resolution, and scan accuracy. Results show that our method is both effective and efficient for the alignment of meshes: it is capable of aligning object meshes with various properties, aligns the majority of meshes without a priori knowledge, and doesn't require a multiview ICP algorithm to improve the final alignment.

3.1 Introduction

As we mentioned in Chapter 2, the coarse alignment of meshes requires the detection of correspondences that represent the overlap between meshes. For just two range scans we need to find the translation and rotation that transforms one range scans to the other such that the object surface they have in common becomes aligned. The main difficulty in this pair-wise alignment is to find surface regions that truly overlap, which is a hard problem because every two meshes have at least a vertex, line, triangle, or a surface

patch that overlap nicely although they represent different parts of the real object. Depending on the object two meshes can have the same shape but from different sides of the object, take for instance two opposite scans of a sphere. Besides that two meshes can be wrongly aligned, two truly overlapping meshes may not align well due to different resolutions, noise, outliers and missing data. For more than two range scans we need to find the translations and rotations that aligns them all in a globally correct way, while taking into account these pair-wise problems. When the range scans are *ordered*, we know which scans are supposed to have overlap and we only need to establish the correspondence. For *unordered* sets of range scans, a multiview alignment algorithm needs to find the globally correct alignment of the range scans without additional knowledge.

The automatic multiview alignment of unordered range scans is essential for the fully automatic reconstruction of 3D models out of real world objects and only a few algorithms have been proposed to perform this coarse alignment. Huber and Hebert [50] describe a framework in which so-called spin-images are used to find correspondences among all possible pairs of meshes. Then they construct a minimal spanning graph that determines- which highly confident pair-wise transformations should be applied to construct a globally correct alignment. Recently, Novatnack and Nishino [77] proposed the use of scale-invariant local shape descriptors instead of spin-images within Huber's framework to align meshes. Mian et al. [71] developed a method that extracts volumetric grids of local surface patches from all meshes and matches those to establish potential alignments between them. A correspondence tree is constructed by selecting the mesh with most data points as the root and iteratively adding meshes with enough correspondence as leafs to the tree. In case a new leaf causes the tree to fail a global consistency check, it is not added.

For a set of N meshes, the framework of Huber requires N^2 and the framework of Mian $O(N^2)$ mesh-to-mesh comparisons. More importantly, these methods are using large sets of complex shape descriptors, such as spin-images and 3D surface patches. Afterwards, these methods apply a multiview ICP algorithm to refine their coarse alignment of meshes, which has a worst case complexity of $O(N^2)$ as well.

3.1.1 Contribution

In this chapter we present a new method to align unordered sets of meshes. The main idea of our method is to select small groups of meshes, which can be aligned using Principal Component Analysis (PCA) in combination with an Iterative Closest Point (ICP) algorithm. Our method selects groups of four meshes (quadruples) that represent the front, right, back, and left views of an object's pose. To select the optimal quadruples, sets of 2D silhouette features are extracted from all meshes, and matched.

This new method has a number of advantages. Firstly, we are able to align mesh sets of objects scanned using different acquisition systems, objects with few and many protrusions, objects with smooth and rough surfaces, and mesh sets with little and much noise. Secondly, our method is able to automatically align all meshes from 12 out of 25 datasets, and 80% of the total amount of meshes, without a priori knowledge. Thirdly, our approach applies $O(N^2)$ mesh-to-mesh comparisons using low cost operations only. The rest of the algorithm, which is the most time consuming part, requires $O(N)$ mesh-to-mesh comparisons and obtains a high quality alignment similar to an $O(N^2)$ multi-

view ICP algorithm in less time.

3.2 Method

This section describes an efficient way to align a set of 3D meshes $\mathbf{M_{i\in(1\ldots N)}}$. The approach is to find quadruples of meshes first, that represent the object's front, right, back, and left side of a certain pose. Therefore, we center all meshes and extract features from the boundary of projection silhouettes (Sect. 3.2.1) and match them to the features of other meshes (Sect. 3.2.2). In Sect. 3.2.3 we describe how quadruples are selected and how the four meshes are coarsely aligned within such quadruples. The coarse alignment of meshes in a quadruple is refined and the quadruple itself is verified (Sect. 3.2.4). During the verification of a quadruple, the quadruple is rejected or accepted, or a subgroup of three of its meshes is accepted. The accepted groups of either three of four meshes are aligned in Sect. 3.2.5 which returns the final alignment of meshes.

Before starting the feature extraction, we resample the meshes for efficiency reasons. The meshes $\mathbf{M_i}$ are resampled towards both fine meshes $\mathbf{FM_i}$ and coarse meshes $\mathbf{CM_i}$. See Sect. 3.2.6 for implementation details.

3.2.1 Feature extraction

For each of the fine meshes $\mathbf{FM_i}$, a set of features is extracted from two of its projections. Given that the Z direction of the XYZ reference system is pointing towards the user, one set of features is extracted for the mesh mapped to the XY-plane ($\mathbf{FM_i^{xy}}$), and one for the mesh mapped to the YZ-plane ($\mathbf{FM_i^{yz}}$). We use no projection of a mesh onto the XZ-plane. This projection could be used to find a top or bottom view. However, to scan a top or bottom view the object has to be toppled over towards a new stable pose, which should result in a new group of different side views. So, we use only the XY- and YZ-projections. For mesh $\mathbf{FM_i^{xy}}$ we extract the vertices on the boundary that two opposite meshes (i.e. opposite model views) most likely have in common, which are the right-, top-, left-, and bottommost set of silhouette vertices of the projected mesh (Fig. 3.1). We refer to this set of vertices as the XY-features. For mesh $\mathbf{FM_i^{yz}}$ only the frontmost silhouette vertices are selected as its YZ-features, which may correspond to a subset of the XY-features as described in the next section.

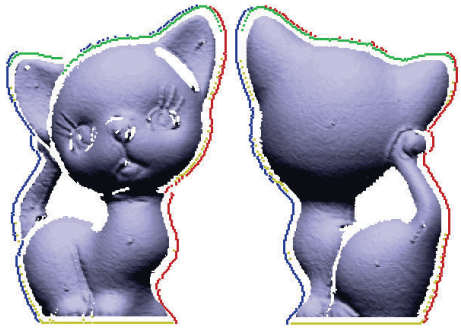


Figure 3.1: The extracted features (in color) from two opposite meshes shown with an offset. Vertices from holes and cavities are not extracted, which makes the two sets of features comparable.

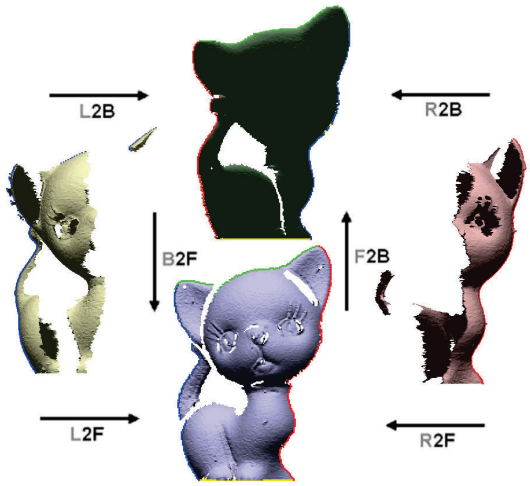


Figure 3.2: Features (in color) from the left, right and back view are matched to the features of the front view. Note that the features from the left view correspond to the leftmost features of the front view.

3.2.2 Feature matching

For each fine mesh $\mathbf{FM_i}$, we try to find a left, right and back side by comparing the mesh to all other meshes using the extracted sets of features. Mesh $\mathbf{FM_i}$ is selected as the front mesh of a newly created quadruple and all other meshes $\mathbf{FM_j}$ are compared as being a potential right, back, or left side of this quadruple. For a back to front comparison the mirrored set of XY-features of the potential back side is compared to the XY-features of the front. For a side to front comparison the YZ-features are matched to the front's XY-features. The distance measure used in these comparisons is the Root Mean Squares (RMS) distance after i_{max} iterations of the point-to-point ICP algorithm (see Algorithm 1 applied to two point sets). When a side to front comparison is performed on an actual side-front pair, the side's YZ-features form a subset of the front's XY-features, that is, when we place the feature sets in one plane. Moreover, these features correspond either to the leftmost XY-features in case of a left side, or to the rightmost XY-features in case of a right side (Fig. 3.2).

Because the point-to-point ICP algorithm requires an initial alignment of point sets, the centers of mass of either the entire sets of features or only the specific subsets (in case of a side view) are aligned. For a perfect match the RMS distance after the point-to-point ICP algorithm is zero. Because each mesh is compared to all other meshes as its potential right, back, and left side, this process is quadratic in the number of meshes. However, we have used only low cost operations, that is, the ICP algorithm is applied to silhouette points only.

3.2.3 Quadruple selection

For each fine mesh $\mathbf{FM_i}$ a quadruple is constructed of those four meshes that have the minimal total sum of RMS distances of the following twelve feature set comparisons:

Algorithm 1 point-to-point ICP (PS_1, PS_2)

1: $dist_{min} \leftarrow \infty$
2: **for** $i \leftarrow 1$ **to** i_{max} **if** no convergence **do**
3: $pairs \leftarrow$ for all points in PS_1 find closest point in PS_2
4: $dist \leftarrow$ minimize the RMS distance between the points in $pairs$
5: **update** $dist_{min}$
6: **end for**
7: **return** $dist_{min}$ **and** 2D rigid transformation

front to back, left to right, right to front, back to right, left to back, front to left, and vice versa (Fig. 3.3). As a result, N quadruples are selected. Since the same quadruple of four meshes may be selected up to four times, we retain only the unique $N' \leq N$ quadruples.

During the matching of feature sets we obtain not only a distance value for two sets of features, but a 2D rigid transformation to align the feature sets as well. Each 2D transformation consists of a translation and a rotation in the plane to which the features were mapped. The twelve rigid transformations within a quadruple define a coarse alignment of the quadruple's meshes.

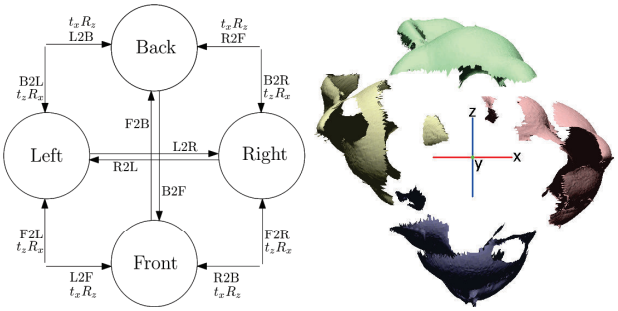


Figure 3.3: The relative transformation of each of the four views consist of: a translation (t) along either the x- or z-axis and a rotation (θ) around the z- or x-axis. F2R is the 2D transformation of the front's YZ-features towards the right side's XY-features rotated 90° around the y-axis for global alignment.

3.2.4 Quadruple verification

Up to here our method selects quadruples of four meshes that represent the object's front, right, back, and left side of a certain pose. So, the selection of an incorrect side view will result in a (partly) incorrect quadruple. Furthermore, the initial alignment obtained so far is not precise enough for general sets of meshes, because it assumes 90 degree rotations between consecutive meshes. Therefore, we need to improve the initial coarse alignment and verify which groups are correct and which are not. To finely align the meshes within a quadruple, we employ Rusinkiewicz's implementation of the point-to-plane ICP algorithm [86] (see Algorithm 2 applied to two meshes M_1 and M_2). This algorithm establishes point-to-point correspondences, but computes point-to-plane distances. The point-to-plane distance is the distance from a point to the plane through the

closest point and oriented perpendicular to the closest point's normal. Point-to-plane pairs with a distance larger than 2.5 times the standard deviations (σ) and without normal compatibility (angle between points normals $> 45°$) are rejected. When either convergence or a stopping criterion is reached the RMS distance is returned. By changing the maximum number of iterations i_{max}, the target number of pairs p_{target}, and the minimum number of pairs p_{min}, the outcome of the algorithm can be influenced.

To improve each of the (N') unique quadruples, the point-to-plane ICP algorithm is applied four times, once for each pair of neighboring meshes in a quadruple. This gives us 3D rigid transformations to transform each mesh towards its two neighbors and a RMS distance for each transformation. The minimum of all ($4N'$) RMS distances is used to accept or reject each 3D transformation. This minimal distance represents an optimal and reachable distance for the alignment of a pair of meshes, that is, under the assumption that all meshes $\mathbf{M_i}$ were scanned using the same laser range scanner and the fact that they were resampled and cleaned in a similar way. 3D transformations with a RMS distance larger than t_{dist} times the minimal distance are rejected. When three or four meshes remain linked by their 3D transformations, then these three or four meshes are finely aligned using the transformations and the *group* is accepted as such. Fig. 3.4 shows several accepted and thus finely aligned groups of an object.

In rare situations, the ICP algorithm might reach the minimum number of corresponding point-pairs with a RMS distance that satisfies the selected t_{dist}, while the alignment is not correct. Because correct groups of meshes have similar volumes for their tight bounding box, we reject groups for which the tight bounding box volume is larger than t_{bb} times the median tight bounding box volume of all groups. Since the bounding box of a PCA normalized object is an approximation of its tight bounding box, we apply this check after PCA normalization in Sect. 3.2.5. See Sect. 3.2.6 for the values for i_{max}, p_{target}, p_{min}, t_{dist} and t_{bb}.

Algorithm 2 point-to-plane ICP (M_1, M_2)

1: $dist_{min} \leftarrow \infty$
2: **for** $i \leftarrow 1$ **to** i_{max} **if** no convergence **do**
3: $pairs \leftarrow$ closest point pairs of p_{target} randomly selected vertices from M_1 to M_2 and M_2 to M_1
4: $pairs \leftarrow pairs$ with distance $< 2.5\sigma$ **and** with normal compatibility
5: **if** $\#pairs < p_{min}$ **return failure**
6: $dist \leftarrow$ minimize the RMS distance of point-to-plane pairs in $pairs$
7: **update** $dist_{min}$
8: **end for**
9: **return** $dist_{min}$ **and** 3D rigid transformation

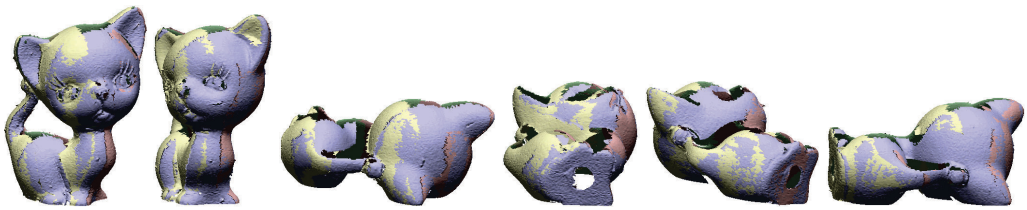


Figure 3.4: Six different, accepted, and finely aligned groups obtained after quadruple verification.

3.2.5 Group alignment

At this stage we have verified and finely aligned groups of either three or four meshes for different poses of the object. These groups cover large parts of the object's surface, which enables us to employ a technique to align incomplete models. To do so, we merge the data of each group's meshes into one (incomplete) model, and then pose normalize the models and obtain their correct alignment using the point-to-plane ICP algorithm as explained below. Because an incomplete model consists of approximately three or four times the number of vertices compared to one fine mesh $\mathbf{FM_i}$, we merge the coarse meshes $\mathbf{CM_i}$ instead.

An incomplete model is PCA normalized by using its center of mass as the coordinate system's origin and its principal axes of variation as the coordinate system's axes. A well known problem of the PCA normalization is the existence of eight possible isometries to align the three principal axes to the x-, y- and z-axis. However, four of those involve mirroring, which ought to be excluded. The incomplete model with most vertices is selected to align the other models to, using our PCA/ICP algorithm (see Algorithm 3 applied to two models/meshes). This algorithm employs the point-to-plane ICP algorithm to each of the valid PCA isometries to find the optimal alignment of two models. In case of an object with three distinct eigenvalues, this algorithm has to try four different isometries. When an object has two similar eigenvalues, then swapping the two corresponding eigenvectors might be a necessity to find the optimal alignment. In case of even three similar eigenvalues the algorithm requires a maximum of 24 valid PCA isometries. Since the point-to-plane ICP algorithm with a large number of target pairs returns only a low RMS distance for the correct alignment of two models, a cut-off criterion can be used to limit the number of tried PCA isometries. This is accomplished by returning the minimal distance and the corresponding 3D rigid transformation once the algorithm reaches a state in which the minimal distance is sufficiently smaller than the maximal encountered ICP distance, determined by threshold t_{dist} (see Sect. 3.2.6). As a result the algorithm requires at least two and at most 24 applied point-to-plane ICP algorithms. With the alignment of the accepted groups the final alignment of meshes is completed. When a single mesh is included in multiple groups, its 3D transformation with the lowest ICP distance is retained.

3.2.6 Implementation

Meshes obtained by a laser range scanner can be highly over-sampled. For a time efficient method, we size normalize and resample the meshes $\mathbf{M_i}$. The resampling is done by extracting a synthetic range image from all meshes using a fixed resolution (i.e. storing its z-buffer). This resolution is automatically selected, such that the largest mesh is resampled using 20,000 sample points. These synthetic range images are converted into our fine meshes $\mathbf{FM_i}$ by connecting adjacent range samples. Similarly we use only $\frac{1}{16}$-th of the extracted range samples to obtain our coarse meshes $\mathbf{CM_i}$. The meshes $\mathbf{FM_i}$ and $\mathbf{CM_i}$ are cleaned by removing faces with an edge longer than t_e ($t_e = 3.8$) times their resolution and removing connected components with less than t_f ($t_f = 20$) faces divided by the mesh resolution. The advantage of this remeshing process is that each vertex in a coarser mesh still represents a point on the originally constructed surface,

Algorithm 3 PCA/ICP (M_1, M_2)

$dist_{min} \leftarrow \infty$, $dist_{max} \leftarrow 0$
for $i \leftarrow 1$ **to** 6 **do**
  $M_0 \leftarrow M_1$
  $dist \leftarrow$ RMS from point-to-plane ICP(M_0,M_2)
  update $dist_{min}$ **and** $dist_{max}$
  if $t_{dist} \times dist_{min} < dist_{max}$ **break**
  $M_z \leftarrow$ rotate M_1 180° around the z-axis
  $dist \leftarrow$ RMS from point-to-plane ICP(M_z,M_2)
  update $dist_{min}$ **and** $dist_{max}$
  if $t_{dist} \times dist_{min} < dist_{max}$ **break**
  $M_y \leftarrow$ rotate M_1 180° around the y-axis
  $dist \leftarrow$ RMS from point-to-plane ICP(M_y,M_2)
  update $dist_{min}$ **and** $dist_{max}$
  if $t_{dist} \times dist_{min} < dist_{max}$ **break**
  $M_x \leftarrow$ rotate M_1 180° around the x-axis
  $dist \leftarrow$ RMS from point-to-plane ICP(M_x,M_2)
  update $dist_{min}$ **and** $dist_{max}$
  if $t_{dist} \times dist_{min} < dist_{max}$ **break**
  if i = 1 **then** swap most similar eigenvectors
  else rotate M_1 90° to a new unique pose.
end for
return $dist_{min}$ **and** 3D rigid transformation

and that the extraction of right-, top-, left- and bottommost sets of vertices becomes a trivial process. A disadvantage is that very slender object parts may not be sampled and become excluded from a coarser mesh. See Fig. 3.5 for some resampling results.

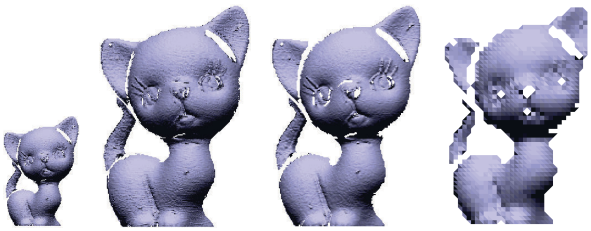


Figure 3.5: From left to right: the original, the resized, the finely resampled, and the coarsely resampled meshes of the *C2 kitten* object.

To match sets of features with the use of Algorithm 1 a maximum number of four iterations is sufficient. For Algorithm 2 we use 30 iterations for i_{max}, and a target number of pairs (p_{target}) and a minimum number of pairs (p_{min}) equal to, respectively, 80% and 6% of the maximum amount of vertices of the two meshes M_1 and M_2. During the verification of quadruples we remove 3D transformations with a distance $t_{dist} = 2.5$ times the minimal RMS distance, and reject groups with a tight bounding box volume larger than $t_{bb} = 1.25$ times the median tight bounding box volume of all groups. Finally, we use a cut-off criterion of $t_{dist} = 3.8$ in Algorithm 3.

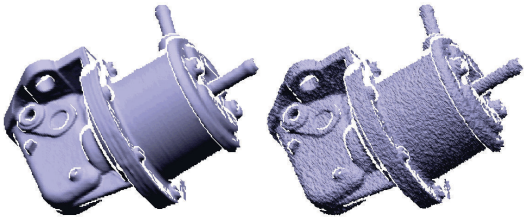


Figure 3.6: Adding synthetic noise to a clean range scan.

3.3 Datasets

To test the effectiveness of our method we apply it to five cluster (C1-C5) of increasing difficulty, with a total number of 25 objects. The first cluster consists of four existing 3D models from which 64 meshes were generated as follows. All models were scaled to a 200 mm sized model and transformed to eight unique poses. For each model's pose a range image (i.e. z-buffer) was generated for eight different sides, rotating the model in steps of 45° around the Y-axis. The range images were converted to meshes by connecting adjacent vertices and the meshes were cleaned by removing faces with a normal almost perpendicular ($\alpha_\Delta < 10^\circ$) to the scan direction. To model scanning noise (Fig. 3.6), we randomly displaced each vertex within range $[-\frac{1}{2}\eta, \frac{1}{2}\eta]$ in its normal direction, with η equal to the average edge length in the mesh (we used Trimesh2 for this [86]). The original cluster models are used as ground truth models, to compare the quality of our final alignment to that of a multiview ICP algorithm.

The second cluster consists of seven objects scanned using the Roland LPX-250 laser scanner. This device scans objects orthogonally in the scan direction and has a rotation table, which we employed to obtain sets of either four or eight range scans per object's pose.

The third cluster contains a subset of three objects from the first cluster. These objects were scanned with a Minolta vi-910 for four, eight or twelve views per pose with approximately the same angle between sequentially scanned views.

The fourth cluster consists of Mian's four objects [71]. It is used to show the applicability of our method to sets of range scans without an exact number of views per pose.

To compare our results with Mian's alignment results [72] we included a fifth cluster from the Stuttgart Range Image Database [97] that he used in his work: a cluster of seven 3D models from which per model 66 range images were synthetically generated. Again the range images were converted into meshes and cleaned, and noise was added.

The models from cluster C1 and most of the models and scans from clusters C2 and C3 are available in the AIM@SHAPE shape repository [1].

Object	Our method		MeshAlign	
	RMS(mm)	time(sec)	RMS(mm)	time(sec)
C1 bimba	1.333	93	1.332	340
C1 elephant	2.683	96	2.645	408
C1 greek	3.112	99	3.095	439
C1 oil-pump	2.833	98	2.811	363

Table 3.1: The quality of the alignments and the time to obtain them. Our method was applied to unaligned scans and MeshAlign to our alignment. The quality was measured using the RMS distance from aligned vertices to ground truth models.

3.4 Results

In Sect. 3.4.1 we evaluate the fine alignments obtained by our method, and we show that the quality of our alignments is comparable to the alignment obtained by a multiview ICP algorithm. Since our final alignments are of high quality, the meshes are directly merged towards a single surface mesh (Sect. 3.4.2). In Sections 3.4.3 and 3.4.4 we describe the effectiveness and efficiency of our method.

(a) C1 oil-pump (64/64) (b) C2 kitten (24/24) (c) C3 memento (32/36) (d) C4 dino2 (16/21) (e) C5 pit-bull (42/66)

Figure 3.7: One final alignment of each cluster C1-C5. The fraction of successfully aligned meshes $\mathbf{FM_i}$ is shown.

3.4.1 Fine alignment quality

For each cluster an object's final alignment (of meshes $\mathbf{FM_i}$) is shown in Fig. 3.7, the other alignments are of similar quality. Because the alignment of grouped meshes were refined during quadruple verification and all accepted groups were finely aligned using the PCA/ICP algorithm, this figure shows qualitative good final alignments.

For the models in cluster C1, we compared the quality of our alignments to the results of a multiview ICP algorithm. For this comparison we applied MeshAlign (v.2) [53] to our alignment to see if the alignment improves. MeshAlign applies Pulli's [82] multiview ICP algorithm to improve the initial alignment of large datasets. The quality of an alignment is quantified using Metro [32], which in our case computed the RMS distance of all vertices of the aligned meshes towards the ground truth model. Results from Table 3.1 show that MeshAlign's multiview ICP algorithm barely improves (less than 1.5%) the alignment we obtained using our method. In other words, our method returns a highly accurate alignment similar to the result of a multiview ICP algorithm.

3.4.2 Merged model quality

When an accurate fine alignment of meshes is obtained, a merge method can construct a single surface out of the partially overlapping meshes. We used MeshMerge [53] to merge our sets of meshes. MeshMerge builds a carefully weighted distance field for each mesh, and blends all the distance fields together in a seamless way in a single volumetric representation. The final surface is reconstructed through the standard Marching Cubes algorithm [64].

The merged models in Fig. 3.8 show qualitative good models, even though some models show holes due to missing scan data or to the lack of aligned meshes (Fig. 3.8(l)). Notice that the models from clusters C1 and C5 are still noisy due to the added noise. Results for the *biplane* are based on the alignment of the original meshes, rather than resampled data, to avoid data loss near slender parts (such as the wings in side views). To inspect the full models we refer to [98].

3.4.3 Effectiveness

The results in Fig. 3.8 and Table 3.2 show that our method is able to align mesh sets of objects scanned using different acquisition systems (different clusters), objects with few and many protrusions (e.g. *C2 buste* vs. *C2 memento*), objects with smooth and rough surfaces (e.g. *C3 pierrot* vs. *C3 warrior*), and mesh sets with little and much noise (e.g. *C3 memento* vs. *C5 pit-bull*). Furthermore, we see that our method aligns all meshes from C1 and C2, which are the meshes from the optimal acquisition process. For the datasets in clusters C3 and C4 the majority of meshes are aligned. The difficulty with the meshes in C3, is that the acquisition system's software generated rigorously cleaned meshes, which eliminated useful data. Our method automatically aligns a significant number of meshes (upto 51) for the noisy datasets in cluster C5. Note that our method performs better on the scan reconstructed models *bone*, *dino3*, *dragon*, *frog* and *pit-bull*, than on the highly symmetric CAD models *porsche* and *biplane*. In total our method aligns all meshes from 12 out of 25 datasets and 80% of the total number of meshes.

The results of Mian's method shown in Table 3.2 are based on his automatic pairwise registration algorithm [72]. To obtain the correct alignment of these meshes, he had to select an *ordered* subset of (18 to 26) meshes for which a priori knowledge had to be defined about the overlap of views. Knowing which view pairs had overlap, his method was able to align all selected (noiseless) meshes. To align all 66 meshes, he would have to order all meshes while our method is able to automatically align large numbers of meshes (with noise) without a priori knowledge.

3.4.4 Efficiency

Our method aligns datasets of N meshes $\mathbf{M_{i\in(1\ldots N)}}$. For each mesh we synthetically generate a range image from which we construct a fine mesh $\mathbf{FM_i}$, a coarser mesh $\mathbf{CM_i}$ and two sets of 2D features. Then each fine mesh is matched to all other fine meshes using the 2D features, which is quadratic in the number of meshes. It takes constant time to generate quadruples in which four meshes are coarsely aligned to each other. Similar quadruples are removed, leaving N' quadruples of meshes to be verified. During

Object	#Meshes (1041)	#Unique Quadruples	#Accepted Groups	#ICPs in PCA/ICP	Time (sec)	#Aligned Our (828)	#Aligned Mian
C1 bimba	64	16	16	44	93	64	-
C1 oilpump	64	16	16	39	96	64	-
C1 greek	64	16	16	43	99	64	-
C1 elephant	64	16	16	38	98	64	-
C2 angel	16	4	4	16	16	16	-
C2 buste	16	4	4	6	15	16	-
C2 kitten	24	6	6	14	22	24	-
C2 pierrot	20	5	5	14	23	20	-
C2 memento	20	5	5	12	19	20	-
C2 warrior	24	6	6	16	23	24	-
C2 bozbezbozzel	44	15	15	52	73	44	-
C3 pierrot	24	12	10	77	67	15	-
C3 memento	36	11	9	23	44	32	-
C3 warrior	24	10	8	22	33	24	-
C4 chef	22	14	14	30	72	21	-
C4 chicken	16	6	5	10	31	12	-
C4 dino1	16	13	13	40	66	13	-
C4 dino2	21	16	13	37	71	16	-
C5 bone	66	52	50	119	253	51	23
C5 dino3	66	48	42	100	221	44	21
C5 dragon	66	51	46	113	248	47	21
C5 frog	66	51	46	113	248	47	19
C5 pit-bull	66	51	42	105	224	42	22
C5 porsche	66	36	12	37	137	29	18
C5 biplane*	66	40	9	18	265	15	26

Table 3.2: Details of our alignment method. *The original meshes were used.

the verification of all quadruples, four mesh-to-mesh comparisons are applied involving a point-to-plane ICP algorithm, which is of $O(N')$. After verification there are N'' accepted finely aligned groups of either three or four meshes left. The coarse meshes $\mathbf{CM_i}$ are "merged", that is in this case, considered to be the surfaces of one mesh without additional computations. Each merged group has fewer vertices than a single fine mesh, because a group consists of three or four coarse meshes and each coarse mesh $\mathbf{CM_i}$ has approximately $\frac{1}{16}$ of the fine mesh's vertices. Finally, we apply N'' times the PCA/ICP algorithm for which we use at most 24 times the point-to-plane ICP algorithm, in other words, we apply at most $24N''$ mesh-to-mesh comparisons, which is of $O(N'')$. Since $N'' \leq N' \leq N$ we have a total number of $O(N)$ mesh-to-mesh comparisons after the selection of quadruples.

The N^2 mesh-to-mesh comparisons during the matching of features are based on low cost operations only. Experiments show that the time to match the sets of 2D features is much less than the time to verify the quadruples, even for the datasets with 66 meshes. Thus, up to 66 meshes the group verification with a number of $O(N')$ comparisons is the bottleneck. However, the group verification is also the main reason why we don't need to apply a multiview ICP algorithm, which would require an additional $O(N^2)$ mesh comparisons. For comparison, the time that MeshAlign's multiview ICP algorithm required to "improve" the alignment of the 64 meshes of the *C1 bimba* model was 340 seconds, which is even more than the total time of 93 seconds (see Table 3.2) our algorithm required to obtain an alignment of similar quality starting from the *bimba*'s

unaligned meshes.

Our method's feature extraction requires less than one second, the verification of a group two to three seconds, and the alignment of groups nearly a second per accepted group. For the alignment of the majority of 66 noisy scans, our method needed two to five minutes in total. Sets of 64 meshes from cluster C1 were even aligned in less than two minutes. All timings are based on a Pentium IV 2,8 GHz with 520 MB internal memory.

Obviously, the total number of applied point-to-plane ICPs (Algorithm 3) shown in Table 3.2 is related to the number of accepted groups. In general two to five point-to-plane ICPs were required to align an accepted group. In Sect. 3.2.5 we already stated that this number could increase when the object has similar eigenvalues. This is what happened for *C3 pierrot*. The effect on *C2 pierrot* is however much less, because its data is more complete.

3.5 Discussion

A limitation of our method is that each quadruple (and thus each scan) should cover more or less the entire object, because accepted quadruples are aligned using their center of masses and principal axes. Scans that cover only a small part of an object's view, or scans without proper left and right neighbors are most likely not aligned. Although the matching of features can correct for small rotations between neighbors, the method performs best on horizontally scanned meshes. This also explains the difference in performance for clusters C1 and C5: the latter consists of scans without any pose assumptions, while the first has for each pose eight scans (i.e. two quadruples) with the same bottom-up direction. Nevertheless, our method performs very well on the scans from cluster C5.

3.6 Concluding remarks

In this chapter, we presented a new and efficient method to perform both the coarse and fine alignment of meshes. Our method selects for each mesh three other meshes that represents its left, right, and back side neighbors. Such quadruples are then verified and the accepted groups of three or four meshes form incomplete 3D models of an object pose. These incomplete models are pose normalized and with the selection of the optimal PCA isometry (based on its ICP-error) these models can be correctly aligned. With the alignment of the incomplete models we obtain our final alignment.

Our method is capable of aligning various sets of meshes both effectively and efficiently. Effectively, because 80% of the total amount of meshes were automatically aligned and even all meshes of 12 out of 25 datasets were completely aligned, without a priori knowledge. Efficiently, because our method requires $O(N^2)$ mesh-to-mesh comparisons using low cost 2D silhouette feature matching. The quadruple verification and group alignment use high cost operations, but require only $O(N)$ mesh comparisons. In contrast, other coarse alignment systems apply a multiview ICP algorithm to refine the obtained coarse alignment, which requires $O(N^2)$ high cost mesh comparisons with a final alignment of similar quality as a result.

(a) C1 bimba $\frac{64}{64}$ (b) C1 elephant $\frac{64}{64}$ (c) C1 greek $\frac{64}{64}$ (d) C1 oil-pump $\frac{64}{64}$ (e) C2 angel $\frac{16}{16}$

(f) C2 buste $\frac{16}{16}$ (g) C2 kitten $\frac{24}{24}$ (h) C2 bozbezboz. $\frac{44}{44}$ (i) C2 pierrot $\frac{20}{20}$ (j) C2 memento $\frac{20}{20}$

(k) C2 warrior $\frac{24}{24}$ (l) C3 pierrot $\frac{15}{24}$ (m) C3 memento $\frac{32}{36}$ (n) C3 warrior $\frac{24}{24}$ (o) C4 chef $\frac{21}{22}$

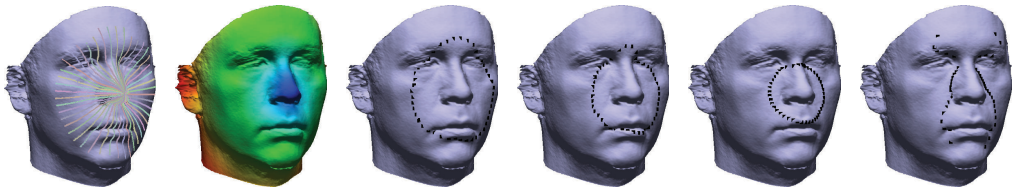
(p) C4 chicken $\frac{12}{16}$ (q) C4 dino1 $\frac{13}{16}$ (r) C4 dino2 $\frac{16}{21}$ (s) C5 bone $\frac{51}{66}$ (t) C5 dino3 $\frac{44}{66}$

(u) C5 dragon $\frac{47}{66}$ (v) C5 frog $\frac{47}{66}$ (w) C5 pit-bull $\frac{42}{66}$ (x) C5 prosche $\frac{29}{66}$ (y) C5 biplane $\frac{15}{66}$*

Figure 3.8: The final reconstructed 3D models of clusters C1-C5. The surface merging was performed with MeshMerge on the fine meshes $\mathbf{FM_i}$. The fraction of successfully aligned meshes used to reconstruct a model is shown. *The original meshes were used.

Chapter 4

Face matching



In this chapter, we use 3D laser range scans for biometric purposes. By scanning the face surface with a laser range scanner, we acquire a 3D surface mesh of the face which differs from one person to another. Because of the ease with which the 3D face data is acquired, the matching of 3D face surfaces is an interesting method to compare individuals. The two main applications for face matching are face recognition and face retrieval. Face recognition is divided into face identification, i.e. identify the person in a face image, and face verification (or authentication), i.e. verify that an input face corresponds to a given claimed identity. For face verification only two faces are matched, and a threshold has to decide whether the match score is good enough to pass the verification test (e.g. at airport check-ins). For face identification, we have an input face and a dataset of known faces, and we need to find the best match from this dataset. Face retrieval is an extension of face identification, in which we consider not only the best match, but a range of best matches (e.g. for searching criminal records). When we want to use 3D face scans for these applications, the 3D face matching algorithm must achieve high recognition and retrieval rates in a time-efficient manner. Since a laser range scanner rapidly captures thousands of depth measures, the comparison to all faces in a database using all measurements becomes an intractable task, and efficient face matching algorithms are required.

Among the many 3D face matching techniques that have been developed are variants of 3D facial curve matching, which reduce the amount of face data to one or a few 3D curves for efficiency reasons. The face's central profile, for instance, proved to work well. However, the selection of the optimal set of 3D curves and the best way to match them has not been researched systematically. In this chapter, we implement a 3D face matching framework that enables profile and contour based face matching. Using this framework we evaluate profile and contour types including those described in the literature, and select subsets of facial curves for effective and efficient face matching.

4.1 Introduction

Before the recent developments in 3D laser scanning, the difficult task of automated face recognition was based on the comparison of 2D images. To automatically recognize a person in different images requires a system to select and match the proper set of corresponding facial features. For a 2D face recognition system to be generally applicable, it needs to cope with variances in digitizers (e.g. color, resolution, and accuracy), subjects (pose, coverage, and expression), and settings (lighting, scaling, and background). The introduction of 3D laser scanning in this area proved to be very useful, because of its invariance to setting conditions: illumination has little influence during the acquisition, the 3D measurements result in actual sized objects, and the depth information can easily separate foreground from background. 3D face information has found its application in face retrieval, face recognition, and biometrics.

4.1.1 Related work

The task of recognizing 3D faces has been approached with many different techniques as described in surveys of Bowyer et al. [21] and Scheenstra et al. [92]. Bowyer et al. [21] divide the 3D face recognition challenge into 3D face recognition and multi-modal 3D + 2D face recognition. Although face recognition may benefit from the 2D texture information, our work can be categorized as a 3D shape based method.

An algorithm that is often applied in the context of 3D face recognition, is the Iterative Closest Point (ICP) [16] algorithm. This algorithm is able to align and evaluate merely the overlapping parts of two surfaces, which makes it robust to scanning deficiencies, such as missing data and outliers. However, in case of a facial expression, the acquired 3D face surface suffers from non-rigid deformations. This makes the direct application of the ICP algorithm for face recognition less reliable as shown in [8]. Mian et al. [68] proposed to extract expression insensitive regions of the face, and uses a variant of the ICP algorithm for the matching. Cook et al. [34] apply the ICP algorithm to align two 3D faces and analyze the distribution of closest point distances using Gaussian mixture models to decide if the two faces belong to the same subject. Chang et al. [27] extract multiple (overlapping) regions around the nose and match these surfaces using ICP to find similar faces. Recently, Faltemier et al. [42] developed a method that normalizes the pose of each face, extracts a predefined set of 38 surface patches, and matches pairs of patches among two scans using the ICP algorithm. To deal with facial expressions, Lu et al. [65] apply expression models to deform a neutral scan such that it fits an expression scan. For the fitting and matching they use the ICP algorithm. The main problem of the above ICP based methods, is that the time costly ICP algorithm is applied during the actual matching of 3D faces, which makes them impractical for face recognition in large sets of 3D face scans.

To improve on the face matching time, many researchers have focused on the extraction of salient features that can be matched time-efficiently. Al-Osaimi et al. [3] developed a method that combines local and global geometric information of the face in 33 2D field histograms, applies principal component analysis (PCA) to each histogram for data reduction, and constructs a single feature vector per face scan. These feature vectors were used to recognize faces without expressions, which is still an active field

of research. In methods of Blanz et al. [18] and Amberg et al. [5], a 3D morphable face model is fitted to the scan data. In the fitting process, the model's coefficients are adjusted such that the face model morphs towards the scan data. In the end, the coefficients are used in a feature vector for face recognition.

In this chapter, we focus on 3D face recognition with the use of 3D curves extracted from the 3D geometry of the face. Several *facial curve based* methods have been proposed in the past. The common goal of these methods is to extract a set of 3D curves that can be effectively used for 3D face recognition. In the work of Li et al. [63], the authors extract the central profile curve and a depth contour curve from 2D depth images. They show that the recognition rate for the combination of the two curves is higher than for each curve individually. Gökberk et al. [46] published recognition results based on sets of seven vertical profile curves. Samir et al. [90] proposed a face matching method that extracts several depth contours. To match two faces, they compute the minimum energy to bend the extracted depth contours of one face to their corresponding depth contours of the other face. In [91], the same authors propose a framework that describes face surfaces with the use of geodesic contour curves and impose a Riemannian structure that measures the required energy to bend one surface to the other. They use their method to optimally deform one surface to the other, and intent to use it for face recognition purposes. Similarly, Bronstein et al. [22] compute surface geodesics for two faces, deform the surfaces such that the geodesic distances become Euclidean ones, and compare the new surfaces using a moments-based distance measure. Instead of geodesic contour curves, Berretti et al. [15] use geodesic stripes and their spatial relationship to identify faces.

All of the above mentioned curve-based methods require one or more reference points, such as the tip of the nose, to start the extraction of facial curves from. Furthermore, these methods require a normalized pose of the face scans, except for those methods that use pose invariant surface geodesics. To accurately locate such reference points, information on the face's pose is used. Several methods to locate the tip of the nose and to normalize the face's pose have been proposed in the literature. Li et al. [63] assume that the tip of the nose is the point closest to the scanning device. They assume an upward pose and facial symmetry to further optimize the face's pose. Mian et al. [68], assume an upward pose of the scanned subject and determine for a set of horizontal slices, what the most protrusive point is for the intersection curve. The most protrusive point among the tried slices is selected as the tip of the nose and facial symmetry is employed to normalize the face's pose. Gökberk et al. [46], use ICP to align a face scan to a face template. The assigned landmarks on the template are transferred to the scan data and facial symmetry is used to further normalize the pose and improve on the landmark locations. Chang et al. [27], subsample frontal depth images to a point where surface curvature can be effectively used for nose tip localization. Pit, peak, and saddle regions that correspond to eye corners, nose tip, and nose bridge are used to normalize the pose of the face. Such curvature information is also applied in [15, 22].

The assumption of the tip of the nose being the vertex with the highest z-value is often wrong, because different poses, hair, clothes, noise and even expressions may interfere. Assuming an upward pose of the face in which the nose tip is the most protrusive point, does not hold in case of full head scans in which an ear or hair could easily be the most protrusive point. When the ICP algorithm is used to align a scan to a

face template in order to normalize the pose, an initial alignment is required. One could use the center of mass and the principal axes of data variance, for the initial alignment, but this only works when the scan is highly similar to the template. The robustness of surface curvature heavily depends on the data density and the level of noise. The way Chang et al. [27] subsample the data density applies to depth images only.

Xu et al. [120] also pointed out some of these problems, and proposed a bottom-up approach to select the tip of the nose in a robust manner. As a first step, they select vertices among the scan that are most protrusive within a small sphere (radius 20 mm). Secondly, they use a Support Vector Machine (SVM) to select vertices that locally resemble a nose tip. Finally, they select the location with most nose tip candidates in its neighborhood as the tip of the nose. Starting from the selected nose tip, they track the nose ridge to normalize the face's pose. This method carefully discards potential nose tip locations based on local surface properties, but it fails in some simple cases where a more global notion of a face could easily improve the results. In this work we present a method that locates the tip of the nose and normalizes the face's pose at the same time using local nose tip properties (as in [120]) in combination with a more global shape template (as in [46]).

To compare face recognition techniques, Face Recognition Grand Challenge sets [79] are publicly available. A more general SHape REtrieval Contest (SHREC) [113] has been organized in the past years to evaluate the effectiveness of 3D-shape retrieval algorithms. This contest considers 3D faces as a subset of the generic graphical models track, but also as a separate 3D face retrieval track. In this chapter, we analyze 3D face models for the interrelated tasks of both face retrieval and face recognition.

4.1.2 Contribution

The contributions of this chapter are the following. First, we introduce a new face pose normalization method that is applicable to face, full head and even full body scans in any given orientation. This method overcomes the limitations of previous methods, and it only requires a triangulated point cloud containing the tip of the nose. Second, we present a 3D face matching framework to extract and match 3D face curves and optimize its settings. Thirdly, we evaluate sets of profiles and contours including those described in the literature. Fourthly, we present new combinations of curves to perform both effective and time-efficient face retrieval. One of these combinations with only eight curves of 90 face samples each, achieved a mean average precision (MAP) of 0.70 and 93% recognition rate (RR) on the SHREC'08 face set [113] and a MAP of 0.96 and 98% RR on the UND face set [26].

Our face pose normalization (Sect. 4.4.1) fits 3D templates to the scan data and uses the inverse transformation of the optimal fit to normalize the face's pose. The tip of the nose is extracted from the scan data in the process. Our 3D face matching framework (Sect. 4.5) uses the nose tip as its origin and extracts a set of profile curves over the face surface. Then, it extracts samples along the profiles, which are used to determine the similarity of faces. In Sect. 4.5.4, we combine such samples in profile and contour features and select sets of features for effective and efficient face matching.

In this chapter, we trained our feature sets on realistic synthetic data created with a morphable face model (Sect. 4.6 and 4.7), and do the testing on real face scans (Sect. 4.8).

Feature sets that perform well on the synthetic face models are used for face retrieval in two popular datasets of real face scans. To cope with scanner noise and missing data, we apply an automatic mesh improvement algorithm. Furthermore, we experimented with two different distance measures, and a percentage of best matching curve samples for expression invariance. In Sect. 4.9, we compare selected feature sets with an ICP and a depth map method, and with results described in the literature.

4.2 Morphable face model

The 3D morphable face model that we use is that of the USF Human ID 3D Database [109]. This statistical point distribution model (PDM) was built from 100 cylindrical 3D face scans with neutral expressions from which n=75,972 correspondences were selected using an optic flow algorithm. Each face shape S_i was described using the set of correspondences $S = (x_1, y_1, z_1, \ldots, x_n, y_n, z_n)^T \in \Re^{3n}$ and a mean face $\bar{S}$ was determined. Principal Component Analysis (PCA) was applied to these 100 sets S_i to obtain m eigenvectors of the PDM [108]. Because there are only 100 faces in the n dimensional face space, there are at most m=99 meaningful eigenvectors. The mean face $\bar{S}$, the eigenvectors $s_i = (\Delta x_1, \Delta y_1, \Delta z_1, \ldots, \Delta x_n, \Delta y_n, \Delta z_n)^T$, the eigenvalues λ_i ($\sigma_i^2 = \lambda_i$) and weights w_i are used to model new face instances according to $S_{inst} = \bar{S} + \sum_{i=1}^{m} w_i \sigma_i s_i$. Weight w_i, also referred to as coefficient, represents the number of standard deviations a face instance morphs along eigenvector s_i. Since the connectivity of the n correspondences in the PDM is known, each instance is a triangular mesh with proper topology and without holes. In chapter 7, we explain a new method to build a morphable face model from scan data.

4.3 Datasets

In this chapter we compare 3D faces from several datasets. The training sets were constructed using the morphable face model previously described. The commonly used test sets are collections of face scans that were acquired using laser range scanners.

Training set A. The 3D faces of training set A were generated with the morphable face model. To construct the query set, we created seven random instances (q) of the morphable model by assigning m random weights w_i within the range [-1.5,1.5]. Each of the queries was morphed to two other random instances (i_1 and i_2) of the morphable model to create new relevant faces (r). Five intensity levels of morphing were applied, namely a 90-10, 80-20, 70-30, 60-40, 50-50 weighting scheme for the m corresponding weights (e.g. $w_i(r) = 0.6w_i(q) + 0.4w_i(i_1)$). So, for each query we have eleven relevant models including the query. The final training set consists of seven queries and 176 face instances, that is, 77 relevant models and 99 random instances. A new random pose was assigned to each instance.

Training set A is used to investigate the properties of single curves based on four sampling strategies, to lower the number of profile and contour samples for effective and efficient face retrieval, and the selection of features sets similar to those described in the literature.

Training set B. The 3D faces of training set B were generated with the same morphable face model. At first, one thousand face instances were randomly created of which 64 were selected as a query q, 64 as instance i_1, and another 64 as i_2. To construct four highly relevant faces (r_h) each query was morphed in four directions with 60-40 weighting and four marginally relevant faces (r_m) using 40-60 weighing. Each query face was morphed towards i_1 and i_2, to the mean face $\bar{S}$ and away from $\bar{S}$. These additional 512 ($64(r_h + r_m)$) face models were added to the dataset, resulting in 1512 unique 3D face models and a query set of 64 faces. A new random pose was assigned to each instance, to introduce a non-trivial pose normalization problem. This dataset was also used in SHREC'07 - Shape Retrieval Contest of 3D Face Models [112]. The larger embedding and the morphing toward and away from the mean face are the major differences with training set A.

Training set B is used to reassess the features sets that were selected using training set A, and to select specific sets of features that perform well. With these feature sets we perform face retrieval in our two test sets.

Test set C. Test set C is the database from SHREC'08 - Shape Retrieval Contest of 3D Face Scans [113], which is a subset of the GavabDB [73]. This set differs from the training sets by having real laser range scans that suffer from noise and holes, and that each subject was scanned for different poses and expressions. The SHREC'08 set consists of Minolta Vi-700 laser range scans from 61 different subjects. The subjects, of which 45 are male and 16 are female, are all Caucasian. Each subject was scanned for different poses and expressions. The neutral scans include two different frontal scans, one scan while looking up (≈+35°), and one scan while looking down (≈-35°). The expression scans include one with a smile, one with a pronounced laugh, and an "arbitrary expression" freely chosen by the subject.

Test set D. The fourth dataset we use is the University of Notre Dame (UND) Biometrics Database [26]. This set consists of 953 2D depth images and corresponding 2D color textures from 277 different subjects acquired using the Minolta Vi-900 laser range scanner. All except ten scans were used in the Face Recognition Grand Challenge (FRGC v.1). In general, the set contains frontal scans of the face with a neutral expression. However, as shown in [67] the scans show small variations in pose and expression and capture both the face and shoulders areas. To obtain 3D triangulated surface meshes, the 2D depth images were projected to 3D with the adjacent depth samples connected with triangles. The color information was neglected.

4.4 Preprocessing

Like many face matching algorithms, our feature extraction algorithm is pose sensitive and requires the tip of the nose as a reference point. What we need to do is to normalize the pose of each 3D face and to detect the tip of the nose. The face scans in the different datasets vary in resolution and accuracy, pose and expression, and coverage of the face. We developed a method that normalizes the face's pose in a robust manner and a mesh improvement algorithm to improve on the mesh deficiencies in face data.

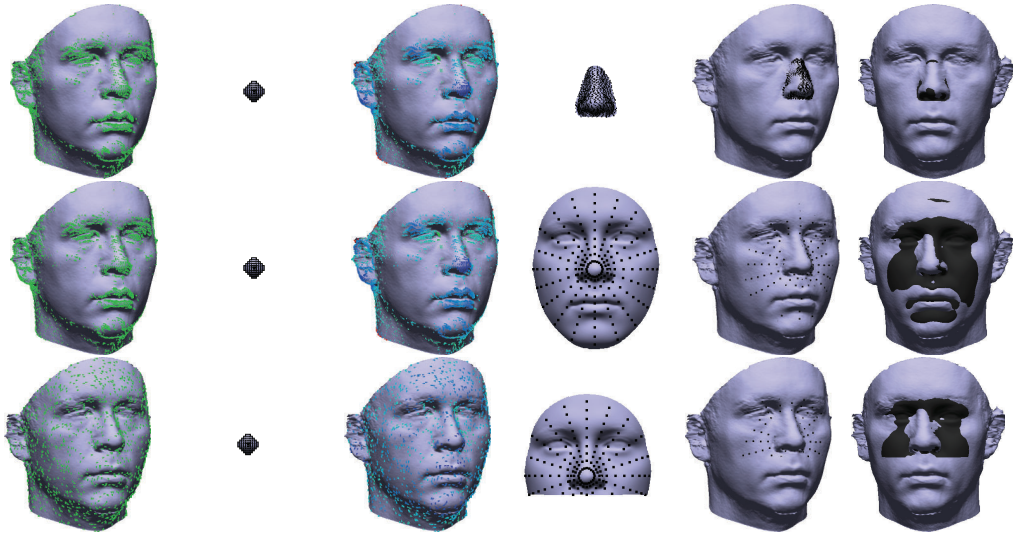


Figure 4.1: Face pose normalization. From left to right, the selection of potential nose tip locations (curvature or random sampling), the small nose tip template t_1, optimal (dark blue) locations for t_1, template t_2 (nose or face features), best t_2 fit, and the normalized pose.

4.4.1 Face pose normalization

Pose normalization is equivalent to correcting the viewing coordinate system that requires a view reference point, a view plane normal, and a view up vector [94]. In 3D face templates we specify the nose tip as view reference point, the gaze direction as view plane normal, and the face's pose as view up vector. By fitting these templates to potential nose tip locations in the scan data, we eventually obtain a new coordinate system in which the face's pose is normalized. In theory each point on a face scan can be considered as a potential nose tip location. The normal direction at a point can be estimated using the mesh' triangles. For the training sets we considered high (positive) curvature areas as potential nose tip locations, because the tip of the nose is generally a location with high curvature. For these sets a large number of potential placements could be excluded using a curvature threshold heuristic. However, depending on the resolution and mesh quality, the curvature information can be less reliable. This is the case for the two test sets. For these scans we randomly sampled the triangular surface mesh such that every $\approx$2.0 mm^2 of the surface is approximately sampled once. After the selection of potential nose tip locations, 3D template matching is applied using a nose tip template t_1 to determine which potential locations locally resemble a nose tip. To the locations where t_1 fits well, a larger template t_2 is fitted to select the actual nose tip and to normalize the pose. How this bottom-up scheme solves the unknown viewing coordinate system is described below and shown in Fig. 4.1.

First template. For each of the potential nose tip locations, we have its position p and normal direction n. The first 3D template t_1 is a *nose tip template* with the known view reference point p_{t_1} and view plane normal n_{t_1}. This template is highly symmetric around its normal n_{t_1}, which allows us to find the view plane normal while ignoring

the view up vector. To fit the nose tip template to the scan data, we place the nose tip template with p_{t_1} on p and with n_{t_1} aligned to n. The alignment is refined using the Iterative Closest Point (ICP) algorithm [16], which minimizes the Root Mean Square (RMS) distance of the template's vertices to their closest points in the scan data. As a result, we have for each potential nose tip a measure of how good t_1 fits that location, but also the view reference point and view plane normal defined in t_1.

Second template. We reduce the number of potential nose tip locations to only a few locations around the face, where t_1 fits well. To these locations we fit a second template t_2 that has a known view reference point p_{t_2}, view plane normal n_{t_2} and view up vector u_{t_2}. Clearly, the optimal fit of this template solves the pose normalization problem. This template is placed on the remaining locations with p_{t_2} on p_{t_1}, n_{t_2} aligned to n_{t_1} and a limited number of different view up vectors u_{t_2}. Since the angle between u_{t_2} and n_{t_2} is known, a view up vector can be instantiated using a rotation θ_{t_2} around n_{t_2}. Because the ICP algorithm is able to correct for small rotations we experimented with a new θ_{t_2} (i.e. view up vector) every thirty degrees. Each placement of t_2 is refined using ICP and the alignment with the lowest RMS distance is selected. The inverse transformation matrix for this optimal fit is used to normalize the face's pose. The point in the scan data closest to p_{t_2} is defined as the tip of the nose and used as the new origin.

Different templates based on the mean face $\bar{S}$ were used as t_2. For training set A, we search for the optimal placement of a nose as template t_2, which we refer to as *nose detection*. For training set B, we search for the optimal placement of either a nose (*nose detection*) or face template (*face detection*) to normalize the pose. This is to link our pose normalization method to the face matching method. For the test sets, we used only the upper half of the face template t_2 to make the pose normalization more robust to facial changes caused by expressions (as shown in [75]). For accurate face matching it is important to acquire the same pose for all face scans of one individual. With our template matching approach we assume that for each individual, these templates have a unique placement around the nose irrespectively to the face's proportion. This is a reasonable assumption, since the ICP algorithm minimizes the RMS distance, which enforces the alignment of typical protrusions such as the nose.

4.4.2 Mesh improvement

The test sets consist of triangle meshes acquired from laser range data and suffer from noise and missing data. Because our current implementation of the feature extraction requires a manifold surface mesh without holes, the noise needs to be removed and the holes interpolated. Furthermore, laser acquisition may even cause spikes around the tip of the nose, which in exceptional cases result in the incorrect selection of the nose tip after our pose normalization. During the mesh improvement in this section we select a new location as the tip of the nose.

In chapter 2, we pointed out that straightforward techniques to interpolate holes in triangle meshes using curvature information or flat triangles often fail in case of complex holes. To guarantee the interpolation of all holes, we operate on 2D depth images instead of triangle meshes (see Fig. 4.2 for illustrations). To do so, we converted the pose normalized triangles meshes (Fig. 4.2a) to 1 mm $\times$ 1 mm depth images using ray casting and then we applied the following image processing techniques. First, we crop

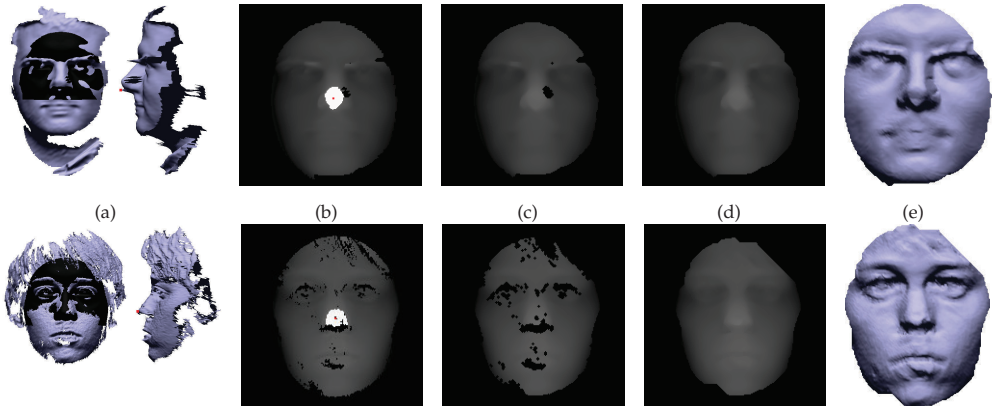


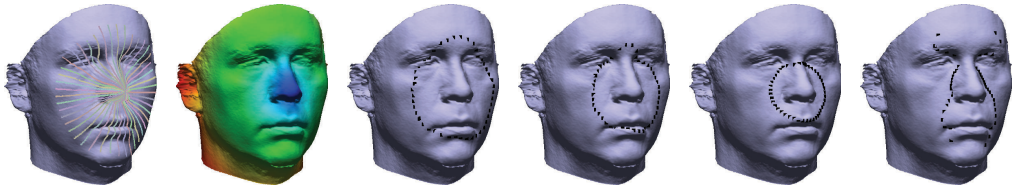
Figure 4.2: Mesh improvement. Scans are pose normalized and the (incorrect) nose tips are detected (a). Each scan is converted to a depth map with a new nose tip (b), noise is removed (c), holes are filled (d), and the final depth map is converted to a new 3D mesh (e).

the face in the 2D depth image ($d(x, y) = z$) using an ellipse of $x^2 + 0.6y^2 = 70^2$ mm, a sphere of $x^2+y^2+z^2 = 100$ mm and a minimal depth of 3 mm before the tip of the nose. To improve on the nose tip location, we select a small region around the initial nose tip and take its center of mass in 2D (Fig. 4.2b). Secondly, to remove noise and spikes in particular, we applied a 7 mm $\times$ 7 mm median filter and a binary erosion. Thirdly, facial symmetry left and right of the nose tip was used to remove non-face data, such as hair and noisy patches, by removing pixels that are at least 10 mm closer to the viewing point than its corresponding pixel on the other side of the face. Fourthly, components with less than 200 pixels are removed (Fig. 4.2c). Finally, holes are filled effectively by linearly interpolating data around the missing data. In more detail, for each missing depth value we trace its left and right boundary and linearly interpolate these depth values in a horizontal manner. Likewise, we interpolate missing data vertically and along the two diagonals. In the end, we average the (at most) four potential depth values to fill the holes (Fig. 4.2d). Extrapolation of depth values is avoided, because this interpolation requires, per direction, two valid depth values of a boundary. The final 2D depth image converted to a 3D mesh with its new nose tip centered in the origin (Fig. 4.2e).

4.5 Face matching framework

Starting from the tip of the nose in a pose normalized face, our framework extracts profile curves over the face surface in different directions. These sets of profile curves are used to determine the similarity of two faces. To match two profile curves, we match a set of samples along the curves. When combined, the samples in all profiles with the same constraints build up a face contour. The facial "Z-contour" [63, 90], for instance, is the curve that contains the samples from all profiles that have the same Z-value. The definition of a contour type determines which samples are extracted along the profiles (see Sect. 4.5.2).

To compute the similarity of two faces $\mathbf{A}$ and $\mathbf{B}$, we extract N_c samples for each



(a) (b) (c) (d) (e) (f)

Figure 4.3: From each face we extract N_p profile curves (a). A G-contour (c) is formed by selecting a sample on each profile (a) that has the same geodesic distance to the tip of the nose (colored in b). form a G-contour (c). Other contour curves are the C-contour (d), XY-contour (e), and Z-contour (f).

of the N_p profiles. Such a sample $\mathbf{A}_{ij}$ is defined as the intersection(s) of profile i and contour j. Because the profiles and their contour samples are extracted in a structured way, we can assume that these $N_p \cdot N_c$ samples correspond for faces $\mathbf{A}$ and $\mathbf{B}$. The distances between these corresponding samples introduce a dissimilarity. We use this information in a 3D face matching framework that consists of the generic formula

$$d(\mathbf{A}, \mathbf{B}) = \frac{1}{N} \sum_{i=1}^{N_p} \sum_{j=1}^{N_c} d_s(\mathbf{A}_{ij}, \mathbf{B}_{ij}) \tag{4.1}$$

which must be instantiated with the following parameters:

- The number of profiles N_p.
- The number of contours N_c.
- The distance measure for two corresponding samples $d_s(\mathbf{A}_{ij}, \mathbf{B}_{ij})$.

If a corresponding sample does not exist due to missing data, then we exclude that sample from the distance computation and lower the normalization factor $N \leq N_p \cdot N_c$. The complexity of our face matching framework depends on the values N_p and N_c. In case both parameters are large, then a lot of face data is used in the comparison, which is highly inefficient. With many profiles N_p and a few samples N_c, the 3D face comparison follows a contour matching approach. With a few profiles N_p and many samples N_c, the comparison follows a profile matching approach. The function $d_s(\mathbf{A}_{ij}, \mathbf{B}_{ij})$ measures the distance between samples that correspond according to the specified contour type. The extraction and matching of feature data is described in the following paragraphs. In Sect. 4.5.4, we use our framework to evaluate different face curves and select the most relevant profiles and contours for effective and efficient face matching.

4.5.1 Profile extraction

To obtain corresponding samples our framework first extracts a set of N_p profiles. A profile is defined as a 3D curve that starts from the tip of the nose and follows a path over the surface mesh with a predefined angle in the XY-plane. Such a path is defined by the intersection points of the mesh's triangles encountered along the way. We extract a profile for every $360/N_p$ degrees in the XY-plane with the tip of the nose as origin. We end a path whenever the Euclidean distance between the current location on the path and the nose tip becomes larger than 90 mm. Beyond this distance, data is less reliable

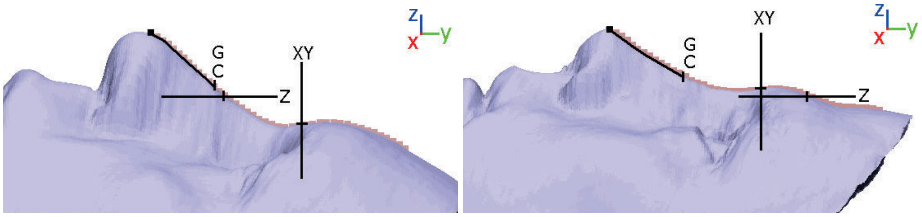


Figure 4.4: The central profile curve of two different faces with a "corresponding" G-, C-, XY- and Z- sample.

because of missing data and hair. Before profile extraction, the pose normalized face is centered with its nose tip at the origin, so that the extracted sets of profiles of two different faces are aligned. We assume a proper topology of the face surface, but to be less restrictive a profile can be defined as the intersection curve of the 3D face with a plane perpendicular to the XY-plane.

4.5.2 Feature data

After the applied pose normalization from Sect. 4.4.1, we can assume that profiles extracted in the same direction correspond. Given two corresponding profile curves A_i and B_i, we extract N_c corresponding samples ($\mathbf{A}_{ij}$ and $\mathbf{B}_{ij}$). Note that all samples are locations on the triangular surface of the face. We specify four different contour samples:

- *G-samples* - samples with a shortest geodesic path of r mm over the surface to the origin.
- *C-samples* - samples with a curve distance of r mm over the profile curve to the origin.
- *XY-samples* - samples with a circular distance of $r = \sqrt{(x^2 + y^2)}$ mm to the origin.
- *Z-samples* - samples with a depth distance of $r = z$ to the origin.

To extract G-samples, we first computed for each vertex its geodesic distance to the origin using the fast marching method [59] and interpolated these measures for the profile paths. When a profile path is sampled using N_c G-samples (with increasing r), we refer to it as a G-profile. A G-contour is the set of N_p G-samples at the same distance r. In Fig. 4.3, the four different contour curves are shown.

4.5.3 Feature matching

The extracted $N_p \cdot N_c$ samples from one face have an assumed one-to-one correspondence to those of an other face (see Figure 4.4). To match those samples we apply a symmetric distance measure that can be used for all four contour types and is rotation invariant. Therefore, we compare samples using their relative distances to the origin (i.e. tip of the nose) instead of their actual coordinates. We define the point-to-point distance (d_p) between a point p from sample $\mathbf{A}_{ij}$ and a point q from sample $\mathbf{B}_{ij}$, using the nose tip (p_{nt}=origin) and the Euclidean distance $e(p, q)$ as:

$$d_p(p, q) = (e(p, p_{nt}) - e(q, p_{nt}))^2.$$

This distance function is not a metric function because it does not satisfy the identity rule. Other distance measures for two 3D coordinates can be used instead. For a fair comparison of contour types, it is important that the samples are matched similarly. This is rather difficult, because a sample $\mathbf{A}_{ij}$ can be more than one point depending on the selected contour type. The Z-contour for instance, can have multiple points p on profile A_i with a similar Z-distance to the origin. Thus a Z-sample can have multiple points, while a C-sample and a XY-sample have at most one point. To deal with multiple points per sample we define the distance d_s between two corresponding samples $\mathbf{A}_{ij}$ and $\mathbf{B}_{ij}$ as the smallest distance between possible point pairs:

$$d_s(\mathbf{A}_{ij}, \mathbf{B}_{ij}) = min_{\forall p \in \mathbf{A}_{ij}, \forall q \in \mathbf{B}_{ij}} d_p(p, q).$$

In case either sample $\mathbf{A}_{ij}$ or $\mathbf{B}_{ij}$ is empty due to missing data, d_s is zero.

4.5.4 Feature selection

Our face matching framework is a useful tool to investigate the performance of profile curves and contour curves for face recognition purposes. In previous work, limited experiments were performed using either one or a few profiles and contours. With our framework we can easily select any set of profile and contour features to perform face matching with. To train and evaluate selected sets of features, we use them to query a training set. For each query, we compute its similarity to all other models in the training set, generating a ranked list of face models sorted on decreasing similarity values in the process. For each ranked list we compute the average precision, which is the average of precisions computed at those ranks where a relevant face is found. The precision (P) and average precision (AP) within the scope of retrieved items are defined as,

$$P(scope) = \frac{\sum_{s=1}^{scope} rel(s)}{scope}$$

$$AP(scope) = \frac{\sum_{s=1}^{scope} rel(s) \times P(s)}{\sum_{s=1}^{scope} rel(s)},$$

where $rel(rank) = 1$ if the face on the specified rank is relevant and zero otherwise. The scope we use equals the size of the dataset. High average precisions indicate the early retrieval of relevant faces, which is a desired property for our feature based face retrieval. The mean average precision (MAP) over all queries is used to assess a selected feature set.

4.6 Training on set A

In this section, training set A is used to investigate the properties of single curves based on different feature data. Furthermore, we experiment with the number of extracted profiles N_p and contours N_c, to find subsets of curves for both effective and efficient face retrieval.

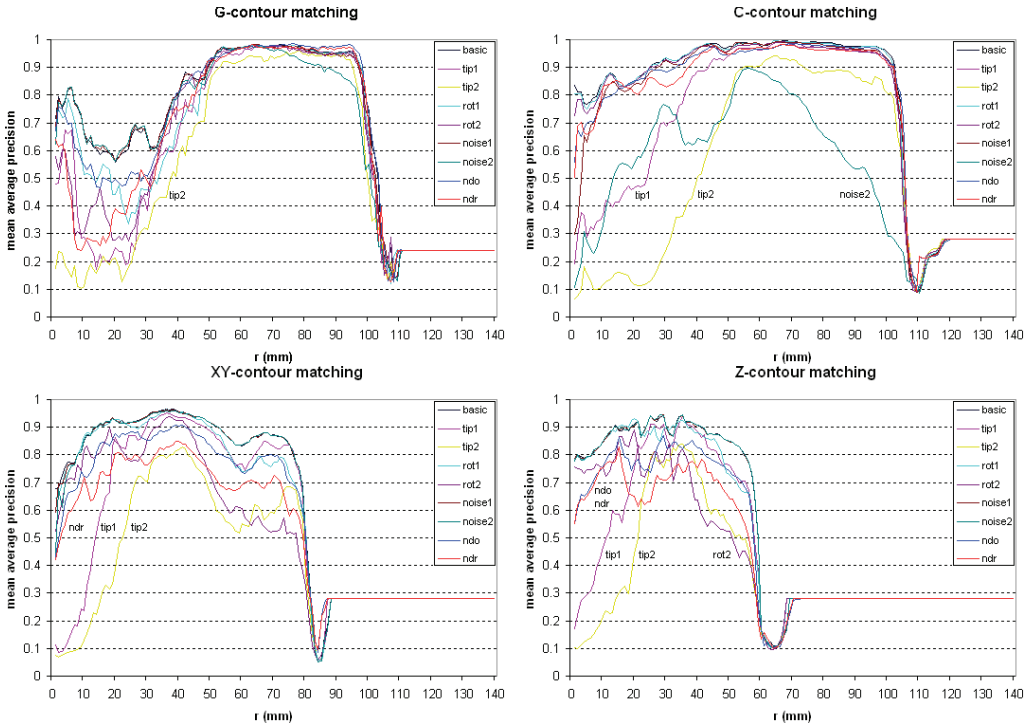


Figure 4.5: Training set A. The mean average precision graphs of single G-, C-, XY-, and Z-contours for sample values 1 to 140 mm under varying conditions.

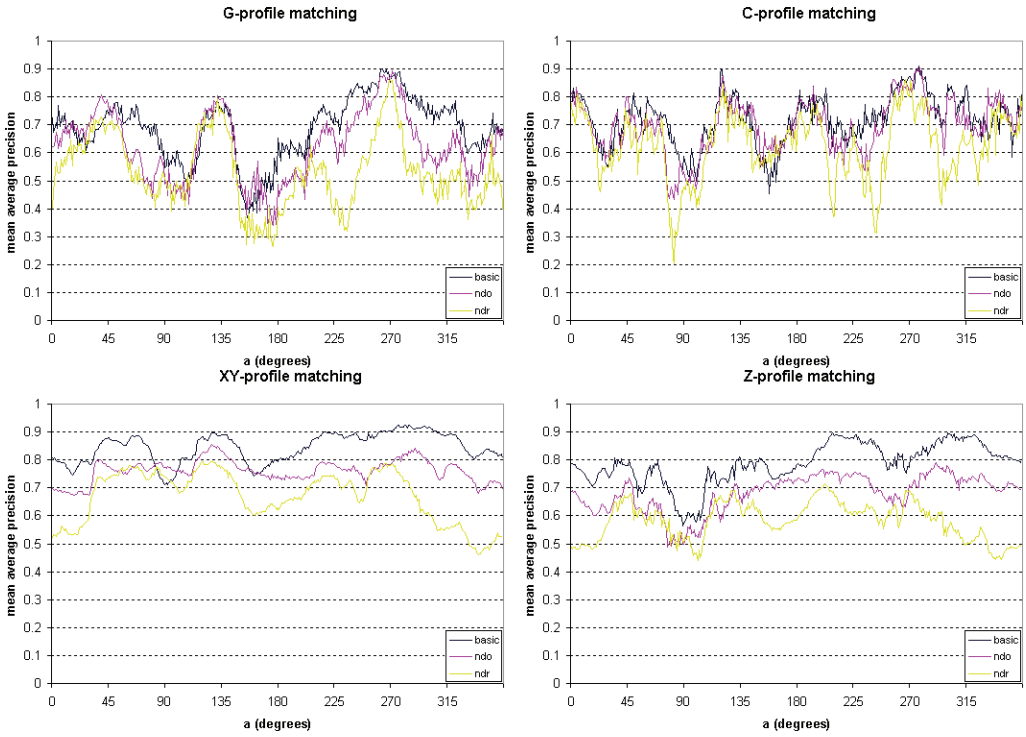


Figure 4.6: Training set A. The mean average precision graphs of single G-, C-, XY-, and Z-profiles for angles 0 to 359 degrees after automatic nose detections.

4.6.1 Single curve matching

For efficient face matching, previous work aims at reducing face information to a single distinctive curve. With our framework we can extract a single contour or profile curve, and assess its performance on our training set. We tested the robustness of single contours under varying conditions that are common in practice, such as small errors in nose tip localization and pose normalization, and different levels of noise. This was done by evaluating the MAP of each contour within the range r=[1,140] mm. Fig. 4.5 shows the following results for each of the contour curves:

- $basic$ - matching original query and database faces with known pose and nose tip location.
- tip - the queries were disrupted with a nose tip displacement, tip_1=2 mm and tip_2=4 mm from the actual nose tip.
- rot - the queries were disrupted with an Euler rotation (ρ,ρ,ρ), rot_1 with ρ=1 and rot_2 with ρ=2 degrees.
- $noise$ - the queries were disrupted with additional noise relative to the average edge length η in the mesh, $noise_1$ with 0.1η and $noise_2$ with 0.2η.
- nd_o - matching original query and database faces after automatic nose detection to normalize the pose and to localize the nose tip (Sect. 4.4.1).
- nd_r - matching randomly rotated query and original database faces after automatic nose detection.

From these results we learn the following:

1. Small changes in r can cause a large decrease in performance.
2. C-contours are more robust to errors in nose tip localization and pose normalization, and XY-, and Z-contours are more robust to noise.
3. G-contours on the outer regions of the face are robust under all these conditions.
4. Each contour type has an *active region* of r=1 mm to the r just before the MAP drops to a minimum, beyond that point a contour lacks sample data because of the face cropping.

The $basic$ results can be used as a reference for the optimal results. The nd_o results are comparable to methods that assume scans to be faced forward. For 3D or 2.5D face retrieval the nd_r results are important, because this involves pose normalization of faces (or head models) under all possible orientations.

To investigate how well a single profile can be used for the purpose of 3D face retrieval we plot the MAP of each sampling strategy per angle within the range a=[0,359]. Each profile is actively sampled with a large number of contour samples, that is 60 to 105 depending on the active region of a contour type. Profile matching suffers from disrupted queries in a similar manner as contour matching. Therefore, we show only the $basic$ results and the results after automatic nose detection (nd_o,nd_r). Fig. 4.6 shows that the maximal performance for a single contour is higher than for a single profile, which means that a single contour can be more descriptive than a single profile.

4.6.2 Multiple curve matching

Single curve matching has regions for which curves are able to obtain high performances, but a small change in range r or angle a can cause a large decrease in per-

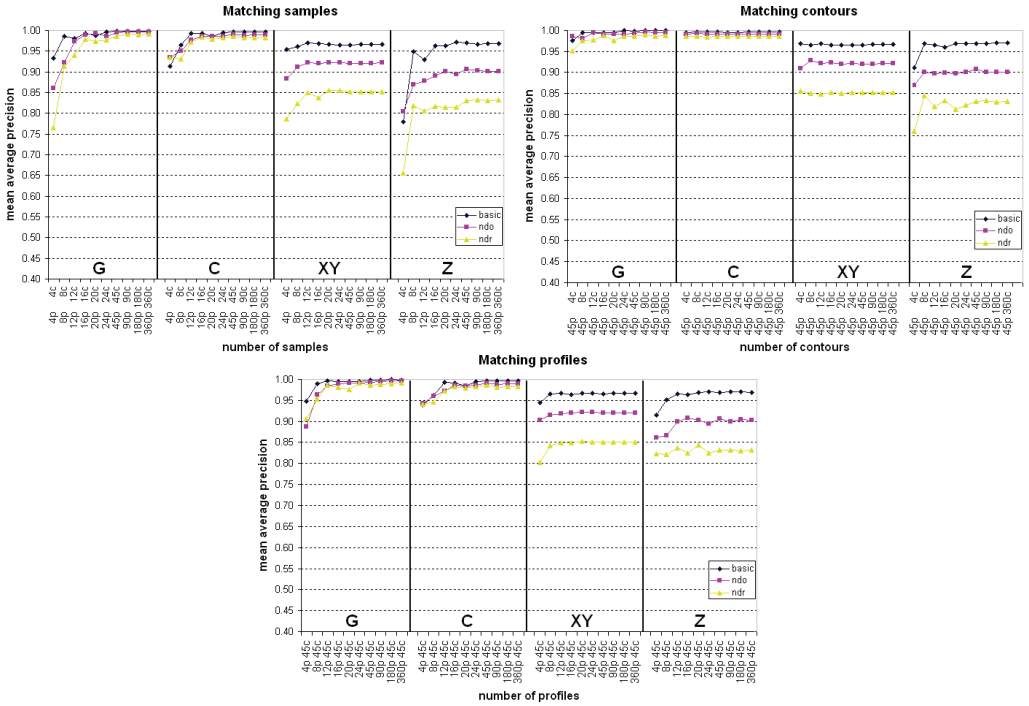


Figure 4.7: The performance while varying the number of samples, contours, or profiles.

formance. In other words, effective face retrieval based on a single curve has a small chance of success. In this section, we assess face matching using multiple curves, based on the $basic$, nd_o, and nd_r results from querying training set A.

To achieve effective face retrieval, using data from multiple curves is essential. However, there is a trade off between the effectiveness and efficiency. Parameters N_p and N_c of our framework determine the amount of samples used to describe a face. A first step is to decrease these numbers to a point were face matching is still effective, but more efficient. To do so, we extracted N_p=360 profile curves and sampled each profile with N_c=360 contour samples equally spaced over the *active region* (see previous section). From these 360 profiles and contours we selected subsets with a decreasing amount of samples $N_p \cdot N_c = n_f \cdot n_f$ with n_f={360,180,90,45,24,20,16,12,8,4}. Note that a set of 360·360 surface samples exceeds the number of vertices in our face models.

From the results in Fig. 4.7 we learn that the number of samples can be reduced from 360·360 to 45·45 without loosing discriminative power. Compared to the number of vertices of a face model, 45·45 samples is already a large reduction of face data. With a number of profiles N_p=45 we can investigate the performance of *multiple contours* by varying N_c={360,180,90,45,24,20,16,12,8,4} and the other way around for the retrieval performance using *multiple profiles*. Fig. 4.7 shows, in general, higher performances for the use of multiple curves compared to the use of a single curve. For a small number of curves the use of multiple contours outperforms the use of multiple profiles.

4.6.3 Central profile and optimal contour

In the work of Li et al. [63] face recognition is performed on 2D depth images using the combination of a single Z-contour at distance z=30 mm with the central profile curve from forehead to chin. With our framework we can perform face matching similarly by selecting the same Z-contour, the XY-profile from nose to forehead, and the XY-profile from nose to chin. Furthermore, we can manually select the best G-, C-, XY-, and Z-contour and the central G-, C-, XY-, and Z-profiles for training set A. We can combine these *manually selected* contours and profiles among different sampling strategies, which we refer to as *hybrid matching*. We selected the following contour curves with the highest MAP for the $basic$ results (see Fig. 4.5): the G-contour at 77mm, the C-contour at 68mm, the XY-contour at 36mm, and the Z-contour at 35mm. These contour curves are shown in Fig. 4.3c-f.

In this section we explore the 16 hybrid combinations of the G-, C-, XY-, and Z-contour (N_p=45) and central G-, C-, XY-, and Z-profiles (N_c=45). Results on training set A (Fig. 4.9) show a high performance for the combinations of the G-, C-, and XY-contour with G-, C-, and XY-profiles. The marked areas show common factors of the results. For the training set the G and C-curves perform best followed by XY-curves and then Z-curves. Li's combination of the two vertical XY-profiles and one Z-contour performs reasonably well, but not as good as our manually selected contours and profiles. Of course the set of three optimal curves may differ per training set.

4.6.4 Optimal contours

From Sect. 4.6.2, we have learned that for a small number of curves the contours are more distinctive than profiles. Instead of combining one contour with two profiles, it makes sense to combine the optimally selected contours. Fig. 4.9 shows the combined performance of the optimally selected contour curves. Results show that the Z-contour has a negative influence on the overall performance of selected features. Nevertheless, the $basic$ results for the Z-contour are high, so its lower performance is probably caused by its lack of robustness to even small changes in the face's pose.

4.6.5 Eight contours

From the results in Sect. 4.6.2 we have learned that eight uniformly selected curves having 45 equally spaced samples has a reasonable performance. Thus, with only 360 samples per face we are already able to perform effective face matching. The single curve properties from Sect. 4.6.1 showed that each sampling type (G, C, XY, and Z) has its strengths and weaknesses. So, it makes sense to investigate the performance of *hybrid matching* using two profiles and one contour based on different sampling types as we did in Sect. 4.6.3. In the following experiment we have used hybrid combinations of eight contour curves in an attempt to improve the performance.

With the use of our framework we generated for each contour type, four equally spaced contours with N_p=45 samples (see Fig. 4.8). These G-, C-, XY-, and Z-contours were then combined into ten unique feature sets. Because the combination of four G-contours with the same four G-contours (and the three other exact matches) is useless,

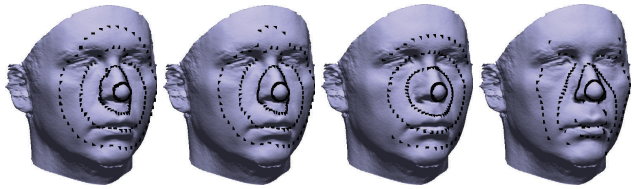


Figure 4.8: Features used in hybrid matching. From left to right, four G-, C-, XY-, and Z-contours.

we use eight equally spaced contours instead. Fig. 4.9 shows the results of the ten unique combinations of G-, C-, XY-, and Z-contours. The results from this experiment show a high performance for sets of eight G-contours, eight C-contours, and the combination of four G-contours and four C-contours. Reasonable results are obtained for combinations of XY-contours with either G- or C-contours.

4.7 Training on set B

In the previous sections we used training set A to investigate the performance of single curves under varying conditions. Training set A was also used show the effect of lowering the number of profiles N_p and contours N_c, such that face matching became more efficient with the same high performance. In this section we evaluate the selected feature sets from sections 4.6.3, 4.6.4, and 4.6.5 on training set B, which has a larger embedding and a larger variety of relevant faces per query. For each set of features we show its $basic$ and nd_r results in Fig. 4.9. To obtain the $basic$ results, we used the ground truth information of this dataset to undo the applied rotations and to select the nose tip locations. These results indicate the optimal performance that can be reached when the correct pose and nose tip are found. The nd_r results were obtained by applying nose detection to faces of the dataset directly. Observations that count for all $basic$ and nd_r results are: (1) The larger embedding and the greater dissimilarity of relevant faces decreases the overall mean average precision from around 0.9 for training set A to around 0.7 for training set B. (2) The performance gap between results after the applied nose detection (nd_r) and the predefined pose and nose tip ($basic$) shows that our 3D face retrieval can be further improved with optimized pose normalization and nose tip localization.

To confirm the latter observation we performed an additional experiment using the face template instead of the nose template and more view up vectors (every ten degrees) as described in Sect. 4.4.1. To evaluate the pose normalization and nose tip localization results, we used the ground truth data to determine the difference of the face's pose and the located nose tip. For local nose detection nd_r the mean and standard deviation of this evaluation are respectively 3.0 $\pm$ 1.9 degrees, and 1.4 $\pm$ 0.95 mm. For global face detection fd_r these results are respectively 2.1 $\pm$ 1.4 degrees, and 1.2 $\pm$ 0.83 mm. Global face detection results in a better face pose normalization and a more accurate nose tip localization, which has a positive effect on the retrieval performance.

Central profile and optimal contour. The combination of the optimally selected G-, C-, and XY-contour with the central XY-profile performs well on training set B, as it did on training set A. Remarkable is that the G- and C-profiles show a relatively large

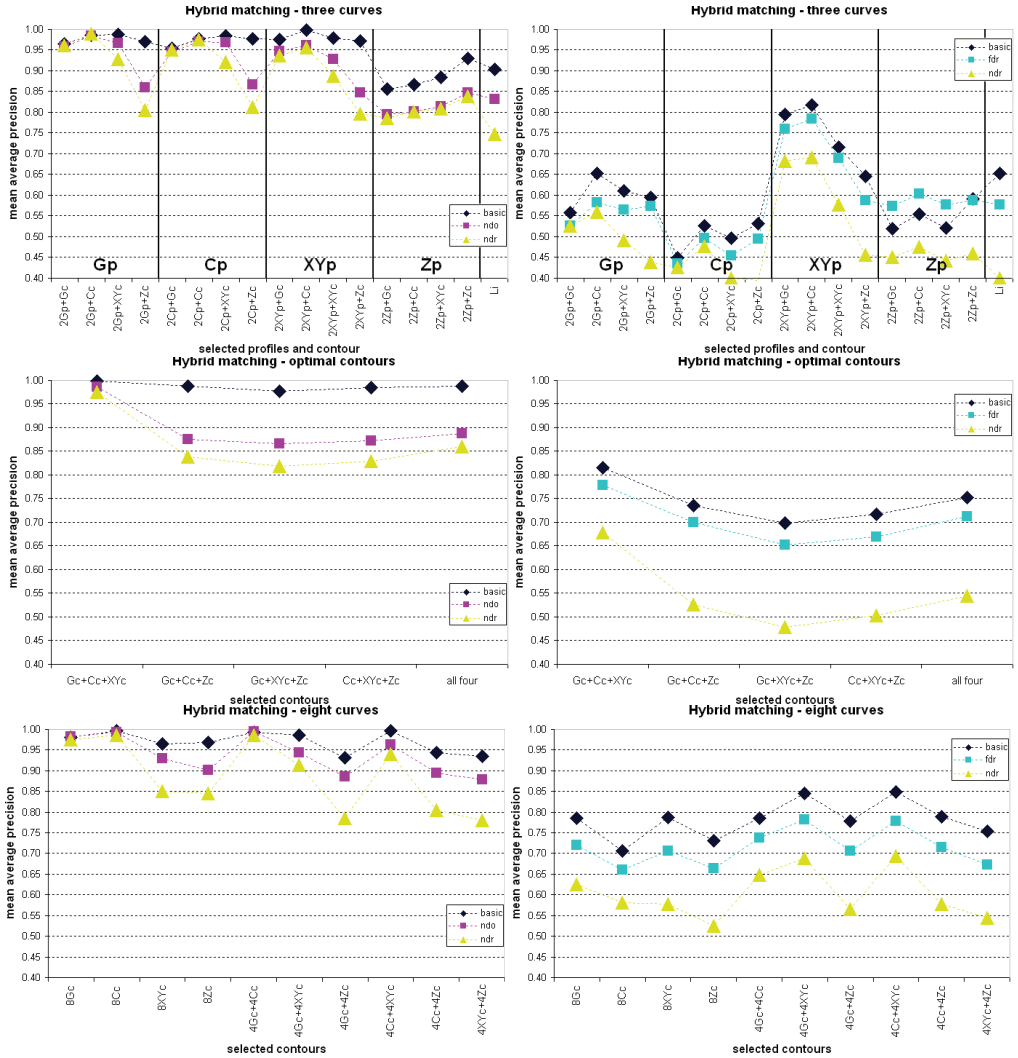


Figure 4.9: Retrieval results on training set A (left) and training set B (right). The framework was applied using combinations of manually selected curves (top, middle), and eight uniformly selected curves (bottom).

decrease in performance compared to the results from training set A. The C-profile performed very well on that training set because all models had a similar level of noise. Training set B on the other hand, contains relevantly classified faces which were morphed towards and away from the mean face introducing different levels of noise. This, and the fact that C-profiles are less robust to noise explains this drop in performance. The decrease of the G-profile's performance can be explained as follows. G-contours close to the tip of the nose, range r=0 mm to r=40 mm, are not effective to retrieve relevant faces (see Fig. 4.5). However, a G-profile contains samples within this range, which makes it less reliable. For the rather small training set A the central G-profiles were discriminative enough, but the larger embedding of training set B caused a lower performance of these curves. The XY-profile, with its constant performance and high robustness to noise, is therefore the best type of profile curve. The combination of the two central XY-profiles and optimal C-contour obtained a MAP of 0.69 for nd_r and even 0.78 for fd_r.

Optimal contours. The results for optimally selected contours on the training set B are similar to those on training set A. Again the highest results are obtained for the combined G-, C-, and XY-contour. The MAP in this case are 0.68 for nd_r and 0.78 for fd_r.

Eight contours. For feature sets of eight uniformly selected contour curves, results shows again a drop in performance for the combinations of C-contours and G-contours. For the eight G-contours, the ones closest to the nose tip have a negative influence on the performance. For the eight C-contours, the noise is again the cause. Nevertheless, the highest results (fd_r=0.78) are obtained for the hybrid combinations of four XY-contours with either four G-contours or four C-contours. Since C-contours and G-contours are very much alike, their combination does not improve the performance. The most important observation is that for each set of eight single type contours, there is a hybrid combination of eight contours with a higher performance. This means that hybrid combinations can improve on the effectiveness of face retrieval, without losing efficiency.

Based on the results of training set B, we selected several feature sets to test on our two test sets with real face scans. For their high efficiency, we selected the combinations of the central XY-profile and our optimal C-contour (2XYp+Cc), the central XY-profile and Li's optimal Z-contour (Li), and our optimal G-,C-, and XY-contour (Gc+Cc+XYc). For their high performance, we selected hybrid combinations of four XY-contours with either four G-contours (4Gc+4XYc) or four C-contours (4Cc+4XYc). For comparison, we selected the sets of eight uniformly selected G-contours (8Gc), G-contours (8Cc), G-contours (8XYc), and Z-contours (8Zc). These results on training set B are also listed in Table 4.1. The $basic$ results shown in this table indicate that the retrieval performance can be further increased, when an even more accurate pose normalization and nose tip localization method is applied.

4.8 Test results

In the previous section, we selected features with a high retrieval performance in a training set with a large embedding. In this section, these selected features are used for the retrieval of relevant faces in test set C and D. The scans in these test sets were all pose normalized and cleaned as described in Sect. 4.4. Test set C contains for each of the 61

features	samples	nd_r	fd_r	$basic$
2XYp + Zc(Li)	135	0.40	0.58	0.65
2XYp + Cc	135	0.69	0.78	0.82
Gc + Cc + XYc	135	0.68	0.78	0.82
8Gc	360	0.63	0.72	0.79
8Cc	360	0.58	0.66	0.71
8XYc	360	0.58	0.71	0.79
8Zc	360	0.52	0.66	0.73
4Gc + 4XYc	360	0.69	0.78	0.85
4Cc + 4XYc	360	0.69	0.78	0.85

Table 4.1: The MAP results of selected features on training set B.

features	test set C		test set D	
	d_{p1}	d_{p2}	d_{p1}	d_{p2}
2XYp + Zc(Li)	0.52	0.55	0.83	0.87
2XYp + Cc	0.59	0.58	0.93	0.93
Gc + Cc + XYc	0.60	0.57	0.94	0.94
8Gc	0.59	0.64	0.94	0.96
8Cc	0.57	0.65	0.89	0.95
8XYc	0.65	0.65	0.95	0.95
8Zc	0.56	0.57	0.89	0.91
4Gc + 4XYc	0.66	0.66	0.94	0.94
4Cc + 4XYc	0.66	0.66	0.94	0.94

Table 4.2: The MAP results on test sets C and D using two different distance measures.

subjects seven different scans. In this data set of 427 scans, each scan is used as a query once. Test set D contains 953 scans, which are all used to query set D. The MAP over all queries is used as a measure to evaluate the retrieval of relevant faces in a certain test set. Relevant faces are scans acquired from the same subject as the query scan.

So far, we have used only a single distance measure d_p to compute the distance between corresponding samples, namely, the squared relative Euclidean distance to the tip of the nose d_{p1},

$$d_s(\mathbf{A}_{ij}, \mathbf{B}_{ij}) = min_{\forall p \in \mathbf{A}_{ij}, \forall q \in \mathbf{B}_{ij}} d_p(p, q),$$

$$d_{p1}(p, q) = (e(p, p_{nt}) - e(q, p_{nt}))^2.$$

Now, we also use the more commonly used squared Euclidean distance d_{p2},

$$d_{p2}(p, q) = (e(p, q))^2.$$

Table 4.2 shows that especially the eight uniformly selected G- and C-contour benefit from distance measure d_{p2}. The performance slightly decreases for our optimally selected contours, which were selected based on the initial distance measure d_{p1}. This is clearly a disadvantage of manually selecting curves.

The retrieval results on test set C are not very high. Up till now, we used our framework to select subsets of contours that can be used for effective and efficient face matching under facial morphing. However, the triangle meshes in test set C are distorted by changes in facial expression. For expression invariant face retrieval, our framework needs to cope with these changes. Samir et al. [90] specifically selected a subset of Z-contours that are reasonably robust for a dataset of six different expressions per person. Mian et al. [68] applied a variant of the ICP algorithm to match only a masked face

features	test set C				test set D				samples	time
	25%	50%	75%	100%	25%	50%	75%	100%		
2XYp+Zc(Li)				0.55				0.86	270	1.3
2XYp+Cc				0.60				0.93	270	1.3
Gc+Cc+XYc	0.60	0.62	0.61	0.60	0.94	0.95	0.95	0.95	270	1.3
8Gc	0.70	0.70	0.70	0.67	0.95	0.96	0.96	0.96	520	3.1
8Cc	0.69	0.69	0.68	0.67	0.95	0.96	0.96	0.95	520	3.1
8XYc	0.67	0.68	0.67	0.65	0.94	0.95	0.95	0.95	520	3.1
8Zc	0.67	0.64	0.61	0.57	0.92	0.94	0.94	0.90	520	3.1
4Gc+4XYc	0.67	0.69	0.68	0.67	0.93	0.95	0.95	0.94	520	3.1
4Cc+4XYc	0.68	0.69	0.69	0.68	0.94	0.95	0.96	0.95	520	3.1
ICP	0.75	0.78	0.73	0.71	0.96	0.97	0.98	0.97	1800	16.8
Depth map	0.49	0.53	0.56	0.61	0.80	0.84	0.87	0.89	1800	7.8

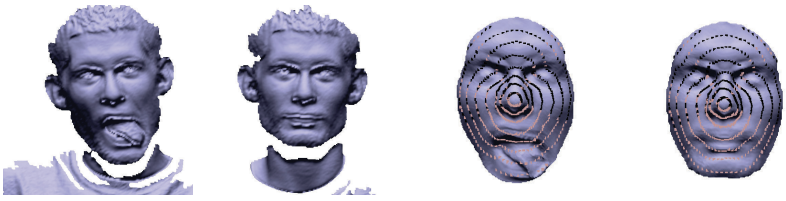
Table 4.3: The MAP results on test sets C and D using contour (N_p=90), ICP, and depth map matching. Timings are reported in seconds per 1000 matches.

region (nose, eyes and forehead) which is assumed to be static under facial expressions. Instead of restricting our framework to a specified subset of facial curves, we can select a percentage of profiles that matches best for two input faces. This way, a person might even be recognized using profiles that go through the mouth, in case that region remains unchanged from one expression to another. To do so, we introduce function f_w in our generic formula (Eq. 4.1), so that a weight can be assigned to specific profile curves:

$$d(\mathbf{A},\mathbf{B}) = \frac{1}{N}\sum_{i=1}^{N_p} f_w \sum_{j=1}^{N_c} d_s(\mathbf{A}_{ij},\mathbf{B}_{ij}) \ . \tag{4.2}$$

Here, we simply use f_w to assign a weight of 1 to a percentage of best matching profiles and a weight of 0 to the other profiles. Percentages of 25%, 50%, 75%, and 100% were used. With this function, profiles in facial areas that changed because of an expression can be neglected for the actual face comparison. We intentionally use a percentage of profiles and not a percentage of samples, because samples close to the nose have a smaller distance than those far from the nose but are often less distinctive for face retrieval. As a result, it is not possible to use this face matching scheme for single profile curves such as the central profile (2XYp+Cc and 2XYp+Zc(Li)). Matching two single contours of faces $\mathbf{A}$ and $\mathbf{B}$ using 50% of its best matching 'profiles', means that only half of the contour's corresponding samples are used to compute the similarity. The number of profiles N_p is increased to N_p=90, to compensate for the reduction of matched data. The retrieval results based on measure d_{p2} and various percentages are shown in Table 4.3.

Results show that the use of a smaller percentage of best matching profiles is an easy way to improve on expression invariant face retrieval. For test set C, the average performance of contour features improves from 0.64 at 100% to 0.67 at 50%. For test set D that has fewer expression scans, the effect is less significant with 0.94 at 100% and 0.95 at 50%. Fig. 4.10 shows an example of expression invariant face retrieval.



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

Figure 4.10: Expression invariance. A face scan with expression (a) that has a relevant scan on top of the ranked list (b) for the set of eight G-contours, because only 50% of its best profiles (in black) were matched (c,d).

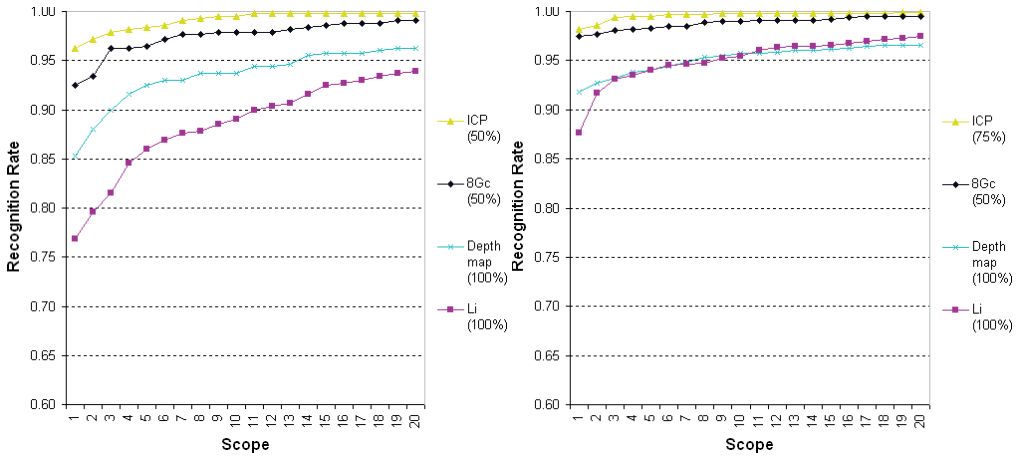


Figure 4.11: The CMC curves for test sets C and D using facial curves, ICP, and depth map matching.

4.9 Method comparison

The results achieved on test set C can be compared with the results from SHREC'08 [113]. Test set D (UND) is used under various conditions in previous work. For a direct comparison of our facial curves method, we implemented two comparable methods.

After the mesh improvement of Sect. 4.4.2, a cleaned depth map is acquired, which is also represented by a triangle mesh (see Fig. 4.2). The first method directly compares the distances of the depth maps (after aligning the nose tips) and computes the RMS distance for a percentage of the best matching samples. Only foreground pixel-pairs are considered. Note that this *depth map matching* is very similar to the comparison of a large set of XY-features. The second method is an *ICP matching* method. This method uses in each iteration a percentage of the closest point pairs to minimize the RMS distance between two triangular meshes, refining their alignment in the process. To have a symmetric distance measure, closest point pairs (p,p') are selected for face $\mathbf{A}$ to $\mathbf{B}$ and vice versa. Closest point pairs for which p' belongs to the surface boundary of a face scan are not used in the distance measure. To have a time-efficient measure, the RMS distance after just four iterations was used to match two faces, only vertex-vertex pairs were considered, and kD-trees were used for fast closest point operations. These are all common variations of the ICP algorithm [88]. Note that, the accurate pose normalization allows for the small number of iterations.

Despite all efforts to make the *depth map matching* and *ICP matching* time-efficient, they were too slow for practical use. Therefore, we subsampled the 1 mm $\times$ 1 mm depth maps of Fig. 4.2 to 3 mm $\times$ 3 mm depth maps, and corresponding surface meshes with approximately 1800 valid data points. For a fair comparison, our curve matching was based on these lower resolution surface meshes as well, which even turned out to result in higher MAPs. For all methods we use the squared distance between samples, which means we again use d_{p2} for our curve matching. We report the MAP results based on percentages 25%, 50%, 75%, and 100% for test sets C and D, in Tab. 4.3. Results in this table show that from all selected sets of profiles and contours, the set of eight uniformly selected G-contours (50%) performs best, closely followed by the other sets of eight contours except the eight Z-contours. This set of contours outperforms the depth map matching in both efficiency as effectiveness. ICP matching has a higher performance, but already requires 16.8 sec. to query a dataset of 1000 faces once.

The test sets C and D are usually used in the context of 3D face recognition. To compare the performance of selected feature set with the performance of other methods described in the literature, we determine recognition rates. The *recognition rate* (RR) is the ratio of relevant faces (other than the query face) retrieved on top of the ranked lists. When a face matching system retrieves for each query one of its relevant database models on top of the list, the RR is 100%. To increase the flexibility of a face recognition system, a relevant face within a small scope could be allowed for a successful recognition instead. The *cumulative match characteristic* (CMC) curves in Fig. 4.11 show the recognition rates for these larger scopes. The faster a CMC curve approaches one, the better that matching algorithm is.

The results we achieved on test set C, can be compared to the results reported in SHREC'08. With a MAP of 0.70 and 93% RR, our set of eight G-contours (50%) performs reasonably well. These results are higher than the facial curves of Li (2XYp+Zc) with a

MAP of 0.55 and 74% RR, higher than our depth map method (100%) with a MAP of 0.61 and 85% RR, but lower than our ICP based method (50%) with a MAP of 0.78 and 96% RR. These differences in performance are also shown in Fig. 4.11 using CMC curves. In these graphs, the first point of each curve shows the reported rank-first recognition rates. Compared to the results reported in [100], we notice that Morphable model based face matching by Amberg and the use of iso-geodesic stripes by Berretti have a higher performance with MAPs varying from 0.65 up to 0.94. Moment invariants by Xu, and region based matching by Nair have a lower performance with MAPs varying from 0.46 up to 0.66.

For test set D we achieved a MAP of 0.96 and 98% RR using eight G-contours (50%), Li's curves 0.86 and 88%, our depth map method (100%) 0.89 and 92%, and our ICP based method (75%) 0.98 and 98%, respectively. The recognition rates were computed over 876 queries, because 77 subjects have only one scan in this data set. The CMC curves of these methods are also shown in Fig. 4.11. For comparison, Blanz et al. [18] achieved a 96% RR for 150 queries in a subset of 150 faces using 1000 morphable model coefficients. Samir et al. [90] reported a 90.4% RR for 270 queries in a subset of 470 scans using a facial depth curves. Mian et al. [67] reported a 86.4% RR for 277 queries in a subset of 277 scans using tensor matching.

4.10 Concluding remarks

In this chapter we presented a new pose normalization method and a 3D face matching framework. Pose normalization is performed by fitting 3D templates to the scan data and using the inverse transformation of the best fit to normalize the pose. The fitted template is used to extract the tip of the nose. Starting from the tip of the nose we extracted a set of profile curves, which were sampled using G-, C-, XY-, and Z-samples. The number of profiles, the number of contours samples, and the distance measure are the parameters to instantiate our framework. According to the selected settings, our framework extracts corresponding samples from faces and matches them using the defined distance measure.

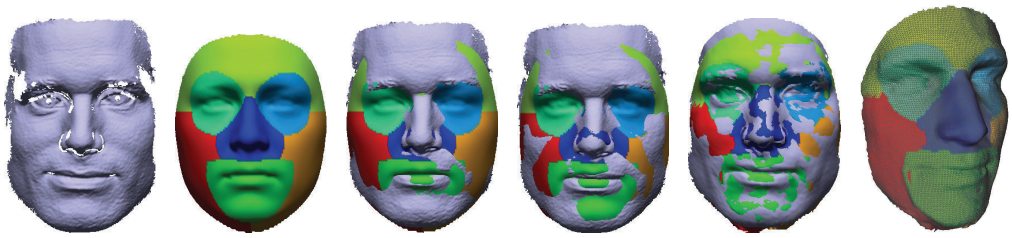
We have optimized our framework's settings for efficient and effective face matching. Furthermore, we examined the properties and the face retrieval performance of single facial curves, uniform selected curves, manually selected curves, and hybrid combinations, using different types of curve samples. This evaluation was done for training set A and to a certain extend for training set B as well. Based on these results we were able to select a few feature sets with a high retrieval performance. These feature sets were used in two different test sets with real face scans, namely the SHREC'08 and UND face sets. After pose normalization and face scan improvement, we applied our feature sets for the retrieval of relevant face scans. Besides that, we also applied a depth map and an ICP based method. To cope with the expressions in the face scans, we used a percentage of face data that matched best.

The combination of the central profile with our optimally selected G-contour performed better than the same central profile with Li's optimally selected Z-contour. With the same amount of feature points, we achieved a higher performance using an optimally selected G-, C-, and XY-contour. In the end, the eight uniformly selected G-

contours using 50% of the N_p=90 best matching profiles performed slightly better than other combinations of profiles and contours. With a MAP of 0.70 and 93% RR on test set C (SHREC'08), and a MAP of 0.96 and 98% RR on test set D (UND), our eight G-contours were more effective and more efficient than our depth map method. Our efficient implementation of an ICP based comparison method achieved a higher performance, but is still less time-efficient than our selected set of contours. This is caused by the iterative selection of corresponding samples during face matching, whereas curve matching allows for the off-line selection of corresponding samples and thus efficient face matching. In SHREC'08, MAPs were achieved ranging from 0.46 up to 0.94 and RRs of 63.5% up to 99.5%. Compared to these results our efficient facial curve based method performs well. For the UND dataset, which contains face scans of higher quality and less facial expressions, our facial curve based method outperforms other methods.

Chapter 5

Face modeling



The framework we presented in the previous chapter extracts and matches curves from manifold face surfaces without holes. To acquire such face surfaces from 3D scan data, we interpolated the holes in the depth images using 2D imaging techniques. However, we did not take the a priori knowledge of the face shape into account, which could improve the face surface in missing areas, and regions covered by facial hair or severe noise.

In this chapter, we present an automatic and efficient method to fit a statistical deformation model of the human face to 3D scan data. In a global to local fitting scheme, the shape parameters of this model are optimized such that the produced instance of the model accurately fits the 3D scan data of the input face. To increase the expressiveness of the model and to produce a tighter fit of the model, our method fits a set of predefined face components and blends these components afterwards. Quantitative evaluation shows an improvement of the fitting results when multiple components are used instead of one. Compared to existing methods, our fully automatic method achieves a higher accuracy of the fitting results. The accurately generated face instances are manifold meshes without noise and holes, and can be effectively used for 3D face recognition: We achieve 100% correct identification for 876 queries in the UND face set, 98% for 244 queries in the GAVAB face set, and 98% for 700 queries in the BU-3DFE face set. Our results show that model coefficient based face matching outperforms contour curve and landmark based face matching, and is more time efficient than contour curve matching.

5.1 Introduction

The use of 3D scan data for face recognition purposes has become a popular research area. With high recognition rates reported for several large sets of 3D face scans [18, 67, 90, 106], the 3D shape information of the face proved to be a useful contribution to person identification. The major advantage of 3D scan data over 2D color data, is that variations in scaling and illumination have less influence on the appearance of the acquired face data. However, scan data suffers from noise and missing data due to self-occlusion. To deal with these problems, 3D face recognition methods should be invariant to noise and missing data, or the noise has to be removed and the holes interpolated. Alternatively, data could be captured from multiple sides, but this requires complex data acquisition. In this chapter we present a method that produces an accurate fit of a statistical 3D shape model of the face to the scan data. We show that the 3D geometry of the generated face instances, which are without noise and holes, can be effectively used for 3D face recognition.

5.1.1 Related work

In the previous chapter we described several methods to perform 3D face recognition. Some of these techniques are based on 3D geodesic surface information, such as the methods of Bronstein et al. [22] and Berretti et al. [15]. The geodesic distance between two points on a surface is the length of the shortest path between two points. To compute accurate 3D geodesic distances for face recognition purposes, a 3D face without noise and without holes is desired. Since this is typically not the case with laser range scans, the noise has to be removed and the holes in the 3D surface interpolated. However, the success of basic noise removal techniques, such as Laplacian smoothing is very much dependent on the resolution and density of the scan data. Straightforward techniques to interpolate holes using curvature information or flat triangles often fail in case of complex holes, as pointed out in [39]. The use of a deformation model to approximate new scan data and interpolate missing data is a gentle way to regulate flaws in scan data.

A well known *statistical deformation model* specifically designed for surface meshes of 3D faces, is the 3D morphable face model of Blanz and Vetter [19]. This statistical model was built from 3D face scans with dense correspondences to which Principal Component Analysis (PCA) was applied. In their early work, Blanz and Vetter [19] fit this 3D morphable face model to 2D color images and cylindrical depth images from the $Cyberware^{TM}$ scanner. In each iteration of their fitting procedure, the model parameters are adjusted to obtain a new 3D face instance, which is projected to 2D cylindrical image space allowing the comparison of its color values (or depth values) to the input image. The parameters are optimized using a stochastic Newton algorithm. More recently, Blanz et al. [18] proposed a method to fit their 3D morphable face model to more common textured depth images. In the fitting process, a cost function is minimized using both color and depth values after the projection of the 3D model to 2D image space. To initialize their fitting method, they manually select seven corresponding face features on their model and in the depth scan. A morphable model of expressions was proposed by Lu et al. [65]. Starting from an existing neutral scan, they use their expression model

Figure 5.1: Face segmentation. The depth image (left) is converted to a surface mesh (middle). The surface mesh is cleaned, the tip of the nose is detected and the face segmented (right, in pink).

to adjust the vertices in a small region around the nose to obtain a better fit of the neutral scan to a scan with a certain expression. Amberg et al. [5] built a PCA model from 270 identity vectors and a PCA model from 135 expression vectors and combined the two into a single morphable face model. They fitted this model to 3D scans of both the UND and GAVAB face sets, and use the acquired model coefficients for expression invariant face matching with considerable success.

Non-statistical deformation models were proposed as well. Huang et al. [49] proposed a global to local deformation framework to deform a shape with an arbitrary dimension (2D, 3D or higher) to a new shape of the same class. They show their framework's applicability to 3D faces, for which they deform an incomplete source face to a target face. Kakadiaris et al. [56] deform an annotated face model to scan data. Their deformation is driven by triangles of the scan data attracting the vertices of the model. The deformation is restrained by a stiffness, mass and damping matrix, which control the resistance, velocity and acceleration of the model's vertices. The advantage of such deformable faces is that they are not limited to the statistical changes of the example shapes, so the deformation has less restrictions. However, this is also their disadvantage, because these models cannot rely on statistics in case of noise and missing data.

5.1.2 Contribution

First, we present a fully automatic algorithm to efficiently optimize the parameters of the morphable face model, creating a new face instance that accurately fits the 3D geometry of the scan data. Unlike other methods, ours needs no manual initialization, so that batch processing of large data sets has become feasible. Second, we quantitatively evaluate our fitted face models and show that the use of multiple components improves the fitting process. Thirdly, we show that our model fitting method is more accurate than existing methods. Fourthly, we show that the accurately generated face instances can be effectively used for 3D face recognition.

5.2 Datasets

In this chapter, we fit the morphable face model that we defined in the previous chapter as $S_{inst} = \bar{S} + \sum_{i=1}^{m} w_i \sigma_i s_i$ to 3D scan data. By doing this, we obtain a clean model of the face scan, that we use to identify 3D faces. The scans that we fit the morphable face model to, are the 3D face scans of the UND [26], a subset of the GAVAB [73, 100] and a subset of the BU-3DFE [121] databases. The UND set contains 953 frontal range

scans of 277 different subjects with mostly neutral expression. The GAVAB set consists of nine low quality scans for each of its 61 subject, including scans for different poses and expressions. From this set we selected, per subject, four neutral scans, namely the two frontal scans and the scans in which subjects look up and down. Acquired scan data from these poses differ in point cloud density, completeness and relatively small facial changes. The BU-3DFE set was developed for facial expression classification. This set contains one neutral scan and 24 expression scans having different intensity levels, for each of its 100 subjects. From this set we selected the neutral scans and the low level expression scans (anger, disgust, fear, happiness, sadness, surprise at level 1).

Although the currently used morphable model is based on faces with neutral expressions only, it makes sense to investigate the performance of our face model fitting in case of changes in pose and expressions. These variations in 3D scan data, which are typical for a non-cooperative scan environment, allows us to evaluate our 3D face recognition methods.

We aim at 3D face recognition, so we need to segment the face from each scan. For that, we employ our pose normalization method from Section 4.4 [105] that normalizes the pose of the face and localizes the tip of the nose. Before pose normalization, we applied a few basic preprocessing steps to the scan data: the 2D depth images were converted to triangle meshes by connecting the adjacent depth samples with triangles, slender triangles and singularities were removed, and only considerably large connected components were retained. Afterwards, the face is segmented by removing the scan data with a Euclidean distance larger than 110 mm from the nose tip. The face segmentation is visualized in Fig. 5.1.

5.3 Face model fitting

In general, 3D range scans suffer from noise, outliers, and missing data and their resolution may vary. The problem with single face scans, the GAVAB scans in particular, is that large areas of the face are missing, which are hard to fill using simple hole filling techniques. When the morphable face model is fitted to a 3D face scan, a model is obtained that has no holes, has a proper topology, and has an assured resolution. By adjusting the m=99 weights w_i for the eigenvectors, the morphable model creates a new face instance. To fit the morphable model to 3D scan data, we need to find the optimal set of m weights w_i. In this section, we describe a fully automatic method that efficiently finds a proper model of the face scan in the m-dimensional space.

5.3.1 Distance measure

To evaluate if an instance of the morphable face model is a good approximation of the 3D face scan, we use the Root Mean Square (RMS) distance of the instance's vertices to their closest points in the face scan. For each vertex point (p) from the instance (M_1), we find the vertex point (p') in the scan data (M_2) with the minimal Euclidean distance

$$e_{min}(p, M_2) = min_{p' \in M_2} d(p, p') \tag{5.1}$$

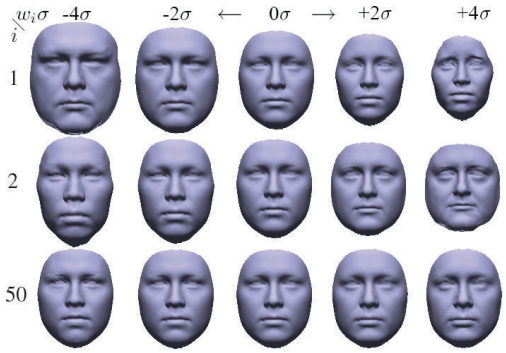


Figure 5.2: Face morphing along eigenvectors starting from the mean face (center column). Different weights for the principal eigenvectors (e.g. i=1,2) changes the global face shape. For latter eigenvectors the shape changes locally (e.g. i=50).

using a kD-tree. The RMS distance is then measured between M_1 and M_2 as:

$$d_{rms}(M_1, M_2) = \sqrt{\frac{1}{n}\sum_{i=1}^{n} e_{min}(p_i, M_2)^2} \tag{5.2}$$

using n vertices from M_1. Closest point pairs (p,p') for which p' belongs to the boundary of the face scan, are not used in the distance measure.

As described in previous chapter (Sect. 4.2), the morphable face model has n=75,972 vertices. These vertices cover the face, neck and ear regions and its resolution in the upward direction is three times higher than in its sideways direction. Because the running time of our measure is dependent on the number of vertices, we use for the model fitting only those vertices that cover the face (data within 110 mm from the tip of the nose) and not the neck and ears. To obtain a more uniform resolution for the model, we reduced the upward resolution to one third of the original model. The number of vertices used in the fitting process is now reduced to n=12,964 vertices, a sample every $\approx$2.6 mm^2 of the face area. Note that we do not change or recompute the PCA model, we simply select a set of vertex indices.

5.3.2 Iterative face fitting

With the defined distance measure for an instance of our compressed morphable face model, the m-dimensional space can be searched for the optimal instance. The fitting is done by choosing a set of m weights w_i, adjusting the position of the instance's vertices according to $S_{inst} = \bar{S} + \sum_{i=1}^{m} w_i\sigma_i s_i$, measuring the RMS-distance of the new instance to the scan data, selecting new weights and continue until the optimal instance is found. Knowing that each instance is evaluated using a large number of vertices, an exhaustive search for the optimal set of m weights is too computationally expensive.

A common method to solve large combinatorial optimization problems is *simulated annealing* (SA) [60]. In our case, random m-dimensional vectors could be generated which represent different *morphs* for a current face instance. A morph that brings the

current instance closer to the scan data is accepted (downhill), and otherwise it is either accepted (uphill to avoid local minima) or rejected with a certain probability. In each iteration, the length of the m-dimensional morph vector can be reduced as implementation of the "temperature" scheme. The problem with such a naive SA approach is that most random m-dimensional morph vectors are uphill. In particular close to the optimal solution, a morph vector is often rejected, which makes it hard to produce an accurate fit. Besides this inefficiency, it doesn't take the eigensystem of the morphable face model into account.

Instead, we use an *iterative downhill walk* along the consecutive eigenvectors from a current instance towards the optimal solution. Starting from the mean face $\bar{S}$ ($\forall_{i=1}^{m} w_i = 0$), try new values for w_1 and keep the best fit, then try new values for w_2 and keep the best fit, and continue until the face is morphed downhill along all m eigenvectors. Then iterate this process with a *smaller search space* for w_i. The advantage in computation costs of this method is twofold. First, the discrete number of morphs in the selected search space directly defines the number of rejected morphs per iteration. Second, optimizing one w_i at a time means only a one (instead of m) dimensional modification of the current face instance $S_{new} = S_{prev} + (w_{new} - w_{prev})\sigma_i s_i$.

Because the first eigenvectors induce the fitting of global face properties (e.g. face height and width) and the last eigenvectors change local face properties (e.g. nose length and width), each iteration follows a global to local fitting scheme (see Fig. 5.2). To avoid local minima, two strategies are applied. (1) The selected w_i in one iteration is not evaluated in the next iteration, forcing a new (similar) path through the m-dimensional space. (2) The vertices of the morphable face model are uniformly divided over three sets and in each iteration a different set is modified and evaluated. Only in the first and last iteration all vertices are evaluated. Notice that this also reduces the number of vertices to fit and thus the computation costs.

The fitting process starts with the mean face and morphs in place towards the scan data, which means that the scan data should be well aligned to the mean face. To do so, the segmented and pose normalized face is placed with its center of mass on the center of mass of the mean face, and finely aligned using the Iterative Closest Point (ICP) algorithm [16]. The ICP algorithm iteratively minimizes the RMS distance between vertices. To further improve the effectiveness of the fitting process, our approach is applied in a *coarse fitting* and a *fine fitting* step.

5.3.3 Coarse fitting

The more the face scan differs from the mean face $\bar{S}$, the less reliable the initial alignment of the scan data to the mean face is. Therefore, the mean face is coarsely fitted to the scan data by adjusting the weights of the first ten principal eigenvectors (m_{max}=10) in a single iteration (k_{max}=1) with 10 different values for w_{new}=[-1.35,-1.05,...,1.05,1.35] as in Algorithm ModelFitting($\bar{S}$,$scan$). Fitting the model by optimizing the first ten eigenvectors results in the face instance S_{coarse}, with global face properties similar to those of the scan data. After that, the alignment of the scan to S_{coarse} is further improved with the ICP algorithm.

5.3.4 Fine fitting

Starting with the improved alignment, we again fit the model to the scan data. This time the model fitting algorithm is applied using all eigenvectors (m_{max}=m) and multiple iterations (k_{max}=9). In the first iteration of Algorithm ModelFitting($\bar{S}$,$scan$), 10 new weight values w_{new} are tried for each eigenvector, to cover a large range of facial variety. The best w_{new} for every sequential eigenvector is used to morph the instance closer to the face scan. In the following k_{max}-1 iterations only four new weight values w_{new} are tried around w_i with a range w_{range} equal to w_{incr} of the previous iteration. By iteratively searching for a better w_i in a smaller range, the weights are continuously optimized. Local minima are avoided as described in Sect. 5.3.2. The range of the first iteration and the number of new weights tried in each next iteration were empirically selected as good settings.

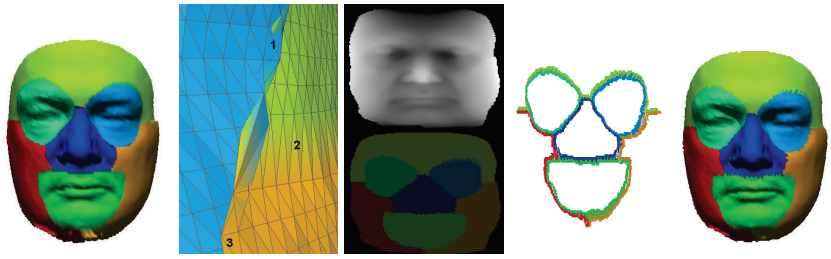
Algorithm 4 ModelFitting(S_{inst}, $scan$)

$w_{range} = 1.5$, $w_{incr} = 0.3$
for $k \leftarrow 1$ **to** k_{max} **do**
  select vertices (uniform subset of component)
  for $i \leftarrow 1$ **to** m_{max} **do**
    $w_{min} = w_i - w_{range} + \frac{1}{2}w_{incr}$
    $w_{max} = w_i + w_{range} - \frac{1}{2}w_{incr}$
    for $w_{new} \leftarrow w_{min}$ **to** w_{max} **do**
      morph S_{inst} with w_{new}
      $d_{rms}(S_{inst}, scan)$ smaller $\rightarrow$ keep w_{new}
      undo morph
      $w_{new} = w_{new} + w_{incr}$
    morph S_{inst} with $w_i \leftarrow$ best w_{new}
  $w_{range} = w_{incr}$, $w_{incr} = \frac{1}{2}w_{incr}$
return S_{inst}

5.3.5 Multiple components

Knowing that the morphable model was generated from 100 3D face scans, an increase of its expressiveness is most likely necessary to cover a large population. To increase the expressiveness, also Blanz and Vetter [19] proposed to independently fit different components of the face, namely the eyes, nose, mouth, and the surrounding region. Because each component is defined by its own linear combination of shape parameters, a larger variety of faces can be generated with the same model. The fine fitting scheme from the previous section was developed to be applicable to either the morphable face model as a whole, but also to individual components of this model.

Component selection. All face instances generated with the morphable model are assumed to be in correspondence, so a component is simply a subset of vertices in the mean shape $\bar{S}$ (or any other instance). We define seven components in our adjusted morphable face model (see Fig. 5.3). Starting with the improved alignment, we can individually fit each of the components to the scan data using the fine fitting scheme, obtaining a higher precision of the fitting process (as shown in Sect. 5.5.1). Individual components for the left and right eyes and cheeks were selected, so that our method



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

Figure 5.3: Multiple components (a) may intersect (b1), move apart (b2), or move across (b3). Simulating a cylindrical scan (c) and smoothing the new border vertices (d) solves these problems (e).

applies to non-symmetric faces as well. For comparison, we also used a face model with four components, one for the eyes, one for the nose, one for the mouth, and one for the forehead and cheeks. The use of multiple components has no influence on the fitting time, because the total number of vertices remains the same and only the selected vertices are modified and evaluated.

Component blending. A drawback of fitting each component separately is that inconsistencies may appear at the borders of the components. During the fine fitting, the border triangles of two components may start to intersect, move apart, or move across (Fig. 5.3). The connectivity of the complete mesh remains the same, so two components moving apart remain connected with elongated triangles at their borders. We solve these inconsistencies by means of a post-processing step, as described in more detail below.

Knowing that the morphable face model is created from cylindrical range scans and that the position of the face instance doesn't change, it is easy to synthetically rescan the generated face instance. Each triangle of the generated face instance S_{fine} is assigned to a component (Fig. 5.3a). A cylindrical scanner is simulated, obtaining a cylindrical depth image $d(\theta, y)$ with a surface sample for angle θ, height y with radius distance d from the y-axis through the center of mass of $\bar{S}$ (Fig. 5.3c). Basically, each sample is the intersection point of a horizontal ray with its closest triangle, so we still know to which component it belongs. The cylindrical depth image is converted to a 3D triangle mesh by connecting the adjacent samples and projecting the cylindrical coordinates to 3D. This new mesh S'_{fine} has a guaranteed resolution depending on the step sizes of θ and y, and the sampling solves the problem of intersecting and stretching triangles. However, ridges may still appear at borders where components moved across. Therefore, Laplacian smoothing is applied to the border vertices and their neighbors (Fig. 5.3d). Laplacian smoothing moves each vertex towards the center of mass of its connected vertices. Finally, data further then 110 mm from the tip of the nose is removed to have the final model S_{final} (Fig. 5.3e) correspond to the segmented face. In Sect. 5.5.1, we evaluate both the single and multiple component fits.

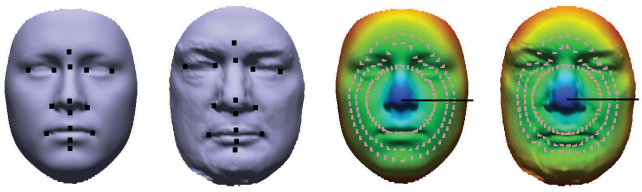


Figure 5.4: The similarity of two 3D faces is determined using one-to-one correspondences, with on the left 15 corresponding landmarks and on the right 135 corresponding contour samples. The optimal XY-, C-, and G-contour curves (inner to outer) were extracted, for which the G-contour uses the (colored) geodesic distances. The line shown in black is one of the N_p profiles.

5.4 Face matching

Our model fitting algorithm determines a set of model coefficients that morphs the mean face to a clean model instance that resembles the 3D face scan. Based on this model instance, we use three different methods to perform face matching. Two methods use the newly created 3D geometry as input, namely the landmarks based and contour based methods. The third method uses the model coefficients as a feature vector to describe the generated face instance.

Landmarks. All vertices of two different instances of the morphable model are assumed to have a one-to-one correspondence. Assuming that facial landmarks such as the tip of the nose, corners of the eyes, etc. are morphed towards the correct position in the scan data, we can use them to match two 3D faces. So, we assigned 15 anthropometric landmarks to the mean face and obtain their new locations by fitting the model to the scan data. To match two faces $\mathbf{A}$ and $\mathbf{B}$ we use the sets of c=15 corresponding landmark locations:

$$d_{corr}(\mathbf{A}, \mathbf{B}) = \sum_{i=1}^{c} d_p(a_i, b_i) \tag{5.3}$$

where distance d_p between two correspondences a_i and b_i is the squared difference in Euclidean distance e to the nose tip landmark p_{nt}:

$$d_p(a_i, b_i) = (e(a_i, p_{nt}) - e(b_i, p_{nt}))^2 \tag{5.4}$$

Contour curves. Another approach is to fit the model to scans $\mathbf{A}$ and $\mathbf{B}$ and use the new clean geometry as input for a more complex 3D face recognition method. To perform 3D face recognition, we extract from each fitted face instance three 3D facial contour curves, and match only these curves to find similar faces.

As we did in Section 4.6.3, we select the XY-contour at r=36 mm, the C-contour at r=68 mm, and the G-contour at r=77 mm. The information of each 3D face instance is now reduced to a set of 135 ($3 \times N_p$) 3D sample points, with one-to-one correspondence to the same set of 135 samples in a different face instance. The similarity of faces $\mathbf{A}$ and $\mathbf{B}$ is defined by the sum of Euclidean distances for the corresponding samples.

Model coefficients. The iterative model fitting process determines an optimal weight w_i for each of the m eigenvectors. These weights, or model coefficients, multiplied by σ

describe a path along the linearly independent eigenvectors through the m dimensional face space. For two similar scans one can assume these paths are alike, which means that the set of m model coefficients can be used as a feature vector for face matching.

In case of multiple components, each component has its own set of m model coefficients. In [18], sets of model coefficients where simply concatenated to a single coefficient vector. Here, we also concatenate the coefficient vectors of multiple components. To determine the similarity of faces with these coefficient vectors, we use four distance measures, namely the L_1 and L_2 distance before and after normalizing the vector's length. In [5], the authors assume that caricatures of an identity lie on a vector from the origin to any identity and use the angle between two coefficient vectors as a distance measure. Normalizing the length of the coefficient vectors and computing the L_2 distance has this same effect and results in the same ranking of retrieved faces.

5.5 Results

In this section we evaluate our fitting results for the UND, GAVAB, and BU-3DFE datasets. We perform a qualitative and quantitative evaluation of the acquired model fits and compare the results with other model fitting methods. To prove that the use of multiple components improves the fitting accuracy over a single component, we compare the quantitative measures and relate the fitting accuracy to face recognition by applying different face matching algorithms to the produced fits. By applying and comparing different face matching methods, we end up with a complete 3D face recognition system with high recognition rates for all three datasets.

Starting with the 3D face scans from a dataset, we apply our face segmentation method (Sect. 5.2). Our face segmentation method correctly normalized the pose of all face scans and adequately extracted the tip of the nose in each of them. For the 953 scans of the UND face set, we evaluated the tip of the nose extraction by computing the average distance and standard deviation of the 953 automatically selected nose tips to our manually selected nose tips, which was 2.3 $\pm$ 1.2 mm. Since our model fitting method aligns the face scan to the mean face and at a later stage to the coarsely fitted face instance, these results are good enough.

5.5.1 Face model fitting

In this section we evaluate the face model fitting as follows. Each segmented face was aligned to $\bar{S}$ and the coarse fitting method of Sect. 5.3.3 was applied. After the improved alignment of the scan data to S_{coarse}, the fine fitting method of Sect. 5.3.4 was applied to either the entire face or to each of the individual components. For a fair comparison the same post-processing steps (Sect. 5.3.5) were applied to the final S_{fine} instances. Fig. 5.5 shows qualitative better fits when multiple components are used instead of a single component. Globally, by looking at the more frequent surface interpenetration of the fitted model and face scan, which means a tighter fit. Locally, by looking at facial features, such as the nose, lips and eyes. Note that our fitting method correctly neglects facial hair and interpolates holes, which is often a problem for 3D face recognition methods.

dataset	model	min	max	mean	sd
UND	1 comp	0.51	1.73	0.79	0.13
	4 comp	0.39	1.57	0.66	0.11
	7 comp	0.39	1.50	0.64	0.10
GAVAB	1 comp	0.94	3.45	1.22	0.19
	4 comp	0.80	2.30	1.06	0.14
	7 comp	0.80	2.24	1.05	0.14
BU-3DFE	1 comp	0.92	1.97	1.20	0.18
	4 comp	0.87	1.77	1.09	0.15
	7 comp	0.87	1.78	1.08	0.16

Table 5.1: RMS errors (mm) of output models to input scans.

dataset	model	min	max	mean	sd
UND	1 comp	0.40	1.65	0.65	0.15
	4 comp	0.28	1.33	0.47	0.11
	7 comp	0.26	1.20	0.43	0.10
GAVAB	1 comp	0.49	3.03	0.79	0.20
	4 comp	0.35	1.91	0.55	0.14
	7 comp	0.35	1.69	0.53	0.12
BU-3DFE	1 comp	0.37	1.58	0.63	0.17
	4 comp	0.25	0.99	0.43	0.10
	7 comp	0.23	0.92	0.39	0.11

Table 5.2: Projection errors (mm) of output models to input scans.

To quantitatively evaluate the produced fits, we determined the RMS distance (Eq. 5.2) for each of the fitted models to their face scan $d_{rms}(S_{final}, scan)$. To report merely the measurements in overlapping face regions, the points paired with boundary points are not included. Also outliers, point-pairs with a distance larger than 10 mm, are not taken into account. The RMS errors are shown in Table 5.1. Note that the UND scans have a higher resolution and thus smaller point-to-point distances, the RMS distances are therefore lower for the UND set than for the GAVAB and BU-3DFE sets.

Blanz et al. [18] reported the accuracy of their model fitting method using the average depth error between the depth images of the input scan and the output model, neglecting point-pairs with a distance larger than 10 mm. To compare the accuracy of our method with their method, we produced cylindrical depth images (as in Fig. 5.3c) for both the segmented face scan and the fitted model and computed the average depth error $|d_{scan}(\theta, y) - d_{final}(\theta, y)|$, excluding the outliers. Because of the surface mesh resampling, these projection errors (Table 5.2) are resolution independent. The GAVAB set has more acquisition artifacts causing higher projection errors, with high maximum projection errors in particular. The available BU-3DFE scans were heavily smoothed right after the acquisition process, causing lower projection errors than the high resolution UND scans.

The errors reported in Table 5.1 and Table 5.2, are all in favor of the multiple component fits with an average RMS gain of 0.2 mm per point pair. However, only a marginal gain in accuracy is acquired when seven components are used instead of four. So, with the use of multiple components we can increase the model's expressiveness to some extend.

Comparison. Blanz et al. [18] reported a mean depth error over 300 UND scans of 1.02 mm when they neglected outliers. For our fitted single component to UND scans

the error $d_{avr.depth}$ is 0.65 mm, which is already more accurate. For the fitted multiple components these errors are 0.47 and 0.43, for four and seven components respectively.

Our time to process a raw scan requires ca. 3 seconds for the face segmentation, ca. 1 second for the coarse fitting, and ca. 30 seconds for the fine fitting on a Pentium IV 2.8 GHz. Blanz method reported ca. 4 minutes on a 3.4 GHz Xeon processor, but includes texture fitting as well. Huang et al. [49] report for their deformation model a matching error of 1.2 mm after a processing time of 4.6 minutes. Recently, Amberg et al. [5] proposed a competitive fitting time of 40 to 90 seconds for their face model with 310 model coefficients and 11.000 vertices.

5.5.2 Face matching

As described in Sect. 5.4, we can use the morphed face instances to perform 3D face recognition. For this experiment, we computed the 953×953, 244×244, and 700×700 dissimilarity matrices and sorted the ranked lists of face models on decreasing similarity. From these ranked lists, we computed the recognition rate (RR), the mean average precision (MAP) and the verification rate at 0.1% false acceptance rate (VR@0.1%FAR). A person is recognized (or identified) when the face retrieved on top of the ranked list (excluding the query) belongs to the same subject as the query. For 77 subjects in the UND set only a single face instance is available which cannot be identified, so for this set the RR is based on the remaining 876 queries. The mean average precision (MAP) of the ranked lists are also reported, to elaborate on the retrieval of all relevant faces, i.e. all faces from the same subject. Instead of focusing on 3D face retrieval application, one could use 3D face matching for imposter detection as well. For an imposter/client detection system, all face matches with a dissimilarity above a carefully selected threshold are rejected. Lowering this threshold means that more imposters are successfully rejected, but also that less clients are accepted. We use the dissimilarity threshold at which the false acceptance rate is 0.1%, which is also used in the face recognition vendor test. Because the VR@0.1%FAR depends on similarity values, it is not only important to have relevant faces on top of the ranked lists, but also that their similarity values are alike and differ from irrelevant faces.

Since, the VR@0.1%FAR evaluation measure depends on the acquired similarity values there are several ways to influence this measure. Rank aggregation with the use of Consensus Voting or Borda Count [42], for instance, reassigns similarity values based on the ranking. This way one can abstract from the actual similarity values, which allows for the selection of a different imposter threshold and change the VR@0.1%FAR. Of course, a rank based threshold can not be used in case of a one-to-one face matching, that is, a scenario in which someone's identity must be confirmed or rejected. The application domain for rank-based measures is the one-to-many face matching, that is, a scenario in which we search for the most similar face in a large database.

In case of the face matching based on model coefficients, we assume that caricatures of an identity lie on a vector from the origin to any identity. If we normalize the lengths of these vectors, we neglect the caricatures and focus on the identity. This normalization step also regulates the similarity values and thus influences the VR@0.1%FAR. In Table 5.3, we report the face matching results based on the L_1 and L_2 distances between coefficient vectors, before and after length normalization. Remarkable is the significant in-

dataset	measure	RR	MAP	VR@0.1%FAR
UND	L1	1.00	0.99	0.94
	L2	1.00	0.99	0.94
	L1 norm*	1.00	1.00	0.99
	L2 norm*	1.00	1.00	0.98
GAVAB	L1	0.97	0.91	0.69
	L2	0.95	0.90	0.69
	L1 norm*	0.97	0.94	0.85
	L2 norm*	0.97	0.93	0.84
BU-3DFE	L1	0.99	0.90	0.59
	L2	0.99	0.88	0.56
	L1 norm*	0.98	0.94	0.80
	L2 norm*	0.97	0.93	0.78

Table 5.3: The effect of normalizing coefficient vectors to unit length. *A significant increase of the VR@0.1%FAR is achieved by matching the normalized coefficient vectors.

crease of the VR@0.1%FAR for the normalized coefficient vectors, whereas the rankings are similar as shown by the MAPs. Although we show in Table 5.3 only the results for the face model fitting using seven components, it also holds for the one and four component case. Because the L_1 distance between normalized coefficient vectors slightly outperforms the L_2 distance measure, we use this measure whenever we evaluate the performance of model coefficients.

Face retrieval and verification results based on anthropometric landmarks, contour curves, and model coefficients are shown in Table 5.4. To each set of face scans we fitted the morphable face model using one, four, and seven components. Each fitted component produces a 99 dimensional model coefficient vector with a different face instance as a result. The performance of our face matching depends on both the number of components as well as the applied feature set. The two main observations are that (1) the coefficient based method outperforms the landmark based and contour based methods, and (2) that the use of multiple components can increase the performance of 3D face matching. In the next paragraphs we elaborate on these observations.

The automatically selected anthropometric landmarks have a reasonable performance on the UND face scans, but are not reliable enough for effective 3D face matching in the two other sets. The contours perform well for retrieval and verification purposes in the UND face set. However, their performance drops significantly for the other two sets, because the contour curves cannot be effectively used in case of facial deformations. The use of the model coefficients consistently outperforms both the landmarks based and contour based face matching. Besides the difference in performance, the three methods differ in running time as well. The landmark based method matches two faces using only 15 coordinates, whereas the contour based method matches two faces using 135 coordinates. The coefficient based method matches faces using 99 weights times the number of fitted components. So, the coefficient based method using four components has the approximately the same running time as the contour based method.

The observation that multiple (four or seven) components increases the performance of our face matching holds for all results except the landmark based and contour based methods in the GAVAB set. The problem with this set is that a low quality scan of a person looking up or down causes artifacts on and around the nose. In such cases a more accurate fit of the face model's nose harms, because the performance of landmark

dataset	features	model	RR	MAP	VR@0.1%FAR
UND	landmarks	1 comp	0.85	0.86	0.71
	landmarks	4 comp	0.90	0.89	0.74
	landmarks	7 comp	0.89	0.90	0.77
	contours	1 comp	0.95	0.94	0.85
	contours	4 comp	0.97	0.92	0.98
	contours	7 comp	0.98	0.97	0.92
	coefficients	1 comp	0.98	0.98	0.95
	coefficients	4 comp	1.00	1.00	0.98
	coefficients	7 comp	1.00	1.00	0.99
GAVAB	landmarks	1 comp	0.72	0.73	0.46
	landmarks	4 comp	0.60	0.66	0.38
	landmarks	7 comp	0.64	0.67	0.43
	contours	1 comp	0.91	0.86	0.67
	contours	4 comp	0.82	0.79	0.58
	contours	7 comp	0.83	0.80	0.59
	coefficients	1 comp	0.96	0.93	0.82
	coefficients	4 comp	0.98	0.95	0.88
	coefficients	7 comp	0.97	0.94	0.85
BU-3DFE	landmarks	1 comp	0.61	0.49	0.27
	landmarks	4 comp	0.63	0.50	0.28
	landmarks	7 comp	0.66	0.54	0.31
	contours	1 comp	0.84	0.66	0.43
	contours	4 comp	0.88	0.67	0.43
	contours	7 comp	0.88	0.67	0.44
	coefficients	1 comp	0.96	0.88	0.73
	coefficients	4 comp	0.98	0.92	0.78
	coefficients	7 comp	0.98	0.94	0.80

Table 5.4: Performances measures based on landmarks, contour curves, and coefficient vectors for single and multiple component fits.

and contour based methods are heavily dependent on an accurate selection of the nose tip. Although the face matching improves from the single to multiple component case, there is no consensus for the four or seven component case. The use of either four or seven components causes either a marginal increase or decrease of the evaluation scores. Although, face matching with the use of 1000 model coefficients is usually referred to as time efficient, one could argue to use four components instead of seven, because the number of coefficients is smaller.

Comparison. Blanz et al. [18] achieved a 96% RR for 150 queries in a set of 150 faces (from the FRGC v.1). To determine the similarity of two face instances, they computed the scalar product of the 1000 obtained model coefficients. In the previous chapter, we achieved 95% RR on the UND set using the three selected contour curves, and 98% RR with an ICP-based method.

5.6 Concluding remarks

Where other methods need manual initialization, we presented a fully automatic 3D face morphing method that produces a fast and accurate fit for the morphable face model to 3D scan data. Based on a global to local fitting scheme the face model is coarsely fitted to the automatically segmented 3D face scan. After the coarse fitting, the

face model is either finely fitted as a single component or as a set of individual components. Inconsistencies at the borders are resolved using an easy to implement post-processing method. Our results show that the use of multiple components produces a tighter fit of the face model to the face scan, but assigned anthropometric landmarks may lose their reliability for 3D face identification.

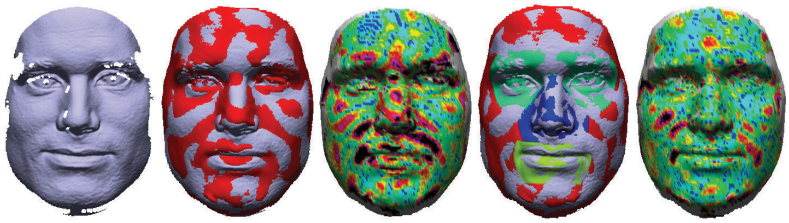
Face matching using facial contours shows higher recognition rates based on the multiple component fits than for the single component fits. This means that the obtained 3D geometry after fitting multiple components has a higher accuracy. Our face modeling approach in combination with the three selected contour curves achieves 98% RR on the UND set. In the previous chapter, we used imaging techniques to fill the holes in the face scan and the same contour curves to achieve 95% RR on the UND set.

The obtained model coefficient that were used to produce the accurate face instances, turned out to have the highest performance. With the use of four components, we achieve 100% correct identification for 876 queries in the UND face set, 98% for 244 queries in the GAVAB face set, and 98% for 700 queries in the BU-3DFE face set. These high recognition rates based on normalized coefficient vectors, proves that our model fitting method successfully fits the morphable face model consistently to scan data with varying poses and expressions.

Figure 5.5: Fitted face models S_{final} based on a single component (1st and 3rd column) and multiple components (2nd and 4rd column) to scan data in blue. Results from the front and side view, show a qualitative better fit of the multiple components to the scan data. The last two subjects on the right were also used in [18].

Chapter 6

Bootstrapping a face model



For a 3D face recognition system based on model coefficients, as we presented in the previous chapter, it is of the utmost importance that the statistics of many realistic faces are captured in the morphable model. In case a face cannot be modeled, the automatically acquired model coefficients are unreliable, which hinders the automatic recognition. In this chapter, we present a new bootstrapping algorithm to automatically enhance a 3D morphable face model with new face data. Our algorithm is based on a morphable model fitting method that uses a set of predefined face components. This fitting method produces accurate model fits to 3D face data with noise and holes. In the fitting process, the dense point-to-point correspondences between the scan data and the face model can become less reliable at the border of components. In this chapter, we solve this by introducing a blending technique that improves on the distorted correspondences close to the borders. Afterwards, a new face instance is acquired similar to the 3D scan data and in full correspondence with the face model. These newly generated face instances can then be added to the morphable face model to build a more descriptive one. To avoid our bootstrapping algorithm from needlessly adding redundant face data, we incorporate a redundancy estimation algorithm. We tested our bootstrapping algorithm on a set of scans acquired with different scanning devices, and on the UND data set. Quantitative and qualitative evaluation shows that our algorithm successfully enhances an initial morphable face model with new face data, in a fully automatic manner.

6.1 Introduction

Statistical models of the human face have proven to be an effective tool for person identification of 3D face scans. To build a statistical model, a set of example faces is required

with face features in full correspondence. With such a model, a new face instance can be constructed as a linear combination of the example faces. For 3D face identification, the idea is to use the statistical model to construct a face instance that resembles an input image. The way these example faces are combined linearly to represent an input face, provides both global and local information about the input face, that can be used to classify and identify different input faces. For a statistical face model to be applicable in face recognition systems all over the world, it is important to include example data on all possible face variations. Since this is an intractable task, a flexible model is required that updates itself in case of new example faces. For that, a system should automatically fit the face model to face data, estimate dense and accurate correspondences beyond the linear combinations of current example data, and measure the redundancy of the new example faces. When full correspondence between the face model and the scan data is established and the new face instance is not redundant, it can be added as a new example to the statistical face model, increasing the descriptiveness of the model. The process of using a statistical model to enhance itself automatically, is referred to as *bootstrapping* the synthesis of the model [114]. The difficulty of bootstrapping is that: (1) If the model (as is) fits a new example well, there is no use of adding the new example to the model. This must be automatically verified. (2) If the model doesn't fit the new example, the correspondences are incorrect and the example cannot be added to the model. (3) It should be fully automatic. Nowadays, several statistical models are available, ready to be used and reused. In this chapter we present a bootstrapping algorithm based on an initial statistical model, which automatically fits to new scan data with noise and holes, and which is capable of measuring the redundancy of new example faces.

6.1.1 Related work

The need for bootstrapping statistical models was posed by Vetter et al. [114]. They introduced a bootstrapping algorithm for statistical models, and showed that the use of merely an optic flow algorithm was not enough to establish full correspondence between example faces and a reference face. Instead, they attain an effective bootstrapping algorithm by iteratively fitting the face model, applying the optic flow algorithm, and updating the face model. Blanz and Vetter also used this bootstrapping algorithm in [19] to build a 3D morphable face model.

Their bootstrapping algorithm works well in case of input data with constant properties, but fails when input data is incomplete and when the optic flow algorithm fails. To bootstrap the 3D morphable face model with more general face data, Basso et al. [10] added a smoothness term to regularize the positions of the vertices where the optic flow correspondence is unreliable. In case a 3D morphable face model is not yet available, a reference face can be used as an approximation instead, which is a major advantage.

Amberg et al. [6] proposed a non-rigid Iterative Closest Point (ICP) algorithm to establish dense correspondences between a reference face and face scans, but they need an initial rigid transformation for the reference face based on 14 manually selected landmarks. Afterwards, the reference face and the fitted face instances can be used to construct a new morphable face model.

Basso et al. [11] fit the morphable face model to scan data using implicit representations. They also use multiple components and blend the implicit functions at the borders

of components, but they loose the full point-to-point correspondence in the process. So the fitted examples cannot be added to the morphable model.

Huang et al. [49] proposed a global to local deformation framework to deform a shape with an arbitrary dimension (2D, 3D or higher) to a new shape of the same class. Their method also operates in the space of implicit surfaces, but uses a non-statistical deformation model. They show their framework's applicability to 3D faces, for which they deform an incomplete source face to a target face.

The use of multiple components has been used by Blanz et al. to improve the face model fitting [19] and for face recognition purposes [20], but so far the resulting face instances were not accurate enough to be incorporated in the statistic model. The explicit point-to-point correspondences of the fitted face instance and the statistical model had to be established by techniques based on optic flow or non-rigid ICP.

In the previous chapter, a set of predefined face components was used to increase the descriptiveness of a 3D morphable face model. With the use of multiple components, a tighter fit of the face model was obtained and higher recognition rates were achieved. However, by fitting each component individually, components started to intersect, move apart, or move across. So, afterwards the full point-to-point correspondences between the morphable model and the fitted instance were distorted. The post-processing method to blend the borders of the components introduces a new set of surface samples without correspondence to the model either.

6.1.2 Contribution

We present a new bootstrapping algorithm that can be applied to general 3D face data. Our algorithm automatically detects if a new face scan cannot be sufficiently modeled, establishes full correspondence between the face scan and the model, and enhances the model with this new face data.

Without the use of unreliable optic flow [10] or semi-automatic non-rigid ICP [6], we are able to bootstrap the 3D morphable face model with highly accurate face instances. As a proof of concept, we (1) fit the initial morphable face model to several 3D face scans using multiple components, (2) blend the components at the borders such that accurate point-to-point correspondences with the model are established, (3) add the fitted face instances to the morphable model, and (4) fit the enhanced morphable model to the scan data as one single component. In the end, we compare each single component fit obtained with the enhanced morphable model to the single component fit obtained with the initial morphable model. Qualitative and quantitative evaluation shows that the new face instances have accurate point-to-point correspondences that can be added to the initial morphable face model. By comparing the multiple and single component fit, our bootstrapping algorithm automatically distinguishes between new face data to add and redundant data to reject. This is important to keep both the model fitting and the face recognition with model coefficients time-efficient.

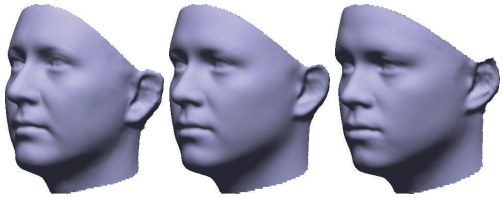


Figure 6.1: Changing weight w_3 {-2,0,+2} causes an unwanted change in the gaze direction.

6.2 Morphable face model

In this chapter, we fit the morphable face model (Sect. 4.2) to 3D scan data to acquire full correspondence between the scan and the model. As we did in the previous chapter (5.3.1) we crop the morphable face model and lower its resolution so that n=12,964 vertices remain for the fitting. We use the new set of correspondences $S = (x_1, y_1, z_1, \ldots, x_n, y_n, z_n)^T \in \Re^{3n}$ to automatically bootstrap the model, in order to increase its expressiveness.

Fig. 6.1 shows the changes of the original face model when the weight of the third eigenvector (w_3) is varied. It can be noticed that the model is tilted upwards and downwards. This variation in one of the first eigenvectors means that the alignment of the 100 sets S_i is not optimal for face identification using model coefficients. To adjust the original model, we realigned each reduced face shape S_i to the mean face $\bar{S}$ of the morphable model using the ICP algorithm, and recomputed the PCA model. Visual inspection of our newly constructed PCA model showed no signs of pose variations.

6.3 Datasets

We fit the morphable face model to 3D scan data from the UND [26], GAVAB [73], BU-3DFE [121], Dutch CAESAR [23], and our local dataset. From all except the UND set, we randomly select four scans yielding a first test set of 16 scans. These scans vary in pose, facial expression, resolution, accuracy, and coverage. This set of 16 face scans is used to test our bootstrapping algorithm. To test the automatic redundancy check, we use a subset of 277 face scans from the UND dataset, namely the first scan of each new subject. To acquire the 3D face data from the scans, we apply the face pose normalization and face segmentation methods described in the previous chapter (Sect. 5.2).

6.4 Bootstrapping algorithm

The main problem in bootstrapping the 3D morphable face model, is that (1) we only want to add example faces that are not covered by the current model, (2) new example faces suffer from noise and missing data, which makes it hard to establish the point-to-point correspondences, and (3) it should be fully automatic. To establish point-to-point correspondences between the 3D morphable face model and new face data with noise and missing data, we apply our model fitting method described in Sect. 5.3 [104]. This

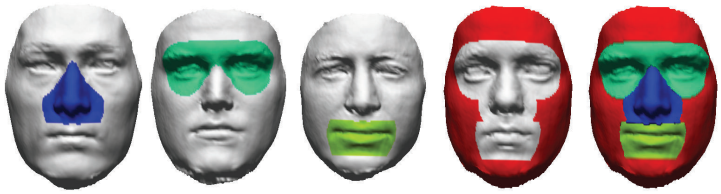


Figure 6.2: Model Fitting. Four face instances S_{comp}, are constructed during the model fitting process each with the focus on a smaller subset of vertices. The composition of these components provides an accurate fit.

method either fits the model as a single component or as a set of predefined components. In case the model is fitted as a single component, the final model fit is in full point-to-point correspondence with the face model, but adds no additional information to the current face model. In case the model is fitted as a set of predefined components, this method produces model fits that go beyond the current statistics of the face model, but the point-to-point correspondence are inaccurate or lost. In this section, we briefly describe the used model fitting method, then we explain our algorithm to establish dense point-to-point correspondences between the multiple component fits and the morphable face model, and finally, we explain how the bootstrapping algorithm can distinguish between new face data to add to the model and redundant data to reject.

6.4.1 Model fitting

The morphable face model that we use (see Sect. 6.2) has m=99 coefficients that can be varied to fit the face model to new scan data. To fit the model, we bring a new 3D face scan into alignment with the 3D face model automatically. First, we automatically normalize the pose of the face, detect the tip of the nose, and segment the face as described in Sect. 6.3. Secondly, we align the face scan to the face model, coarsely by using their nose tips and more accurately using the ICP algorithm. Then we apply the model fitting algorithm as described in Sect. 5.3, using a set of four face components. This model fitting algorithm iteratively adjusts the m coefficients w_i for each component, such that the vertices move closer to the vertices of the scan data. After all components are fitted to the scan data individually, an accurate representation of the new face S_{fine} is acquired. Each component can then be described using the set of m coefficients w_i for the eigenvectors of the face model. However, the fitted face instance S_{fine} may show artifacts at the borders of these fitted component. Note that we use the same PCA model for each component, but with a subset of the model's vertices only.

6.4.2 Correspondence estimation

After the application of the model fitting method, most of the face model's vertices are brought into correspondence with the face scan, but at the component's borders these point-to-point correspondences are less accurate. Only in highly exceptional cases, borders are good enough to bootstrap the face model with S_{fine} directly. To resolve the artifacts at the borders of the individually fitted components, we use their sets of m coefficients w_i that were used to obtain each component. In fact, each set of coefficients

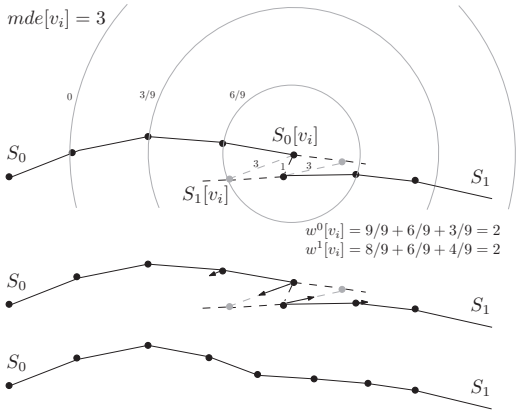


Figure 6.3: Repairing inconsistencies at the borders of components. Vertex $S_0[v_i]$ is moved towards its corresponding point $S_1[v_i]$ using a weighted voting scheme. Close by vertices in both components S_0 and S_1 are used to compute the appropriate weight.

can be used to acquire a full face instance of which the component is simply a predefined subset of vertices. We refer to such a full face instance as S_{comp}. Because we fitted a set of c=4 components, we have c=4 face instances S_{comp}. So the face instance S_{fine} is basically a composition of the c intermediate face instances (Fig. 6.2). To blend the c components, we blend the vertices of the c face instances. In this process, the goal is to determine for each vertex in S_{fine} a new position, such that it has a smooth transition to its neighboring vertices. Once we reach this state, we refer to this final face instance as S_{final}.

Because components can overlap more than several millimeters, Laplacian smoothing of the vertices at the borders would not suffice. The selection of a larger region of triangles and vertices to smooth causes a non-statistical shape deformation and local features may not be preserved. In particular, the places were three or more components overlap, it is hard to regularize the vertex positions using smoothing techniques. Surface stitching techniques as in [107] could be applied to stitch the components, but this would distort the point-to-point correspondences. Mesh fairing using Gaussian curvatures, as in [123] for instance, could smooth some of the border triangles properly. However, these techniques focus on the preservation of sharp features, whereas we want to remove sharp creases caused by overlapping components.

To repair the discontinuities at the borders, we have developed an algorithm, that uses the vertex positions of the intermediate face instances S_{comp}, local neighborhood properties and the PCA model. Since all instances S_{comp} were acquired with the same PCA model, their vertices are all in full correspondence. In case two connected vertices cause a discontinuity in S_{fine}, one can assume that moving each vertex towards the center of mass of its adjacent vertices (as in Laplacian smoothing) slightly improves on the situation. However, propagating this smoothing locally causes the border discontinuities to attract well positioned vertices instead of solving the discontinuity. Propagating such smoothing globally, changes the statistical shape of the face (Fig. 6.4h). Instead, we morph each vertex $S_{fine}[v_i]$ towards its c corresponding positions $S_{comp}[v_i]$. For that we need to (1) detect the local distortions, (2) know which components lie close to which vertices, (3) have for each vertex a weight for each close by component, and (4) recom-

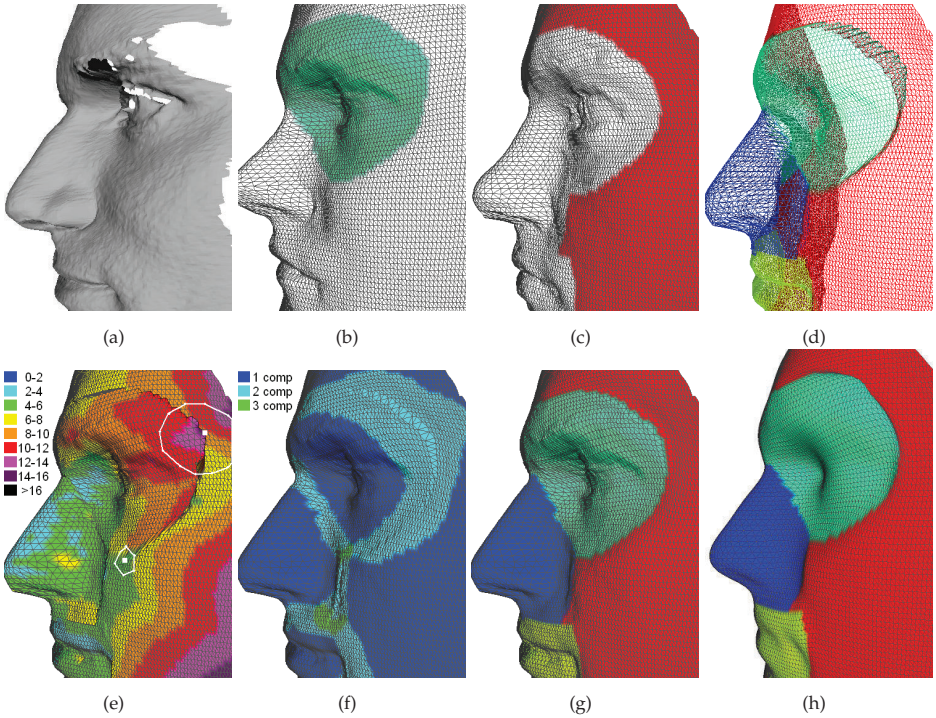


Figure 6.4: Correspondence estimation. By individually fitting components (b,c) to scan data (a), artifacts may occur at the borders of S_{fine} (d). The maximum displacement error (r in mm) for a vertex is an indication of the local mesh distortion (e), towards purple/black means a higher distortion. Using this error, surrounding vertices are selected that all contribute to the voting for a new position of the vertex. After the voting, each vertex is repositioned based on the weighted votes for close by components (f). The final model fit S_{final} (g) presents smooth transitions between the different components. Five iterations of Laplacian smoothing (h) was not enough to repair the artifact close to the eyebrow whereas the face already lost most of its detail.

pute the vertex positions.

First, to detect the local distortion, we determine for each vertex $S_{fine}[v_i]$ its *maximum displacement error* to one of its c corresponding positions $S_{comp}[v_i]$, using

$$mde[v_i] = max_{c \in comp}(e(S_{fine}[v_i], S_c[v_i])),$$

where $e(p, q)$ is the Euclidean distance between two 3D coordinates p and q. When a vertex has approximately the same 3D position in all face instances S_{comp}, its displacement error is small. In that case, we have consensus on its position and thus less need for a repositioning. Second, close by vertices are selected ($S_{fine}[v_j]$) using a sphere of radius $r[v_i]$ around $S_{fine}[v_i]$, where radius $r[v_i]$ equals the displacement error times three, $r[v_i] = 3 \cdot mde[v_i]$. For a vertex $S_{fine}[v_i]$ on a component border, this set of close by vertices include vertices from different components. If all close by vertices belong to the same component as the current vertex nothing will change. Thirdly, each close by vertex $S_{fine}[v_j]$ adds a weighted vote for the component it belongs to. This weight decreases linearly with the distance of vertex $S_{fine}[v_j]$ to $S_{fine}[v_i]$. The maximum weight of one

Figure 6.5: Redundancy estimation. From left to right, the face scan, S_{single}, the residual errors of S_{single}, S_{mult}, the residual errors of S_{mult}, and the color map for the point-to-point distances (in mm).

is for the vertex $S_{fine}[v_i]$ itself, and decreases linearly to zero for radius $r[v_i]$. Because components can move away from each other, radius $r[v_i]$ must be at least two times larger than the maximum displacement error, otherwise nothing changes. In the end, a vertex next to the border of the two components, has a number of close by vertices that vote for its own component and a number of close by vertices that vote for the other component. Then, vertex $S_{fine}[v_i]$ is morphed towards its corresponding position $S_{comp}[v_i]$ in the other component according to these weighted votes. For example, if a vertex (weighted vote of 1) has four close by vertices with weighted votes of 0.25 of which two vertices lie on an other component S_{comp}, then we have a 3-to-1 vote and vertex $S_{fine}[v_i]$ if morphed for 25% towards $S_{comp}[v_i]$. Another example in 2D is shown in Fig. 6.3. In case close by vertices belong to three or more components, the vertex $S_{fine}[v_i]$ is morphed to three or more different positions $S_{comp}[v_i]$.

This *weighted voting scheme* for the local geometry of a face instance S_{fine}, results in a proper blending of the components. Since the overall topology of the face instance is retained, and the problematic vertices and their local neighborhood are assigned to new positions, the adjusted face instance S_{fine} is now in full correspondence with the face model. Fig. 6.4, shows the multiple component fit S_{fine} based on four components S_{comp} (two are shown), the displacement errors, the blending, and the final result S_{final}.

6.4.3 Redundancy estimation

After the regularization of the vertex positions, the repaired face instance S_{final} can be added to the morphable face model. To do so, we can apply the ICP algorithm to finely align S_{final} to the mean face $\bar{S}$ of the morphable model and include it in the example set of 3D faces from which the morphable face model was built. Then, we can recompute the PCA model using 100+k example faces, and keep the $m + k$ principal eigenvectors and eigenvalues of the face (Sect. 6.2). This way the face properties of a new example face can be added to the statistical face model. With this enhanced model, we should be able to produce accurate model fits with only a single component to face scans similar to those k example scans. In the end, each face scan can be described using $m+k$ model coefficients, and face identification can be performed based on such a $m + k$ feature vector.

The addition of k extra example faces to the current face model, causes an increase of computation costs for both the model fitting and face identification method. So, it is important not to add example faces that are already covered in the current morphable face model. Therefore, we estimate the redundancy of encountered example data. First, the

morphable face model is fitted as a single face component to the 3D scan data, which we refer to as S_{single}. Secondly, we fit the morphable face model using multiple face components with the improved correspondence estimation described above, S_{mult}. After the model fitting process, a residual error between the vertices of the model fit and the scan data remains. The difference of the two residual errors of S_{single} and S_{mult}, can be used to estimate the redundancy of the new face scan. In case the residual error of S_{single} is significantly larger than that of S_{mult}, then the face scan is most likely not contained in the current morphable face model, and we should add S_{mult} to the model.

To compute the residual error of these model fits we use the RMS distance of closest point pairs,

$$d_{rms}(S, scan) = \sqrt{\frac{1}{n}\sum_{i=1}^{n} e_{min}(p_i, scan)^2} \tag{6.1}$$

using all n vertices of S_{single} and S_{mult}. Closest point pairs (p,p') for which p' belongs to the boundary (including holes) of the face scan, are not used in the distance measure. Fig. 6.5 shows for one face scan, the two model fits and their residual error maps. For this example the RMS error for S_{single} is 0.89 mm and for S_{mult} 0.68 mm.

6.5 Results

To elaborate on the performance of our bootstrapping algorithm, we applied it to the dataset of 16 different face scans and to the subset of 277 UND scans. The small set is used to evaluate the model fitting and correspondence estimation algorithm. The UND set is used to test the redundancy estimation.

6.5.1 Correspondence estimation

To evaluate the correspondence estimation, we compare the residual errors of fitted face instances. First, we fit the initial morphable face model as a single component to the segmented face data S_{single}. Secondly, we fit the initial model to the segmented face data using the four components and blend their borders. Thirdly, we add these 16 face instances S_{mult} to the example set and recompute the PCA model, keeping the m=99 principal components of the face. Finally, we fit the enhanced morphable face model to the same segmented face data using a single component S^+_{single}.

In Tab. 6.1, we report the residual errors of the fitted face instances. We use these errors for quantitative evaluation of the produced fits. For all face scans, S_{mult} has a lower residual error than S_{single}, which means that a higher fitting accuracy is achieved with the use of multiple components in combination with the improved correspondence estimation. After the model was enhanced with the 16 face instances S_{mult}, the model was again fitted as a single component to the 16 face scans. Now, all residual errors are lower for S^+_{single} than they were for S_{single}, which means that our bootstrapping algorithm successfully enhanced the morphable face model. For one face scan, the face instance S^+_{single} is even more accurate than S_{mult}. This is possible, because we enhanced the model with the facial variety of 16 faces at once. We expect that the residual errors of S^+_{single} can be lowered further, by iterating the process of (1) fitting the enhanced model

Dataset	S_{single}	S_{mult}	S^{+}_{single}	$S_{single} - S_{mult}$	$S^{+}_{single} - S_{mult}$
GAVAB	1.36	1.24	1.27	**0.13**	0.04
GAVAB	1.34	1.16	1.19	0.18	0.03
GAVAB	2.04	1.59	1.56	**0.45**	-0.03
GAVAB	1.32	1.16	1.19	0.17	0.03
BU-3DFE	1.18	1.01	1.02	0.18	0.02
BU-3DFE	1.28	1.12	1.15	0.16	0.03
BU-3DFE	1.06	0.96	0.96	**0.10**	0.00
BU-3DFE	1.61	1.40	1.49	**0.21**	0.09
local	0.70	0.55	0.58	0.15	0.03
local	0.89	0.68	0.69	**0.21**	0.01
local	0.79	0.62	0.67	0.17	0.05
local	0.69	0.55	0.58	**0.14**	0.04
CAESAR	1.86	1.77	1.80	0.09	0.03
CAESAR	1.88	1.77	1.78	**0.11**	0.01
CAESAR	1.81	1.74	1.77	0.07	0.03
CAESAR	1.80	1.75	1.78	**0.05**	0.03

Table 6.1: RMS errors (mm) of output models to input scans. The model fits with the smallest and largest difference in residual errors (bold) are show in Fig. 6.6.

using multiple components, (2) replacing the 16 face instances S^{+}_{mult}, and (3) building a new PCA model. In fact, we tried it for the face scan in Fig. 6.5 and lowered its RMS distance for S_{mult} (0.68 mm) and S^{+}_{single} (0.69 mm) to 0.64 mm for S^{+}_{mult}. So, iteratively replacing the 16 instances of the enhanced model with their improved instances S^{+}_{mult}, will probably help to some extend. Note that the residual errors are lowest for our local set, which contains the highest resolution scans, and the highest errors for the low resolution CAESAR faces. This is due to the RMS distance that measures point-to-point distances. Because we are interested in the difference between residual errors, this works fine, otherwise, one could use a surface mesh comparison tool instead. In chapter 2, we used *metro* [32] for that.

In Fig. 6.6, we show some of the resulting model fits and their distance maps acquired with our bootstrapping algorithm. In this figure, we show the two faces per dataset that achieved the smallest and largest difference in residual errors in the same order as in Tab. 6.1. Visual inspections of the fitted models shows an improved single component fit of the enhanced model to the scan data (S^{+}_{single}), compared to the single component fit using the initial morphable model (S_{single}). This can be seen in the residual error maps as well. That the bootstrapping algorithm successfully incorporated the 16 face scans in the morphable face model, can be seen by face instances S_{mult} and S^{+}_{single}, which are very similar. The initial morphable face model consisted of neutral expression scans only. Nevertheless, the use of multiple components allows for correspondence estimation among some expression data as well.

6.5.2 Redundancy estimation

To distinguish between new and redundant face data, we computed the residual errors for face instances S_{single} and S_{mult} using the RMS distance measure. In Tab. 6.1, we reported the RMS errors for our set of 16 faces. These differences in RMS error for $S_{single} - S_{mult}$ vary between 0.05 and 0.45. The maximum difference of 0.45 was achieved

Figure 6.6: Automatic correspondence estimation. From left to right, the segmented and pose normalized faces, the single component fit S_{single}, its distance map, the multiple component fit S_{mult}, its distance map, and the single component fit S^{+}_{single} with its distance map. Faces in rows two, four, and five were selected as being new to the model.

Figure 6.7: Automatic bootstrapping and redundancy estimation. From left to right, the segmented and pose normalized faces, the single component fit S_{single}, its distance map, and the multiple component fit S_{mult} with its distance map. The instances S_{mult} were automatically selected as being new.

for the sad looking person on row four in Fig. 6.6. With the use of a threshold t for the difference in RMS error, we decide whether a face is redundant or new. Based on the visual inspection of the faces in Fig. 6.6, we decided to select t=0.17 for our experiments. In case the RMS difference is higher than t, we consider a face to be new and otherwise as redundant. With this threshold, we classify only the faces in row two, four, and five as being new.

We applied our bootstrapping algorithm to the 277 UND scans, and let our algorithm automatically select potential faces to add to the model, without actually adding them. This way we can see which face scans (persons) are new to the model. For these persons, we may assume a difficulty in identifying them, because a coefficient based recognition system may confuse that person with a different person that has those coefficients. Out of the 277 UND scans, 35 scans were found as being new to the system, that is, having a decrease in RMS error of $S_{single} - S_{mult}$ higher than threshold t. Some of these produced fits are shown in Fig. 6.7. Most of the selected faces have indeed new face features and should be added to the morphable face model. However, some of the faces that are covered by facial hair produce less reliable fits. To improve on these fits, one could apply a skin detection algorithm and remove the hair beforehand.

6.6 Concluding remarks

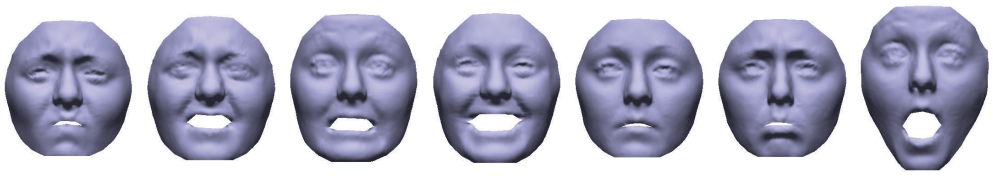
In this chapter we presented a method for establishing accurate correspondences among 3D face scans. With the use of an initial morphable face model and a set of predefined components, we are able to produce accurate model fits to 3D face data with noise and holes. Afterwards, the components still have defects on their border, for which we presented a new blending technique that retains the full correspondence between the face instance and the morphable model. These newly generated face instances can be added to the example set from which a new and enhanced morphable model can be built. We tested our fully automatic bootstrapping algorithm on a set of 16 new face scans, to show that the new morphable face model has indeed an enhanced expressiveness. By adding data to an initial morphable face model, we can fit the model to face scans of a larger population and reducing the confusion among identities. Therefore, the building of a strong morphable face model is essential for the difficult task of 3D face recognition. However, adding data to a morphable model increases the computation costs for both model fitting and face comparison. Therefore, we developed an automatic redundancy check, which discriminates between new face data to add to the model and redundant face data to reject. Based on the initial 16 scans, we selected a residual error threshold for the automatic redundancy check. Then, we applied our bootstrapping algorithm to 277 UND scans. Our bootstrapping algorithm successfully established full correspondence between these scans and the initial morphable model, and selected 35 scans that were new to the model.

With our new bootstrapping algorithm, we are able to successfully update an initial face model, which we use to produce more accurate fits to new scan data. Compared to previous work, our algorithm is fully automatic, reuses initial face statistics, checks for redundancy, and retains the full correspondence even in case of noisy scan data with holes. Our algorithm successfully enhances the neutral morphable face model with

new (close to) neutral face scans. It should also apply to e.g. a surprised face model and surprised scans, but not to a neutral face model and surprised scans. To establish correspondences among face scans with different expressions, new automatic algorithms are required. In the next chapter, we target this problem by semi-automatically building morphable expression models, and using those to automatically establish correspondences among scans with different expressions. In turn, it should be possible to automatically bootstrap each of the expression models with new face data, as we did in this chapter.

Chapter 7

Facial expression modeling



In Chapters 5 and 6, we used an existing morphable face model to model 3D scan data. This statistical model consists of face data with neutral expressions only, and with the bootstrapping algorithm we presented in the previous chapter, we could extend the model with small expression deformations only. Adding these expression deformations to the same morphable face model causes the statistical spaces of identities and expressions to interfere, which means that the model coefficients are no longer reliable for face identification. As a result, we should be able to fit the model to a neutral and an expression scan of the same person, but we cannot use the coefficients to identify one with the other.

This chapter presents a new automatic and efficient method to fit a statistical expression model of the human face to 3D scan data. To achieve expression invariant face matching we incorporated expression-specific deformation models in the fitting method. In a global to local fitting scheme, the identity and expression coefficients of this model are adjusted such that the produced instance of the model accurately fits the 3D scan data of the input face. Quantitative evaluation shows that the expression deformation as well as a set of predefined face components improve on the fitting results. 3D face matching experiments on the publicly available UND, GAVAB, BU-3DFE, FRGC v.2 datasets show high recognition rates of respectively 99%, 98%, 100%, and 97% with the use of the identity coefficients. Results show that not only the coefficients that belong to the globally optimized model fit perform well, but that the coefficients of four locally optimized model fits can produce similar recognition rates. Finding the optimal model fit is hard and loosening this requirement could make a system more robust.

7.1 Introduction

Statistical models of the human face have proven to be an effective tool for person identification of 3D face scans. To build a statistical model, a set of example faces is required with face features in full correspondence. With such a model, a new face instance can be constructed as a linear combination of the example faces. For 3D face identification, the idea is to use the statistical model to construct a face instance that resembles an input image. The way these example faces are combined linearly to represent an input face, provides both global and local information about the input face, that can be used to classify and identify different input faces. Expressions are a problem, because they change the resemblance of the input faces.

7.1.1 Related work

Most of the early 3D face recognition methods focused on variants of the Iterative Closest Point (ICP) [16] algorithm to find similarity between two 3D face scans. As 3D face recognition became more challenging with larger datasets and expression scans, the ICP-based methods showed two main disadvantages. The non-rigid expression deformations forced the ICP-based methods to rely on smaller face regions such as the nose and forehead, and the computational expensive face matching lowered its practical use. Methods of Faltemier et al. [42] and Mian et al. [69] reported high recognition rates based on nose regions in combinations with ICP. For efficient face matching, the extraction of person specific features became the new area of interest.

With high recognition rates, low computational costs during face matching, and high robustness to noise and missing data, *3D morphable face model* based methods prove to perform well. To build a 3D morphable face model, dense correspondence are required among a set of 3D example faces. The mean face and the statistical variation of these faces can be computed using Principal Component Analysis (PCA). Using the statistical face variations, the mean face can be deformed to fit the noisy scan data. The way such a model is deformed (larger, wider, longer nose, etc.), provides information on the geometric shape properties of the input face. The coefficients that induce these deformations form a relatively small feature vector for efficient face matching. For reliable model coefficients, the model deformation must be independent of changes in the face pose. Therefore, the model fitting is often combined with an ICP algorithm to compensate for the rigid transformation between closest point features. Because both the model fitting and the ICP algorithm are local optimization methods, a coarse alignment between the scan data and the model should be automatically established first.

Blanz and Vetter use a 3D morphable face model to model 3D faces out of 2D images [19]. In [18], Blanz et al. fit a morphable model to 3D scan data and use the deformation weights (or model coefficients) to recognize faces with neutral expressions. In each iteration of their stochastic Newton algorithm, the current model instance is projected to 2D image space and the model coefficients are adjusted according to the difference in texture and depth values. For the coarse alignment and to initiate the morphable model, they manually select seven corresponding face features on their model and in the depth scan.

Amberg et al. [5] build a PCA model from 270 identity vectors and a PCA model

from 135 expression vectors and combined the two into a single morphable face model. Their method fits this model to 3D scan data by iteratively finding closest point pairs to improve on the alignment, the identity deformation, and the expression deformation at the same time. Their local optimization method, which does not guarantee convergence to the global minimum, returns a set of identity coefficients that perform well in terms of face recognition. For the initial alignment of the scan to the model, they use our automatic face pose normalization method from Sect. 4.4.1.

Lu and Jain [65] train a morphable expression model for each expression in their test set. Starting from an existing neutral scan, they fit each of their expression models separately to adjust the vertices in a small region around the nose to lower the ICP error between that particular neutral scan and an expression scan. The expression model that produces the most accurate fit is used to deform the neutral scan. For the initial alignment they use three automatically detected feature points. For the fitting, they combine the accurate ICP alignment for the rigid transformation with the fast *eigenspace projection* [108] for the expression deformation. This process is iterated until convergence and the lowest residual error is used as the dissimilarity score between the neutral scan and the new scan. Although the authors use PCA models, their method can be classified as an ICP based method, because the fitting procedure has to be repeated for every pair of face scans in the dataset. The expression models are merely used to improve on the ICP fitting procedure.

Mpiperis et al. [74] build a bilinear PCA model for the BU-3DFE dataset suitable for both expression and identity recognition *after* a face scan is brought into full correspondence with the model. To establish full correspondence, they detect the boundary of the mouth, elastically deform a low resolution face mesh to the scan data (considering the mouth), and subdivide the mesh for denser correspondences. The bilinear PCA model is solely used to map the full correspondence to expression and identity coefficients that are either used for expression classification or person identification.

Kakadiaris et al. [56] deform an annotated subdivision face model to scan data. Their non-statistical deformation is driven by triangles of the scan data attracting the vertices of the model. The deformation is restrained by a stiffness, mass and damping matrix, which control the resistance, velocity and acceleration of the model's vertices. They use the newly created geometry for wavelet analysis and achieve state of the art recognition results on the Face Recognition Grand Challenge (FRGC) [79].

7.1.2 Contribution

Firstly, we introduce seven morphable expression models, for the "expressions" anger, disgust, fear, happiness, sadness, surprise, and inflated cheeks. Secondly, we use a new morphable identity model to perform expression invariant 3D face identification, in combination with the expression model. Starting with a dataset of neutral scans, expression scans, and a small set of annotated landmarks, we describe how to build a strong multi-resolution morphable model for both identity and expression variations of the human face. A new feature in this modeling method is the decoupling of the pose normalization and deformation modeling, so that the model fitting becomes highly time-efficient. Thirdly, we introduce a new model fitting method that combines eigenspace sampling, eigenspace projection, and predefined face components. This method is able

to produce accurate fits for the morphable identity model in combination with the best expression model to new expression scans. The statistics captured in the morphable face model allows for the robust handling of noise and holes. Afterwards, the final expression instance and its model coefficients can be used as the complete and noiseless representation of the expression scan, to automatically extract the facial landmarks, to bootstrap the face model, to remove the expression, and for expression invariant face recognition. Fourthly, for 3D face recognition with model coefficients, we propose a new multiple minima approach and compare the results with the global minimum approach. Local minima in facespace are easier to find and their locations provide valuable information for face identification.

Results show that (1) our method can be applied with considerable success to a large range of datasets with high recognition rates of 99%, 98%, 100%, and 97%, for the UND, GAVAB, BU-3DFE, FRGC v.2 datasets, (2) the use of expression models is essential for a high performance, (3) the use of multiple components (MC) improves on the single component (SC) results, (4) in case of scan data with lower quality, as in the GAVAB dataset, the multiple minima (MM) approach can improve the system's performance. (5) the time-efficiency of our complete 3D face recognition system allows for its application in face authentication and face retrieval scenarios.

7.2 Datasets

In this chapter, we use the 3D face scans of the UND [26], the GAVAB [73, 100], the BU-3DFE [121], the FRGC v.2 [79], and the USF Human ID 3D [109] databases. As opposed to Chapter 5 we have now access to the FRGC v.2 and the USF datasets, and we use all of the BU-3DFE scans. The UND set, from the University of Notre Dame, contains 953 frontal range scans of 277 different subjects with mostly neutral expression. The GAVAB set consists of nine low quality scans of which we use seven for each of its 61 subjects as in [100]. The BU-3DFE set, from the Binghamton University, was developed for facial expression classification. This set contains one neutral scan and 24 expression scans having different intensity levels for each of its 100 subjects. The FRGC v.2 set, of the Face Recognition Grand Challenge, is often used for the evaluation of 3D face recognition systems and contains 4007 high quality 3D face scans of 466 different subjects. Almost half of these scans show an expression varying from a smile or frown to a pronounced laugh or a surprised look. The USF Human ID 3D database, from the University of South Florida, contains 136 high quality full head scans without expressions.

We aim at 3D face modeling and recognition, and therefore we need to segment the face from each scan. For that, we employ our pose normalization method from Sect. 4.4.1 that takes as input a triangular surface mesh and outputs the normalized pose of the face with the tip of the nose in the origin. The face is segmented by removing the scan data with a Euclidean distance larger than 130 mm from the nose tip. In several scans of the FRGC v.2, the frontal pose was not completely recovered due to hair covering the face. To further improve on the face's pose, an average nose template (shown in Fig. 7.2) is aligned to each segmented face and the inverse transformation applied to the scan. This template was selected for its high expression invariance as described in [69].

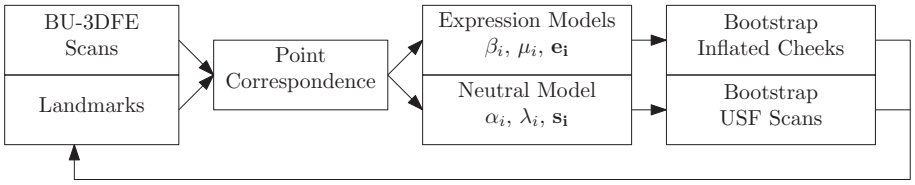


Figure 7.1: Flow chart of the semi-automatic model building.

Qualitative evaluation showed that the tip of the nose was found in all 3D scans, except for two scans of the FRGC v.2 which did not have a nose (2 failures out of 8023 scans). Mian et al. [69] report an initial nose detection failure of 85 out of 4950 FRGC v.1 and v.2 scans, and a final pose correction failure for eight of them. Faltemier et al. [42] report a failure of 72 out of 4007 face scans.

7.3 Morphable face model

In this chapter, we use a new morphable face model built from both 3D neutral and expression scans of the human face. We fit this model to 3D scan data in such a way that expressions can be removed and subjects identified in an expression invariant manner. To build a morphable face model with expressions, an example set of subjects showing various expressions is required. For that, we use the BU-3DFE [17] dataset, from which we select the 100 neutral scans and 600 expression scans at their highest intensity level. The BU-3DFE set was developed for facial expression classification. This set contains one neutral scan and 24 expression scans having different intensity levels, for each of its 100 subjects. From this set we selected the neutral scans and the highest intensity level expression scans (anger, disgust, fear, happiness, sadness, surprise at level 4). The goal is to model a neutral face model from a dense set of correspondences, and a neutral-to-expression model for each of the expressions anger, disgust, fear, happiness, sadness and surprise. The neutral face model, which is built from the 100 neutral scans, captures the identity of different subjects, whereas a neutral-to-expression model captures the facial changes caused by a certain expression.

A morphable face model is a type of statistical point distribution model (PDM) [35], where the points are facial features that have a different distribution among different faces. Building a morphable face model, requires n dense correspondences $S = (x_1, y_1, z_1, \ldots, x_n, y_n, z_n)^T \in \Re^{3n}$ among a set of input face scans. Principal Component Analysis (PCA) is used to capture the statistical distribution of these correspondences among the input faces. Because the automatic estimation of reliable dense correspondences among noisy face scans with expressions is still unsolved, we propose a semi-automatic correspondence estimation that requires 26 facial landmarks. With the use of these 26 landmarks, we construct a low resolution mesh that is projected to the cylindrical depth image of a 3D face scan. By subdividing the triangles of the low resolution mesh, a multi-resolution representation of the face is constructed. At each level, we assume that the vertices between different subjects or expressions correspond. The correspondences at the highest level are used to build a neutral 3D morphable face model as well as a morphable expression model for each of the expressions. Because the man-

ual annotation of facial landmarks in 3D face scans is often a major disadvantage in statistical modeling, we explain in Sect. 7.3.6 how our initial morphable face model can be used to enhance itself with new 3D scan data. This *automatic bootstrapping* is a useful tool to limit the user input. The flow chart of our semi-automatic modeling approach is shown in Fig. 7.1. Our semi-automatic model building consists of the following steps:

1. Manual annotation of facial landmarks, including nose, eyes, eyebrows, and mouth.
2. Cylindrical depth image construction.
3. Multi-resolution face mesh construction.
4. Building the morphable identity model.
5. Building the morphable expression models.
6. Automatic bootstrapping the morphable model.
7. Data reduction.
8. Component selection.

7.3.1 Landmark annotation

In each of the 700 pose normalized (raw) BU-3DFE scans, we manually selected the same sequence of 26 facial landmarks as an initial set of correspondences. These landmarks include locations on the nose, mouth, eyes, and eyebrows, and provide a coarse notion of facial changes among different identities and expressions. This is the only user input throughout this chapter. In fact, most of these landmarks were already annotated in the BU-3DFE set and the nose tip was detected automatically.

7.3.2 Cylindrical depth image

Knowing that almost all face scans (even with facial hair and expressions) can be correctly pose normalized after the final alignment to an average nose template (Sect. 7.2), it makes sense to build the morphable face model based on face scans in the coordinate system of this nose template. Each BU-3DFE scan was brought into alignment with the reference nose template, which has the desired pose and its nose tip in the origin. Although the nose template was accurately fitted to the face scans, this doesn't mean that the nose tip of the face scan is aligned to the nose tip in the template. A smaller nose, for instance, has its tip behind the template (lower z-value) and a larger nose in front of the template (higher z-value). To produce a cylindrical depth image $d(\theta, y)$ for each of the face scans, we simulate a cylindrical laser range scanner. To cover most of the face, the nose template and the aligned face scans are moved 80 mm along the positive z-axis. A surface sample is acquired for each angle θ at each height y with radius distance d to the y-axis of the coordinate system. Basically, we cast a horizontal ray at height y with an angle θ in the xz-plane from the y-axis to the face scan, and store the distance to the furthest ray-triangle intersection. Because we model the face only, we scan the front half of the cylinder i.e. the angles $\theta = [180°, 360°]$. The step size for θ is $0.4°$ (450 angles) and the step size for y is 0.5 mm, producing a high resolution 2D cylindrical depth image. The 26 annotated landmarks are projected to the cylindrical depth image, by assigning them to the closest ray. Note that the cylindrical depth image can be converted to a 3D triangle mesh by connecting the adjacent samples and projecting the cylindrical coordinates to 3D.

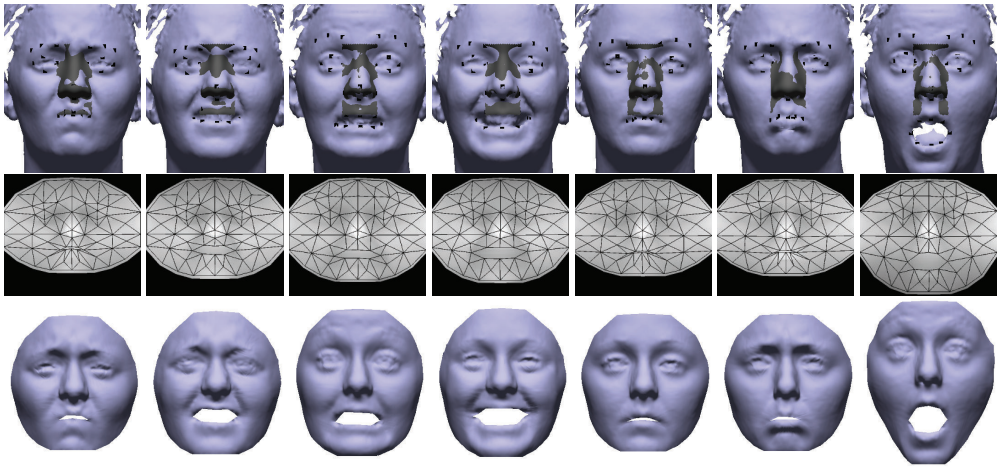


Figure 7.2: Semi-automatic model building. The pose normalized faces are annotated with landmarks (first row) that correspond among different expressions and different subjects to construct an initial face mesh as a layer over the cylindrical depth image (second row). A subdivision scheme is applied to acquire dense 3D face meshes (third row).

7.3.3 Multi-resolution face mesh

To construct the low resolution face mesh, we extend the initial set of landmarks using predefined locations on the cylindrical depth map, such as the location ($270°$, nose tip$_y$+180) on the forehead, the location ($270°$, lower lip$_y$-70) on the chin, leftmost location ($180°$, upper lip$_y$) and rightmost location ($360°$, upper lip$_y$). This way a coarse 2D mesh is constructed as an overlay on the cylindrical depth image (Fig. 7.2). By using relative locations, we ensure that all necessary face features (chin, forehead, cheeks) are captured, whereas the cylindrical depth images provide radial depth variation among different subjects. Alternatively, a more geometry guided approach could be used instead. Large triangles are avoided in the coarse mesh, by adding extra vertices in sparse density areas. The final low resolution mesh consists of 68 vertices and 110 triangles. To improve on the cylindrical depth map quality, depth values outside the face polygon and inside the mouth polygon are removed, and depth values are interpolated and extrapolated to fill the gaps. With the use of this underlying depth map, the face mesh can be projected to 3D. To construct a higher resolution face mesh, we subdivide each triangle in the low resolution mesh into four smaller congruent triangles. In an iterative manner, we construct five meshes with a resolution up to 28160 triangles. The highest resolution mesh is projected in 3D using the cylindrical depth image and speckle noise is removed by a single iteration of Laplacian smoothing. The advantage of the subdivision scheme is that each vertex in a lower resolution mesh has the same index number in the highest resolution mesh, which means that the highest resolution mesh can be used as the final multi-resolution face mesh.

In the end we have acquired for each input face a set of dense correspondences $S = (x_1, y_1, z_1, \ldots, x_n, y_n, z_n)^T \in \Re^{3n}$, with n=14288 vertices of which the first 26 were annotated. Valid transitions for lower resolution faces are in this case, n=68, n=246, n=932, n=3624. The multi-resolution mesh construction is shown in Fig. 7.2. In addi-

tion to the multi-resolution mesh, we also construct a mirrored version. Therefore, we interchange the coordinates of the left and right side landmarks, mirror the cylindrical depth map in the y-axis, and redo the multi-resolution mesh construction. With these additional faces, the variability of the morphable face model increases. Also, the statistical mean face becomes fully symmetric around the y-axis, because facial asymmetry is modeled in both directions.

7.3.4 Morphable identity model

Building an identity based face model requires a training set of neutral faces of different subjects in full correspondence. For that we use the m=200 (original and mirrored) neutral face instances $S = (x_1, y_1, z_1, \ldots, x_n, y_n, z_n)^T \in \Re^{3n}$, with n=14288. Principal Component Analysis (PCA) is applied to these neutral face instances S_i to acquire an orthogonal coordinate system in which each face can be described as a linear combination of principal shape vectors, i.e. the eigenvectors of the eigenspace. Turk and Pentland [108] also described how to compute the "eigenfaces" that define this "face space", for 2D intensity images.

Each of the m=200 face instances S_i is described as a one-dimensional vector of size $3n$, and the average of these vectors is the mean face shape $\bar{S}$. The mean shape $\bar{S}$ is extracted from each face instance S_i, and these shape deformation vectors are stored in a matrix $A[S_1 - \bar{S}, S_2 - \bar{S}, \ldots, S_m - \bar{S}]$ of size $3n \times m$. To compute the eigenfaces, a covariance matrix $C = AA^T$ is constructed from which the eigenvectors and eigenvalues should be extracted. However, the covariance matrix of size $3n \times 3n$ can be too large for practical use of the Singular Value Decomposition (SVD). Instead of directly computing the eigenvectors of C, we can compute the eigenvectors and the eigenvalues of C using the smaller matrix $L = A^T A$ of size $m \times m$, and multiply A with the eigenvectors ($\mathbf{v_i}$) of L to acquire the final set of eigenvectors $\mathbf{s_i}$ for C:

$$\begin{aligned} L\mathbf{v_i} &= \lambda_i \mathbf{v_i} \\ A^T A\mathbf{v_i} &= \lambda_i \mathbf{v_i} \\ AA^T A\mathbf{v_i} &= \lambda_i A\mathbf{v_i} \\ C(A\mathbf{v_i}) &= \lambda_i (A\mathbf{v_i}) \\ C\mathbf{s_i} &= \lambda_i \mathbf{s_i} \end{aligned}$$

Note that the non-null eigenvalues λ_i of C and L are the same. There are at most $m-1$ valid eigenvectors, which are sorted in the descending order of their corresponding eigenvalues. The eigenvectors $\mathbf{s_i} = (\Delta x_1, \Delta y_1, \Delta z_1, \ldots, \Delta x_n, \Delta y_n, \Delta z_n)^T$, the eigenvalues λ_i and identity coefficients α_i are used to model an identity vector according to

$$S_{id} = \sum_{i=1}^{m-1} \alpha_i \sqrt{\lambda_i} \cdot \mathbf{s_i} \ .$$

Adding this identity vector to the mean face $\bar{S}$ results in a 3D face with a new identity, $S_{inst} = \bar{S} + S_{id}$. The identity coefficient α_i represents the number of standard deviations $\sigma_i = \sqrt{\lambda_i}$ that a face instance morphs along eigenvector $\mathbf{s_i}$. To determine the

coefficient α_i for face instances S_i, one can subtract $\bar{S}$ and project its residual identity vector S_{id} into face space:

$$\alpha_i = \frac{1}{\sqrt{\lambda_i}}(\mathbf{s_i}^T S_{id}) \ .$$

The projection of the identity vector onto each eigenvector returns the least-squares solution defined as $\boldsymbol{\alpha} = (\tilde{S}^T\tilde{S})^{-1}(\tilde{S}^T S_{id})$, because the columns in matrix $\tilde{S} = [\mathbf{s_1}, \mathbf{s_2}, \ldots, \mathbf{s_{m-1}}]$ are orthogonal [61]. Without the use of PCA (as in [65]), one must solve the least-squares solution for the (linearly independent) face instances S_i according to $\boldsymbol{\alpha} = (A^T A)^{-1}(A^T S_{id})$, which is computationally more expensive. In the end, the vector $\boldsymbol{\alpha}$ can be used to describe a subject in face space, and as a feature vector for the recognition of 3D faces.

7.3.5 Morphable expression model

Building an expression model requires full correspondence between all the neutral faces and the sets of expression faces. Matrix A is now initiated with the difference between the expression face E_i and neutral face S_i of subject i. The computation of the eigenvalues and eigenvectors for matrix $A[E_1 - S_1, E_2 - S_2, \ldots, E_m - S_m]$ remains the same. The eigenvectors $\mathbf{e_i} = (\Delta x_1, \Delta y_1, \Delta z_1, \ldots, \Delta x_n, \Delta y_n, \Delta z_n)^T$, the eigenvalues μ_i ($\sigma_i^2 = \mu_i$) and weights β_i are used to model an expression vector according to

$$S_{expr} = \sum_{i=1}^{m-1} \beta_i \sqrt{\mu_i} \cdot \mathbf{e_i} \ .$$

Adding an expression vector S_{expr} to a neutral face instance S_{inst}, results in a 3D face with a certain expression, $S_{expr.inst} = \bar{S} + S_{id} + S_{expr}$.

After the correspondence estimation and mirroring, our training set consists of 200 neutral faces and 1200 expression faces in full correspondence. At this point, we could either build a generic model including all expressions or a specific model for each of the expressions, anger, disgust, fear, happiness, sadness, and surprise. In the work of Lu and Jain [65], experiments with an expression-generic and expression-specific models show that the latter outperforms the former. Because their example faces are different and the expressions were only modeled in a small area around the nose, we decided to use expression-specific models as well. For each of the expressions we build a new model (β_i, μ_i, $\mathbf{e_i}$), which we use to add an expression to the neutral face instance S_{inst}, but also to remove an expression from $S_{expr.inst}$.

7.3.6 Automatic bootstrapping

For face recognition purposes it is important to have an identity model S_{id} that describes a large human population. The face space allows for the interpolation between example faces and the extrapolation outside its statistical boundary, but only to some extend. In case a subject cannot be sufficiently described in face space, its identity coefficients α_i become unreliable. However, manually annotating more 3D face scans in order to improve the identity model has its limitations. Instead, we automatically bootstrap the identity model with 134 (beardless) scans of the USF Human ID 3D database. For that,

the morphable face model is fitted to the scan data as described in Sect. 7.4. Secondly, the 26 facial landmarks are extracted, the cylindrical depth image constructed, and the multi-resolution face mesh created as described above. Finally, we rebuild the identity model S_{id} with the 468 original and mirrored sets of correspondences S_i.

Because the FRGC v.2 dataset contains not only scans with the aforementioned expressions, but also scans with inflated cheeks, we select twenty subjects with inflated cheeks and their neutral faces, and build an "expression" model of these scans. For that, we use our automatic bootstrapping algorithm to establish correspondences between these forty FRGC scans and the initial identity model of 200 faces. PCA is applied to the twenty regular and twenty mirrored expression vectors to build the expression model S_{expr} for cheek inflation.

The automatic bootstrapping method that we use here, does not require a perfect fit of the entire model as in Chapter 6, but just for the 26 annotated landmarks. However, to get these landmarks in place, it helps to fit the full morphable model and not just the vertices that correspond to such landmarks.

7.3.7 Data reduction

In Fig. 7.3, the mean face $\bar{S}$ is deformed along the first two eigenvectors of either the identity model or an expression model. Because each of the expressions causes a similar face deformation among the training subjects, the first eigenvector is more or less the vector to move from the cluster of neutral faces to the cluster of expression faces. A negative coefficient β_1 for eigenvector $\mathbf{e_1}$ means that we move away from an expression cluster, causing unrealistic changes. Note that a weight of β_1=2 is already an exaggeration of the expression. The first eigenvector of all expression models except the sad model causes a larger shape deformation than the first eigenvector of the identity model, which in turn causes a larger shape deformation than the second eigenvector of the expression models. This can be easily seen by comparing the eigenvalues λ_i and μ_i of the deformation models, which are larger in case of a larger shape deformation. During the morphable model fitting it is important to optimize the large shape deformations before the smaller shape deformations. Because the smallest eigenvectors are the least significant and add merely noise to a model instance, we reduce the number of eigenvectors for the identity model to m_s=80 and for each expression model to m_e=6.

7.3.8 Component selection

Each eigenvector of a morphable model defines a translation vector for each vertex in the model. With the use of a mask vector one can simply turn a vertex in the model on or off. A vertex that is not selected, is not adjusted nor evaluated during the model fitting. So, the use of a lower resolution for the *multi-resolution* face model speeds up the fitting process. Additionally, we can select *predefined components* such as the nose, eyes, mouth, and the rest of the face and refine each of these components individually to have a larger face variety with the same model [104]. With our expression deformation models, we can find regions of the face that are *invariant to expressions*. When the expression coefficients β_i of an expression model are all set to one, each vertex is translated to a new position. These translation vectors have different lengths, depending on

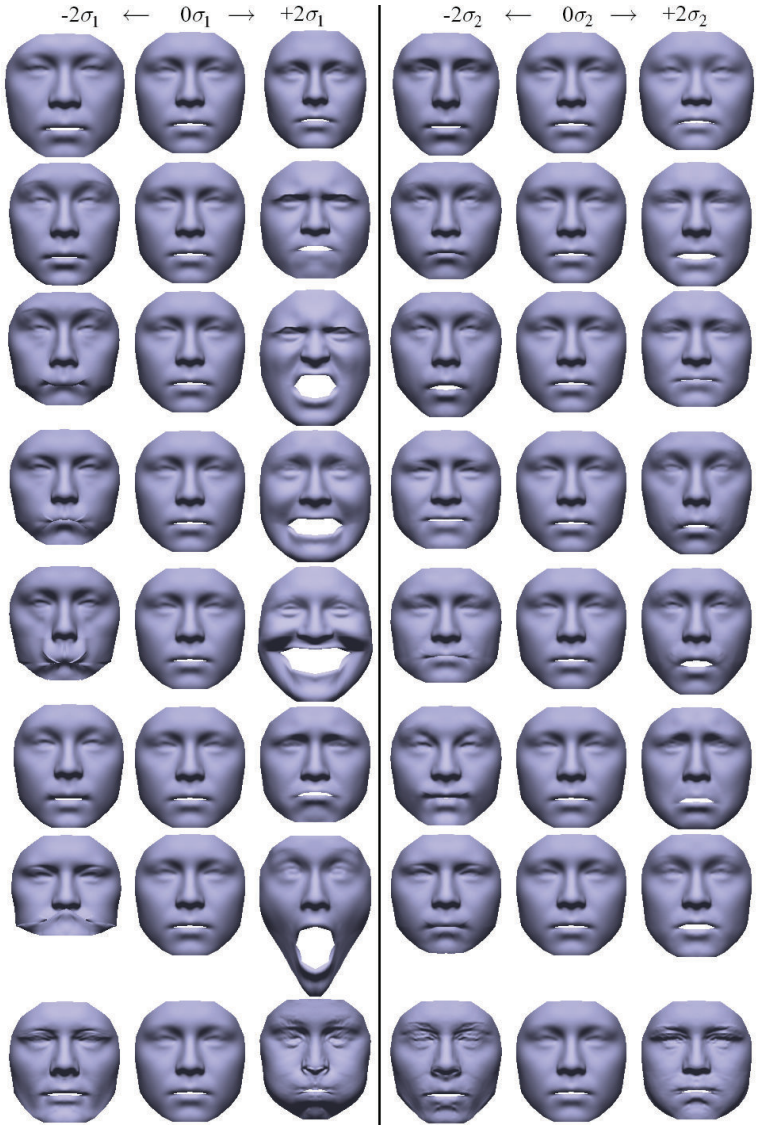


Figure 7.3: Face deformation along the first eigenvector (left) and the second eigenvector (right). Starting from the mean face, a model based shape deformation is applied by changing the coefficients α_1, α_2, β_1, or β_2 to -2 or +2. From top to bottom, the deformation is based on the identity model ($\sigma = \alpha$), and the expression models ($\sigma = \beta$) anger, disgust, fear, happiness, sadness, surprise, and cheek inflation. The deformation β_1=-2 results in unrealistic faces.

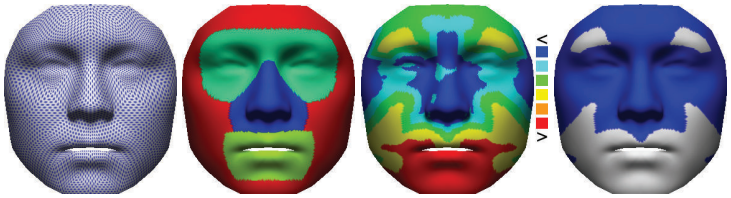


Figure 7.4: Component selection. From left to right, n=3624 lower resolution vertices, four face components, expression deformation intensity, and 60% most static vertices under various expressions.

the selected expression model and the expression invariance of that particular vertex. If the maximum displacement, over all expression models, is stored for each vertex, we can determine face regions that are more static under different expressions. We select 60% of the vertices with the smallest maximum vertex displacements as the static face component. A static face component can be used to coarsely estimate the identity coefficients before estimating the expression coefficients. Fig. 7.4 shows these sets of selected vertex indices.

7.4 Morphable model fitting

The task of the model fitting algorithm is to find the face instance $S_{expr.inst}$ in the high dimensional face space that produces the best point-to-point correspondence with a new face scan. Additionally, the model fitting algorithm should be robust to noise and perform well even when large areas of the face are missing. To regulate the scan density, each face surface is cylindrically rescanned (as in Sect. 7.3) to a uniform resolution ($\Delta\theta$=1.3, Δy=1.3) with approximately 16,000 vertices. Slender triangles and small connected components are removed. When accurate point-to-point correspondences are established between the morphable face model and the scan data, then the identity coefficient vector $\boldsymbol{\alpha}$ can be used for face recognition, or the expression deformation S_{expr} can be subtracted from $S_{expr.inst}$ to produce a neutral face instance S_{inst} for geometry based face recognition. In this section, we describe a fully automatic method that efficiently optimizes the weights $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ to obtain a model instance from the high dimensional face space that accurately fits the face scan. To evaluate if an instance of the morphable face model is a good approximation of the 3D face scan, we use the Root Mean Square (RMS) distance of closest point pairs,

$$d_{rms}(S_{expr.inst}, scan) = \sqrt{\frac{1}{n}\sum_{i=1}^{n} e_{min}(p_i, scan)^2} \tag{7.1}$$

using n vertices of $S_{expr.inst}$. Closest point pairs (p,p') for which p' belongs to the boundary (including holes) of the face scan are not used in the distance measure.

Several methods for 3D morphable model fitting have been proposed in the literature [5, 18, 65, 104]. These methods consider two transformations, a rigid transformation to align the model with the scan data and a non-rigid deformation to deform the model

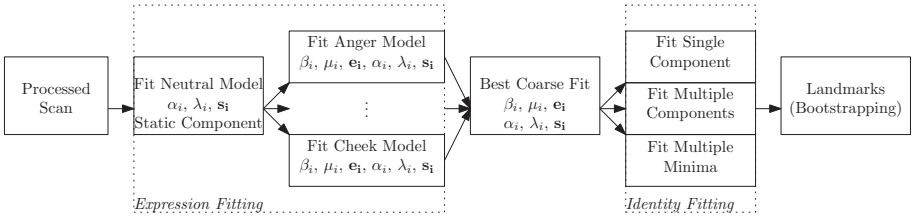


Figure 7.5: Flow chart of the combined identity and expression model fitting.

Figure 7.6: Example faces of the morphable model fitting. From left to right, the processed scan, the static region fit, the selected coarse fit (surprised), the full face refinement, the multiple component refinement, the automatically acquired landmarks, and the neutralized face model.

to the scan data. As described in Section 7.2, our method computes the rigid transformation only once, with the use of a pose normalization method that aligns the scan data with an average nose template. This transformation is kept constant during the fitting process for fast model fitting. The model fitting is separated into an *expression optimization* step to select the best expression model and an *identity optimization* step to find the global minimum (and local minima) in the identity space. The morphable model fitting is shown in Fig. 7.5 as a flow chart and in Fig. 7.6 based on an example face. Both the expression and identity fitting use the two *coefficient selection* algorithms described below.

7.4.1 Coefficient selection

With the face scan aligned to the average nose template, we can compute the closest point pairs between the mean instances $\bar{S}$ and the scan. These closest point correspondences will only be reliable when the scan closely resembles the mean face. To improve on the set of correspondences, the model coefficients of the principal identity vectors and the expression vectors are adjusted iteratively using *eigenspace sampling* (algorithm ESSamp). After a number of iterations, the correspondences of $S_{expr.inst}$ are reliable enough to apply *eigenspace projection* (algorithm ESProj) and to evaluate the fit using d_{rms}. Fig. 7.7 shows a schematic view of the iterative search through (2D) coefficient space. To optimize expressions, algorithms ESSamp and ESProj use β_i, μ_i, $\mathbf{e_i}$ instead.

The *eigenspace sampling* iteratively selects model coefficients, that morphs the face model closer to the scan data. This algorithm simply tries four new coefficients for each sequential eigenvector $\mathbf{s_i}$, and keeps the one that produces the smallest RMS distance (similar to the model fitting algorithm in Chapter 5). By reducing the search space α_{range} in each iteration, the algorithm produces a more accurate fit in each of the iterations. Because the first eigenvectors induce the fitting of global face properties and the last

Algorithm 5 ESSamp(S_i, $scan$)

for $k \leftarrow 1$ **to** k_{max} **do**
  $\alpha_{incr} = \frac{2}{3}\alpha_{range}$ (4 samples in full range)
  for $i \leftarrow i_{min}$ **to** i_{max} **do**
    for $\alpha_i' \leftarrow \alpha_i - \alpha_{range}$ **to** $\alpha_i + \alpha_{range}$ **do**
      update $S_i' = S_i + ((\alpha_i' - \alpha_i)\sqrt{\lambda_i} \cdot \mathbf{s_i})$
      $d_{rms}(S_i', scan)$ smaller $\rightarrow \alpha_{opt} = \alpha_i'$
      $\alpha_i' = \alpha_i' + \alpha_{incr}$
    update $S_i = S_i + ((\alpha_{opt} - \alpha_i)\sqrt{\lambda_i} \cdot \mathbf{s_i})$
  $\alpha_{range} = \frac{5}{4}\alpha_{incr}$ (slight overlap)
return $d_{rms}(S_i, scan)$

Algorithm 6 ESProj(S_i, $scan$)

for $k \leftarrow 1$ **to** k_{max} **do**
  select sets of correspondences S_i and S_{scan}
  compute residual deformation vector $S_i - S_{scan}$
  for $i \leftarrow i_{min}$ **to** i_{max} **do**
    $\alpha_i' = \alpha_i + \frac{1}{\sqrt{\lambda_i}}((S_i - S_{scan})^T \mathbf{s_i})$
  update $S_i = \sum_{i=1}^{m} \alpha_i' \sqrt{\lambda_i} \cdot \mathbf{s_i}$
return $d_{rms}(S_i, scan)$

eigenvectors change local face properties, each iteration follows a global to local fitting scheme. To avoid local minima in face space, we try four new coefficient values in each iteration and we use in following iterations a slightly larger range α_{range} than the latest increment α_{incr}.

The *eigenspace projection* refines the set of correspondences that are selected with the eigenspace sampling method. Before the projection, we have to establish $n' \leq n$ closest point correspondences from instance $S_{expr.inst}$ to the scan data, where each point-pair describes the direction to which the vertex of $S_{expr.inst}$ should move for a tighter fit. The number of correspondences n' is usually smaller than the number of vertices of the morphable face model n, because we use a multi-resolution scheme and not every vertex in the model has a closest compatible point in the scan data. The sets of n' correspondences $S_{expr.inst}$ and S_{scan} are subtracted and in case $n' < n$, the missing correspondences are replaced with zero vectors to retain full correspondence with the morphable model. The residual deformation vector can be projected either into the eigenspace of the expression model or into the eigenspace of identities:

$$\beta_i' = \beta_i + \frac{1}{\sqrt{\mu_i}}((S_{expr.inst} - S_{scan})^T \mathbf{e_i}) \ ,$$

$$\alpha_i' = \alpha_i + \frac{1}{\sqrt{\lambda_i}}((S_{expr.inst} - S_{scan})^T \mathbf{s_i}) \ .$$

Projecting the residual deformation vector onto the eigenvectors of the orthogonal eigenspace is the fastest and easiest way to obtain the least-squares solution for the given set of correspondences (Sect. 7.3.4). Afterwards, a new set of closest point

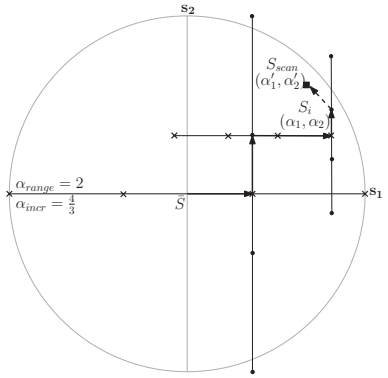


Figure 7.7: Searching the coefficient space using eigenspace sampling (solid arrows) and eigenspace projection (dashed arrow). At each mark, the model instance S_i is updated and correspondence with the $scan$ estimated.

correspondences can be selected and the residual deformation is projected into the eigenspaces again. This process converges to a local optimum within a few iterations (k_{max}). As a result, the model coefficients are refined.

7.4.2 Expression fitting

The main difficulty in model fitting is that neither the expression coefficients nor the identity can be optimized without optimizing the other. When the model is fitted to a new scan with an unknown expression, it makes sense to coarsely estimate the identity based on expression invariant regions and then to select the best expression model and search for its optimal expression parameters.

Starting with the morphable mean face $\bar{S}$, the identity coefficients $\boldsymbol{\alpha}$ are improved using algorithms ESSamp and ESProj based on the static face component (Fig. 7.4). The former algorithm iteratively improves coefficients (i_{min},i_{max}) α_1 up to α_4, α_5 up to α_8, and α_9 up to α_{12}. To cover a large range of facial variety, we use a large range of coefficients α_{range}=2, and k_{max}=4 iterations. The established correspondences are refined with algorithm ESProj to obtain the coarsely fitted face instance S_{coarse}.

To find the expression instance $S_{expr.inst}$ that fits the scan data best, we need to find the optimal combination of identity and expression coefficient vectors $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$. Moreover, to select the appropriate expression model, we need an optimized fit for each of the expression models in combination with S_{coarse}. For that, we select an expression model and three different coefficients $\beta_1 = \{0.0, 0.5, 1.0\}$ for its first expression vector $\mathbf{e_1}$ and apply this deformation to S_{coarse}. Note that this first expression vector causes the largest shape deformation, and that its weight should be positive (Sect. 7.3.7). Starting from each of these instances $S_{expr.inst}$, the coefficients β_2 up to β_6 are refined with algorithm ESSamp using α_{range}=2 and k_{max}=4. Then α_1 to α_4, β_1 to β_6, α_5 to α_8, and α_9 to α_{12} are refined with algorithm ESSamp using α_{range}=$\frac{1}{2}$ and k_{max}=4.

For each expression model and for each coefficient β_1, a combined identity/expression fit is acquired with ESSamp. Each of these coarse fits is then projected onto the eigenspace of expressions and then to the eigenspace of identities to refine the estab-

lished correspondences with all model coefficients in algorithm ESProj (k_{max}=1). The best fit, i.e. the instance with the smallest d_{rms} distance, is selected as the *best coarse fit* as shown in Fig. 7.5. To further refine the expression parameters $\boldsymbol{\beta}$ of the selected fit, the residuals are projected onto the expression space with algorithm ESProj using k_{max}=5 iterations. This gives us the final expression vector S_{expr}.

7.4.3 Identity fitting

After the expression fitting, we have obtained a coarse identity vector that, in combination with the final expression vector, produces a relatively good fit to the scan data. For the purpose of face recognition, each subject needs a unique expression invariant identity vector $\boldsymbol{\alpha}$. Amberg et al. [5] proposed to produce the best possible fit and to use the decoupled identity vector for face recognition. In the previous chapter, a more accurate fit was produced by fitting predefined face components individually. Here, we use both methods and propose a new descriptor.

To produce the best possible fit for the entire face as one component, we use algorithm ESProj to refine the identity coefficients in k_{max}=5 iterations. This gives us the final identity vector S_{id} and its coefficient vector $\boldsymbol{\alpha}$. This *single-component vector* is used as feature vector for the face matching.

For the multiple component method, we define a subset of vertices for the nose, eyes, mouth and remainder regions and project the residual vector for each component to the identity space using ESProj. This gives us an identity vector S_{id} and a coefficient vector $\boldsymbol{\alpha}$ per component. The coefficient vectors are concatenated to produce a single feature vector for the face matching, which we refer to as the *multi-component vector*. The identity vector S_{id} can be used to bootstrap the model directly, or its facial landmarks can be mapped to the scan data and used to model a new set of correspondences as described in Sect. 7.3.

Since there is no guarantee that the fitting process finds the global optimum in the identity space, we propose to search for a number of local optima and concatenate their coefficient vectors $\boldsymbol{\alpha}$. These locations in m_s-dimensional space should form a unique pattern for each subject that we use for face recognition. To find four local minima, we initiate a face instance as the combination of the mean face with the final expression vector ($\bar{S} + S_{expr}$), and adjust its first two coefficients α_1 and α_2 with {-2,2}. Then algorithm ESSamp is applied using a relatively small α_{range}=$\frac{1}{2}$, and k_{max}=4, to iteratively refine coefficients α_1 up to α_{12} as we did before. Each of the four coarsely fitted $S_{expr.inst}$ is then refined using algorithm ESProj in k_{max}=5 iterations (also without $\boldsymbol{\beta}$). This gives us an identity vector S_{id} and its coefficient vector $\boldsymbol{\alpha}$ for each of the four initializations. The coefficient vectors are concatenated to produce a single feature vector for the face matching, which we refer to as *multi-minima vector*. We could have used more than four local minima, but this would result in a larger feature vector and thus a slower face recognition system.

7.4.4 Implementation

The difficulty in model fitting is the high dimensional face space in which each location represents a detailed face mesh. Exhaustive search for the global optimum is an

intractable task and scan deficiencies and expressions cause local minima in the face space. During the expression model fitting (Sect. 7.4.2), we iteratively fit each expression model using three different values β_1 to be able to select the proper intensity of the expression. Algorithm ESSamp, iteratively improves a small set of coefficients (e.g. α_1 up to α_4) at a time, which allows the algorithm to recover from an incorrect choice for a coefficient at an early stage. Refining either one coefficient iteratively at a time or all coefficients per iteration are two variants that performed less well. For the selection of the best expression model, we fit each model in combination with no more than twelve coefficients of the identity model. With all identity coefficients, several expression models might produce a tight fit to the scan data, but with limited resources the distance d_{rms} is more reliable. To stay within the statistical space of known faces, each model coefficient is kept within range α_i, β_i=[-2,2].

Mpiperis et al. [74] also experienced that especially the mouth region causes local minima in the face space that require additional effort to avoid. One common local minimum is a model instance with a closed mouth while the face scan shows an open mouth. Instead of a dedicated mouth detection algorithm that Mpiperis proposed, we allow the model's vertices in the mouth area to pair up with boundary points of the scan data. In case the face scan has an open mouth and the face model has not, these point pairs are automatically penalized by the distance measure d_{rms}.

For the ESProj algorithm we use closest point-to-point correspondences from the model to the scan data and from the scan data to the model. This results in a higher fitting accuracy [74]. With the use of a kD-tree the closest point correspondences can be found efficiently. For high efficiency, we compute in algorithm ESSamp only the correspondences from model to scan, because the model and its kD-tree change in each iteration. For the eigenspace sampling we consider two closest points pairs to correspond if the distance between them is smaller than 50 mm, for the eigenspace projection we use correspondences closer than 10 mm. We stop traversing a kD-tree, when this criterion can no longer be met.

For time efficiency, algorithm ESSamp is applied using the low resolution face model of n=932 vertices. Algorithm ESProj is applied to a coarse fit using n=3624 vertices and to the final expression and identity vectors using n=14288 vertices. In the end, the time to process a raw scan requires ca. 3 seconds for the face segmentation, ca. 10 seconds to fit all expression models, less than 1 second to improve the coarse identity fit, and ca. 4 seconds to find four local minima on a Pentium IV 2.8 GHz. Note that the fitting of each expression model as well as the search for multiple minima can be done in parallel to further speed up the process.

7.5 Face matching

After the morphable model is fitted to each of the face scans, we have obtained three feature vectors of model coefficients, namely, the single-component vector, the multi-component vector, and the multi-minima vector. For the face matching we use each of these vectors individually to do 3D face recognition. To determine the similarity of faces with these coefficient vectors, we use the L_1 distance between the normalized coefficient vectors. So, the matching of two faces requires only the comparison of either m_s or $4m_s$

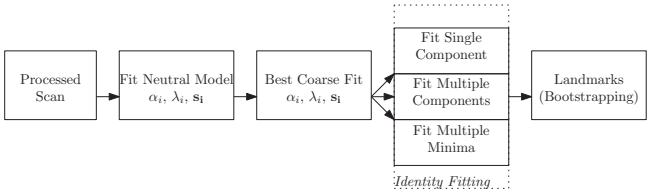


Figure 7.8: Flow chart of the neutral model fitting.

dataset	fit	min	max	mean	sd
UND	SC neutral	0.64	2.34	0.75	0.13
	SC	0.63	2.32	0.75	0.12
	MC	0.62	2.27	0.72	0.12
GAVAB	SC neutral	0.70	1.98	0.89	0.16
	SC	0.70	1.53	0.85	0.10
	MC	0.67	1.36	0.78	0.08
BU-3DFE	SC neutral	0.60	2.32	0.79	0.14
	SC	0.60	1.12	0.71	0.06
	MC	0.59	0.99	0.67	0.05
FRGC v.2	SC neutral	0.64	3.55	0.81	0.19
	SC	0.65	3.39	0.79	0.17
	MC	0.63	3.40	0.75	0.16

Table 7.1: RMS errors (mm) of output models to input scans.

float values, which is extremely time-efficient. For each query, we compute its similarity to all other models in the training set, generating a ranked list of face models sorted on decreasing similarity values in the process. Based on these ranked lists, we compute the recognition rate (RR), the mean average precision (MAP) and the verification rate at 0.1% false acceptance rate (VR@0.1%FAR), as we did in Chapter 5.

7.6 Results

7.6.1 Morphable model fitting

In this section we evaluate the accuracy of the final expression instances $S_{expr.inst}$ both quantitatively and qualitatively. After the identity fitting in Sect. 7.4.3 we have two final face instances, a single component fit (SC) and a multiple component fit (MC). We can evaluate the fits qualitatively by looking at the more frequent surface interpenetration of the fitted model and face scan (Fig. 7.9), which means a tighter fit. Note that our fitting method is robust to missing data and even creates accurate face instances when half of the face is missing. A quantitative evaluation can be done by comparing the residual d_{rms} distances. Table 7.1 shows a decrease in residual error for the multiple component fitting.

To prove that the expression modeling improves the fitting process, we also fitted the neutral model without the additional expression models according to the flow chart in Fig. 7.8. The fitting results are shown in the 2nd and 3rd column of Fig. 7.9, which show a consistent failure in case of expression scans. The higher residual d_{rms} distances are listed in Table 7.1 (SC neutral).

Figure 7.9: Model fitting to processed scans (1st column) using the neutral model only (2nd and 3rd column), a single component (4th and 5th column) and multiple components (6st and 7th column). The last column shows the neutralized face instances $S_{expr.inst} - S_{expr}$ of the 6st column.

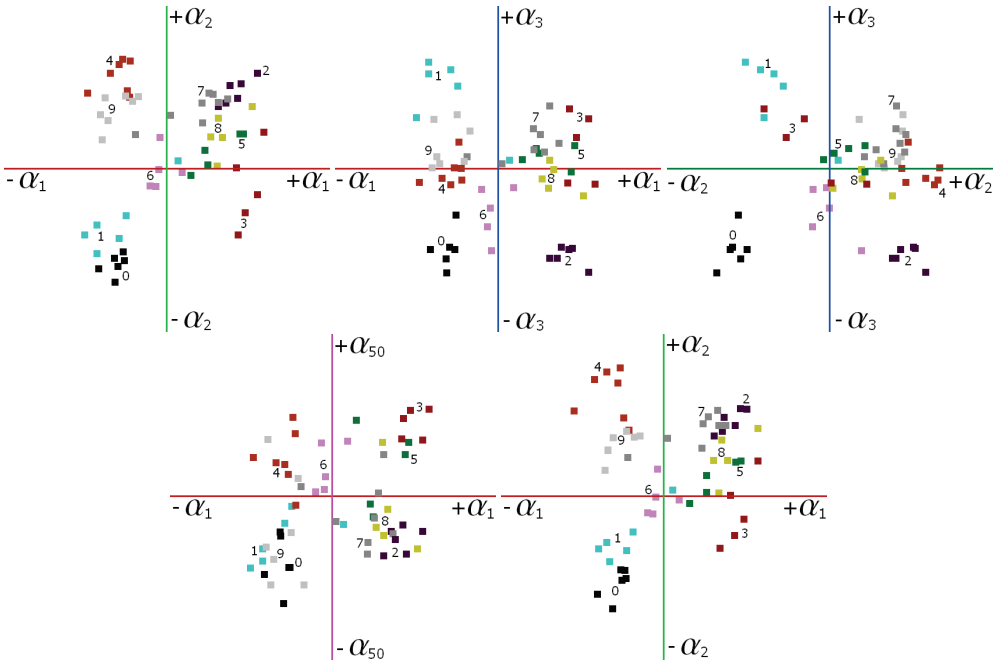


Figure 7.10: Identity clustering in coefficient space. Each coefficient aids the classification of subjects (shown in color). The last graph shows the projection onto α_1 and α_2 after normalizing the length of all vectors $\boldsymbol{\alpha}$.

7.6.2 Face matching

After the fitting process, the expression deformation S_{expr} can be subtracted to neutralize the expression, $S_{expr.inst} - S_{expr} = \bar{S} + S_{id}$. The identity coefficients $\boldsymbol{\alpha}$ that model the identity deformation S_{id} are used for the face matching as explained in Sect. 7.5. For perfect retrieval results, the acquired coefficient vector $\boldsymbol{\alpha}$ for each scan of the same subject should point to the same unique position in the m_s-dimensional coefficient space. To get an impression of the identity clustering in coefficient space, we show in Fig. 7.10 ten random subjects of the UND dataset having more than four scans. The projected coefficient vectors are those acquired with the single component fit. These graphs show that not only the principal eigenvectors are useful to distinguish between different subjects, but that even the fiftieth coefficient contributes to the clustering of subjects.

To evaluate the use of the acquired coefficient vectors $\boldsymbol{\alpha}$ for our single component fit (SC), our multiple component fit (MC), and our multiple minima fit (MM), we computed the recognition rates (RR), the mean average precisions (MAP) and the verification rates at 0.1% false acceptance rate (VR@0.1%FAR) for the subjects in the UND, GAVAB, BU-3DFE, and FRGC v.2 datasets. The results based on the L_1 distance between the normalized coefficient vectors are shown in Table 7.2. Results show that (1) our method can be applied with considerable success to a large range of datasets, (2) the use of expression models is essential for high performance, (3) the use of multiple components (MC) improves on the single component (SC) results, (4) in case of scan

dataset	fit	RR	MAP	VR@0.1%FAR
UND	SC neutral	0.99	0.99	0.99
	SC	0.99	0.98	0.97
	MC	0.99	0.98	0.98
	MM	0.99	0.99	0.97
GAVAB	SC neutral	0.94	0.80	0.53
	SC	0.97	0.90	0.73
	MC	0.97	0.92	0.77
	MM	0.98	0.93	0.80
BU-3DFE	SC neutral	0.98	0.59	0.29
	SC	1.00	0.91	0.80
	MC	1.00	0.92	0.82
	MM	1.00	0.91	0.75
FRGC v.2	SC neutral	0.91	0.80	0.73
	SC	0.97	0.89	0.84
	MC	0.97	0.91	0.87
	MM	0.97	0.91	0.82

Table 7.2: All to all face matching.

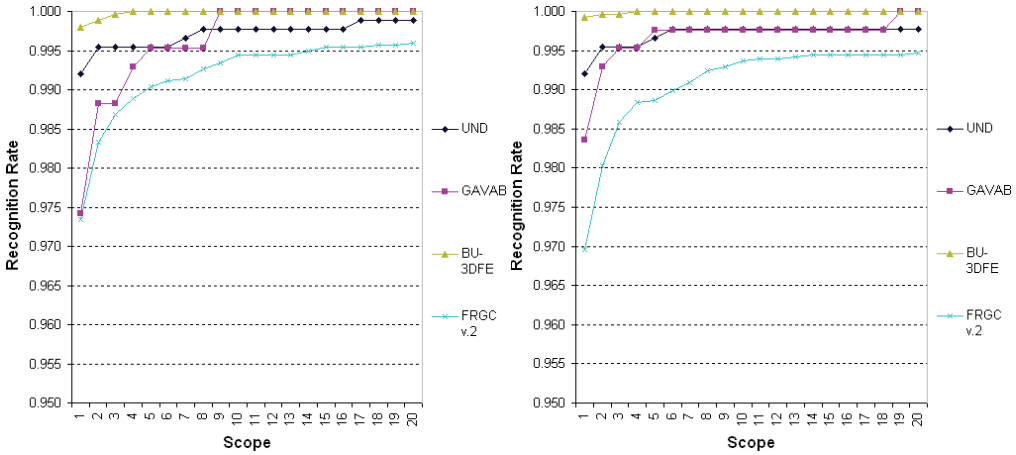


Figure 7.11: The cumulative match characteristic curves for the multiple component (MC) results (left) and mulitple minima (MM) results (right) on the four datasets.

data with lower quality, as in the GAVAB dataset, the multiple minima (MM) approach can improve the system's performance. In Fig. 7.11, we show the *cumulative match characteristic* (CMC) curves for both the multiple component and multiple minima results on the four datasets.

Comparison UND. Several authors report recognition rates for the UND dataset. Blanz et al. [18] achieved a 96% RR for 150 queries in a set of 150 faces. Samir et al. [90] reported 90.4% RR for 270 queries in a set of 470 faces. Mian et al. [67] reported 86.4% RR for 277 queries in a set of 277 scans. Amberg et al. [5] used all 953 scans and achieved 100% RR.

Comparison GAVAB. The GAVAB dataset has been used in the Shape Retrieval Contest 2008 [113] to compare 3D face retrieval methods. Results of different approaches vary between 60% and 100% RR. Recently, Amberg et al. [5] achieved a recognition rate of 99.7% on this dataset. They use a morphable head model that covers the neck and

ears as well, features that may aid the person identification.

Comparison BU-3DFE. Mpiperis et al. [74] performed experiments on the BU-3DFE dataset. They used two methods for the expression invariant face matching, a symmetric bilinear model and geodesic polar coordinates, with respectively 86% and 84% RR. The authors report that their recognition results outperform Bronstein et al.'s [22] canonical image representation.

Comparison FRGC v.2. Lu et al. [65] applied their expression-specific deformation models to only 100 subjects of the FRGC v.2 and report 92% recognition rate and 0.7 VR@0.1%FAR, which is considerably lower than the results with our expression-specific deformation models. Moreover, we do not need a neutral face scan for the deformation nor the computational expensive ICP algorithm for the matching. Other 3D shape based methods that report the VR@0.1%FAR for the all-to-all face matching experiment are, Maurer et al. [66] with 0.78 VR, Mian et al. [69] with 0.87 VR, Cook et al. [33] with 0.92 VR, and Faltemier et al. [42] with 0.93 VR. Most of them use the computational expensive ICP algorithm during face matching and simply neglect data in regions with expressions. Kakadiaris et al. [57] reported a 97% RR and 0.97 VR@0.1%FAR for slightly different experiments.

7.7 Conclusion

We presented a complete 3D expression invariant face recognition system. Starting from raw scan data we applied our face pose normalization method that correctly normalized the pose of 8021 out of 8023 face scans from five publicly available datasets. The two failures were due to a missing nose. Starting from the pose normalized faces scans, we proposed an easy to implement method to semi-automatically build identity and expression models from the BU-3DFE dataset. With the presented model fitting algorithm, we can automatically establish full correspondence with new scan data and bootstrap the identity and expression models. Accordingly, we bootstrap the identity model with scans of the USF dataset and established full correspondence with 40 FRGC v.2 scans to build a deformation model that inflates the cheeks of a neutral face instance. Statistical face models provide strong shape priors that allow for the robust handling of noise and holes. Results show that our morphable face model method can be effectively used for landmark extraction, bootstrapping, face completion, and face matching, which is an advantage over other methods.

The model fitting method that we presented, coarsely fits the identity model in combination with each of the expression models and keeps the overall best fit. Because separate models are used for the identity and expression deformations, the expression can be easily neutralized and the separate identity coefficients can be used for expression invariant face matching. Three identity coefficient vectors were acquired for the face matching, one based on the face as a single component, one for multiple face components, and one for multiple local minima. Compared to the literature, all our coefficient vectors proved to perform very well on the publicly available datasets. The use of multiple face components is an easy way to improve on the face matching performance, and in case of low quality scans the multiple minima vector can be a good alternative. Our system effectively recognizes faces with different expressions from a wide range of data

sets, and is also very time-efficient: After the model is fitted to each scan in at most 17 seconds (linear to the number of scans), our face matching (quadratic to the number of scans) requires only the comparison of either 80 or 360 float values in our experiments. Therefore, our method can be very well applied to authentication scenarios (e.g. airport check-ins) as well as face retrieval scenarios (e.g. searching criminal records).

Chapter 8

Conclusions and future research

Throughout this thesis we have used 3D laser range scans for the acquisition, reconstruction, and analysis of 3D shapes. 3D laser range scanning has proven to be a fast and effective way to capture the surface of an object in a computer. Thousands of depth measurements represent a part of the surface geometry as a cloud of 3D points. Geometric algorithms have been developed to turn such 3D point sets into manageable shapes and shape representations for end users or other algorithms. We have used 3D laser range scans to evaluate acquisition and reconstruction systems and algorithms, to fully automate the object reconstruction, to find discriminative face features, for automatic landmark extraction, for face identification with and without expressions, and for the statistical modeling of faces.

In Chapter 2, we explored systems and algorithms for the acquisition and reconstruction pipeline from object to 3D model. Two laser range scanners were evaluated for their accuracy. The most accurate scanner was used to reconstruct real world objects as reference models. Range scans from the other scanner were also used for the object reconstruction, but this time to compare each system's output to the reference model, in order to quantify the system's accuracy. Systems for the fine alignment, merging, and hole filling were evaluated. Although it is difficult to directly compare algorithms for the reconstruction of a 3D model, it is important to understand the consequences of the heuristics used in these algorithms. Variants of the ICP algorithm, for instance, produce highly accurate alignments of meshes, but require a good initial fit. Volumetric merge methods average out noise while carefully blending surface meshes, but the final surface resolution depends on the number of voxels and thus on the (tight) bounding box of the object and the available memory.

We used short range laser scanners to evaluate the reconstruction accuracy of relatively small objects. Scanning patches of a large object with a high resolution short range scanner involves the 3D mosaicing of range scans [62, 81] for which the optimal pipeline may differ due to, for instance, higher computational costs and memory constraints. Future work could focus on the reconstruction speed for real-time user-feedback during the 3D laser range scan acquisition, as Rusinkiewicz et al. [87] did for structured light scanning under the assumption of small object rotations. Such feedback can be used to discover under-sampled areas on the object that would require hole filling techniques to complete the 3D model, techniques that are dependent on the input resolution and the complexity of holes as we have shown in Chapter 2.

As we pointed out in Chapter 2, the main bottleneck in the acquisition and reconstruction pipeline is the coarse alignment of the surface scans, which requires intensive user-interaction. In Chapter 3, we presented an algorithm that automatically aligns tens of the range scans swiftly. Typical for the acquisition of an object that completely fits the scanner's field of view is that users randomly capture different object views, as opposed to systematically scanning the surface of large objects. When the order of the acquired views is unknown, we cannot assume two sequentially scanned meshes to overlap. In this case the multiview alignment of unordered range scans needs to be solved. The algorithm that we presented performs both the coarse and fine alignment of such unordered scans. For highly accurate alignments of noisy scan data, an unordered set of N meshes requires $O(N^2)$ mesh-to-mesh comparisons, because the position of each mesh has to be refined with respect to its overlapping meshes. In our method, we lower the computation costs for the N^2 mesh-to-mesh comparisons, and ensure $O(N)$ mesh-to-mesh comparisons for the computational expensive parts of our algorithm. Our method is effective, time-efficient, and has a competitive accuracy compared to the multiview ICP algorithm. Typical for the scanning of large objects is to sequentially capture parts of the object (e.g. from left-to-right and bottom-to-top) with sufficient overlap. The alignment of these ordered patches can be done by exploiting the scanning sequence in combination with a pair-wise alignment algorithm [81]. After this 3D mosaicing, a single view of the entire object is again acquired and the process can be repeated for a new view. Our algorithm can be directly applied to bring the combined scans into alignment. An open issue is to automatically find the orientation and position of patches in a larger 3D mosaic, without a priori knowledge. A research topic that is closely related to the automatic alignment of depth maps acquired from airplanes.

In Chapter 4, we use range scans for 3D face identification. A complete 3D face recognition pipeline was presented that automatically detects, segments, and improves the face surface, and matches 3D faces with the use of profile and contour curves. To select effective combinations of curves as a compact representation of the face for the recognition, a 3D face matching framework was designed. This framework extracts contour curves with different properties and evaluates these curves for their ability to identify persons. To cope with changes in face poses, we presented a 3D template based algorithm that selects the location that locally resembles and globally corresponds to the nose tip. To deal with facial expressions, we automatically select a subset of best matching profile curves. Results show that the number of compared face curves can be greatly reduced without sacrificing facial uniqueness. This is interesting because such a compact representation of a face with a few 3D curves protects the privacy better than storing the full 3D face scan. Selected sets of face curves after training outperformed several existing 3D face recognition methods both in recognition rates and time-efficiency. Our ICP-based method achieves higher recognition rates, but at the expense of higher computation costs.

In Chapter 5, we studied the use of statistical model fitting for the task of 3D face identification. We introduced an algorithm to automatically fit an existing 3D morphable face model to 3D face scans for their identification. The 3D morphable face model is a statistical face model built from example faces and linearly interpolates between these faces to construct new ones. By automatically selecting new model coefficients, our algorithm deforms the 3D face model along the principal axes of data

variance, which statistically changes for instance the size of the face, the position of the eyes, or the length of the nose. We used the morphable face model to acquire a face instance similar to the scan data without holes nor noise and to automatically extract anthropometric landmarks. For the recognition of faces we used the extracted landmarks, contour curves, and model coefficients. Results show the superior performance of the selected model coefficients. We use multiple face components for more accurate fitting and normalize the coefficient vector for higher recognition rates. Unlike other model fitting algorithms, ours needs no manual initialization and achieves higher fitting accuracies. Two clear limitations of this existing morphable model are its limited number of example faces and its limited variation in facial expressions.

In Chapter 6, we presented a new bootstrapping algorithm to automatically enhance the 3D morphable face model with new face data. For a 3D face recognition system based on model coefficients it is important that the statistics of many realistic faces are captured in the morphable model. With this bootstrapping algorithm, we are able to successfully update an initial face model, which we use to produce more accurate fits to new scan data. Compared to previous work, our algorithm is fully automatic, reuses initial face statistics, checks for redundancy, and retains the full correspondence even in case of noisy scan data with holes.

In Chapter 7, we semi-automatically constructed a new identity model and several expression models. By manually selecting anthropometric landmarks as the initial set of correspondences, we ensure that our models have reliable landmarks locations. Dense correspondences are established with the use of a subdivision surface as a layer over a cylindrical depth map. To automatically bootstrap the models, we only need to locate the anthropometric landmarks with our model fitting approach and repeat the same subdivision scheme. We bootstrapped the models and performed expression invariant face recognition by incorporating the expression-specific deformation models in our face modeling approach. In a coarse to fine fitting scheme, the identity and expression coefficients of this model are optimized such that the final face instance accurately fits the 3D face scan with unknown expression. Finding the global optimum in identity space can be difficult, especially when the expression has to be optimized at the same time. Therefore, we investigated the use of multiple local minima in the identity space for the recognition of 3D faces.

Throughout this thesis we presented several contributions to 3D face recognition. In Chapter 4, a successful face pose normalization method was presented, which we further improved in Chapter 7. In approximately three seconds, this method locates the tip of the nose, normalizes the pose of the face, aligns the scan data to an average nose template, and segments the face. This method successfully processed 8021 out of 8023 3D face scans from the publicly available UND, GAVAB, BU-3DFE, and FRGC v.2 datasets. For the two failures, the scan acquisition failed to capture data of the nose. The automatic pose normalization is an important step to automate 3D face matching for authentication scenarios (e.g. airport check-ins) and 3D face retrieval scenarios (e.g. searching criminal records).

The cooperative authentication scenario is a typical application domain for which facial contour curves (Chapter 4) would suffice. To pass a passport control at an airport, people are willing to cooperate to avoid any delay. Automating this process by introducing 3D face verification could even speed-up the control. To get or renew a passport, an

up-to-date 2D photo is required with a neutral expression and a predefined pose, with the same effort a 3D scan of the face (or even the full body) can be made and stored on the passport chip. For higher privacy protection the selected contour curves could be stored instead.

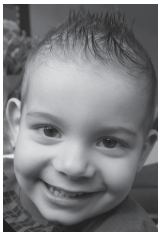
Passport control is a one-to-one matching scenario in which a subject is compared to the person on the document. An entrance security that allows only employees to enter can be designed as a cooperative one-to-many identification scenario. In this scenario, a database of 3D face scans of the employees is constructed and a person that highly resembles one of the database records is allowed to enter. The enhanced neutral 3D morphable face model and the efficient algorithms we presented for the model fitting and face matching (Chapter 5 and 6) can be effectively used in this scenario.

In a non-cooperative scenario a subject may try to avoid the correct identification. The non-rigid face deformation imposed by a facial expression causes many 3D face recognition systems to fail in identifying subjects. The face modeling algorithm presented in Chapter 7 is able to perform expression invariant face recognition, and even to remove an expression from and add an expression to a face instance.

In the past, textured 3D morphable face models have been used to extract neutral 3D faces from single images [19] and 3D expression faces from multiple images [80]. Recently, this concept has been extended to extract 3D expression faces from single images [115]. For reliable results, these methods require manually selected landmarks to instantiate the face model. Automating this process and evaluating the generated face quality using 3D laser range data could boost the 2D/3D face modeling and face recognition with statistical face models.

Up to date, statistical face modeling has been mainly based on 2D and 3D face data of adults, and mostly for a single time frame. The statistical modeling of the aging process in 3D has been under-exposed as well as age invariant face recognition [78]. Especially the face modeling of children has received little attention. A 3D statistical aging model can be used to interpolate and maybe even to predict one's appearance over time, which can be of practical usage to find a missing person and for forensic research. For the images shown below, the appearance prediction in both 2D and 3D would be simply entertaining and, of course, to provide a glimpse of the future.

(a) 2006



(b) 2009

(c) 2009

(d) 2012

Bibliography

[1] AIM@SHAPE. Shape Repository at `http://www.aimatshape.net/resources`, Nov 2006.

[2] L. Akarun, B. Gökberk, and A. Salah. 3D Face Recognition for Biometric Applications. In *EUSIPCO*, 2005.

[3] F. R. Al-Osaimi, M. Bennamoun, and A. Mian. Integration of Local and Global Geometrical Cues for 3D Face Recognition. *Pattern Recognition*, 41(3):1030–1040, 2008.

[4] P. Alliez, D. Cohen-Steiner, Y. Tong, and M. Desbrun. Voronoi-Based Variational Reconstruction of Unoriented Point Sets. In *Symposium on Geometry processing*, pages 39–48, 2007.

[5] B. Amberg, R. Knothe, and T. Vetter. Expression Invariant Face Recognition with a Morphable Model. In *Automatic Face and Gesture Recognition*, 2008.

[6] B. Amberg, S. Romdhani, and T. Vetter. Optimal Step Nonrigid ICP Algorithms for Surface Registration. In *CVPR*, pages 1–8, 2007.

[7] N. Amenta and Y.-J. Kil. Defining Point-set Surfaces. In *Trans. on Graphics*, volume 23, pages 264–270, 2004.

[8] B. B. Amor, K. Ouji, M. Ardabilian, and L. Chen. 3D Face Recognition by ICP-based Shape Matching. In *ICMI*, 2005.

[9] G. Barequet and M. Sharir. Filling Gaps in the Boundary of a Polyhedron. *Computer Aided Geometric Design*, 12(2):207–229, 1995.

[10] C. Basso, P. Paysan, and T. Vetter. Registration of Expressions Data using a 3D Morphable Model. In *Automatic Face and Gesture Recognition*, pages 205–210, 2006.

[11] C. Basso and A. Verri. Fitting 3D Morphable Models Using Implicit Representations. *Journal of Virtual Reality and Broadcasting*, 4(18), 2007.

[12] R. Bergevin, M. Soucy, H. Gagnon, and D. Laurendeau. Towards a General Multiview Registration Technique. *PAMI*, 18(5):540–547, 1996.

[13] F. Bernardini, J. Mittleman, H. Rushmeier, C. Silva, and G. Taubin. The Ball-Pivoting Algorithm for Surface Reconstruction. *Trans. on Visualization and Computer Graphics*, 5(4):349–359, 1999.

[14] F. Bernardini and H. Rushmeier. The 3D Model Acquisition Pipeline. *Computer Graphics Forum*, 21(2):149–172, 2002.

[15] S. Berretti, A. Del Bimbo, P. Pala, and F. Silva Mata. Face Recognition by Matching 2D and 3D Geodesic Distances. In *MCAM07*, pages 444–453, 2007.

[16] P. J. Besl and N. D. McKay. A Method for Registration of 3D Shapes. *PAMI*, 14(2):239–256, 1992.

[17] Binghamton University 3D Facial Expression Database. BU-3DFE.

[18] V. Blanz, K. Scherbaum, and H.-P. Seidel. Fitting a Morphable Model to 3D Scans of Faces. In *ICCV*, pages 1–8, 2007.

[19] V. Blanz and T. Vetter. A Morphable Model for the Synthesis of 3D Faces. In *SIGGRAPH*, pages 187–194, 1999.

[20] V. Blanz and T. Vetter. Face Recognition Based on Fitting a 3D Morphable Model. *PAMI*, 25(9):1063–1074, 2003.

[21] K. W. Bowyer, K. Chang, and P. Flynn. A Survey of Approaches and Challenges in 3D and Multi-modal 3D + 2D Face Recognition. *CVIU*, 101(1):1–15, 2006.

[22] A. M. Bronstein, M. M. Bronstein, and R. Kimmel. Three-Dimensional Face Recognition. *IJCV*, 64(1):5–30, 2005.

[23] CAESAR-survey. The Civilian American and European Surface Anthropometry Resource at `http://store.sae.org/caesar`, Dec 2008.

[24] R. J. Campbell and P. J. Flynn. A Survey of Free-form Object Representation and Recognition Techniques. *CVIU*, 81(2):166–210, 2001.

[25] J. C. Carr, R. K. Beatson, J. B. Cherrie, T. J. Mitchell, W. R. Fright, B. C. McCallum, and T. R. Evans. Reconstruction and Representation of 3D Objects with Radial Basis Functions. In *SIGGRAPH*, pages 67–76, 2001.

[26] K. I. Chang, K. W. Bowyer, and P. J. Flynn. An Evaluation of Multimodal 2D+3D Face Biometrics. *PAMI*, 27(4):619–624, 2005.

[27] K. I. Chang, K. W. Bowyer, and P. J. Flynn. Multiple Nose Region Matching for 3D Face Recognition under Varying Facial Expression. *PAMI*, 28(10):1695–1700, 2006.

[28] C.-S. Chen, Y.-P. Hung, and J.-B. Cheng. RANSAC-based DARCES: a New Approach to Fast Automatic Registration of Partially Overlapping Range Images. *PAMI*, 21(11):1229–1234, 1999.

[29] Y. Chen and G. Medioni. Object Modelling by Registration of Multiple Range Images. *Image and Vision Computing*, 10(3):145–155, 1992.

[30] J. Cheng and H. Don. A Graph Matching Approach to 3-D Point Correspondences. *Int. Journal of Pattern Recognition and Artificial Intelligence*, 5(3):399–412, 1991.

[31] C. S. Chua and R. Jarvis. Point Signatures: A New Representation for 3D Object Recognition. *IJCV*, 25(1):63–85, 1997.

[32] P. Cignoni, C. Rocchini, and R. Scopigno. Metro: Measuring Error on Simplified Surfaces. *Computer Graphics Forum*, 17:167–174, 1998.

[33] J. Cook, V. Chandran, and C. Fookes. 3D Face Recognition using Log-Gabor Templates. In *BMVC*, pages 83–93, 2006.

[34] J. Cook, V. Chandran, S. Sridharan, and C. Fookes. Face Recognition from 3D Data using Iterative Closest Point Algorithm and Gaussian Mixture Models. In *3DPVT*, pages 502–509, 2004.

[35] T. Cootes, G. Edwards, and C. Taylor. Active Appearance Models. *PAMI*, 23(6):681–685, 2001.

[36] S. Cunnington and A. J. Stoddart. N-view Point Set Registration: a Comparison. In *BMVC*, pages 234–244, 1999.

[37] B. Curless and M. Levoy. A Volumetric Method for Building Complex Models from Range Images. In *SIGGRAPH*, volume 30, pages 303–312, 1996.

[38] J. Davis and X. Chen. A Laser Range Scanner Designed for Minimum Calibration Complexity. *3DIM*, 00:91–99, 2001.

[39] J. Davis, S. R. Marschner, M. Garr, and M. Levoy. Filling Holes in Complex Surfaces using Volumetric Diffusion. In *3DPVT*, pages 428–861, 2002.

[40] T. K. Dey and S. Goswami. Provable Surface Reconstruction from Noisy Samples. In *Symposium on Computational Geometry*, pages 330–339, 2004.

[41] D. W. Eggert, A. W. Fitzgibbon, and R. B. Fisher. Simultaneous Registration of Multiple Range Views for use in Reverse Engineering of CAD Models. *CVIU*, 69(3):253–272, 1998.

[42] T. Faltemier, K. Bowyer, and P. Flynn. A Region Ensemble for 3-D Face Recognition. *Trans. on Information Forensics and Security*, 1(3):62–73, 2008.

[43] V. Fiorin, P. Cignoni, and R. Scopigno. Out-of-core MLS Reconstruction. In *Computer Graphics and Imaging*, pages 27–34, 2007.

[44] P. Fitzpatrick. *Euclid's Elements of Geometry*. Richard Fitzpatrick, 2008.

[45] M. Garland and P. S. Heckbert. Surface Simplification using Quadric Error Metrics. In *SIGGRAPH*, pages 209–216, 1997.

[46] B. Gökberk, M. O. Irfanoglu, and L. Akarun. 3D Shape-based Face Representation and Feature Extraction for Face Recognition. *Image and Vision Computing*, 24(8):857–869, 2006.

[47] J. Gwilt. *The Architecture of Marcus Vitruvius Pollio: In Ten Books*. Lockwood, 1874.

[48] K. Higuchi, M. Hebert, and K. Ikeuchi. Building 3-D Models from Unregistered Range Images. *CVGIP-GMIP*, 57(4):315–333, 1995.

[49] X. Huang, N. Paragios, and D. N. Metaxas. Shape Registration in Implicit Spaces Using Information Theory and Free Form Deformations. *PAMI*, 28(8):1303–1318, 2006.

[50] D. Huber and M. Hebert. 3-D Modeling Using a Statistical Sensor Model and Stochastic Search. In *CVPR*, pages 858–865, 2003.

[51] A. Imhausen. Calculating the Daily Bread: Rations in Theory and Practice. *Historia Mathematica*, 30(1):3–16, 2003.

[52] INUS Technology. RapidForm 2004 PP2 at `http://www.rapidform.com`, Nov 2006.

[53] ISTI-CNR Visual Computing Laboratory. MeshAlign, MeshLab, MeshMerge, OM and Metro at `http://vcg.isti.cnr.it`, June 2009.

[54] A. E. Johnson. *Spin Images: A Representation for 3-D Surface Matching*. PhD thesis, Carnegie Mellon University, Pittsburgh, 1997.

[55] T. Ju. Robust Repair of Polygonal Models. *Trans. on Graphics*, 23(3):888–895, 2004.

[56] I. Kakadiaris, G. Passalis, G. Toderici, N. Murtuza, and T. Theoharis. 3D Face Recognition. In *BMVC*, pages 869–878, 2006.

[57] I. A. Kakadiaris, G. Passalis, G. Toderici, M. N. Murtuza, Y. Lu, N. Karampatziakis, and T. Theoharis. Three-Dimensional Face Recognition in the Presence of Facial Expressions: An Annotated Deformable Model Approach. *PAMI*, 29(4):640–649, 2007.

[58] Y. J. Kil, N. Amenta, and B. Mederos. Laser Scanner Super-resolution. In *Point Based Graphics*, 2006.

[59] R. Kimmel and J. Sethian. Computing Geodesic Paths on Manifolds. In *Proc. of National Academy of Sciences*, volume 95(15), pages 8431–8435, 1998.

[60] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by Simulated Annealing. *Science, Number 4598, 13 May 1983*, 220, 4598:671–680, 1983.

[61] D. C. Lay. *Linear Algebra and Its Applications*. Addison-Wesley, 1997.

[62] M. Levoy, K. Pulli, B. Curless, S. Rusinkiewicz, D. Koller, L. Pereira, M. Ginzton, S. Anderson, J. Davis, J. Ginsberg, J. Shade, and D. Fulk. The Digital Michelangelo Project: 3D Scanning of Large Statues. In *SIGGRAPH*, pages 131–144, 2000.

[63] C. Li, A. Barreto, J. Zhai, and C. Chin. Exploring Face Recognition by Combining 3D Profiles and Contours. In *SoutheastCon*, pages 576–579, 2005.

[64] W. E. Lorensen and H. Cline. Marching Cubes: a High Resolution 3D Surface Construction Algorithm. In *SIGGRAPH*, volume 21, pages 303–312, 1987.

[65] X. Lu and A. Jain. Deformation Modeling for Robust 3D Face Matching. *PAMI*, 30(8):1346–1356, 2008.

[66] T. Maurer, D. Guigonis, I. Maslov, B. Pesenti, A. Tsaregorodtsev, D. West, and G. Medioni. Performance of Geometrix ActiveID^{TM} 3D Face Recognition Engine on the FRGC Data. *CVPR Workshop*, 0:154–160, 2005.

[67] A. S. Mian, M. Bennamoun, and R. Owens. Matching Tensors for Pose Invariant Automatic 3D Face Recognition. In *A3DISS*, 2005.

[68] A. S. Mian, M. Bennamoun, and R. Owens. Automatic 3D Face Detection Normalization and Recognition. In *3DPVT*, pages 735–742, 2006.

[69] A. S. Mian, M. Bennamoun, and R. Owens. An Efficient Multimodal 2D-3D Hybrid Approach to Automatic Face Recognition. *PAMI*, 29(11):1927–1943, 2007.

[70] A. S. Mian, M. Bennamoun, and R. A. Owens. From Unordered Range Images to 3D Models: A Fully Automatic Multiview Correspondence Algorithm. In *Theory and Practice of Computer Graphics*, pages 162–166. Computer Society Press, 2004.

[71] A. S. Mian, M. Bennamoun, and R. A. Owens. 3D Model-based Object Recognition and Segmentation in Cluttered Scenes. *PAMI*, 2006.

[72] A. S. Mian, M. Bennamoun, and R. A. Owens. A Novel Representation and Feature Matching Algorithm for Automatic Pairwise Registration of Range Images. *IJCV*, 66(1):19–40, 2006.

[73] A. Moreno and A. Sanchez. GavabDB: a 3D Face Database. In *Workshop on Biometrics on the Internet COST275, Vigo*, pages 77–85, 2004.

[74] I. Mpiperis, S. Malassiotis, and M. G. Strintzis. Bilinear Models for 3-D Face and Facial Expression Recognition. *Trans. on Information Forensics and Security*, 3(3):498–511, 2008.

[75] P. Nair and A. Cavallaro. SHREC'08 Entry: Registration and Retrieval of 3D Faces using a Point Distribution Model. In *Shape Modeling and Applications*, pages 257–258, 2008.

[76] P. J. Neugebauer. Reconstruction of Real-World Objects via Simultaneous Registration and Robust Combination of Multiple Range Images. *Int. Journal of Shape Modeling*, 3(1-2):71–90, 1997.

[77] J. Novatnack and K. Nishino. Scale-Dependent/Invariant Local 3D Shape Descriptors for Fully Automatic Registration of Multiple Sets of Range Images. In *ECCV*, pages 440–453, 2008.

[78] U. Park, Y. Tong, and A. K. Jain. Face Recognition with Temporal Invariance: A 3D Aging Model. In *Automatic Face and Gesture Recognition*, 2008.

[79] P. J. Phillips, P. J. Flynn, T. Scruggs, K. W. Bowyer, and W. Worek. Preliminary Face Recognition Grand Challenge Results. In *Automatic Face and Gesture Recognition*, pages 15–24, 2006.

[80] F. Pighin, J. Hecker, D. Lischinski, R. Szeliski, and D. Salesin. Synthesizing Realistic Facial Expressions from Photographs. In *SIGGRAPH*, pages 75–84, 1998.

[81] P. Pingi, A. Fasano, P. Cignoni, C. Montani, and R. Scopigno. Exploiting the Scanning Sequence for Automatic Registration of Large Sets of Range Maps. *Computer Graphics Forum*, 24(3):517–526, 2005.

[82] K. Pulli. Multiview Registration for Large Data Sets. In *3DIM*, pages 160–168, 1999.

[83] H. T. F. Rhodes. *Alphonse Bertillon: Father of Scientific Detection*. Adelard-Schuman, 1956.

[84] C. Rocchini, P. Cignoni, F. Ganovelli, C. Montani, P. Pingi, and R. Scopigno. The Marching Intersections Algorithm for Merging Range Images. *The Visual Computer*, 20:149–164, 2004.

[85] C. Rocchini, P. Cignoni, C. Montani, P. Pingi, and R. Scopigno. A Low Cost 3D Scanner Based on Structured Light. In *Proc. EG*, volume 20(3), pages 299–308, 2001.

[86] S. Rusinkiewicz. Trimesh2 at `http://www.cs.princeton.edu/gfx/proj/trimesh2/`, Nov 2006.

[87] S. Rusinkiewicz, O. Hall-Holt, and M. Levoy. Real-Time 3D Model Acquisition. *Trans. on Graphics*, 21(3):438–446, 2002.

[88] S. Rusinkiewicz and M. Levoy. Efficient Variants of the ICP Algorithm. In *3DIM*, pages 145–152, 2001.

[89] W. Saleem, O. Schall, G. Patane, A. Belyaev, and S. H. On Stochastic Methods for Surface Reconstruction. *The Visual Computer*, 23(6):381–395, 2007.

[90] C. Samir, A. Srivastava, and M. Daoudi. Three-Dimensional Face Recognition using Shapes of Facial Curves. *PAMI*, 28(11):1858–1863, 2006.

[91] C. Samir, A. Srivastava, M. Daoudi, and E. Klassen. An Intrinsic Framework for Analysis of Facial Surfaces. *IJCV*, 82(1):80–95, 2009.

[92] A. Scheenstra, A. Ruifrok, and R. C. Veltkamp. A Survey of 3D Face Recognition Methods. In *AVBPA*, pages 891–899, 2005.

[93] A. Sharf, M. Alexa, and D. Cohen-Or. Context-Based Surface Completion. *Trans. on Graphics*, 23(3):878–887, 2004.

[94] M. Slater, A. Steed, and Y. Chrysanthou. *Computer Graphics and Virtual Environments: From Realism to Real-Time*. Addison-Wesley Publishers, 2001.

[95] Stanford Computer Graphics Laboratory. 3D Scanning Repository, Scanalyze, VripPack, and VolFill at `http://graphics.stanford.edu/software/`, Nov 2006.

[96] A. J. Stoddart and A. Hilton. Registration of Multiple Point Sets. In *Pattern Recognition*, volume A, pages 40–44, 1996.

[97] Stuttgart. Range Image Database at `http://range.informatik.uni-stuttgart.de/htdocs/html/` , Nov 2006.

[98] F. B. ter Haar. Quadruple Alignment at `http://give-lab.cs.uu.nl/quadruple-alignment/` , Nov 2006.

[99] F. B. ter Haar, P. Cignoni, P. Min, and R. C. Veltkamp. A Comparison of Systems and Tools for 3D Scanning. In *3D Digital Imaging and Modeling: Applications of Heritage, Industry, Medicine and Land*, 2005.

[100] F. B. ter Haar, M. Daoudi, and R. C. Veltkamp. SHape REtrieval Contest 2008: 3D Face Scans. In *Shape Modeling and Applications (SMI)*, pages 225–226, 2008.

[101] F. B. ter Haar and R. C. Veltkamp. Alignment, Merging and Hole Filling Experiments with 3D Range Scans. Technical Report UU-CS-2005-047, Utrecht University, 2005.

[102] F. B. ter Haar and R. C. Veltkamp. A 3D Face Matching Framework. Technical Report UU-CS-2007-047, Utrecht University, 2007.

[103] F. B. ter Haar and R. C. Veltkamp. Automatic Multiview Quadruple Alignment of Unordered Range Scans. In *Shape Modeling and Applications (SMI)*, pages 137–146, 2007.

[104] F. B. ter Haar and R. C. Veltkamp. 3D Face Model Fitting for Recognition. In *European Conference on Computer Vision (ECCV)*, pages 652–664, 2008.

[105] F. B. ter Haar and R. C. Veltkamp. A 3D Face Matching Framework. In *Shape Modeling and Applications (SMI)*, pages 103–110, 2008.

[106] F. B. ter Haar and R. C. Veltkamp. A 3D Face Matching Framework for Facial Curves. *Graphical Models*, 71(2):77–91, 2009.

[107] G. Turk and M. Levoy. Zippered Polygon Meshes from Range Images. In *SIGGRAPH*, pages 311–318, 1994.

[108] M. Turk and A. Pentland. Face Recognition using Eigenfaces. In *CVPR*, pages 586–591, 1991.

[109] University of South Florida, Sudeep Sarkar. USF HumanID 3D Face Database.

[110] R. C. Veltkamp. *Closed Object Boundaries from Scattered Points*. Lecture Notes in Computer Science 885. Springer, 1994. ISBN 3-540-58808-6.

[111] R. C. Veltkamp, R. Ruijsenaars, M. Spagnuolo, R. van Zwol, and F. B. ter Haar. SHREC2006: 3D Shape Retrieval Contest. Technical Report UU-CS-2006-030, Utrecht University, 2006.

[112] R. C. Veltkamp and F. B. ter Haar. SHREC2007: 3D Shape Retrieval Contest. Technical Report UU-CS-2007-015, Utrecht University, 2007.

[113] R. C. Veltkamp and F. B. ter Haar. SHREC2008: 3D Shape Retrieval Contest. In *Shape Modeling and Applications*, pages 215–263, 2008.

[114] T. Vetter, M. J. Jones, and T. Poggio. A Bootstrapping Algorithm for Learning Linear Models of Object Classes. In *CVPR*, pages 40–46, 1997.

[115] S.-F. Wang and S.-H. Lai. Estimating 3D Face Model and Facial Deformation from a Single Image Based on Expression Manifold Optimization. In *ECCV*, pages 589–602, 2008.

[116] A. Wehra and U. Lohrb. Airborne Laser Scanning - an Introduction and Overview. *Journal of Photogrammetry and Remote Sensing*, 54(2-3):68–82, 1999.

[117] T. Weise, B. Leibe, and L. V. Gool. Fast 3D Scanning with Automatic Motion Compensation. In *CVPR*, pages 1–8, 2007.

[118] T. Whitmarsh, R. C. Veltkamp, M. Spagnuolo, S. Marini, and F. B. ter Haar. Landmark Detection on 3D Face Scans by Facial Model Registration. In *1st Int. Workshop on Shape and Semantics*, pages 71–76, 2006.

[119] J. V. Wyngaerd, L. V. Gool, and M. Proesmans. Invariant-based Registration of Surface Patches. In *ICCV*, pages 301–306, 1999.

[120] C. Xu, T. Tan, Y. Wang, and L. Quan. Combining Local Features for Robust Nose Location in 3D Facial Data. *Pattern Recognition Letters*, 27(13):1487–1494, 2006.

[121] L. Yin, X. Wei, Y. Sun, J. Wang, and M. J. Rosato. A 3D Facial Expression Database For Facial Behavior Research. In *Automatic Face and Gesture Recognition*, pages 211–216, 2006.

[122] L. Zhang, B. Curless, and S. M. Seitz. Rapid Shape Acquisition Using Color Structured Light and Multi-pass Dynamic Programming. In *3DPVT*, pages 24–36, 2002.

[123] H. Zhao and G. Xu. Triangular Surface Mesh Fairing via Gaussian Curvature Flow. *Journal of Computational and Applied Mathematics*, 195(1):300–311, 2006.

[124] Z. Zonghua, P. Xiang, S. Weiqiang, and H. Xiaotang. A Survey of Surface Reconstruction from Multiple Range Images. In *Asia-Pacific Conf. on Systems Integrity and Maintenance*, volume 8, pages 519–523, 2000.

Samenvatting

In mijn onderzoek worden metingen verricht op bestaande voorwerpen met gebruik van driedimensionale (3D) laserscanners, een technologie dat het mogelijk maakt de afstand te meten van de laserscanner tot het oppervlak van een voorwerp. Voor één dieptemeting wordt een laserpunt geprojecteerd op het oppervlak van een voorwerp. Het licht dat terugkaatst kan worden waargenomen door een sensor die onder een bekende hoek naar het voorwerp kijkt. Omdat de kijkhoek en de afstand tussen de sensor en laserscanner bekend zijn, kan de afstand tot het voorwerp worden berekend. Dergelijke laserscanners kunnen duizenden diepte metingen per seconde verrichten, maar meestal van één kant van het voorwerp per scan. In de computer wordt elke meting gerepresenteerd als een driedimensionaal punt. Door gebruik te maken van het scanpatroon kunnen de duizenden punten van één scan eenvoudig worden verbonden met driehoeken, waardoor een gedetailleerd oppervlak ontstaat. Om een compleet model te maken moeten de scans van verschillende kanten als een 3D puzzel in elkaar worden gezet, samengevoegd tot één oppervlak, en de overgebleven gaten in het oppervlak worden gevuld. Voor de snelle bewerking moeten hiervoor meetkundige algoritmes worden ontwikkeld die automatisch te werk gaan. De uiteindelijke computer modellen kunnen worden geanalyseerd en vergeleken op vorm en afmeting, en gebruikt in virtuele omgevingen. In dit proefschrift maken we 3D computer modellen van bestaande voorwerpen, vergelijken gezichten, en analyseren statistische eigenschappen van gezichten en gezichtsexpressies.

In hoofdstuk 2 bestudeer ik het volledige proces van lasermetingen tot een compleet 3D computermodel. Tussentijdse resultaten van het proces worden vergeleken met referentiemodellen om zo een uitspraak te kunnen doen over de nauwkeurigheid van beschikbare algoritmes en systemen. Hierbij worden verschillende algoritmes en systemen vergeleken om voorwerpen te scannen, om scans in elkaar te zetten, om scans samen te voegen, en om gaten te vullen. Een dergelijk vergelijkend onderzoek is niet eerder gedaan en verschaft inzicht in de automatische reconstructie van 3D modellen met grote nauwkeurigheid.

Er zijn veel automatische algoritmes beschikbaar die de gebruiker helpen met de reconstructie van 3D modellen uit scandata, maar niet voor het automatisch in elkaar zetten van de losse ongeordende scans, wat (interactief) een tijdrovende klus is voor een gebruiker. Er bestaan veel algoritmes die twee *overlappende* scans precies op elkaar kunnen plaatsen. Het moeilijkste is om die scans te vinden die daadwerkelijk over-

lap hebben en deze tegelijkertijd in elkaar te passen op een (globaal gezien) correcte manier. Het automatiseren van dit proces op een efficiënte manier is essentieel om snel vormen te kunnen reconstrueren. In hoofdstuk 3 presenteer ik een nieuw algoritme dat meerdere ongeordende scans tegelijkertijd in elkaar plaatst met hoge precisie, doeltreffendheid en efficiëntie.

In hoofdstuk 4 gebruik ik 3D laserscans voor biometrische doeleinden. Door iemands gezicht te scannen met een laserscanner wordt de geometrie van het 3D gezichtsoppervlak verkregen, dat per persoon verschilt. Met geometrische algoritmes kunnen zulke gezichten worden geanalyseerd en vergeleken, wat het belangrijkste element is in 3D gezichtsherkenningssystemen. Een dergelijk systeem slaat 3D gezichtsscans, waarvan de identiteit bekend is, op in een gegevensbank en vergelijkt nieuwe gezichtsscans met alle opgeslagen gezichten om de meest gelijkende te vinden. De beste match kan worden gebruikt om de identiteit van de nieuwe scan te achterhalen of als de nieuwe scan als onbekend aan te duiden, wanneer deze match niet goed genoeg is. Een 3D gezichtsvergelijkingssysteem overwint een aantal beperkingen van bestaande 2D gezichtsvergelijkingssystemen, met veelal hetzelfde gebruiksgemak. Ik presenteer een compleet systeem, dat automatisch het gezicht in 3D scans detecteert, uitknipt, verbetert, en vervolgens de gezichtsvergelijking doet met verschillende profiel- en omtreklijnen. Om nuttige combinaties van lijnen op het gezicht te vinden als compacte representatie, wordt een raamwerk ontworpen dat 3D lijnen met elkaar vergelijkt. Dit raamwerk gebruikt verschillende type lijnen van het gezichtsoppervlak en evalueert hun doeltreffendheid voor de persoonsherkenning.

In hoofdstuk 5 presenteer ik een algoritme om automatisch een 3D gezichtsmodel te vervormen naar 3D gezichtsscans om daar vervolgens persoonsherkenning op toe te passen. Dit 3D gezichtsmodel is een statistisch model, gemaakt uit bestaande 3D voorbeeldgezichten en interpoleert lineair tussen deze gezichten om nieuwe gezichten te kunnen construeren. Met gebruik van modelcoëfficiënten, vervormt het model langs de hoofdassen van de dataspreiding, waarbij bijvoorbeeld de grootte van het gezicht, de positie van de ogen, ofwel de lengte van de neus verandert, terwijl er rekening wordt gehouden met de onderliggende statistiek van het model. Door dit model te vervormen naar het gezicht, wordt een gelijkend 3D gezicht gecreëerd zonder gaten en zonder ruis, wat vaak een probleem is voor 3D gezichtsvergelijkende algoritmes. Ik evalueer de nauwkeurigheid van de automatisch vervormde gezichten en laat zien dat mijn methode preciezer is dan bestaande methoden. Door het gezichtsmodel te verdelen in meerdere gezichtscomponenten, wordt het resultaat nog beter. Het is ook mogelijk antropometrische herkenningspunten (zoals het neuspuntje en de ooghoeken) aan te brengen op het vervormbare gezichtsmodel. Deze worden dan automatisch naar hun statistisch betrouwbare locaties in de verschillende gezichtsscans gebracht. Bovendien geeft de manier waarop het model wordt vervormd om het te laten passen op een scan, een idee van de geometrische eigenschappen van het gezicht, welke verborgen zitten in de modelcoëfficiënten. Voor de gezichtsherkenning, gebruik ik de omtreklijnen uit hoofdstuk 4, de automatisch gedetecteerde herkenningspunten, en de verkregen modelcoëfficiënten. Ik laat zien dat de modelcoëfficiënten de beste prestaties leveren, met de maximale gezichtsherkenning voor de UND gegevensbank met 3D gezichtsscans.

In hoofdstuk 6 presenteer ik een nieuw algoritme om automatisch het 3D vervormbaar gezichtsmodel uit te breiden met nieuwe gezichten. Voor een 3D gezichtsver-

gelijkingssysteem gebaseerd op model coëfficiënten, is het uitermate belangrijk dat de eigenschappen van veel realistische gezichten in het model zitten. In het geval dat een gezicht niet (voldoende) gemodelleerd kan worden, zijn de automatisch verkregen modelcoëfficiënten minder betrouwbaar, hetgeen de identificatie van personen kan verhinderen. Het algoritme dat ik presenteer detecteert of een nieuwe scan onvoldoende gemodelleerd kan worden, zorgt dat dit toch lukt, en voegt de nieuwe data toe aan het model. Het nieuwe aan deze methode is het gebruik van meerdere gezichtscomponenten om het model beter op de scan te passen, een algoritme dat eventuele modelleringsfouten corrigeert, en een automatische check om nieuwe gezichten te herkennen en alleen deze toe te voegen aan het model.

In hoofdstuk 7 bouw ik een nieuw statistisch gezichtsmodel waarin ook expressies gemodelleerd kunnen worden, hiermee kunnen gezichten worden geïdentificeerd onafhankelijk van expressies. In een globaal naar locaal proces, worden de identiteit- en expressiecoëfficiënten van het model aangepast zodat het verkregen gezicht precies past op de 3D scan van een gezicht met een expressie. Kwantitatieve evaluatie laat zien dat de expressievervorming en ook het gebruik van voorgedefinieerde gezichtscomponenten het pasresultaat verbetert. 3D gezichtsherkenningsexperimenten op de publiekelijk beschikbare UND, GAVAB, BU-3DFE, en FRGC v.2 gegevensbanken laten hoge herkenningspercentages van 99%, 98%, 100%, and 97% zien wanneer de identiteitscoëfficiënten worden gebruikt. Resultaten laten zien dat niet alleen de coëfficiënten behorend bij het globaal goed passende model goed presteren, maar dat de coëfficiënten van vier locaal goed passende modellen een vergelijkbaar resultaat kunnen behalen. Aangezien het vinden van het optimaal passende model een rekenintensief proces is, kan het laten vieren van deze strenge eis een gezichtsherkenningssysteem sneller maken.

In dit onderzoek heb ik automatische rekenmethodes ontwikkeld voor het maken van computermodellen uit 3D laserscans van algemene voorwerpen, en van gezichten in het bijzonder. De resultaten kunnen worden toegepast door musea die hun culturele erfgoed in 3D willen vastleggen of zelfs reproduceren met een 3D printer, en door de entertainmentindustrie om snel karakters van klei naar computermodellen om te zetten. De gezichtsmodellen die ik heb ontwikkeld kunnen naar een 3D gezichtsscan worden vervormd, om zo de eigenschappen van dat gezicht te bepalen. Op basis daarvan kunnen we gezichtsvergelijking doen met een precisie tussen de 97% en 100%. Dit kan gebruikt worden voor beveiliging, forensische toepassingen, en het ontwerpen van een goed passend gasmasker.

Curriculum Vitae

Frank Bart ter Haar was born on January 1st 1980 in Vlaardingen, The Netherlands. After moving to Woerden, he received his high school diploma in 1998 at the Kalsbeek Scholengemeenschap. From 1999 to 2004, he studied computer science at the Utrecht University, passing both the propaedeutic exam and the doctoral exam *cum laude*. His Master's thesis entitled "Automatic localization of the optic disc in digital colour images of the human retina" was completed at the Image Sciences Institute. In 2004, he started as a Ph.D. student under the supervision of Prof. dr. Remco C. Veltkamp at the same university. In 2009, he completed this thesis there. An up-to-date curriculum vitae can be found at `http://www.linkedin.com/in/frankterhaar`.

Acknowledgments

Finishing a PhD thesis cannot be accomplished without the proper guidance, feedback, working environment, education, perseverance, and motivation.

Remco Veltkamp, thank you for guiding me during the course of my PhD. Both your keen eye for detail and your overall view have proven to be of great help during these years.

I thank the members of the reading committee, Hein Daanen, Mohamed Daoudi, Bianca Falcidieno, Erik Jansen, and Mark Overmars for taking the time to read this thesis.

A word of thanks goes to all my colleagues within the AIM@SHAPE network and at the ICS department for having a good time at conferences, summer schools, international project meetings, and the daily breaks. I was very lucky to have Reinier van Leuken, Martijn Bosma, Arno Kamphuis, and Geert-Jan Giezeman as office mates.

I'm grateful to my parents Bart ter Haar and Alice van der Steege for providing me the means for education and passing on their perseverance, for which in turn I thank my grandparents for doing the same. Furthermore, I'm proud to have my two brothers Sander and Bart-jan as paranymphs during the defense of this thesis.

As for the motivation part, I would like to express my gratitude to all friends and family for their support and interest, but most of all to Femke and my sons Yoran and Niels. Yoran and Niels thanks for your little distractions, they really kept me going. Femke, thank you for your inspiration and endless support during the past years and for those to come.