Constructing Anatomically Accurate Face Models using Computed Tomography and Cyberware data

by

Faisal Zubair Qureshi

A thesis submitted in conformity with the requirements
for the degree of Master of Science
Graduate Department of Computer Science
University of Toronto

Copyright © 2000 by Faisal Zubair Qureshi

Abstract

Constructing Anatomically Accurate Face Models using Computed Tomography and Cyberware data

Faisal Zubair Qureshi

Master of Science

Graduate Department of Computer Science

University of Toronto

2000

Facial animation and cranio-facial surgery simulation both stand to benefit from the development of anatomically accurate computer models of the human face. State-of-the-art biomechanical models of the face have shown promise in animation, but they are inadequate for the purposes of cranio-facial surgery simulation. The goal of this thesis is to develop an improved facial model, using Cyberware data which captures the external structure and appearance of the face and head, as well as computed tomography (CT) data which captures the internal structure of facial soft and hard tissues. To this end, we develop algorithms to (1) register the CT and Cyberware datasets, (2) extract from the CT data a skull subsurface which serves as a foundation of the soft-tissue model, and (3) compute thickness of facial soft-tissues over the face.

Dedication

I dedicate this thesis to Tahira.

Acknowledgements

I thank my advisor, Professor Demetri Terzopoulos, for the many fresh ideas that he suggested, for encouraging me to work to the best of my abilities and for providing the resources and creating the environment that made this work possible. I also thank Dr. Tim McInerney for his guidance and support. I very much appreciate the suggestions and assistance that I received from fellow graduate student Victor Lee. I would also like to thank Dr. Jacques-Olivier Lachuad for his assistance and valuable ideas about this work, and for providing me with CT datasets and the marching cubes program.

I am grateful to Dr. Jun Ohya, who invited me to work at the Advanced Telecommunications Research (ATR) Labs in Kyoto, Japan, for his ideas and direction. Part of the work presented here was done under his supervision. I thank Dr. Shigeo Morishima for providing lots of ideas during our long discussions at ATR.

I would like to thank my fellow graduate students, Kiam Choo, Chakra Chennubhotla, Alex Vasilescu, Bonny Biswas, and Robert Majka. They made working in the lab a lot more fun and stimulating. They provided much insight during many long, involved conversations. Chakra helped me start exploring and using many of the computer and research resources.

My thanks to Kathy Yen for her assistance with departmental and university documents.

Finally, I am grateful to my parents and my siblings for their love, support and encouragement. They instilled in me a love of knowledge and a love of life.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The human face plays a key role in interpersonal relationships. Even very subtle malformations of facial geometry can significantly affect subjective appearance and compromise individual aesthetics. Faces can be disfigured by birth defects, injury and/or disease. Cranio-facial surgery can help restore facial appearance and functionality. These operations require considerable pre-operative planning, and it is often difficult for surgeons to predict post-operative appearance. A cranio-facial surgery simulation system can help surgeons plan many such operations.

Recent advances in the fields of computer graphics, visualization, and medical image analysis have made surgical planning and simulation realizable, including the planning and simulation of cranio-facial surgeries. Our long-term goal in cranio-facial surgery simulation is to develop a software system that will enable surgeons to explore the biomechanical implications of surgical alternatives in a high fidelity virtual reality environment. We envision that surgeons will be able to simulate the actual operation, including the cutting, moving and re-alignment of facial hard and soft tissue. The system should also allow surgeons to compare pre- and post-operative facial appearance and functionality.

A cranio-facial surgery simulation system of this sort would benefit from an anatomically accurate, functional model of the patient's face. In the field of computer graphics,

Lee, Terzopoulos and Waters (1995) propose a physics-based facial model for the purposes of facial animation. They construct a 3-layered biomechanical model of the facial skin with contractile actuators that model the muscles of facial expression embedded with anatomical consistency and rooted in an estimated skull subsurface. Their approach has demonstrated impressive results in the context of facial animation; however, the model of Lee *et al.* has two main shortcomings from the point of view of surgery simulation: (1) it does not take into account variations in skin thickness in different regions of the face and (2) it does not include an anatomically accurate skull substructure. Hence, a functional facial model which can be used for surgery simulation as well as for animation remains to be achieved.

1.1 Goals

In this thesis, we extend the facial model proposed by Lee *et al.* (1995) by incorporating skin thickness information plus a more accurate skull subsurface. Patient data is acquired from a *Computed Tomography* (CT) scanner and a Cyberware, Inc., *3D Color Digitizer*. The CT scan captures internal facial skin thickness information and skull geometry, while the Cyberware scan captures external, epidermal geometry and photometry (color/texture) information. We extract polygonal representations of the skin and skull from the CT data using the Marching Cubes algorithm (Lorensen and Cline1987). We then adapt the facial mesh introduced by Lee *et al.* to the Cyberware data, and register the two datasets. Once the two datasets are registered, we can compute the facial skin thickness and construct the facial model. Figure 1.1 illustrates the steps of the facial model construction process.

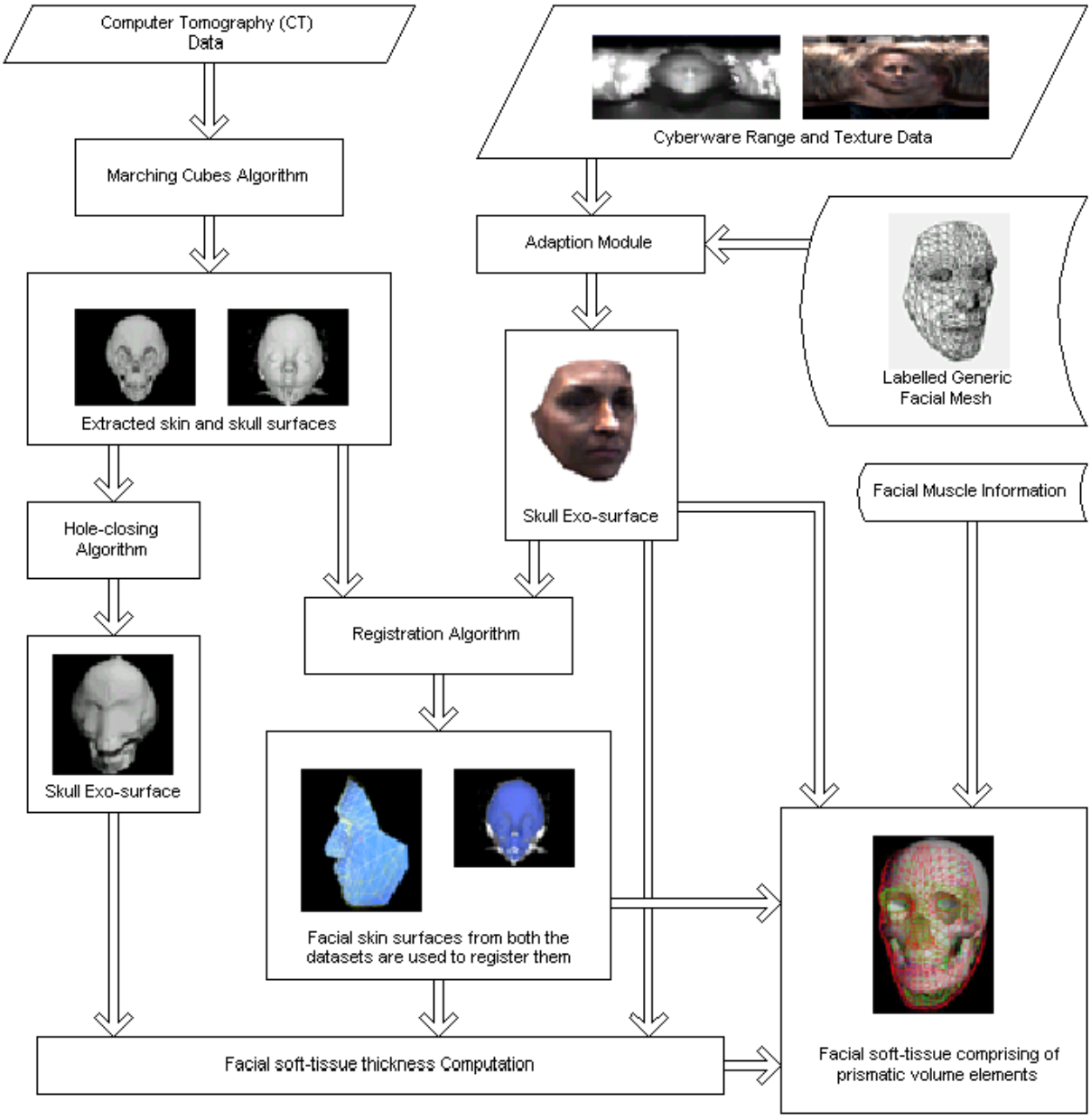


Figure 1.1: The facial soft-tissue construction process

1.2 Contributions

This thesis develops an approach for constructing anatomically accurate facial soft-tissue models from CT and Cyberware data.

The skull, extracted from CT data using the Marching Cubes algorithm, contains holes which are anatomically correct but pose difficulties in the construction of the facial soft-tissue model. We propose a technique for generating a smooth exosurface from the skull. This surface serves as the skeletal foundation for our model.

We also develop an interactive surface-based algorithm for registering CT and Cyberware datasets. This algorithm estimates an affine transformation matrix which registers volumes of interest across the two datasets. The algorithm is able to register any non-coplanar surfaces related through a global, affine transformation.

We use CT and Cyberware datasets to compute facial soft-tissue thickness, and construct a soft-tissue model whose structure is identical to the one proposed by Lee *et al.* (1995)—a 3-layered biomechanical model of with contractile muscles embedded at anatomically correct positions. By incorporating subsidiary models of eyes, eyelids, teeth and neck, as was done by Lee *et al.*, our new model can potentially be used for facial animation. However, our new model also incorporates a more accurate skull subsurface which is extracted from the CT data. Currently, our model is still not entirely adequate for advanced surgery simulation because it lacks a solid model for the underlying skull; however, this thesis makes progress towards the long-range goals stated in the previous section.

1.3 Thesis Outline

In Chapter 2, we review the literature on the topics of *isosurface extraction* and *registration.*

In Chapter 3, we develop an algorithm for extracting the skin and skull from CT data.

We also present an algorithm to extract the smooth skull exosurface from the CT data. Furthermore, we explain the process of adapting the generic mesh to the Cyberware data.

In Chapter 4, we develop an interactive surface-based registration algorithm that can register CT and Cyberware datasets.

In Chapter 5, we propose a scheme for computing skin thickness and constructing the facial soft-tissue model from CT and Cyberware data.

In Chapter 6, we summarize our work and discuss future directions.

Appendix A describes the marching cubes algorithm (Lorensen and Cline1987).

Appendix B describes an algorithm for ray-triangle intersection.

Appendix C presents a detailed empirical study of the registration algorithm developed in Chapter 4.

Chapter 2

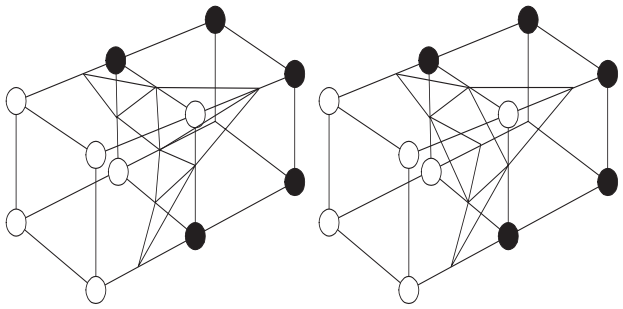
Literature Review

Our scheme of constructing facial soft-tissue model requires computer tomography (CT) and Cyberware datasets to be precisely registered. We extract skin surface from CT data and uses it to register the two datasets. We also extract skull surface from CT data and use it to compute facial soft-tissue thickness. Therefore, we are interested in techniques available for extracting surfaces from three dimensional datasets (*isosurface extraction*), and registering datasets (*registration*). In this section we will review the literature on these topics.

2.1 Isosurface Extraction

Extracting and displaying isosurfaces of volume data is useful in many areas. Three dimensional (3-D) data is difficult to visualize, and rendering surfaces within volume data is one of the main tools available for this purpose. The most common method for representing a generic surface is with connected polygons, and hence most isosurface extraction algorithms employ a polygonal surface representation.

Lorensen and Cline's *marching cubes* (MC) algorithm (Lorensen and Cline1987) extracts a polygonal representation of an isosurface by reducing the problem to that of triangulating a single cube which is intersected by the surface. The isosurface-cube inter-

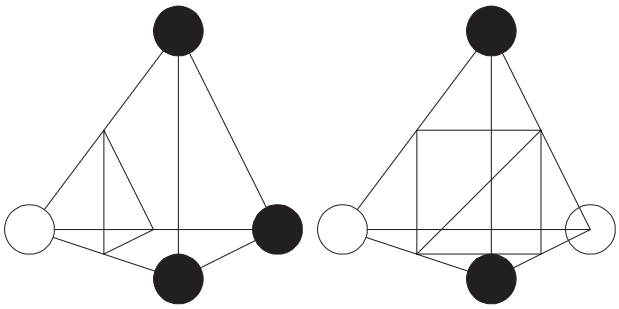


(a) Consistent (b) Inconsistent

Figure 2.1: **Marching Cubes.** The use of complementary symmetry can generate topological inconsistencies (a), which can be corrected by using an alternative triangulation for the complementary case (b) .

sections are estimated along the edges of the cube, by linear interpolation of the sampled volume data at each corner. The intersection points are joined by one or more triangles to form a polygonized surface patch. The whole isosurface can be triangulated by "marching" this cube through the data and creating surface patches whenever the cube intersects the isosurface. Appendix A presents details of the algorithm. The MC algorithm has become the method of choice for isosurface extraction; however, it may extract topologically inconsistent surfaces (see Figure 2.1) (Durst1988; Wilhelms and Gelder1990) and it generates redundant polygons (Li and Agathoklis1997).

The original MC algorithm produces a large number of polygons to represent an isosurface. Reducing the number of polygons can increase computational efficiency. The polygons can be reduced by adapting their sizes to the shape of the isosurface, using fewer, larger sized, polygons for low-curvature regions (Schroeder, Zarge and E.1992; Turk1992; Muller and Stark1993; Schmidt1993). A common technique to reduce the number of triangles is to apply a mesh simplification algorithm on the generated mesh (Schroeder, Zarge and E.1992). Mesh simplification methods have recently been reviewed in (Cignoni, Montani and Scopigno1998). Some other methods involve modifications of



(a) One point (b) Two points

Figure 2.2: **Marching Tetrahedra.** The two cases where triangulation is required, and the resulting triangulations.

the original MC algorithm. Yun and Park (1992) create compact polygon meshes by reducing the number of MC-lookup-table entries from 22 to 5. In the original MC algorithm, the data values of the cube-vertices are interpolated to compute the surface-cube intersections, which creates degenerate and redundant triangles. The *Discretized marching cubes* (DMC) (Montani, Scateni and Scopigno1994) algorithm solves this problem by discretizing the locations of a triangle's vertices at 13 points within a cube, one at the cube's center plus the 12 mid points of its 12 edges. The algorithm then produces triangular and rectangular meshes, and a post-isosurface-generation process is applied to merge coplanar polygons. Li and Agathoklis (1997) extend this algorithm b y merging coplanar triangles in the lookup table; therefore, no post-processing is required.

As mentioned earlier, the original MC algorithm may not produce surfaces which are topologically consistent with the original data. This is due to the arbitrary choice of the triangulation for a cube, with any face containing two opposite corners on one side of the isosurface, and two on the other. If the same triangulations are chosen for both cubes containing this face, the resulting surface will have holes (Durst1988; Wilhelms and Gelder1990; Kalvin et al.1991; Treece, Prager and Gee1998). Many methods have been proposed to address this problem, as reviewed in (Ning and Bloomenthal1993; Wilhelms

and Gelder1990). Triangulation scheme which biases the topology towards one side of the surface resolves this problem (Montani, Scateni and Scopigno1994).

Tessellating the space with tetrahedra rather than cubes also resolves the topological ambiguity. The *marching tetrahedra* (MT) (Guéziec and Hummel1995) algorithm constructs a tetrahedral tessellation by decomposing each cube into five tetrahedra. The symmetry of subdivision of the cube has to alternate between cubes to align the faces of the tetrahedra, and this introduces additional ambiguity. Chan and Purisma (1998) have suggested a tessellation of space into tetrahedra based on a body centered lattice. Unlike the subdivided cube method, the resulting tetrahedra are all identical and there is no ambiguity in the tessellation. The main disadvantage of tetrahedral schemes is that they create an even larger number of triangles than the original MC for a given data set. This problem can be ameliorated by applying a mesh simplification algorithm on the constructed triangulation. Treece *et al.* (1998) have recently proposed a *regularized marching tetrahedra* (RMT) algorithm. This algorithm generates isosurfaces which are topological consistent with the data. The generated isosurfaces also have fewer triangles than those generated by the original MT algorithm.

2.2 Registration

Images acquired through different modalities are usually in different coordinate systems. When acquired from a single object, these images often contain complementary information about the object. It is therefore useful to combine these images to get a better understanding of the object. To this end, we need to map the space of one image onto that of the other. This process is called *registration*. Registration techniques are reviewed in (van del Elsen, Pol and A.1993; Toga and Banerjee1993; Maintz and Viergever1998). For our purpose, registration algorithms can be divided into three broad categories:

- Point-set based registration methods

- Voxel based registration methods
- Segmentation based registration methods

2.2.1 Point-Set based Registration Methods

Point-set based registration methods register images by establishing point correspondences across images. The selection of point correspondences, the difficult part of the registration problem, requires an anatomical (Hill et al.1991) or a geometrical (Thirion1994; Feldmar and Ayache1994; Thirion1996; Feldmar and Ayache1996) criterion. Once the point correspondences are set up, a transformation is computed which brings the first set of points closer, in some predefined sense, to the second set of points. The computed transformation is then used to register the two images. The point sets are sparse compared to the actual images; therefore, finding the transformation is fast. These methods usually assume that the two images, and hence the point sets, are related through a rigid or affine transformation.

The problem of least-squares fitting of two point sets is addressed in (Faugeras and Hebert1983; Lin et al.1986; Arun, Huang and Blostein1987). Assuming that the correspondences between the two point sets are known and that the point sets are related to each other through a rigid transformation i.e., a rotation and a translation, the transformation parameters are computed using an iterative technique (Faugeras and Hebert1983; Lin et al.1986), or a non-iterative algorithm based on *singular value decomposition* (SVD). The computed parameters are best in the least-squares sense.

The *Iterative closest-point matching* (ICP) algorithm, proposed by Besl and Mckay (1992), is an automatic surface registration algorithm. The ICP algorithm requires a procedure for finding the closest point on the surface to a given point. This algorithm iteratively converges to the nearest local minimum of the mean square distance metric (for proof, see (Besl and McKay1992)). The algorithm assumes that one of the two

surfaces is in point form. This method does not require any derivative estimation or local feature extraction. A drawback of the ICP algorithm is that it fails to register the two surfaces if the point set representing one surface includes a significant number of data points that do not correspond to any point on the other surface. The *Iterative point matching* (Zhang1994) algorithm solves this problem by using the matching criterion where each point in the point set representing the first surface is assigned a value 1 or 0 depending whether or not it has a reasonable match in the second surface. This is determined by a dynamically chosen maximum tolerable distance between a point in the point set representing the first surface and its match on the second surface. Feldmar and Ayache (1994; 1996) have extended the ICP algorithm to perform affine and locally-affine registration of free-form surfaces. Chen and Medioni (1992) propose an algorithm for registering multiple range images in order to create a complete model of an object. They choose one hundred points, called *control points*, on a regular grid on the first range image. The surface normal for each control point is calculated, and the intersection of the surface normal to the surface in the second image is chosen as the closest point.

2.2.2 Voxel based Registration Methods

Voxel based registration methods are more reliable than point set methods because they use the whole image content without prior image reduction or image segmentation throughout the registration process. The only downfall is the associated computational cost, since these methods have to deal with a large amount of data. These methods define registration parameters (i.e., transformations), as a function of some similarity measure between the two images and attempt to perform the registration by maximizing this similarity measure. Suitable similarity measures are reviewed in (Penney et al.1998).

Alpert *et al.* (1990) propose a method to register *computed tomography* (CT), *positron emission tomography* (PET) and/or *magnetic resonance* (MR) images. The user delineates exactly the same volume structures in both images. The volumes are treated as

rigid bodies and their respective *center-of-masses* (COM) and principal axes are computed. The COM's are aligned by a translation and the principal axes are aligned by a rotation about the COM. The computations involved are simple and fast and the quality of the registration is determined by how accurately the volume structures are delineated. This technique is non-iterative and will fail to register properly if a portion of one of the images is missing. Van del Elsen *et al.* (1994) use correlation between pixel values (gray level values) of the two images for registration. This approach yields good results in the case of mono-modal images—images acquired through the same modality. In multi-modal image registration, however, statistical similarity measures must be defined. Woods *et al.* (1992) propose a semi-automatic algorithm for registering inter-subject PET images by minimizing the standard deviation of the corresponding pixels in both images. The user provides an initial guess for the registration parameters. Woods *et al.* (1993) extend this algorithm to register inter-subject MR and PET images. In addition to providing the initial guess for registration parameters, the user is required to edit the scalp, the skull and the meninges from the MR image. Wells *et al.* (1996) maximize a measure based on mutual information between the two i mages to register MR- with CT- and PET-images; they compute a rigid transformation.

2.2.3 Segmentation based Registration Methods

In these methods, anatomically similar structures are first segmented from both images. These structures are then registered, and the original images are registered using the same transformations. The accuracy of these methods depends upon the segmentation step, which is usually fast and computationally inexpensive. Many segmentation based registration methods have been proposed for registering head images. The segmented structures can be points, curves or surfaces.

The *head-hat* registration method, introduced by Pelizzari *et al.* (1989), is currently in clinical use. In the head-hat approach, CT, MR and/or PET head images are reg-

istered using 3-D surface models constructed by extracting the external surface of the head from these images. An iterative algorithm minimizes the mean-squared distance between the points on one surface called the *hat* and the other surface called the *head* in order to compute the transformation which brings the two 3-D surface models into congruence. Human interaction is not required to identify correspondences between the two surfaces. This method works well even when both surfaces are partially overlapped, and also if a portion of one of the surfaces is missing (Turkington et al.1993; Turkington et al.1995). The volumes-of-interest in the images are then registered using the computed transformation.

Hill *et al.* (1991) identify 6 to 12 point-like anatomical features in CT and MR images. These point-correspondences are used to compute the transformation needed to register the volumes of interest in the images. Thirion (1996) uses feature points which are invariant under rigid transformations (Thirion1994). The *marching lines* algorithm (Thirion and Gourdon1996) is used to extract these feature points, called extremal points. Once the extremal points are extracted, the possible sets of correspondence pairs are set up and the transformation matrices are computed. The extremal points are sparse compared to the images; therefore, only a few transformation matrices have to be computed. Moreover, a geometric invariance constraint further reduces the set of possible transformations. The remaining transformations are then studied in a systematic manner to find the best transformation.

Chapter 3

Extraction of Surfaces

This chapter explains the first phase of our approach to construct a volumetric facial soft-skin model, in which we fit the generic mesh to the Cyberware data and we extract the skin and skull surfaces from the CT data. The extracted skin surface and the adapted generic mesh are aligned to register the CT and the Cyberware data.

In Section 3.1 we explain the algorithm proposed by Lee *et al.* (1995) to adapt the generic facial mesh to the Cyberware data. Section 3.2 discusses the extraction of the skin and skull surfaces from the CT data using the MC algorithm. The MC algorithm extracts both the inner and outer surfaces of the skull. For facial skin tissue modeling only the outer skull surface is required. The skull, moreover, contains holes—the eye orbits, nasal cavities and around the upper part of the mandible (see Figure 3.5). These features are anatomically correct, however, they along with artifacts in the data, such as spikes resulting from X-ray reflections off tooth fillings, make the facial skin thickness computations and the dynamic simulation of the physics-based facial model difficult. Section 3.2.1 describes a new algorithm to remove these artifacts and fill the holes in order to construct a smooth skull exosurface.

3.1 Fitting a Generic Face Mesh to Cyberware Data

A Cyberware data set consists of two registered images of the subject: a range image (Figure 3.1(a)) which is a topographic map that records the distance of the surface points from the sensor, and a reflectance (RGB) image (Figure 3.1(b)) which registers the color of the surface at those points. The images are in cylindrical coordinates, where longitudes (0-360 degrees) are along the x-axis while the vertical height is along the y-axis. The algorithm for adapting the facial mesh to the Cyberware data consists of two steps. In the first step, defects in the captured images are removed. In the next step, the generic facial mesh is deformed using a feature-based matching scheme. The deformed mesh captures the facial geometry as a triangulated surface. The triangulated surface is then textured with the reflectance image to create a realistic facsimile of the subject's face.

3.1.1 Interpolating Missing Cyberware Data

The Cyberware scanner fails to capture range information in the hair area, the chin area, inside the nostrils and even in the pupils as the laser beam tends to disperse in these regions (see figure 3.1(a)). Missing points are assigned a value "VOID". We use a membrane interpolation method to compute reasonable values for these points (Terzopoulos1988). The range values of the "VOID" points are initially set to zero and adjusted through repeated averaging with neighboring range values. The relaxation algorithm iterates until the maximum change in the interpolated values drops below some small, pre-specified constant ϵ. The algorithm is as follows:

Interpolating Range Data

1: Let w be the width and h be the height of the range map.

2: Set the flags $F(i,j)$, where $i = 1, 2, \cdots, w$ and $j = 1, 2, \cdots, h$, of all "VOID" points $p(i,j)$ to be "false".

3: Set $p(i,j) = 0$ for all "VOID" points.

4: **repeat**

5: Let $m_f = 0$

6: **for all** Points $p(i,j)$ with $F(i,j) =$ "$false$" **do**

7: Set $o = p(i,j)$

8: Assume $p(u,v) = 0$, when $u \notin \{1,2,\cdots,w\}$ or $v \notin \{1,2,\cdots,h\}$, and set $n = \frac{p(i,j+1)+p(i,j-1)+p(i+1,j)+p(i-1,j)}{4}$

9: Set $m_f = \max((n-o), m_f)$

10: Set $p(i,j) = n$

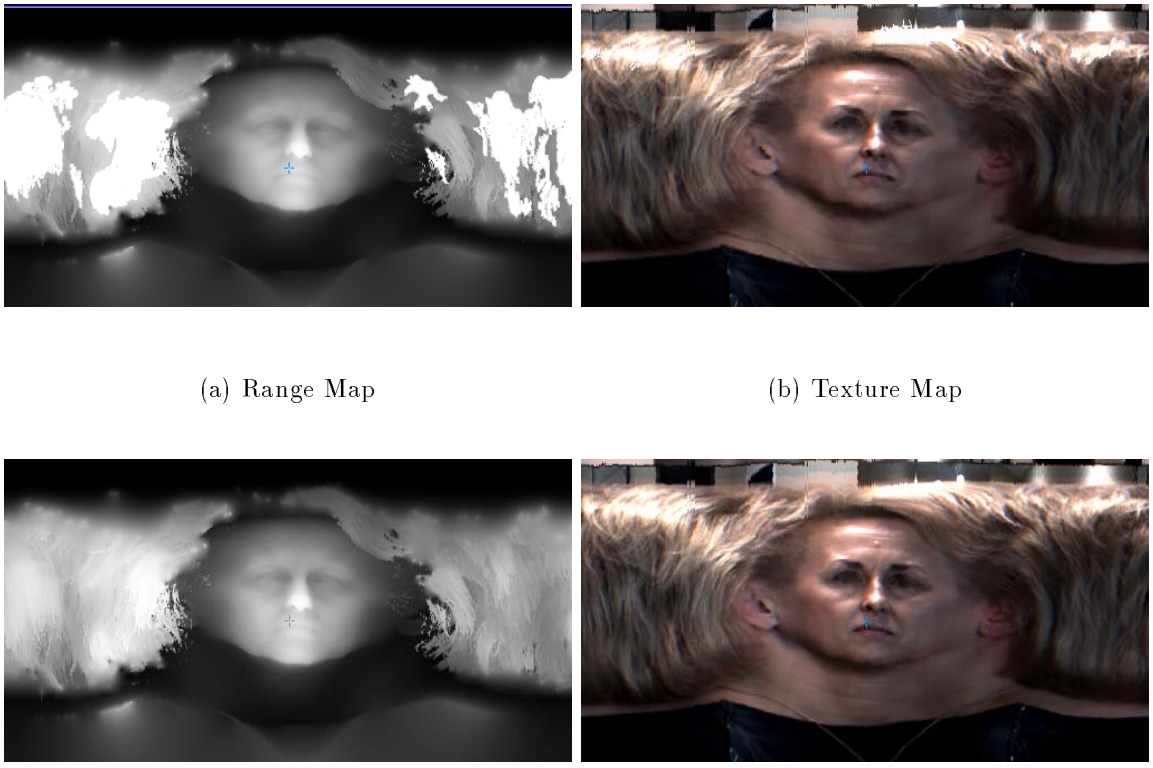
11: **end for**

12: **until** $m_f < \epsilon$

Figure 3.1(c) illustrates the interpolated range data. The reflectance image may contains "VOID" values which indicate that the color values at these points are unobservable (see Figure 3.1(b)). The membrane relaxation algorithm is used to interpolate color values at these points by repeated averaging of neighboring known colors. Figure 3.1(d) shows the interpolated texture image.

3.1.2 Computing the Laplacian Map

Although the head structure in the cylindrical laser range data is distorted along the longitudinal direction, important features such as the slope changes of the nose, forehead, chin and the contours of the mouth, eyes, and nose are still discernible. We apply a modified Laplacian operator to the range map to locate these features. Figure 3.2 shows the Laplacian field computed over the range map in Figure 3.1(c). The local maxima of the modified Laplacian reveal the boundaries of the lips, eyes, and the chin.



(a) Range Map

(b) Texture Map

(c) Interpolated Range Map

(d) Interpolated Texture Map

Figure 3.1: Cyberware range (a) and reflectance (b) data, showing regions of missing values. Interpolated Cyberware range (c) and reflectance (d) data.

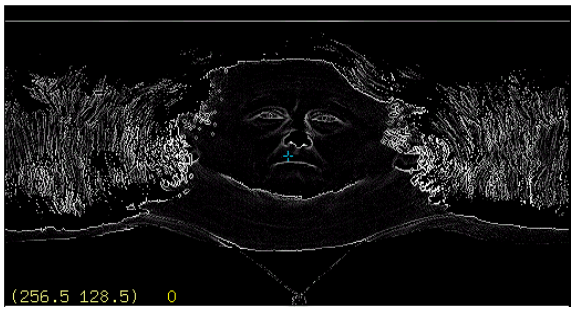


Figure 3.2: Laplacian field computed over the range data

3.1.3 Mesh Adaptation

We deform an irregularly triangulated generic face mesh (see Figures 3.3 and 3.4(a)) to fit the Cyberware data. This generic mesh has well-defined features which form the basis for accurate feature-based adaptation to the scanned data and automatic scaling and positioning of the facial muscles. This mesh also produces an efficient triangulation with more triangles in the highly curved and/or highly articulate regions of the face and larger triangles elsewhere. The feature nodes including the eye contours, nose contours, chin contours, and the mouth contours are labeled. Given the range image and its Laplacian field, the mesh adaptation algorithm fits the generic mesh to the range data according to the following steps:

Mesh Adaptation Algorithm

1: Locate nose tip.

2: Locate chin tip.

3: Locate mouth contour.

4: Locate ears.

5: Locate eyes.

6: Activate spring forces.

7: Adapt hair mesh.

8: Adapt body mesh.

9: Store texture coordinates.

The range image is sampled at the location of the nodes of the adapted face mesh to capture the facial geometry. The node positions also provide texture map coordinates that are used to map the full resolution color image onto the triangles. Ideally the subject's face should have a relaxed expression when they are being scanned, but Lee *et al.* (1995) describe a method for dealing with the case if the subject's mouth is open.

Figure 3.4(b) shows the adapted generic mesh.

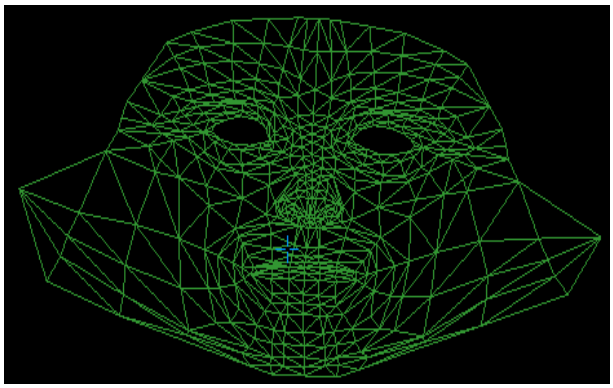
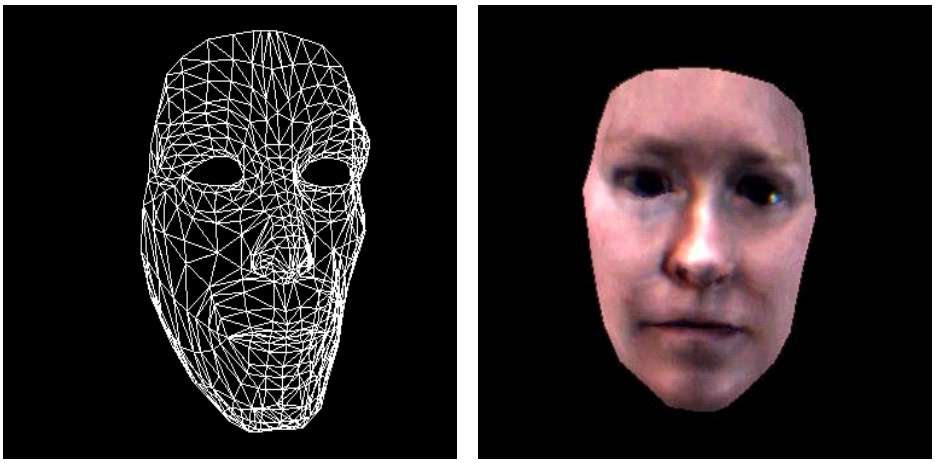


Figure 3.3: Generic Facial Mesh mapped onto Cylindrical Coordinates



(a) Triangulated face mesh, before adaptation

(b) Textured mapped face mesh, after adaptation

Figure 3.4: Generic Facial Mesh, before and after adaptation using algorithm proposed by Lee *et al.* [1995].

3.2 Extracting Bone and Skin Surfaces from CT Data

The CT data consists of contiguous slices from a three dimensional array. A number of geometric representations for different anatomical structures can be extracted from this data. We use the *Marching Cubes* (MC) algorithm (see Appendix A) to create a triangulated representation. Carefully selected thresholds isolate the bone and the skin tissue densities and are used to create geometric isosurfaces for the skull and the epidermal skin tissue. Figure 3.5 shows the skull and skin surfaces created from a sample CT dataset.

As discussed earlier, it is desirable to have a continuous and smooth skull exosurface. In the next section we present the algorithm for generating a smooth and continuous skull exosurface.

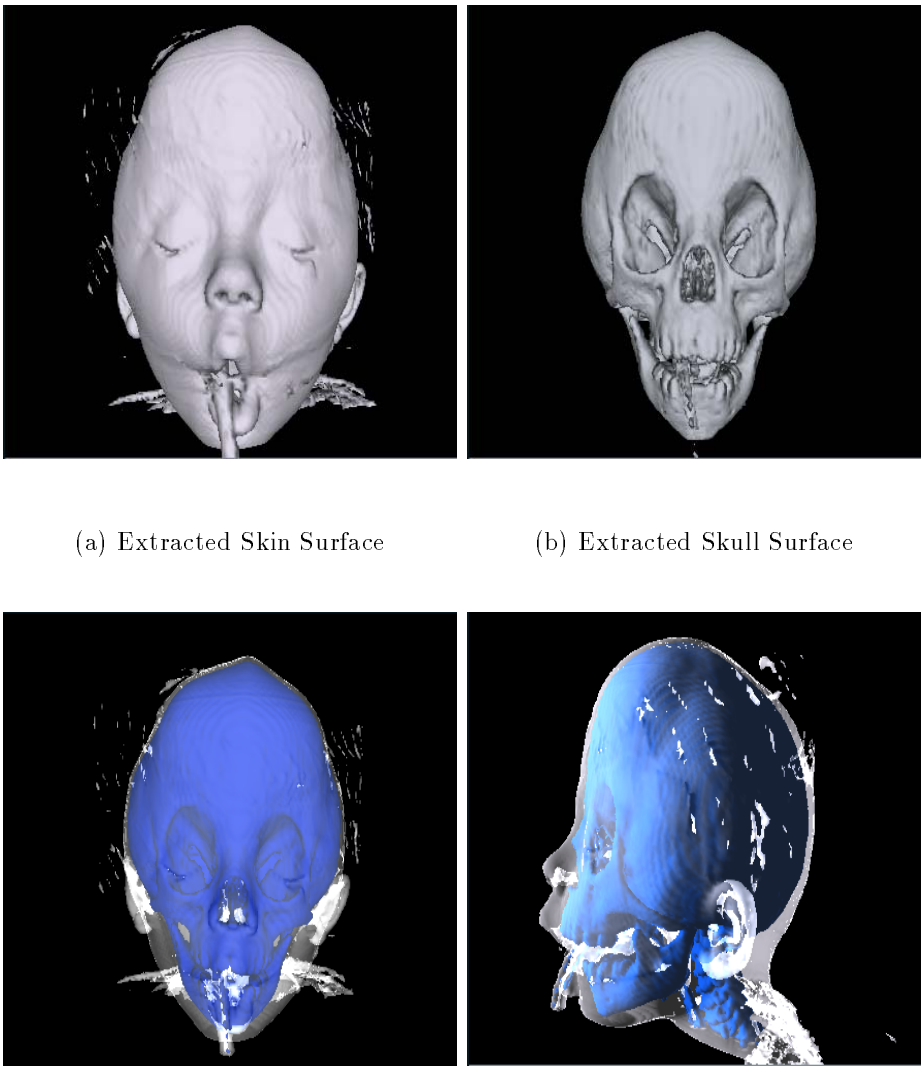
3.2.1 Extracting a Smooth Skull Exosurface

We start by mapping the skull onto the cylindrical coordinates. The human face, being for the most part convex, maps conveniently into the cylindrical coordinates. The process of mapping the skull into the cylindrical coordinates is analogous to constructing a range map of the skull. This mapping has significant advantages in the sense that 3-D data can be manipulated in 2-D space. A 3-D world coordinate, $\mathbf{p} = (x, y, z)$, is mapped into cylindrical coordinates $\mathbf{c} = (\theta, y)$ as follows:

$$\mathbf{c} = (atan2(x, z), y),$$

where *atan2* is the C language math library function returning arctangent of x/z in the range $-\pi$ to π. The inverse transformation is $\mathbf{p} = (r\cos(\theta), y, r\sin(\theta))$.

We construct a range map of the skull by sampling points on the outermost surface of the skull. Figure 3.6(a) shows the range map for the skull in Figure 3.5(b). The following steps generate a range map for the skull surface:



(a) Extracted Skin Surface

(b) Extracted Skull Surface

(c) The skin and Skull Surface displayed together

(d) Another view

Figure 3.5: A particular implementation of the *Marching Cubes* algorithm (Courtesy J. O. Lachaud) was used to extract the above surfaces. For the skin surface, the threshold value was set to 10 and for skull surface it was set to 70. The threshold values were chosen manually. The CT dataset was provided by Lachaud. In (c) and (d), the skull is colored blue and the skin is rendered with transperancy for easy visualization.

Steps:

1: Rotate and translate the skull so that the y-axis passes through the top of the skull and between the lower part of the mandible.
2: Generate N_p equally spaced planes parallel to the $x-z$ plane. N_p is the height of the range map.
3: **for all** Planes **do**
4: Cast N_r rays radially outwards from the center of the plane (the point where y-axis intersects this plane). N_r is the width of the range map.
5: **for all** Rays **do**
6: Set y = y-value for the plane.
7: Set θ = angle between the ray and the negative z-axis
8: Intersect a ray with the skull (see Appendix B for details).
9: **if** the ray intersects the skull **then**
10: Set r=radius of the intersected point which is farthest from the center of the plane.
11: **else**
12: Set $r = 0$.
13: **end if**
14: **end for**
15: **end for**

The range map thus created is used to generate face-masks, by modifying it in an image manipulation package, such as Adobe Inc's Photoshop. We construct a regular triangulated mesh (Figure 3.7) in which every vertex correspond to a pixel in the range map, and each vertex is assigned (r, θ, y) values depending upon the corresponding pixel. The regions containing the artifacts, holes, and cavities are identified in the face mask, and the algorithm in Section 3.1.1 stretches a membrane over these regions. We then modify vertices of the regular triangulated mesh to reflect the new pixel values in the

(a) Range map (b) *fillmask* (c) *discardmask*

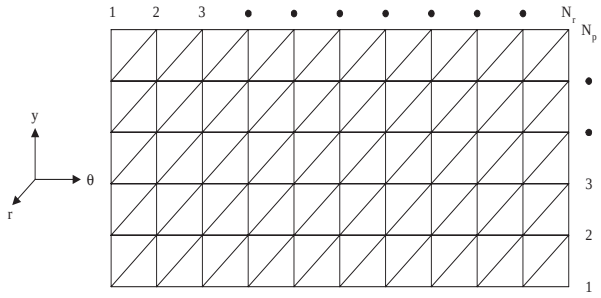
Figure 3.6: Range map created by sampling points (x, y, z) on the outer surface of the skull and converting sampled points into cylindrical coordinates (θ, r, y). The range map is 50×50; i.e., 2500 points are sampled on the skull surface (a). The *fillmask* (b) and the *discardmask* (c) are generated from the height map (a) in Adobe Photoshop.

range map. This results in a triangulated mesh approximating the exosurface of the skeletal foundation without any holes and cavities. Note that the original skull illustrated in Figure 3.5(b) contains 247100 triangles, whereas the skull exosurface illustrated in Figure 3.8 contains approximately 4700 triangles. The complexity of the constructed exosurface is independent of the complexity of the original skull extracted from the CT data, and depends solely upon the height and width of the range map. In general, the number of triangles in the constructed exosurface is less than twice the height of the range map times the width of the range map. The number of triangles in the constructed exosurface is small; therefore, we do not need to apply a mesh simplification algorithm on the original skull (Figure 3.5(b)).

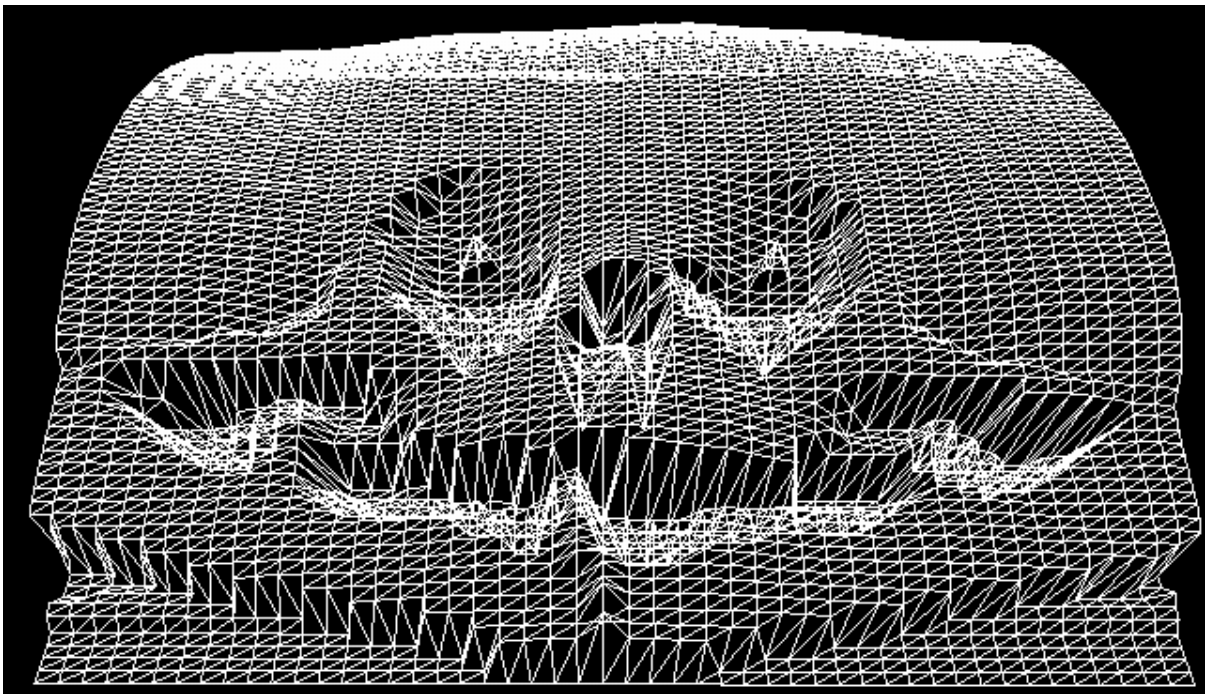
Currently, we are using two kinds of face masks, as follows;

- *discardmask*: The algorithm simply discards any region painted black in the *discardmask*. For the purposes of facial animation, we are not interested in the back of the skull which can be discarded using this mask.

- *fillmask*: The algorithm attempts to stretch membrane over regions painted black



(a)



(b)

Figure 3.7: Regular triangulated mesh, here each vertex corresponds to a pixel in range map, (r, θ, y) values for vertices are computed using the corresponding pixels (a). Regular triangulated mesh constructed using the range map in Figure 3.6(a), it has 2500 vertices and 4802 triangles (b).

in the *fillmask*. A *fillmask* can be used to identify undesirable regions on the skull, such as the eye orbits, the nasal cavities and other artifacts.

Hole-Filling Algorithm

We now explain the algorithm for constructing the skull exosurface. This algorithm takes the set of points sampled on the outermost surface of the skull along with the face masks and creates a triangulated representation of the skull exosurface. The points in the undesirable regions are assigned a value "FILL". We use the membrane interpolation method (Terzopoulos1988) to compute reasonable range values for these points. The r values for the "FILL" points are initially set to zero and adjusted through repeated averaging with neighboring r values. The relaxation algorithm iterates until the maximum change in the interpolated values drops below some small pre-specified constant ϵ. Once all the points are handled i.e., their r values are computed, the points are converted into Cartesian coordinates and triangles are generated. The algorithm consists of following steps:

1: Let S be an empty triangulation.

2: Range map: $r = (\theta, y)$, $\theta = \{\theta_1, \theta_2, \cdots, \theta_{N_r}\}$ where $0^\circ \leq \theta_i < \theta_{i+1} \leq 360^\circ$, $i = \{1, 2, 3, \cdots, N_r - 1\}$ and $y = \{y_1, y_2, \cdots, y_{N_c}\}$ where $y_{\max} \geq y_j > y_{j+1} \geq y_{\min}$, $j = \{1, 2, 3, \cdots, N_c - 1\}$.

3: For all points in undesirable regions, set flag = "FILL"

4: Set the r values of all "FILL" points to 0

5: Compute the r values of all "FILL" points using the mesh relaxation algorithm described in Section 3.1.1.

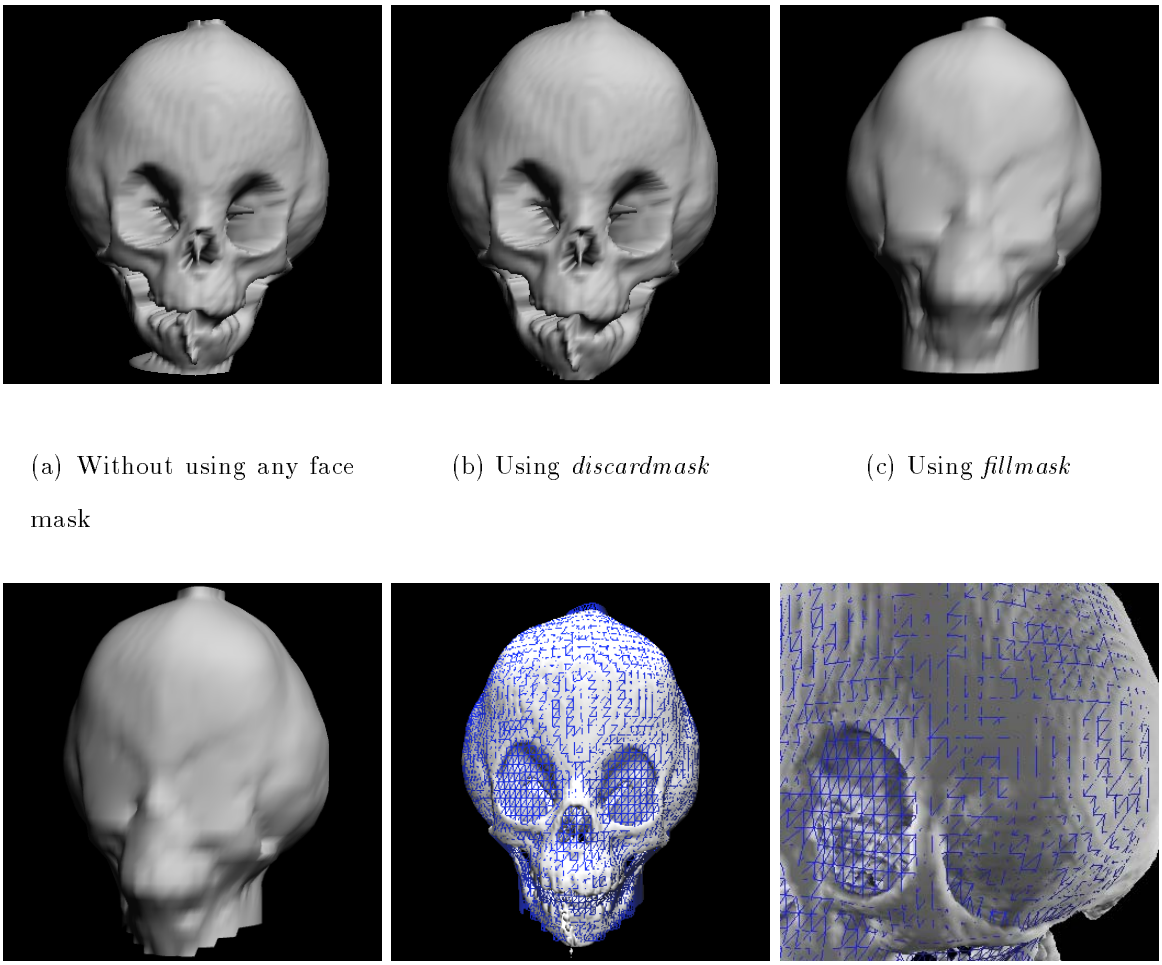
6: **for** $\theta_i \in \{\theta_1, \theta_2, \cdots, \theta_{N_r-1}\}$ **do**

7: **for** $y_j \in \{y_1, y_2, \cdots, y_{N_c-1}\}$ **do**

8: Assume $r_{(i,j)} = (\theta_i, y_j)$, set $p_{(i,j)} = (r_{(i,j)} \cos(\theta_i), y_j, r_{(i,j)} \sin(\theta_i))$.

9: Set $t^1_{(i,j)} = \{p_{(i,j)}, p_{(i,j+1)}, p_{(i+1,j)}\}$.

10: Set $t^2_{(i,j)} = \{p_{(i,j+1)}, p_{(i+1,j+1)}, p_{(i+1,j)}\}$.



(a) Without using any face mask

(b) Using *discardmask*

(c) Using *fillmask*

(d) Using both *fillmask* and *discardmask*

(e) The skull and the exosurface

(f) A close-up view of the skull and the exosurface

Figure 3.8: Skull exosurface constructed using the Hole-Filling algorithm.

11: Add triangles $t^1_{(i,j)}$ and $t^2_{(i,j)}$ to S.

12: **end for**

13: **end for**

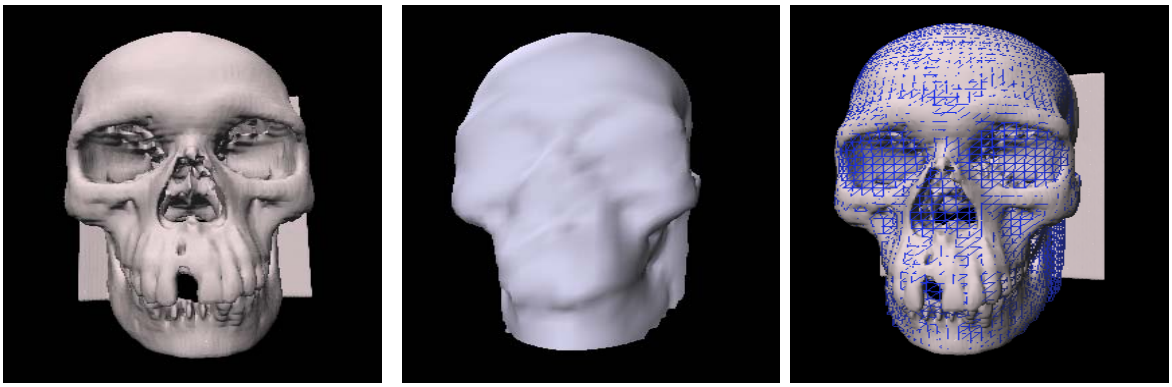
S is the triangulated exosurface. Figure 3.8(a) illustrates the skull exosurface constructed from the range map shown in Figure 3.6(a). The outermost surface of the skull is segmented; however, the regions around the eye orbits, nasal cavities and the upper part of the mandible still contains cavities. Moreover, an artifact of the CT data is also

visible in the chin region. A carefully designed *fillmask* can be used to smooth these regions. The user can generate a *fillmask* by painting these regions black, thereby forcing the algorithm to stretch a membrane over these regions. This smoothes out the holes and removes the artifact. Figure 3.8(d) illustrates the skull exosurface constructed using the *fillmask* shown in Figure 3.6(b). The smoothing out of the cavities is visually confirmed by comparing the two surfaces (Figure 3.8(a) and Figure 3.8(d)).

The exosurface in Figure 3.8(d) contains 4172 triangles, whereas the actual skull contains 237736 triangles. The range map used to construct the exosurface is 50×50 (Figure 3.6(a)). The range map computation takes 2-3 hours on a 450 MHz Pentium-II machine running Windows NT 4.0. Once the range map is computed, it takes about 3-5 minutes to compute the skull exosurface. Figure 3.8(e) displays the skull and the exosurface together.

3.3 Summary

We have described a method for a adapting the generic facial mesh to Cyberware data. We have also developed a scheme for generating smooth skull exosurface from CT data. Our approach has produced acceptable results in practice. In each case, the constructed exosurface is smooth and it closely approximates the outer-surface of the skull. The number of triangles in the exosurface is much smaller than that for the skull produced by the MC algorithm. The method also requires very little user intervention. The user needs to generate the face masks to identify the undesirable regions; the face masks can be generated from the range map in 5-10 minutes using any image manipulation package. We use Adobe Photoshop for this purpose. Computing the range map for the skull is the most expensive step, it takes 4-5 hours of processing time, on a Pentium-II 450 MHz machine running Windows NT 4.0, to generate a 100×100 range map from a skull containing 400000 triangles. The current implementation is $O(mn)$ where m is



(a) Skull (b) Generated Exosurface (c) The skull and the exosurface

(d) Range Map (e) *fillmask* (f) *discardmask*

Figure 3.9: The MC algorithm extracts the skull from the CT data using the isosurface value = 75 (a). This skull has 477708 triangles. We compute a 50×50 range map from this skull (d). *fillmask* (e) and *discardmask* (f) are generated from the range map and the exosurface is constructed using the Hole-Filling algorithm, this surface contains 4172 triangles (b).

the number of rays and n is the number of triangles. The user can choose to generate a range map of any size, however, a 50×50 range map is enough to capture the structure of the skull. Figure 3.9 illustrates the skull produced by the MC algorithm along with the skull exosurface constructed using our approach.

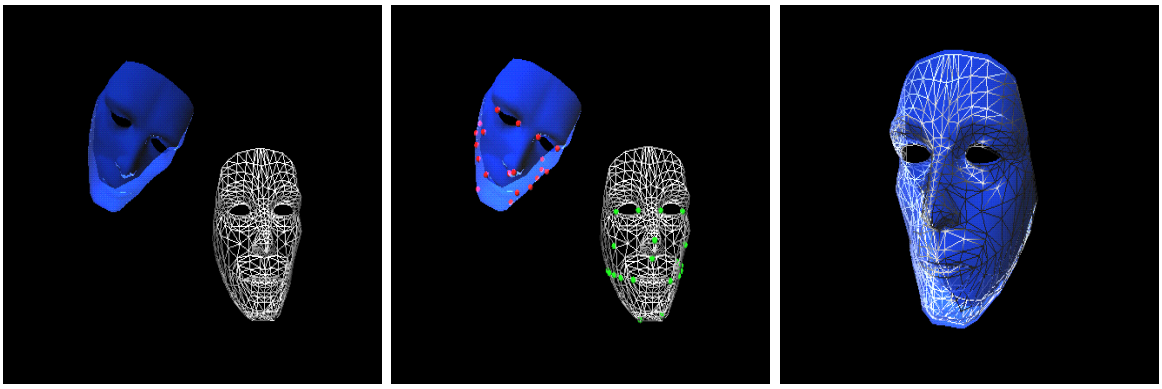
Chapter 4

Registration of the CT and Cyberware data

The next step in the construction of the volumetric facial soft-tissue model involves registering the CT and the Cyberware data. The extracted skin surface and the adapted facial mesh are used to register these datasets. This chapter describes the registration process.

We propose a surface-based registration technique which computes a global affine transformation that aligns the skin surface and the facial mesh assuming both the surfaces are triangulated. The transformation is applied on the facial mesh to register the CT and Cyberware data. Our approach falls under the category of segmentation-based registration methods discussed in Section 2.2.3, since first we segment anatomically similar structures from the CT and Cyberware data and then use these structures to register the two datasets.

The user interactively identifies landmark points on both surfaces, thereby generating two sets of points. The point-correspondences are established across these point-sets, and the algorithm computes the transformation matrix by minimizing the least-squares distances between the corresponding points. Once the transformation is computed, the



(a) Before registration

(b) The points picked for registering the two face meshes

(c) The registered meshes

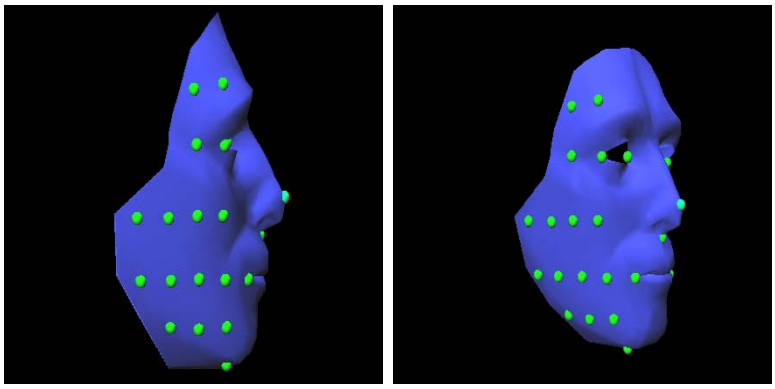
Figure 4.1: Two generic face meshes, before and after registration

two surfaces are spatially aligned by transforming each vertex of the adapted facial mesh.

It is assumed that the CT and the Cyberware data are acquired from the same person; therefore, the skin surface and the adapted facial mesh have similar local structure and only global scaling, rotation and translation are required to correctly register them. We, therefore, only compute the global affine transformation. The method is specifically designed to register the CT and the Cyberware data; however, it is applicable to any 3-D surface registration problem where the surfaces have similar local structure. Figure 4.1 illustrates two similar facial meshes before and after the registration.

4.1 Choice of the Landmarks Points

For the purpose of registration, each surface is represented by a carefully chosen set of point landmarks. It is important for accurate registration to capture as much of the surface structure in this point-set as possible. First, the desirable regions in both the surfaces are identified—a desirable region has similar local structure in both the surfaces and has enough visual cues to set up the point correspondences correctly. Second, the



(a) Side View (b) Another view

Figure 4.2: A scheme of picking points on a surface

points are picked in these desirable regions and point correspondences across the two surfaces are set up.

We do not have any texture information for the facial skin surface extracted from the CT data; therefore, we cannot use the texture information to set up the point-correspondences. We use anatomical features, the tip of nose, the corner of eyes and the lips, in the desirable regions of both surfaces to pick points and set up the correspondence pairs. Figure 4.2 shows the points picked on a surface. We need at least four non-coplanar pairs of points to compute the transformation matrix; however, in practice we usually pick 15 to 20 points, since 4 points do not adequately capture the complex structure of the human face.

4.2 Computing the Transformation Matrix

Let P be the set of N points picked on the facial skin surface extracted from the CT data and Q be the set of N points picked on the generic facial mesh adapted to the Cyberware data. Assume that $\mathbf{p}_i \in P$ and $\mathbf{q}_i \in Q$, $i = \{1, 2, 3, ..., N\}$, are corresponding points in the two point-sets P and Q. Since the two surfaces are related through a global

affine transformation, the point-sets P and Q are also related through the same affine transformation.

An affine transformation can be represented by a 4×4 matrix in the projective space. We convert the points $\mathbf{p}_i \in P$ and $\mathbf{q}_i \in Q$, to the homogeneous coordinates, generating sets of homogeneous points P' and Q' s.t. $\forall \mathbf{p}_i \in P$, $\tilde{\mathbf{p}}_i = [\mathbf{p}_i^T, 1]^T \in P'$ and $\forall \mathbf{q}_i \in Q$, $\tilde{\mathbf{q}}_i = [\mathbf{q}_i^T, 1]^T \in Q'$, and if $\mathbf{p}_i$ and $\mathbf{q}_i$ are corresponding points in P and Q then $\tilde{\mathbf{p}}_i$ and $\tilde{\mathbf{q}}_i$ are corresponding points in P' and Q'. P' and Q' are also related through an affine transformation:

$$\tilde{\mathbf{p}}_i = \mathbf{A}\tilde{\mathbf{q}}_\mathbf{i} + \mathbf{n}_i, \tag{4.1}$$

where $\mathbf{A}$ is a 4×4 affine transformation matrix and $\mathbf{n}_i$ is a noise vector. We seek a matrix $\mathbf{A}$ so as to minimize

$$E = \sum_{i=1}^{N} \|\tilde{\mathbf{p}}_i - \mathbf{A}\tilde{\mathbf{q}}_\mathbf{i}\|^2 \tag{4.2}$$

We describe a non-iterative algorithm based on *Singular Value Decomposition* (SVD) to solve (4.2) (Press et al.1992). We begin by converting the above problem into a *linear least-square* problem. Rewriting (4.1) as follows:

$$\tilde{\mathbf{p}}_i = \mathbf{A}\tilde{\mathbf{q}}_\mathbf{i} = \begin{bmatrix} \mathbf{a}_1 \\ \mathbf{a}_2 \\ \mathbf{a}_3 \\ \mathbf{a}_4 \end{bmatrix} \tilde{\mathbf{q}}_i = \begin{bmatrix} \mathbf{a}_1 \cdot \tilde{\mathbf{q}}_i \\ \mathbf{a}_2 \cdot \tilde{\mathbf{q}}_i \\ \mathbf{a}_3 \cdot \tilde{\mathbf{q}}_i \\ \mathbf{a}_4 \cdot \tilde{\mathbf{q}}_i \end{bmatrix} = \begin{bmatrix} \tilde{\mathbf{q}}_i^T & 0 & 0 & 0 \\ 0 & \tilde{\mathbf{q}}_i^T & 0 & 0 \\ 0 & 0 & \tilde{\mathbf{q}}_i^T & 0 \\ 0 & 0 & 0 & \tilde{\mathbf{q}}_i^T \end{bmatrix} \begin{bmatrix} \mathbf{a}_1^T \\ \mathbf{a}_2^T \\ \mathbf{a}_3^T \\ \mathbf{a}_4^T \end{bmatrix}, \tag{4.3}$$

where $\mathbf{a}_i$ is the ith row of the matrix $\mathbf{A}$. Let $\mathbf{a}_{(16\times1)} = [\mathbf{a}_1\mathbf{a}_2\mathbf{a}_3\mathbf{a}_4]^T$.

$$\tilde{\mathbf{p}}_{i,(4\times1)} = \mathbf{M}_{i,(4\times16)}\mathbf{a}_{(16\times1)}. \tag{4.4}$$

Equation (4.4) shows a system of linear equations. Here, the number of unknowns is 16 and the number of equations is $K = 4$. The number of equations K can be smaller than, equal to, or greater than 16 depending upon the number of corresponding points N in the two sets, usually $K = 4N$. In general, the number of equations K is greater than N

since we choose around 15 to 20 points on both surfaces. The set of linear equations is, therefore, *over-determined*. We compute a compromise solution which comes closest to satisfying all the equations in the *least-squares* sense. For $N > 1$, (4.4) becomes

$$\tilde{\mathbf{p}} = \mathbf{M}\mathbf{a} \tag{4.5}$$

where $\tilde{\mathbf{p}}_{i,(4N\times 1)} = \left[\tilde{\mathbf{p}}_1^T \tilde{\mathbf{p}}_2^T \tilde{\mathbf{p}}_3^T \cdots \tilde{\mathbf{p}}_N^T\right]^T$, $\mathbf{M}_{(4N\times 16)} = \left[[\mathbf{M}_1]^T [\mathbf{M}_2]^T [\mathbf{M}_3]^T \cdots [\mathbf{M}_N]^T\right]^T$ and $\mathbf{a}_{16\times 1}$ is the vector of unknowns.

We use SVD to solve this system. Since, we have 16 unknowns, we choose at least 4 pairs of points so that the system is not *under-determined*. The SVD decomposes $\mathbf{M}$ into a column-orthogonal matrix $\mathbf{U}$, a diagonal matrix $\mathbf{W}$ with positive or zero elements (singular values) and an orthogonal matrix $\mathbf{V}$.

$$\mathbf{M} = \mathbf{U} \cdot \mathbf{W} \cdot \mathbf{V}^T \tag{4.6}$$

The solution for equation 4.5 is

$$\begin{aligned} \mathbf{a} &= \mathbf{M}^{-1}\tilde{\mathbf{p}} \\ \mathbf{M}^{-1} &= \mathbf{V} \cdot diag\left(\frac{1}{w_i}\right) \cdot \mathbf{U}^T \end{aligned} \tag{4.7}$$

where w_i's are the diagonal elements of $\mathbf{W}$. This solution of $\mathbf{a}$ is the best in the *least-squares* sense (for a proof see (Press et al.1992)).

4.2.1 Discussion

Affine transformation can take care of the scaling, translation, and the rotation; however, it introduces an undesirable artifact—*skew*. Although the SVD tries to give a solution every time, a particular choice of the corresponding points can be such that it is impossible to compute a unique affine transformation. We have the following three possibilities:

- If the points picked on the surfaces are not coplanar, the solution is unique.

- If the points picked on the surfaces are coplanar, there are two solutions which will give identical results (a unique *rotation* and a unique *reflection*).

- If the points picked on the surfaces are collinear, there are infinitely many solutions (infinite *rotations* and *reflections*).

The above situations do not arise in our application due to the structure of the human face; however, one should be aware of these limitations.

4.3 Surface Transformation

Once the transformation matrix is computed, the next step is to transform the adapted facial mesh surface. Since the facial mesh is a triangulation, only the vertices of the triangles need be transformed to transform the whole surface.

A triangle of the facial mesh is defined as a triplet $\{v_1, v_2, v_3\}$, here $v_1, v_2, v_3 \in I$ and $0 \leq v_1, v_2, v_3 < t$, such that v_1, v_2, v_3 are indices into the vertex array $V(x) : I \rightarrow R^3$. The following algorithm transforms all the vertices' coordinates, thereby transforming the surface:

1: $i = 0$

2: **for all** $i \geq 0$ and $i < t$ **do**

3: $\quad \mathbf{v}_h = [V(i)^T \ 1]^T$

4: $\quad \mathbf{v}_h^{new} = \mathbf{A}\mathbf{v}_h$

5: $\quad V(i) \leftarrow \left(\frac{\mathbf{v}_h^{new}[1]}{\mathbf{v}_h^{new}[4]}, \frac{\mathbf{v}_h^{new}[2]}{\mathbf{v}_h^{new}[4]}, \frac{\mathbf{v}_h^{new}[3]}{\mathbf{v}_h^{new}[4]} \right)^T$

6: **end for**

Figure 4.1(c) illustrates the registered facial surfaces. The points shown in Figure 4.1(b) are used to set up the correspondences and an affine transformation is computed (as discussed in Section 4.2). The transparent surface is then transformed to align the two surfaces.

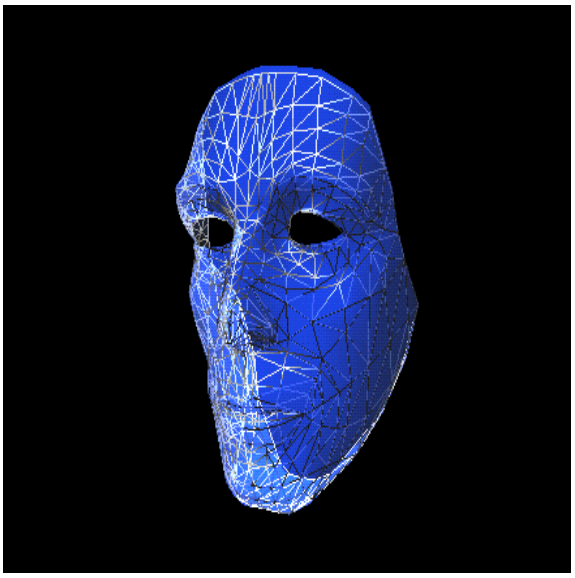
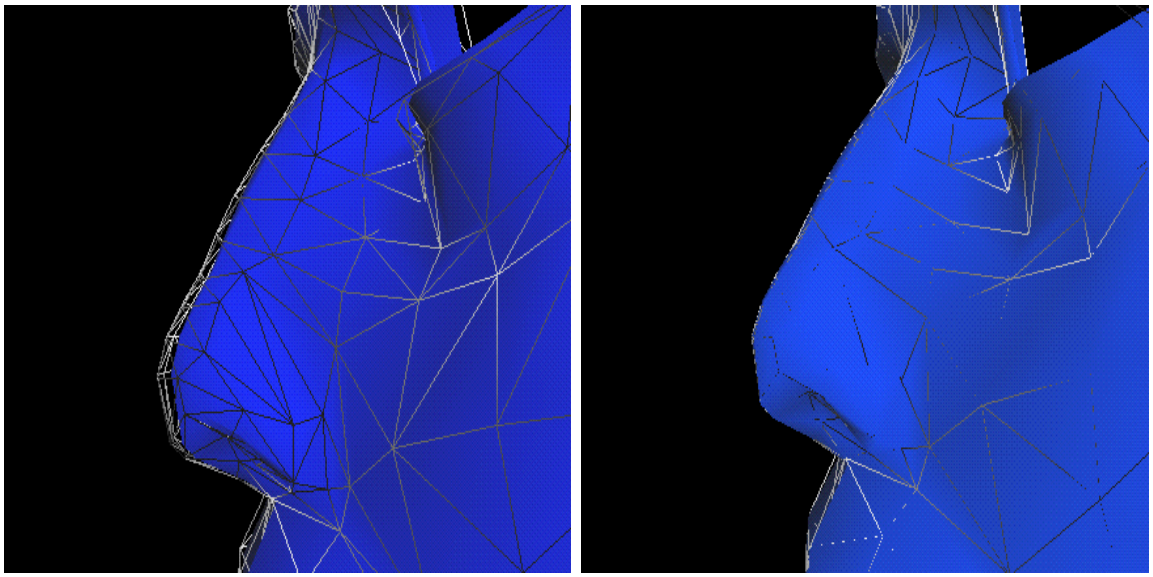


Figure 4.3: The registered facial surfaces after deforming the transparent surface

4.4 Surface Deformation

Once the facial mesh is transformed, we observe that the registration quality is further improved by locally deforming the mesh. The mesh is deformed by replacing its vertex coordinates with their respective closest points on the other surface. This heuristic does not always give good results. If the surfaces are mis-aligned, deforming a surface further deteriorates the registration quality, but careful use of the heuristic yields good results. Figure 4.4(a) illustrates a close-up view of the two registered surfaces before applying the deformation. Comparing it with Figure 4.4(b), which shows the same close-up view after applying the deformations to one of the surfaces, clearly shows that the registration quality has improved in the nose region. Figure 4.3 illustrates the two registered surfaces after applying the local deformations. A surface is deformed using the following steps:

1: **for all** Vertices v of the first surface **do**
2: $\quad \mathbf{v}' =$ The point on the second surface that is closest to $\mathbf{v}$
3: $\quad \mathbf{v} = \mathbf{v}'$
4: **end for**



(a) Before (b) After

Figure 4.4: The registered face mesh, before and after applying local surface deformations

4.5 The Quality of Registration

A good method to ascertain the registration quality is to visually inspect the registration results; however, a more objective quantitative measure of the registration quality is the distance between the two surfaces. We define the distance between the two surfaces to be the mean of the shortest distances between the points on the first surface and the second surface. We propose *inter-surface-distance* (ISD) as a measure of registration quality.

Steps for Computing the ISD

1: Let S_1 and S_2 are the first and second surfaces respectively. v_1 is a vertex of S_1 and v_2 is a vertex of S_2. n_1 and n_2 are total number of vertices in S_1 and S_2 respectively.

2: $d_1 = \sum_{\forall v_1} \|v_1 - v_2'\|$, where v_2' is the point on S_2 that is closest to v_1.

3: $d_2 = \sum_{\forall v_2} \|v_2 - v_1'\|$, where v_1' is the point on S_1 that is closest to v_2.

4: $\mathrm{ISD}(S_1,S_2) = \frac{n_2 d_1 + n_1 d_2}{2 n_1 n_2}$

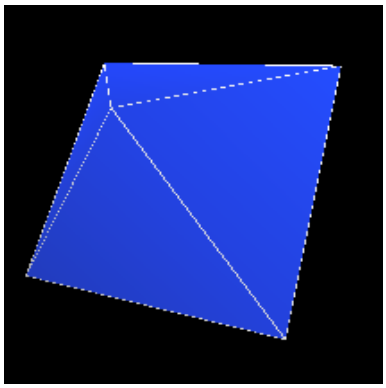


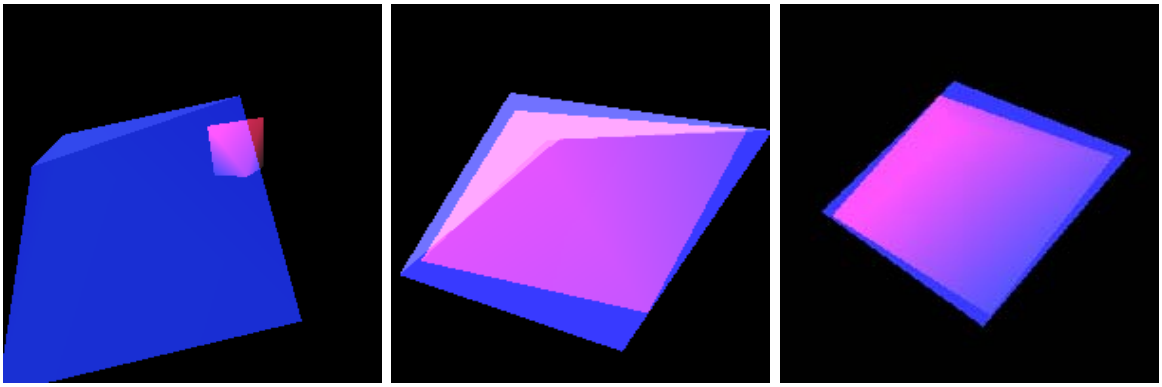
Figure 4.5: Surface 1

Rotation (X,Y,Z)	Translation (X,Y,Z)	Scaling (X,Y,Z)
(90°,15°,5°)	(0,10,-10)	(1,1.5,1.3)

Table 4.1: Transformation Values used for generating Surface 2 and Surface 4

We have found ISD to be a good indicator of the registration quality. For any two surfaces, ISD increases steadily as we increase the mis-alignment. The surfaces shown in Figure 4.1(a) are completely mis-aligned and their ISD value is 9.85867. After registration (Figure 4.1(c)) the ISD value decreases to 0.0720825, and after locally deforming one of the surfaces (Figure 4.3) the ISD value further decreases to 0.031227. ISD value does not drop to zero because we are deforming only one of the two surfaces, and vertices of the second surface does not necessarily lie on the first surface which prevents ISD from becoming 0.

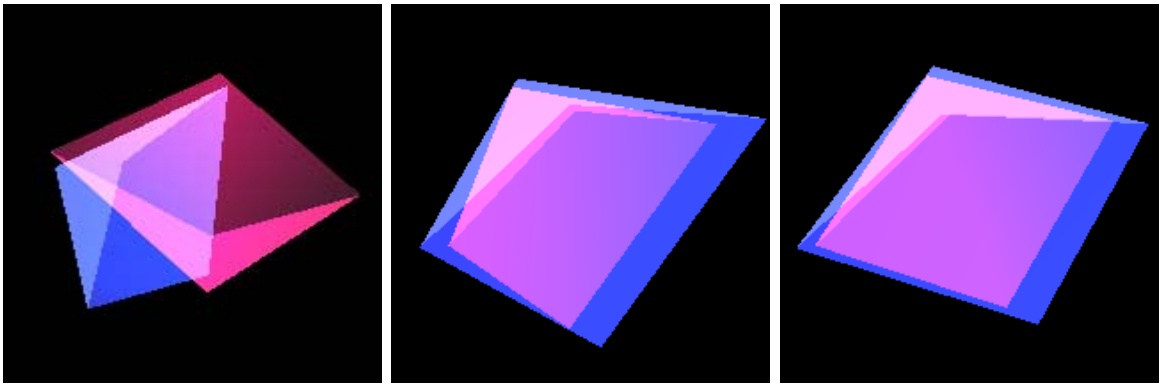
The maximum tolerable ISD value depends upon the surfaces used for the registration, and is proportional to the sizes of the bounding boxes of the surfaces. For any two surfaces, this value can be fixed by visually inspecting the registered surfaces. For the surfaces shown in Figure 4.1(a), the maximum tolerable ISD value before applying the deformations is found to be 0.3. This value was fixed after performing many registration and visually inspecting the results.



(a) Surface 1 and Surface 2 before registration

(b) After registration, ISD = 0.168158

(c) After deformations, ISD = 0.16382



(d) Surface 3 and Surface 4 before registration

(e) After registration, ISD = 3.27

(f) After deformations, ISD = 3.08

Figure 4.6: Pyramid surfaces

We now provide an example which shows that the maximum tolerable ISD value depends upon the sizes of the surfaces. We choose the surface shown in Figure 4.5. We call this surface *Surface 1*. *Surface 1* is transformed, using the transformations shown in Table 4.1. We call the transformed surface *Surface 2*. The maximum tolerable ISD value for *Surface 1* and *Surface 2* is found to be 0.168158. We then scale *Surface 1*. The scaled-up version of *Surface 1* is called *Surface 3*. We transform *Surface 3* using the same transformations, we call the transformed surface *Surface 4*. The maximum tolerable ISD value for *Surface 3* and *Surface 4* is found to be 3.27. In each of the cases, the two surfaces are related through the same global affine transformation. The only difference in the above cases is in the dimensions of the bounding boxes of the surfaces. The dimensions of the bounding box of *Surface 1* are $2 \times 2 \times 1$, and the dimensions of the bounding box of *Surface 3* are $30 \times 30 \times 15$. The increase in the sizes of the bounding boxes have increased the maximum tolerable ISD value.

4.6 Results

We have tested our registration algorithm on different surfaces and in each case the user was able to perform a reasonable job in less than 10 minutes. For the human face, it takes less than 5 minutes to specify the correspondences across the two surfaces. Once the correspondences are specified, the algorithm takes less than 2 minutes to compute the transformation matrix on an SGI ONYX machine. The user can visually inspect the results, compute the ISD value, and apply local deformations. If the registered surfaces are close, application of local deformations usually improves the registration quality. In case of the CT and the Cyberware data, transforming the facial mesh adapted to the Cyberware data registers the two datasets. The performance of the registration algorithm is evaluated against various factors and results are provided in Appendix C. These results indicate that the performance of the algorithm depends solely on the structure of the

surface to be registered—how similar or dissimilar are the surfaces, and on error in the landmarks points—how accurately correspondences are established across the surfaces; and it does not depend upon the initial mis-alignment of the two surfaces.

Chapter 5

Facial Skin Tissue Model Construction

The final phase of the construction of the facial soft-tissue model, involves the computation of the facial skin thickness. The physically-based facial animation model proposed by Lee *et al.* consists of prismatic volume elements (Lee, Terzopoulos and Waters1995). These volume elements are constructed by extending the triangles of the facial mesh adapted to the cyberware data. We also generate the prismatic volume elements by extending the triangles of the facial mesh inwards, but unlike Lee *et al.* (1995), we take into account the actual thickness of the skin. Moreover, our facial model employs a more accurate skeletal foundation. We conjecture that for the purposes of facial animation, the skull exosurface (see Chapter 3) can serve as the skeletal foundation.

In this chapter, we explain the process of computing the facial skin thickness. We also explain the construction of the prismatic volume elements. First, we compute the facial skin thickness for each vertex of the facial mesh. The skin thickness is defined as the distance between the vertex and the underlying skull exosurface along the negative vertex normal. This definition of thickness is invalid in the eyes, the lower part of the jaw and the nasal cavity regions, because in these regions skin thickness is independent of the

distance between the outer surface of the skin and the skull beneath it. We, therefore, compute thickness values for the vertices in these regions by interpolating the thickness values for the neighboring vertices. We use a labeled face mesh, therefore, we know which vertex lies in which region *a priori*.

After computing the skin thickness values for all the vertices, we protrude the triangles towards the underlying skeletal foundation to form the prismatic volume elements. The thickness of each prismatic volume element depends upon the facial skin thickness in that region. Once the prismatic elements are set up, the facial muscles are embedded in the anatomically correct positions. The positions of these facial muscles are known with respect to the generic facial mesh. The facial mesh is then texture mapped to create a realistic looking face.

5.1 Computing Skin Thickness and Constructing Prismatic Volume Elements

We now explain the soft-tissue construction algorithm in more detail. The algorithm assumes that the generic facial mesh is adapted to the CT data, the surface representations of the skull and the epidermis are extracted from the CT data, the CT and the Cyberware datasets are registered, and the skull exosurface is constructed from the skull. The algorithm is as follows:

1: Set $F = 1$ for all vertices.

2: Identify vertices in the eyes, the lower part of the jaw and the nasal cavity regions, and set their $F = 0$ and also set their $O_v = 0$.

3: **for all** Vertices v with F = 1 **do**

4: Project a ray r along the negative normal of vertex v.

5: Intersect r with the skull exosurface.

6: **if** (Intersection of r with the skull exosurface is successful) AND (Angle between r and the normal of the intersected triangle is less than 90°) **then**

7: Let the intersected point be p. The skin thickness for vertex v is, $S_v = \sqrt{(v-p)^2}$.

8: **else**

9: Set $F = 2$ and $O_v = 0$ for vertex v.

10: **end if**

11: **end for**

12: Set skin thickness values for all vertices with $F \in \{0, 2\}$ equal to 0.

13: **repeat**

14: Set $d = 0$

15: **for all** Vertices v with $F \in \{0, 2\}$ **do**

16: $O_v = S_v$

17: S_v = average of the thickness values of the neighbors with $F = 1$.

18: Set $d = \max(d, S_v - O_v)$

19: **end for**

20: **until** $d < \epsilon$.

Here, ϵ is some pre-defined value.

The next phase consists of extruding the triangles towards the skull exosurface to construct the prismatic volume elements, as follows:

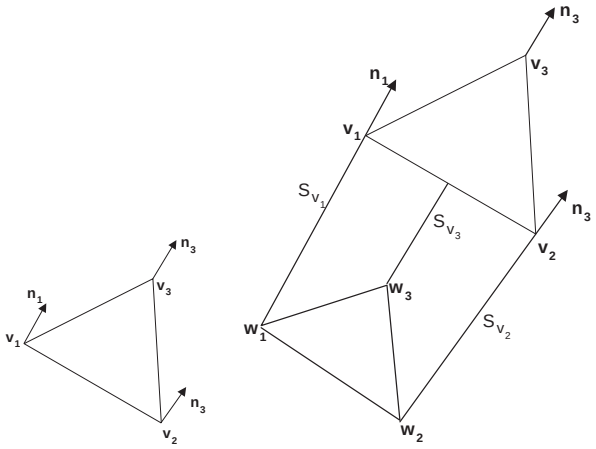
1: **for all** Triangles t in the face mesh **do**

2: Let $\mathbf{v}_1$, $\mathbf{v}_2$ and $\mathbf{v}_3$ be the vertices of the triangle t, and let $S_{\mathbf{v}_1}$, $S_{\mathbf{v}_2}$ and $S_{\mathbf{v}_3}$ be the facial skin thickness value for the three vertices respectively. Let $\mathbf{n}_1$, $\mathbf{n}_2$ and $\mathbf{n}_3$ be the vertex normals for t.

3: $\mathbf{w}_i = \mathbf{v}_i - S_{\mathbf{v}_i}\mathbf{n}_i$ for $i = \{1, 2, 3\}$.

4: The set $\{\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3, \mathbf{w}_1, \mathbf{w}_2, \mathbf{w}_3\}$ defines the prismatic volume element for triangle t (see Figure 5.1).

5: **end for**



(a) Triangle (b) Prismatic Volume Elements

Figure 5.1: A triangle and the corresponding prismatic volume element. The three vertices of the triangle (a) are extruded along the negative normal direction to construct the prismatic element (b).

5.2 Discussion and Results

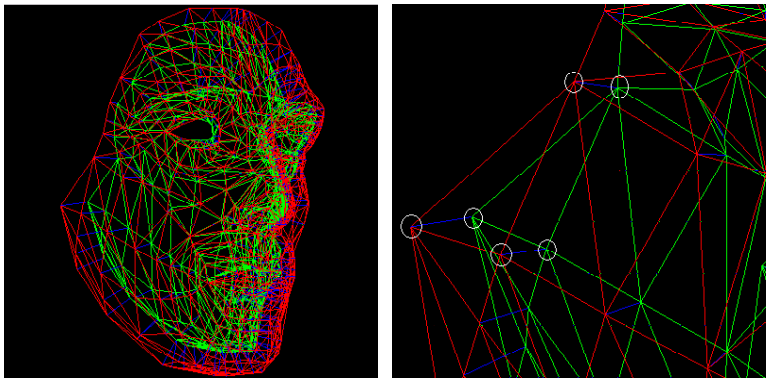
Figure 5.4 illustrates the facial soft-tissue model constructed from the generic mesh (Figure 3.4(a)) and the skull (Figure 3.9(a)). In Figure 5.4, the red wireframe represents the generic facial mesh, whereas, the blue lines represents the thickness values for the vertices of the generic mesh. Currently, we do not have CT and Cyberware data from the same person; therefore, to test the algorithm we manually deform the generic mesh to approximate the epidermis surface for skull. Given CT data and Cyberware data of the same person, we will use the adapted facial mesh instead. First, we construct the smooth exosurface from the skull, using the hole-filling algorithm described in Chapter 3 for this purpose. Second, we compute the facial skin thickness values for all the vertices of the generic mesh and construct the prismatic elements.

We also construct the facial soft-tissue model for the generic skull in Figure 5.3. Once again, we transform the generic facial mesh to closely approximate the facial skin for the

generic skull. We first construct the exosurface for the skull. Next, we compute the skin thickness and construct the prismatic volume elements.

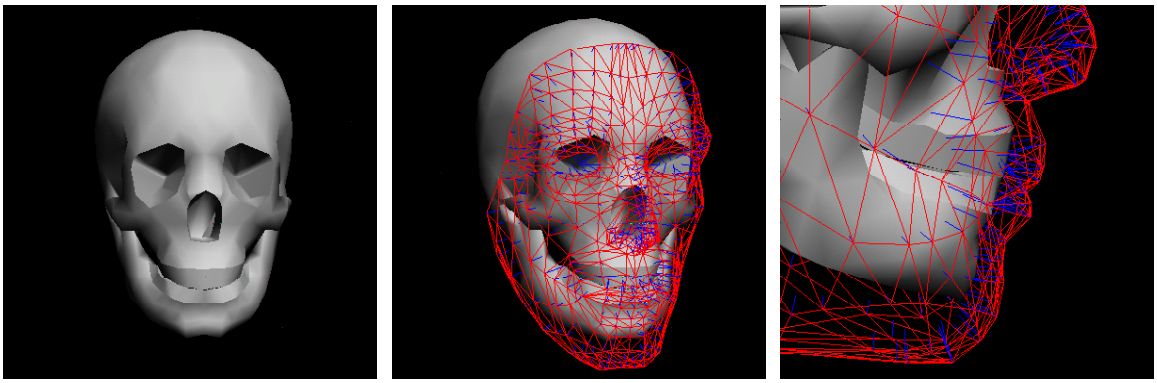
Visual inspection of the results suggests that the computed skin thickness values for the vertices are reasonable. Figure 5.4 and Figure 5.3 illustrate that the thickness values for the cheek vertices is larger compared to the thickness values for the forehead vertices, as expected. The number of prismatic elements generated by our approach is equal to the number of triangles in the facial mesh.

A shortcoming of our approach is that it can generate inverted prismatic elements in high curvature regions where facial skin thickness is large—the inverted prisms are found around the lips and the eyes. Not more than 3 to 5 percent of the prisms are inverted. The inverted prisms may pose problems when used in the Lee *et al.* facial animation system (Lee, Terzopoulos and Waters1995). Inverted prisms can be corrected by importing the 3-D soft-tissue model in to a 3-D modelling package. The facial soft-tissue model is constructed so that it may be incorporated into the facial animation system of Lee *et al.* (1995). At this point we have not tested the performance of our model, however, we conjecture that our model will produce more realistic facial animations.



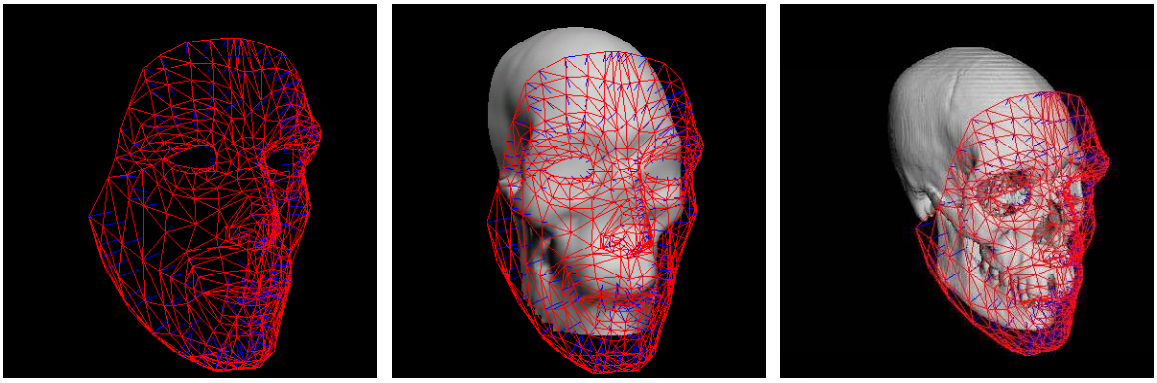
(a) Prismatic mesh (b) A prismatic volume element

Figure 5.2: Facial soft-tissue model consisting of prismatic volume elements. The generic facial mesh and the skull in Figure 3.9(a) are used to construct this model. Note that the thickness of the prismatic elements vary in different regions of the face, as expected.

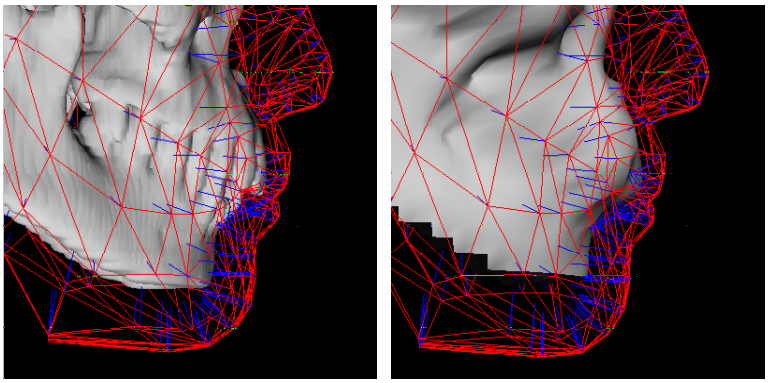


(a) (b) (c)

Figure 5.3: Facial soft-tissue model constructed from the generic mesh and the generic skull (a). Constructed model is displayed with the underlying skull (b).



(a) (b) (c)



(d) (e)

Figure 5.4: The facial soft-tissue model is displayed together with the underlying skull (c). Facial skin thickness value associated with each thickness is shown in (a). The exosurface constructed from the skull is used to compute the thickness value for the vertices of the generic mesh (b). Figure (d) illustrates the soft-tissue model with the underlying skull and Figure (e) illustrates the soft-tissue model with the underlying skull exosurface. Note that for the vertices in the nose region, the blue lines (these lines represent the thickness values) do not extend all the way to the underlying skull structure (d).

Chapter 6

Conclusions

6.1 Summary

With the advent of powerful new graphics systems, we will soon have facial models which not only can be animated, but which can also serve in cranio-facial surgery simulation systems. The motivation for our work has been to take another step towards such facial models. In this regard, we have aimed to develop anatomically accurate, biomechanically simulated models of the face. In particular, we have extended the facial animation model introduced by Lee, Terzopoulos, and Waters (1995) by incorporating facial soft-tissue thickness information and an accurate skull exosurface. To this end, we have developed algorithms to construct a more accurate soft-tissue model using information from CT and Cyberware datasets. We have also developed a surface based registration algorithm to register CT and Cyberware datasets. The results of the facial model construction scheme were found to be acceptable. Moreover, the steps involved in the process are simple and require only modest user interaction.

6.2 Future Directions

Our new model can be used for facial animation and we conjecture that because of the more accurate hard tissue geometry and skin thickness it will produce more realistic facial deformations. To date, however, our model has the following shortcomings:

1. Our current facial model cannot be used in advanced cranio-facial surgery simulation, because it lacks a solid model of the skull. A solid skull model is necessary for simulating operations like the cutting, moving and re-alignment of facial hard-tissues, and advanced cranio-facial surgery can not be carried out without these operations.

2. We also need to automate the process of facial soft-tissue construction. Currently, user intervention is required to generate face masks from the range map of the skull. We would need to develop image processing techniques to address this problem.

3. Another major improvement in the current scheme will be to automate the registration algorithm. Currently, the user establishes the point correspondences between the two surfaces. We would like a system which automatically establishes the correspondences. Snakes (Kass, Witkin and Terzopoulos1988) could be used to extract features on both the surfaces—the adapted facial mesh and the facial skin surface could be extracted from the CT data, and correspondences established using these features. For now, we have studied the performance of the registration algorithm on synthetic datasets; however, we need to evaluate the performance on real datasets.

4. In the final phase of the facial model construction, inverted prisms may be generated and user intervention is required to correct such prisms. We would like to automate this process.

5. The current implementation of surface-ray intersection algorithm is $O(mn)$ where n is the number of rays and m is the number of triangles, however there are many

ray-tracing optimizations that are applicable to this problem. We need to speed-up this algorithm.

We need to study the performance of the proposed scheme on actual clinical datasets—i.e., CT and Cyberware datasets acquired from the same person. Average measurements are available for tissue thickness among various ethnic groups, and these can provide first-order verification of our results. We also need to compare the animation results for our model with those of other existing facial animation techniques.

Future work will correct the shortcomings and ultimately lead to an anatomically accurate facial model which will produce more realistic facial animations and may also be used for the purposes of cranio-facial surgery simulation. Ultimately, we hope to develop a cranio-facial surgery simulation system which will provide surgeons with tools to operate upon facial models constructed from patients.

Appendix A

Marching Cubes

The *Marching Cubes* (MC) algorithm (Lorensen and Cline1987) extracts *isosurfaces* from 3-D datasets, where the isosurface threshold values are specified by the user. The 3-D dataset is usually stored as 2-D slices. The MC algorithm sets up logical cubes, choosing four pixels each from the adjacent slices (see Figure A.1). Each cube is then tested to determine whether or not the surface passes through it. The cube's vertices are assigned values 1 or 0 depending on whether or not the vertices' data values are greater than the threshold value. The vertices, which are assigned a value 1 are inside or on the isosurface. Once all the vertices are marked, we find which cubes intersect the surface. If all the vertices of a cube are not assigned the same value (1 or 0) than the cube intersects the surface. The next step is to determine the topology of the surface within the intersected cubes. There are eight vertices in each cube and two states for each vertex; therefore, there are only $2^8 = 256$ ways a surface can intersect a cube. 256 cases are reduced to 14 different cases using two different symmetries of the cube. The symmetries are:

- Complimentary Symmetry (if the vertex values are swapped; i.e., a vertex having a value 1 is assigned a value 0 and vice-versa).

- Rotational Symmetry (if the cube is rotated).

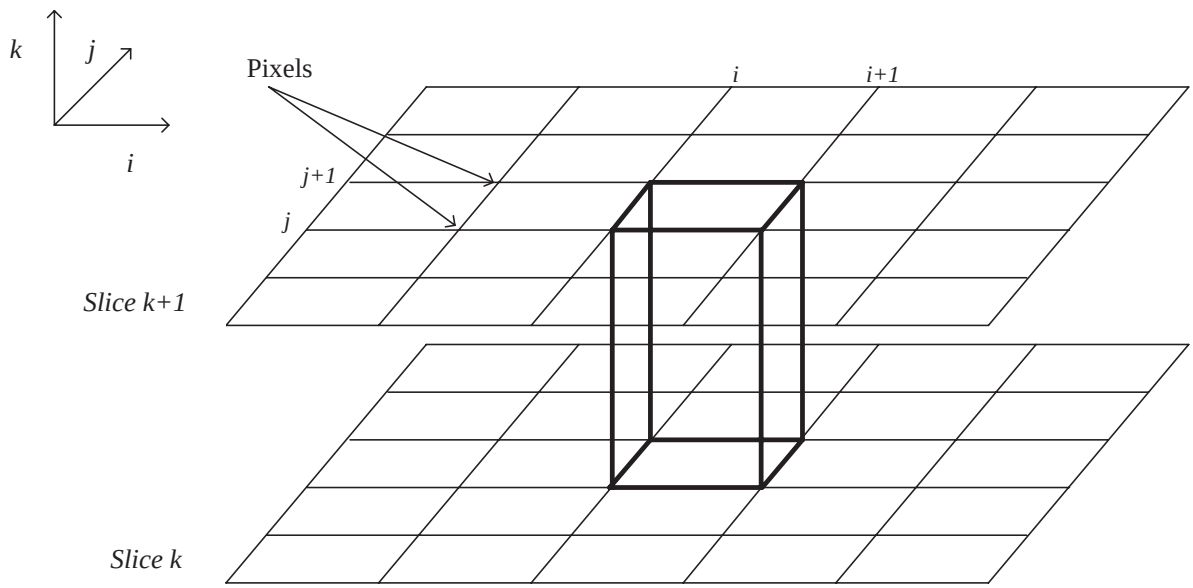


Figure A.1: A logical cube using eight vertices; four from two adjacent slices

Figure A.2 shows the 14 triangulation patterns. A lookup table identifies the intersected edges of the cube given the values (0 or 1) of its 8 vertices. An eight bit value, containing one bit for each vertex, is computed for each cube and used as an index in the lookup table to retrieve edge intersection information for the cube. Data values of the vertices defining the intersected edge are linearly interpolated along the edge to compute the surface-edge intersection point. As seen in the Figure A.2, the algorithm produces at least one and as many as four triangles per intersected cube. Next, normals for each triangle vertex are computed. A surface of constant density has a zero gradient component along the surface tangent. Therefore, the gradient vector is normal to the surface. The gradient vector is the derivative of the density function:

$$\overrightarrow{g}(x,y,z) = \overrightarrow{\bigtriangledown f}(x,y,z)$$

Gradient vectors are computed at the cube vertices. At a cube vertex (i,j,k), the gradient is estimated using central differences along the three coordinates axes as follows:

$$G_x(i,j,k) = \frac{D(i+1,j,k) - D(i-1,j,k)}{\bigtriangleup x}$$

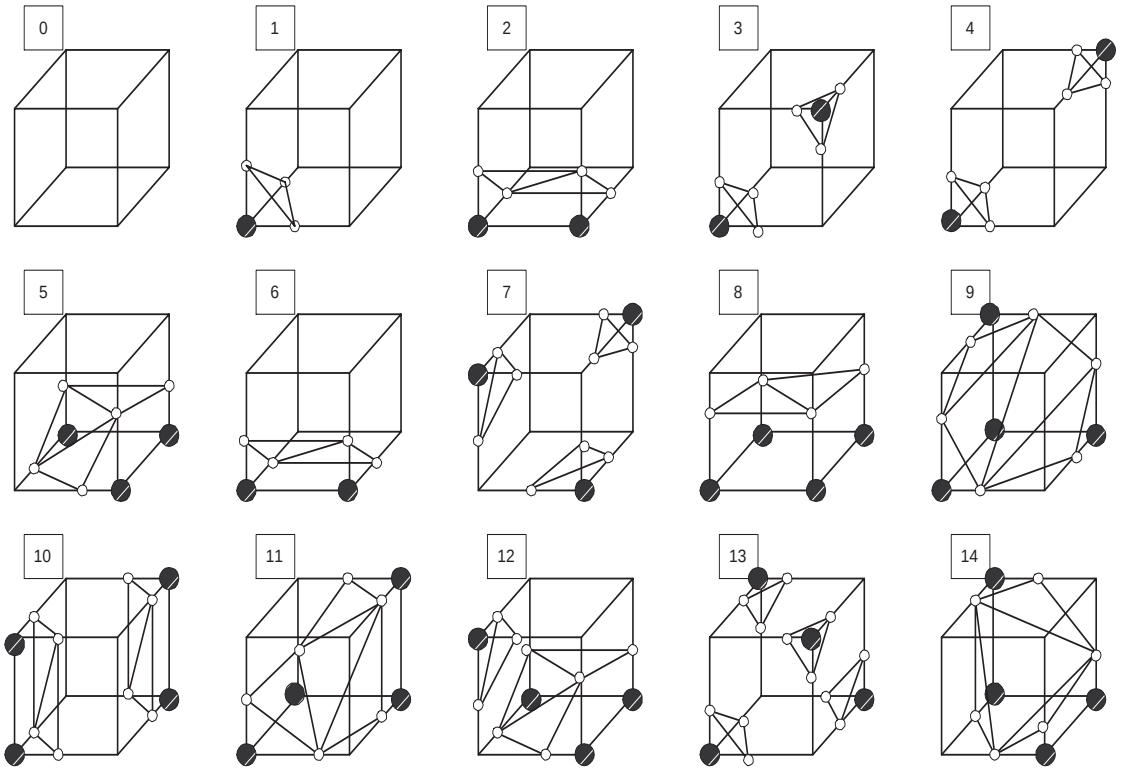


Figure A.2: 15 different cases

$$G_y(i,j,k) = \frac{D(i,j+1,k) - D(i,j-1,k)}{\triangle y}$$

$$G_z(i,j,k) = \frac{D(i,j,k+1) - D(i,j,k-1)}{\triangle z}$$

Here, $D(i,j,k)$ is the data value at pixel (i,j) in slice k and $\triangle x$, $\triangle y$ and $\triangle z$ are the lengths of the cube edges. Gradient vectors at the surface-edge intersection point are computed by linearly interpolating the gradient vectors at the vertices defining the edge, and normalized.

Appendix B

Triangle-Ray Intersection

This appendix describe a fast algorithm for ray—triangle intersection. This algorithm will first determine if a ray goes through the triangle and then compute the coordinates of the intersection point. See also (Glassner1990).

Step 1: Intersection with the Embedding Plane

A triangle is represented by its vertices $\mathbf{v}_i = (x_{v_i}, y_{v_i}, z_{v_i})$, $(i = 1, 2, 3)$. The normal, $\mathbf{n}$, of the plane containing the triangle is:

$$\mathbf{n} = \mathbf{l} \times \mathbf{m}$$

where $\mathbf{l} = \mathbf{v}_1 - \mathbf{v}_0$, and $\mathbf{m} = \mathbf{v}_2 - \mathbf{v}_0$. For each point $\mathbf{p} = (x_i, y_i, z_i)$ of the plane, the quantity $\mathbf{p} \cdot \mathbf{n}$ is constant. This constant value is computed by the dot product $d = -\mathbf{v}_o \cdot \mathbf{n}$. The implicit representation of the plane is:

$$\mathbf{n} \cdot \mathbf{p} + d = 0 \tag{B.1}$$

Let the origin and direction of the ray be $\mathbf{o}$ and $\mathbf{d}$ respectively, then the parametric representation of the ray is:

$$r(t) = \mathbf{o} + \mathbf{d}t, \tag{B.2}$$

where $t > 0$. Using equations B.1 and B.2 we can compute the value of the parameter t corresponding to the intersection point as follows:

$$t = -\frac{d + n \cdot \mathbf{o}}{\mathbf{n} \cdot \mathbf{d}} \tag{B.3}$$

The value of t is tested as follows:

1. If the triangle and the ray are parallel ($\mathbf{n} \cdot \mathbf{d} = 0$), the intersection is rejected.
2. If $t \leq 0$, the intersection is behind the origin of the ray, and the intersection is rejected.
3. If $t > 0$, the intersection is accepted.

Step 2: Intersecting the Triangle

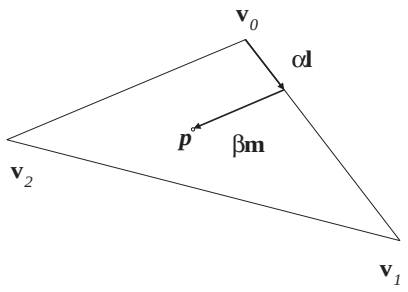


Figure B.1: Parametric representation of the point P w.r.t. the triangle

The point $\mathbf{p}$ in triangle coordinates (see Figure B.1) is given by

$$\mathbf{k} = \alpha\mathbf{l} + \beta\mathbf{m} \tag{B.4}$$

where $\mathbf{k} = \mathbf{p} - \mathbf{v}_0$. The point $\mathbf{p}$ will be inside the triangle $\Delta\mathbf{v}_o\mathbf{v}_1\mathbf{v}_2$ if

$$\alpha \geq 0, \beta \geq 0, \text{ and } \alpha + \beta \leq 1.$$

Equation B.4 can be written as:

$$\begin{cases} x_p - x_0 = \alpha(x_1 - x_0) + \beta(x_2 - x_0) \\ y_p - y_0 = \alpha(y_1 - y_0) + \beta(y_2 - y_0) \\ z_p - z_0 = \alpha(z_1 - z_0) + \beta(z_2 - z_0). \end{cases} \tag{B.5}$$

Equation (B.5) has a unique solution. The system is reduced by projecting the triangle onto one of the primary planes (xy, xz, or yz). If the triangle is perpendicular to one of these planes, its projection onto that plane will be a line. To avoid this problem, we maximize the projection by using the plane perpendicular to the dominant axis of the normal vector. Let

$$i = \begin{cases} x \text{ if } |n_x| = \max(|n_x|, |n_y|, |n_z|) \\ y \text{ if } |n_y| = \max(|n_x|, |n_y|, |n_z|) \\ z \text{ if } |n_z| = \max(|n_x|, |n_y|, |n_z|). \end{cases}$$

Consider g and h (g and $h \in \{x, y, z\}$), the symbol different from i. They represent the primary plane used to project the triangle. Let $\mathbf{q}_i=(u_i, w_i)$ be the two-dimensional coordinates of a vector in this plane; the coordinates of $\mathbf{k}$, $\mathbf{l}$ and $\mathbf{m}$ projected onto this plane are:

$$\begin{aligned} u_0 &= g_p - g_{v_0} \quad & w_0 &= h_p - h_{v_0} \\ u_1 &= g_{v_1} - g_{v_0} \quad & w_1 &= h_{v_1} - h_{v_0} \\ u_2 &= g_{v_2} - g_{v_0} \quad & w_2 &= h_{v_2} - h_{v_0} \end{aligned}$$

Equation (B.5) then reduces to

$$\begin{cases} u_0 = \alpha u_1 + \beta u_2 \\ w_0 = \alpha w_1 + \beta w_2 \end{cases} \tag{B.6}$$

The solutions of (B.6) are

$$\alpha = \frac{\det \begin{bmatrix} u_0 & u_2 \\ w_0 & w_2 \end{bmatrix}}{\det \begin{bmatrix} u_1 & u_2 \\ w_1 & w_2 \end{bmatrix}} \text{ and } \beta = \frac{\det \begin{bmatrix} u_1 & u_0 \\ w_1 & w_0 \end{bmatrix}}{\det \begin{bmatrix} u_1 & u_2 \\ w_1 & w_2 \end{bmatrix}}.$$

Appendix C

Evaluation of the Registration Algorithm

We have investigated the performance of the registration algorithm under the variation of the following factors:

- the local similarities or lack thereof between the two surfaces
- the number of correspondences
- the error in the correspondence pairs
- the initial mis-alignments of the two surfaces

We use the surfaces shown in Figure C.1(a) and Figure C.1(b) to study the algorithm, denoting the first surface S1 and the second surface S_2. Different test cases are set up by varying the number of correspondences, the error in the picked points and the initial mis-alignment of the two surfaces. We now apply the registration algorithm on the test cases and study the results to evaluate its performance. We discuss the results and draw conclusions in the following sections.

Rotation	Translation	Scaling
45,60,90	-2,5,-10	1.2,0.67,0.95

Table C.1: Transformation Parameters

Surfaces Used	ISD (Before)	ISD (After)	(ISD) (After Deformations)
S_1 and S_4	13.09905	0.492979	-
S_1 and S_3	13.6178	0.0854026	0.0539778

Table C.2: Registration results

C.1 Local dissimilarities of the surfaces

The algorithm assumes that the surfaces are related through a global affine transformation. Therefore, when given the task of registering two surfaces, the algorithm only captures the global transformation and does not take into account the local features of the surfaces. Moreover, it does not deform a surface to match the other surface. One consequence of this behavior is that the algorithm can not precisely register the CT and the Cyberware datasets coming from two different individuals, since the facial skin surfaces of different individuals are not related through a global transformation. The following example illustrates this shortcoming of the algorithm. S_1 and S_2 are transformed using the transformation parameters listed in Table C.1. The transformed surfaces are called S_3 and S_4 respectively. We now register S_1 with S_3 and S_4 and compute the ISD values. Table C.2 lists the registration results. Figure C.2 shows the surfaces before and after registration.

In the first case, the two surfaces are the same, and the algorithm has registered them properly. In the second case, the two surfaces are different. The algorithm has captured the global transformations, but the registration quality is much worse. This is also reflected by the higher ISD value for the second case. Visual inspection of the results also shows that the algorithm has failed to match the local details. It should, however,

No.	Rotation	Translation	Scaling
1	$90°, 15°, 5°$	$0, 10, -10$	$1, 1.5, 1.3$
2	$90°, 0°, 0°$	$0, 0, 0$	$0.5, 0.5, 0.73$
3	$10°, 15°, 0°$	$0, 0, 0$	$1, 1, 1$
4	$0°, 0°, 0°$	$0, 0, 0$	$5, 10, 0.3$
5	$0°, 0°, 0°$	$0, 0, -20$	$1.5, 1.5, 1.0$
6	$0°, 0°, 0°$	$10, 5, -10$	$1, 1, 1$
7	$62°, 42°, 4°$	$10, 5, -10$	$1, 1, 1$
8	$0°, 0°, 0°$	$0, 0, 0$	$1, 1, 1$

Table C.3: Transformation Parameters

be noted that the two surfaces are much better aligned after the registration. The algorithm has globally distorted S_3. In some cases, such a distortion may be undesirable. This distortion is an artifact of the affine transformation and one scheme to avoid it is to use a rigid transformation.

C.2 The number of correspondences used, error in the picked points and initial misalignment of the surfaces

We study the effects of the number of correspondences and the error in the picked points on the algorithm using the following methodology. We pick approximately 80 points on S_1 and S_2. We now generate 14 pairs of surfaces with known correspondences by transforming S_1 and S_2 using the transformation parameters shown in Table C.3. The points picked on S_1 and S_2 are also transformed using the same transformation parameters. Each pair of surfaces is assigned a name using the following rule:

- The surface generated by transforming S_1 using the transformation parameters given in the i^{th} row of Table C.3 is called $S_1{}^i$. The pair of S_1 and $S_1{}^i$ is called S_1-$S_1{}^i$, $i = \{1, 2, \cdots, 8\}$.

For each pair of the surfaces, we add a noise vector (n_x, n_y, n_z) to each of the transformed points, where n_x,n_y, and n_z are randomly drawn from the normal distribution with mean 0 and standard deviation σ. The surface pairs are then registered and the *mean-squared-errors* (MSEs) for the corresponding points are calculated. We also compute the ISD values. For each pair, the process is repeated 100 times for each value of σ. We plot the MSE and ISD against σ and the number of correspondences used.

The maximum tolerable ISD for S_j-$S_j{}^i$, $j = \{1, 2\}$ and $i = \{1, 2, \cdots, 8\}$, pairs is 0.3. This value is fixed by visually inspecting the registration results. Figure C.4 shows registration results for S_2-$S_2{}^1$ pair, using 15 points, for various ISD values. It is obvious that the registration quality is unacceptable for large ISD values.

The following discussion refers to the plots given on pages 73–76.

If a small number of correspondences are used, and the value of σ is small, the ISD is more then the MSE. This behavior is to be expected, since the small number of correspondences are unable to capture the total surface information. The affine transformation computes a transformation matrix which brings the corresponding points closer by distorting the over-all surface.

As we increase the number of correspondences, for small values of σ the ISD becomes smaller then the MSE. Visually inspecting the registration results also confirms that using a large number of points improves the registration quality. This behavior indicates that unlike ISD, MSE is not a good indicator of the registration quality. As the number of correspondences is increased beyond a certain number, the quality of the registration does not improve. For the purpose of registering facial skin surfaces, it was found that 15 to 20 correspondences are enough, and increasing the number of correspondences beyond this does not improve the quality.

The registration quality deteriorates steadily as we increase the value of σ. For small value of σ the registration quality is acceptable. The quality of registration actually depends upon the ratio of the size of the bounding box of the surface and the noise in the picked points. The smaller the ratio, the worse is the algorithm's performance. This observation leads to a simple technique for improving the registration quality: we scale-up both surfaces before picking points, thereby reducing the size of the boundary box to the noise ratio.

In the tests we have conducted, the results are found to be independent of the initial mis-alignment. The plots in Figure C.3 supports this claim. They show the contours of maximum tolerable ISD value for S_1 and S_2 for the different test cases. The contours are plotted against the noise level σ and the number of correspondences. For all 16 cases the contours are overlapping. This behavior illustrates that the performance of the registration is independent of the initial mis-alignment.

C.3 Error in the manually picked points

To understand the error in manually picked points, we requested 5 users to pick points on the surfaces (Figure C.1(a) and Figure C.1(b)) and the picked points were stored. The individuals were once again asked to pick the same points. Every individual picked 20 points on each of the surfaces. The standard deviation of the MSE between the corresponding points was computed. It was found to be 0.065. Comparing this value to the plots in Figure C.3, it is clear that the for $\sigma \leq 0.065$, the registration results are acceptable if we choose more than 10 points. The registration quality is acceptable for $\sigma \leq 0.065$ and the number of correspondences ≥ 10 for all the pairs except S_1-S_1^4. This is one of the reasons why in all the above cases we were able to perform acceptable registrations manually. Table C.4 lists the ISD values obtained after manually registering the pairs S_j-$S_j{}^i$, $j = \{1, 2\}$ and $i = \{1, 2, \cdots, 8\}$.

No.	Surface Pair	ISD (before registration)	ISD (after registration)
1	S_1-$S_1{}^1$	10.0915	0.055
2	S_1-$S_1{}^2$	4.4065	0.043
3	S_1-$S_1{}^3$	0.8026	0.067
4	S_1-$S_1{}^4$	15.0230	0.076
5	S_1-$S_1{}^5$	16.7949	0.08
6	S_1-$S_1{}^6$	11.1412	0.092
7	S_1-$S_1{}^7$	14.3850	0.046
8	S_1-$S_1{}^8$	0.0	0.054
9	S_2-$S_2{}^1$	11.3284	0.046
10	S_2-$S_2{}^2$	6.4967	0.071
11	S_2-$S_2{}^3$	1.0184	0.067
12	S_2-$S_2{}^4$	20.5413	0.09
13	S_2-$S_2{}^5$	15.4300	0.048
14	S_2-$S_2{}^6$	9.3540	0.072
15	S_2-$S_2{}^7$	13.2882	0.065
16	S_2-$S_2{}^8$	0.0	0.08

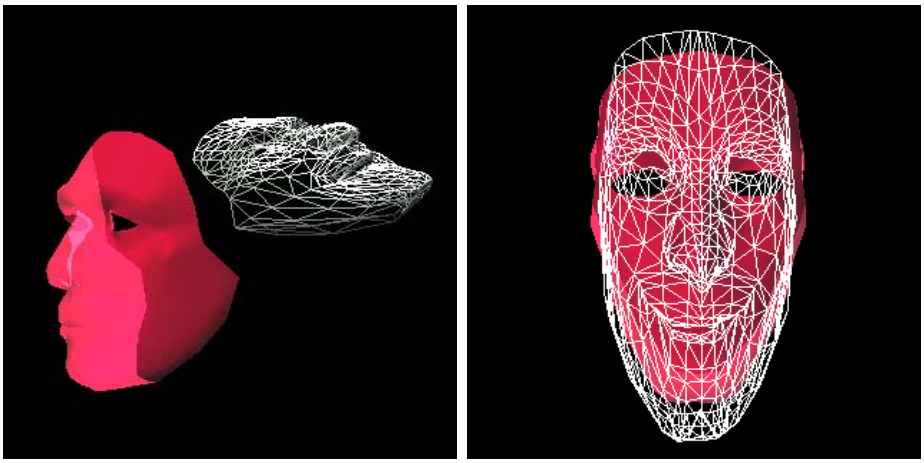
Table C.4: ISD values after manually registering the pairs, Sj-Sj^i, $j = 1, 2$ and $i = 1, 2, \cdots, 8$. 15 correspondences are used on each of the surfaces for the purpose of registration.

(a) S_1

(b) S_2

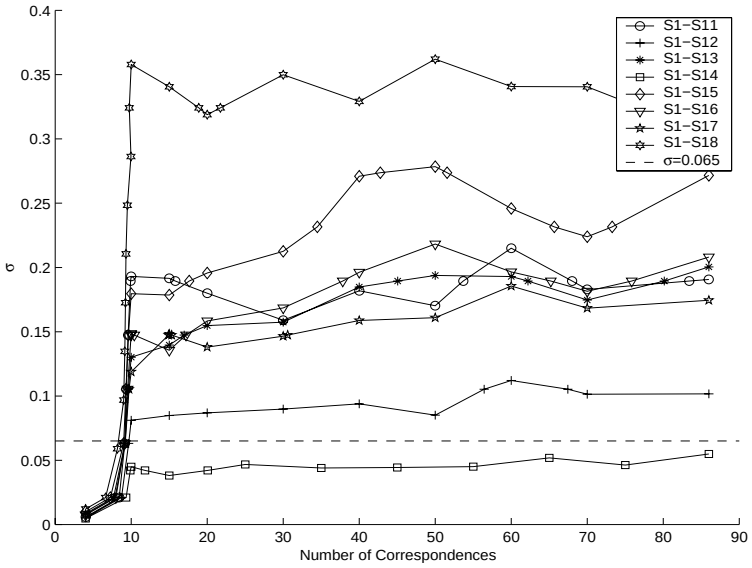
Figure C.1: The facial surfaces used to investigate the registration algorithm

(a) S_1 and S_3 before registration (b) S_1 and S_3 after registration

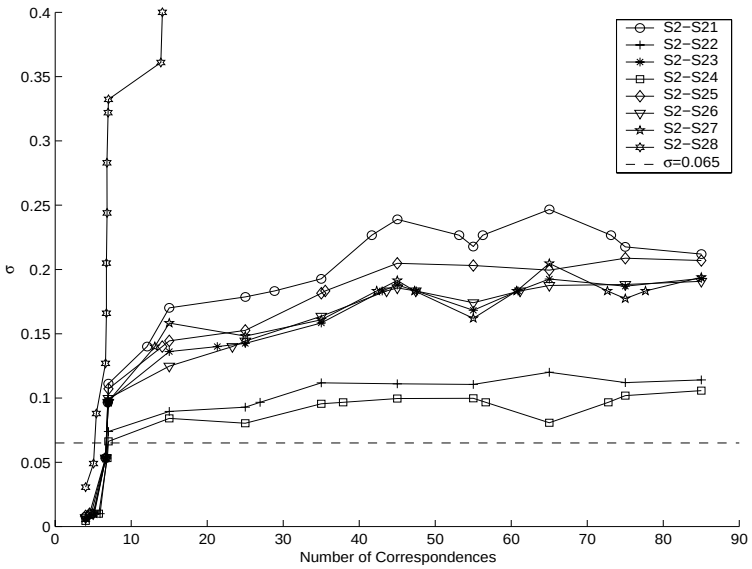


(c) S_1 and S_4 before registration (d) S_1 and S_4 after registration

Figure C.2: Surfaces S_1, S_3 and S_4. S_3 is the transformed version of S_1 (Figure C.1(a)) and S_4 is the transformed version of S_2 (Figure C.1(b))

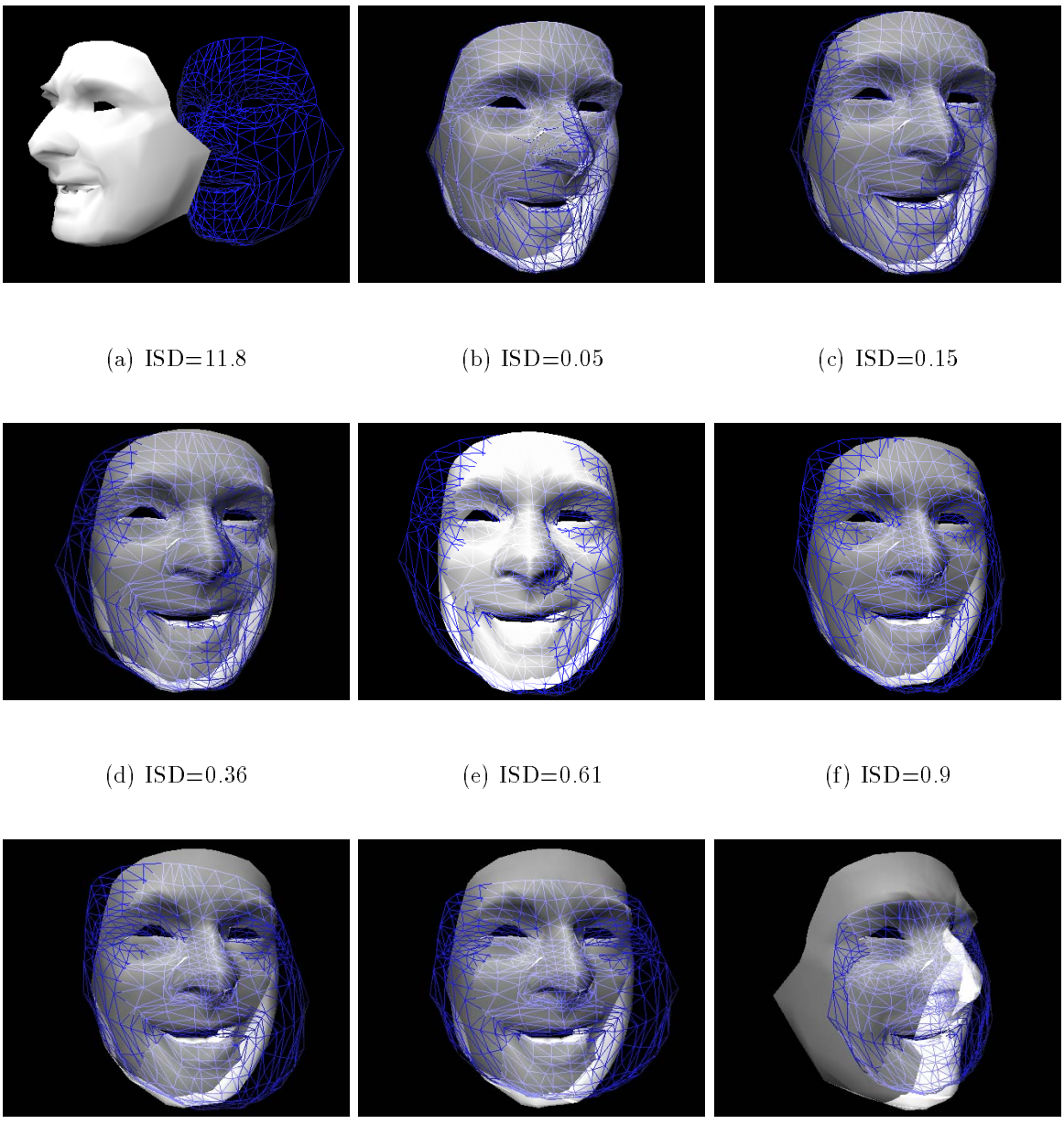


(a) S_1-$S_1{}^i$



(b) S_2-$S_2{}^i$

Figure C.3: Contours of the maximum tolerable ISD values against the number of correspondences and the noise (σ). $i = 1, 2, \cdots, 8$.

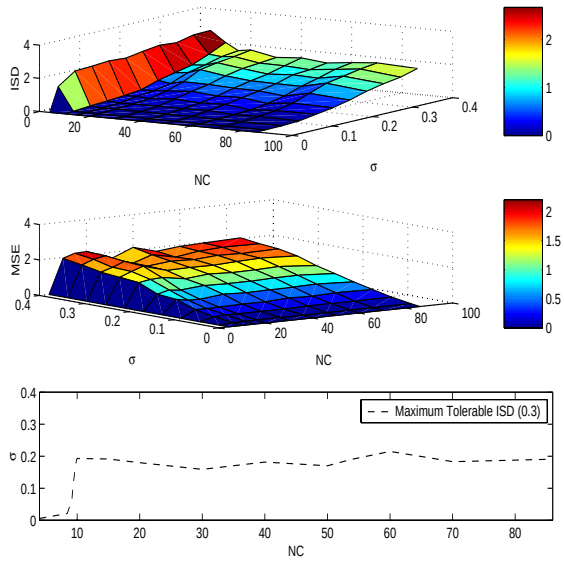


(a) ISD=11.8 (b) ISD=0.05 (c) ISD=0.15

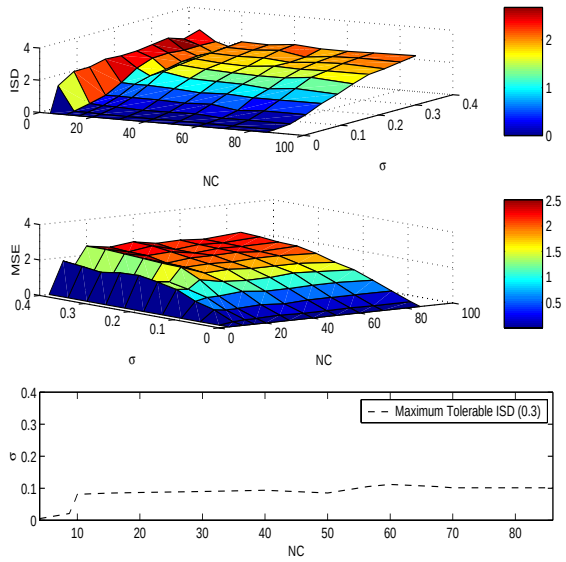
(d) ISD=0.36 (e) ISD=0.61 (f) ISD=0.9

(g) ISD=1.02 (h) ISD=1.23 (i) ISD=1.42

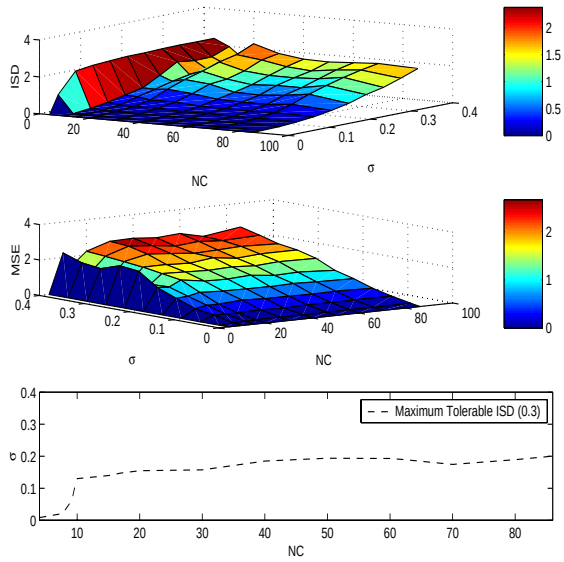
Figure C.4: Registration results for S_2-$S_2{}^1$ pair, using 15 correspondences, for different ISD values.



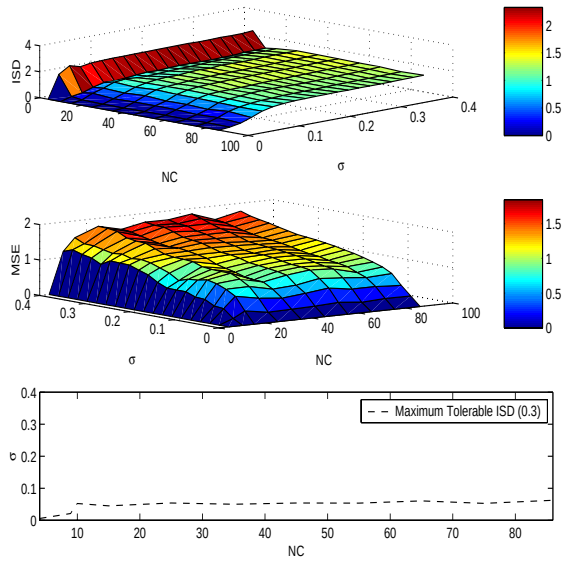
(a) S_1-$S_1{}^1$



(b) S_1-$S_1{}^2$

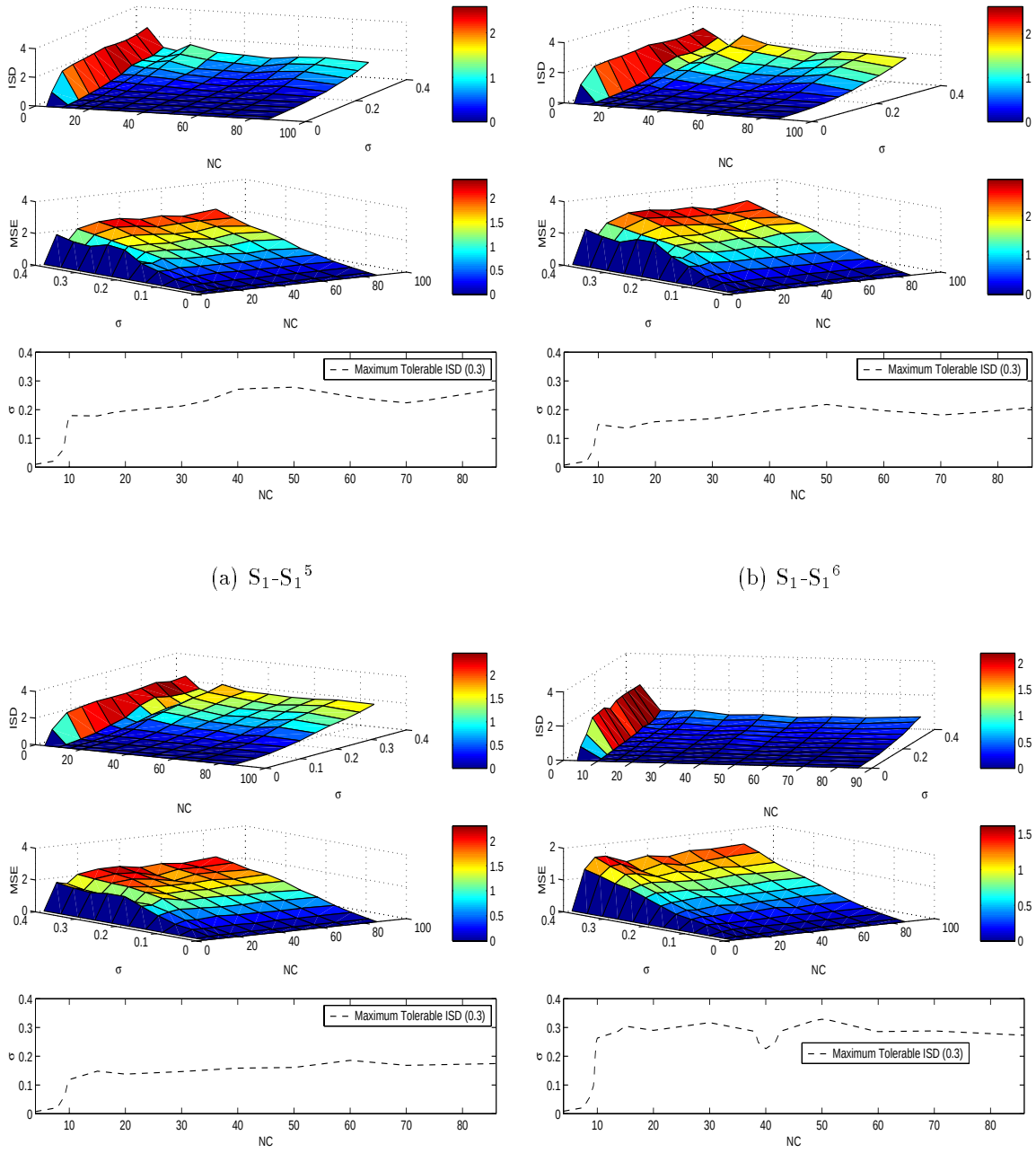


(c) S_1-$S_1{}^3$



(d) S_1-$S_1{}^4$

Figure C.5: ISD and MSE plots against σ and the number of correspondences (NC)



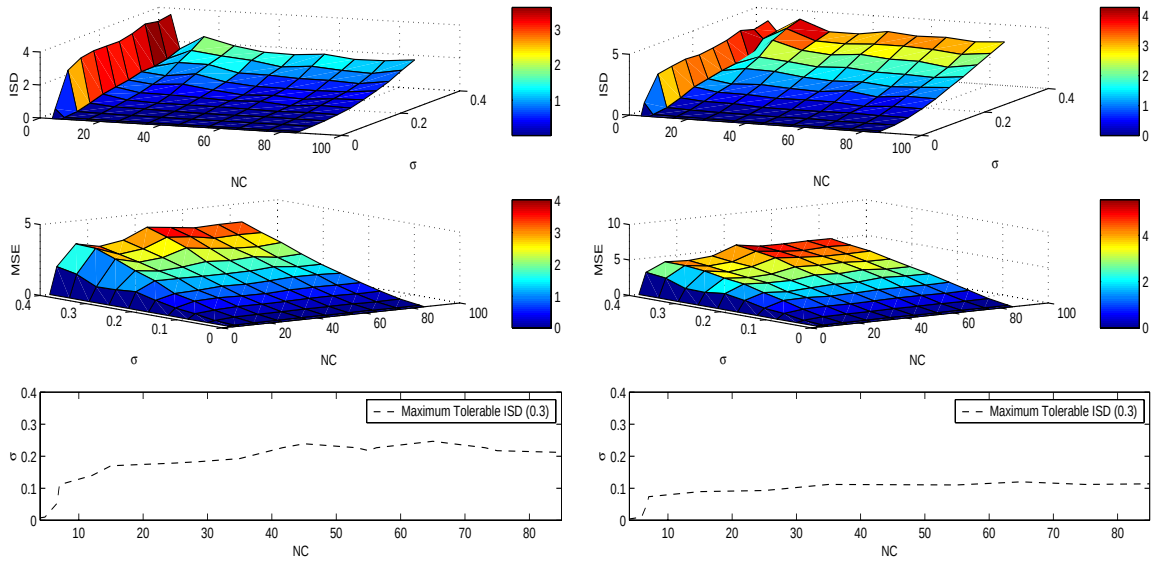
(a) S_1-$S_1{}^5$

(b) S_1-$S_1{}^6$

(c) S_1-$S_1{}^7$

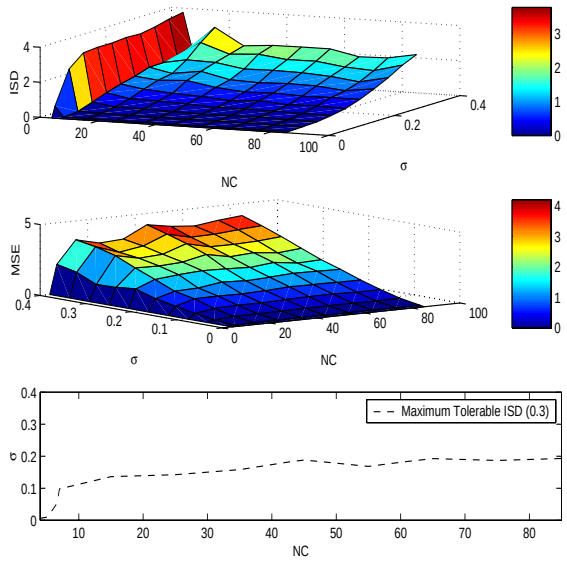
(d) S_1-$S_1{}^8$

Figure C.6: ISD and MSE plots against σ and the number of correspondences (NC)

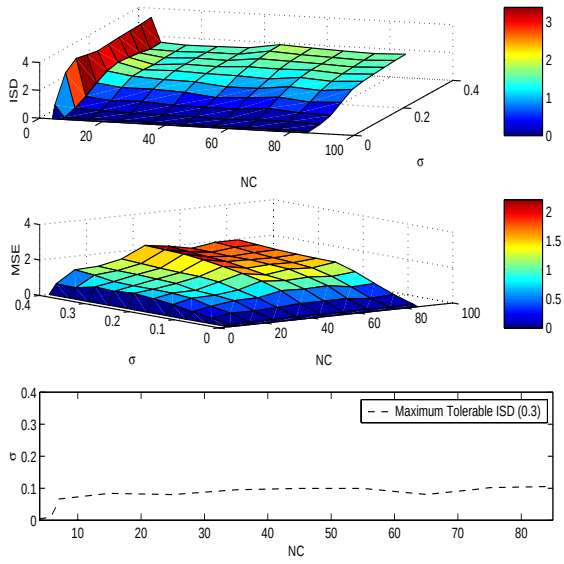


(a) S_2-$S_2{}^1$

(b) S_2-$S_2{}^2$

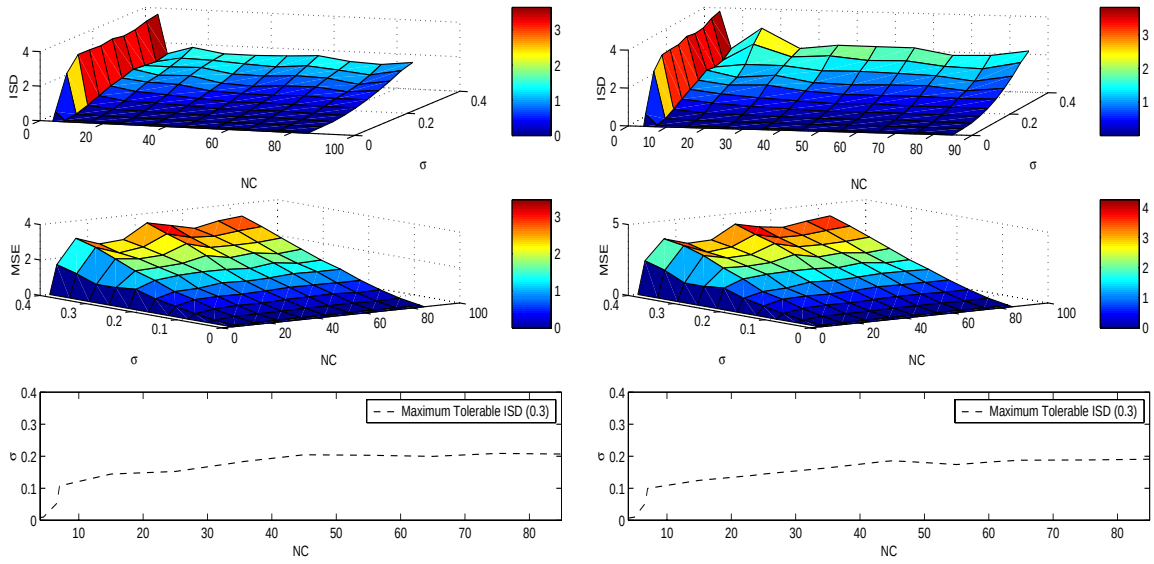


(c) S_2-$S_2{}^3$



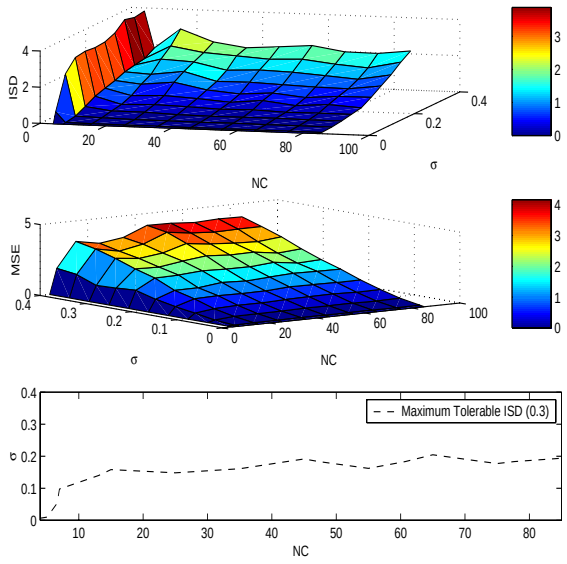
(d) S_2-$S_2{}^4$

Figure C.7: ISD and MSE plots against σ and the number of correspondences (NC)

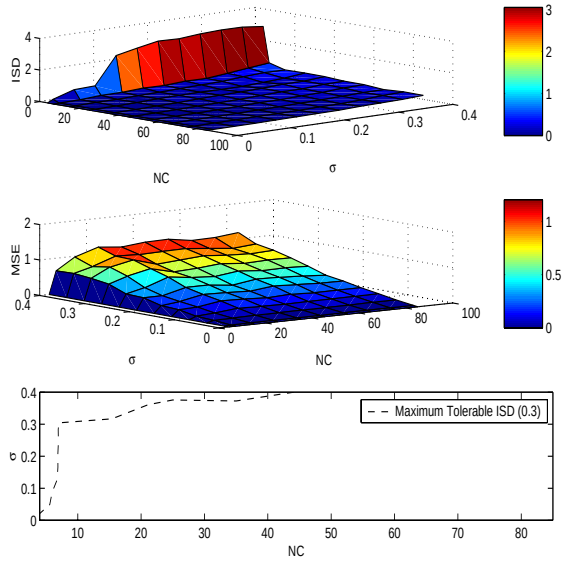


(a) S_2-$S_2{}^5$

(b) S_2-$S_2{}^6$



(c) S_2-$S_2{}^7$



(d) S_2-$S_2{}^8$

Figure C.8: ISD and MSE plots against σ and the number of correspondences (NC)

References

Alpert, N. M., Bradshaw, J. F., Kennedy, D., and Correia, J. A. (1990). The principle axis transformation – a method for image registration. *Journal of Nuclear Medicine*, 31:1717–1722.

Archer, K. (1997). Craniofacial reconstruction using hierarchical b-spline interpolation. Master's thesis, University of British Columbia.

Archer, K., Coughlan, K., Forsey, D., and Struben, S. (1998). Software tools for craniofacial growth and reconstruction. In *Graphics Interface*, pages 73–81.

Arun, K. S., Huang, T. S., and Blostein, S. D. (1987). Least-squares fitting of two 3-D point sets. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-9(5):698–700.

Besl, P. J. and McKay, N. D. (1992). A method for registration of 3-D shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2):239–258.

Chan, S. L. and Purisima, E. O. (1998). A new tetrahedral tesselation scheme for isosurface generation. *Computers and Graphics*, 22(1):83–90.

Chen, Y. and Medioni, G. (1992). Object modeling by registration of multiple range images. *International Journal of Image and Vision Computing*, 10(3):145–155.

Cignoni, P., Montani, C., and Scopigno, R. (1998). A comparison for mesh simplification algorithms. *Computers and Graphics*, 22(1):37–54.

Cyberware Laboratory, Inc. (1990). *4020/RGB 3D Scanner with Color Digitizer*. Monterey, CA.

Durst, M. J. (1988). Letters: Additional reference to marching cubes. *Computer Graphics*, 22.

Faugeras, O. D. and Hebert, M. (1983). A 3-D recognition and positioning algorithm using geometrical matching between primitive surfaces. In Bundy, A., editor, *Proceedings of the 8th International Joint Conference on Artificial Intelligence*, pages 996–1002, Karlsruhe, FRG. William Kaufmann.

Feldmar, J. and Ayache, N. (1994). Localy affine registration of free-form surfaces. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*, pages 496–501, Los Alamitos, CA, USA. IEEE Computer Society Press.

Feldmar, J. and Ayache, N. (1996). Rigid, affine and locally affine registration of free-form surfaces. *International Journal of Computer Vision*, 18(2):99–119.

Glassner, A. S., editor (1990). *Graphics Gems*, volume 1, chapter 7, pages 390–393. Academic Press, Inc.

Guéziec, A. and Hummel, R. (1995). Exploiting Triangulated Surface Extraction Using Tetrahedral Decomposition. *IEEE Transactions on Visualization and Computer Graphics*, 1:328–342.

Hill, D. L. G., Hawkes, D. J., Crossman, J. E., Gleeson, M. J., Cox, T. C. S., Bracey, E. C. M. L., Strong, A. J., and Graves, P. (1991). Registration of MR and CT images for skull base surgery using pointlike anatomical features. *British Journal of Radiology*, 64:1030–1035.

Kalvin, A. D., Cutting, C. B., Haddad, B., and Noz, M. E. (1991). Constructing topologically connected surfaces for the comprehensive analysis of 3D medical structures. In *Medical Imaging V: Image Processing*, volume 1445, pages 247–258. SPIE.

Kass, M., Witkin, A., and Terzopoulos, D. (1988). Snakes: Active contour models. *International Journal of Computer Vision*, 1(4):321–331.

Lee, Y. V., Terzopoulos, D., and Waters, K. (1995). Realistic modeling for facial animation. *Computer Graphics, Annual Conference Series (SIGGRAPH'95 Proceedings)*, pages 55–62.

Li, J. and Agathoklis, P. (1997). An efficiency enhanced isosurface generation algorithm for volume visualization. *The Visual Computer*, 13:391–400.

Lin, Z. C., Huang, T. S., Blostein, S. D., and Margerum, E. A. (1986). Motion estimation from 3-D point sets with and without correspondences. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 194–201. IEEE.

Lorensen, W. E. and Cline, H. E. (1987). Marching cubes: A high resolution 3D surface construction algorithm. *Computer Graphics*, 21(4):163–169.

Maintz, J. B. A. and Viergever, M. A. (1998). A survey of medical image registration. *Medical Image Analysis*, 2(1):1–36.

Montani, C., Scateni, R., and Scopigno, R. (1994). A modified lookup table for implicit disambiguation of marching cubes. *The visual computer*, 10:353–355.

Muller, M. and Stark, M. (1993). Adaptive generation of surfaces in volume data. *The Visual Computer*, 9:182–199.

Ning, P. and Bloomenthal, J. (1993). An evaluation of implicit surface tilers. *IEEE Computer Graphics and Applications*, 13(6):33–41.

Pelizzari, C. A., Chen, G. T. Y., Spelgring, D. R. Weichselbaum, R. R., and Chen, C. T. (1989). Accurate three-dimensional registration of CT, PET, and/or MR images of the brain. *Journal of Computer Assisted Tomography*, 13(1):20–26.

Penney, G. P., Weese, J., Little, J. A., Desmedt, P., Hill, D. L. G., and Hawkes, D. J. (1998). A comparison of similarity measures for use in 2D-3D medical image registration. *IEEE Transactions on Medical Imaging*, 17(4):586–595.

Press, W. H., Trukolsky, S. A., Vetterling, W. T., and Flannery, B. T. (1992). *Numerical Recipies in C: The Art of Scientific Computing*. Cambridge University Press.

Schmidt, M. F. W. (1993). Cutting cubes – visualizing implicit surfaces by adaptive polygonaization. *The Visual Computer*, 10:101–115.

Schroeder, W. J., Zarge, J., and E., L. W. (1992). Decimation of triangle meshes. *Computer Graphics*, 26:65–70.

Terzopoulos, D. (1988). The computation of visible-surface representations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 10(4):417–438.

Thirion, J. P. (1994). Extremal points: Definition and application to 3D image registration. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*, pages 587–592, Los Alamitos, CA, USA. IEEE Computer Society Press.

Thirion, J. P. (1996). New feature points based on geometric invariants for 3D image registration. *International Journal of Computer Vision*, 18(2):121–137.

Thirion, J. P. and Gourdon, A. (1996). The 3D marching lines algorithm. *Graphical Models and Image Processing*, 58(6):503–509.

Toga, A. W. and Banerjee, P. K. (1993). Registration revisited. *Journal of Neuroscience Methods*, 48(1-2):1–13.

Treece, G. M., Prager, R. W., and Gee, A. H. (1998). Regularised marching tetrahedra: improved isosurface extraction. Technical report, Cambridge University Engineering Department.

Turk, G. (1992). Re-tiling polygon surfaces. *Computer Graphics*, 26:79–86.

Turkington, T. G., Hoffman, J. M., Jaszczak, R. J., Macfall, J. R., Harris, C. C., Kilts, C. D., Pelizzari, C. A., and Coleman, R. E. (1995). Accuracy of surface fit registration for PET and MR brain images using full and incomplete brain surfaces. *Journal of Computer Assisted Tomography*, 19:117–124.

Turkington, T. G., Jaszczak, T. J., Pelizzari, C. A., Harris, C. C., Macfall, J. R., Hoffman, J. M., and Coleman, R. E. (1993). Accuracy of registration of PET, SPECT and MR images of a brain phantom. *Journal of Nuclear Medicine*, 34(9):1587–1594.

van del Elsen, P. A., Pol, E. J. D., and A., V. M. (1993). Medical image matching: A review with classification. In *IEEE Engineering in Medicine and Biology Magazine*, volume 12, pages 26–39.

van den Elsen, P. A., Pol, E. J. D., Sumanawaeera, T. S., Hemler, P. F., Napel, S., and Adler, J. R. (1994). Gray value correlation techniques used for automatic matching of CT and MR brain and spine images. In *Proceedings of Visualisation in Biomedical Computing*, pages 227–237, Rochester, MN. SPIE Press, Bellinghan, WA.

Waters, K. (1992). A physical model of facial tissue and muscle articulation derived from computer tomography data. In *Visualization in Biomedical Computing*, volume 1808 of *SPIE*, pages 574–583.

Wells, W. M., Viola, P., Atsumi, H., and Nakajima, S. (1996). Multi-model volume registration by maximization of mutual information. *Medical Image Analysis*, 1(1):35–51.

Wilhelms, J. and Gelder, A. V. (1990). Topological considerations in isosurface generation. Extended abstract. *Computer Graphics*, 24:79–86.

Woods, R. P., Cherry, S. R., and Mazziotta, J. C. (1992). Rapid automated algorithm for aligning and reslicing pet images. *Journal of Computer Assisted Tomography*, 16:620–633.

Woods, R. P., Mazziota, J. C., and Cherry, S. R. (1993). MRI-PET registration with automated algorithm. *Journal of Computer Assisted Tomography*, 17(4):536–546.

Yun, H. and Park, K. H. (1992). Surface modeling method by polgonal primitives for visualizing three-dimensional volume data. *The Visual Computer*, 8:246–259.

Zhang, Z. (1994). Iterative point matching for registration of free-form curves and surfaces. *International Journal of Computer Vision*, 13(2):119–152.