ACCURATE CHARACTERIZATION OF SKIN DEFORMATIONS USING RANGE DATA

by

Jimmy D. Talbot

A Thesis submitted in conformity with the requirements
for the Degree of Master of Science
Department of Computer Science
University of Toronto

© Copyright by Jimmy D. Talbot 1998

Abstract

Generating realistic skin deformations arising from joint movement and muscle contraction is a requirement for producing realistic human character animation. For example, the human arm, hand, or foot change shape in significant ways during motion, ways which are difficult to model accurately with traditional character animation techniques. This thesis suggests a new way to build realistic animated models of the human form. We exploit range image technology to capture the true human form and create parameterized animated surface models based upon this data. The thesis improves in several ways upon the algorithms required to process the range data, as well as presenting a methodology for the required data capture, data integration and surface parameterization. Results are presented for the parameterized flexion of a human arm model.

Acknowledgements

This thesis would not have been possible without the generous contribution in time and ideas of my supervisor, Professor Michiel van de Panne. Professor James Stewart also contributed his time (when it was in such high demand) to read the thesis. Both were of great help in making this document much more understandable.

SPAR Aerospace allowed the use of their laser range scanner for this research. The people at SPAR, particularly Eldon Wong and Perry Newhook, were always willing to help with the scanning and other aspects of the research.

I am grateful to the following authors who have agreed to the inclusion in this thesis of figures from their papers: Marek Teichmann, for Figure 15; Yosihiro Yasumuro, for Figure 16; Dr. Jane Wilhelms, for Figure 17 and Figure 18; and Dr. Ferdi Scheepers, for Figure 19 and Figure 20.

Financial support for this thesis was provided by NSERC and SPAR Aerospace, through the NSERC Postgraduate Industrial Scholarship. I have really appreciated this opportunity to pursue my research interests in the industry.

Table of Contents

Table of Figures

1. INTRODUCTION

Humans and other animals are among the most interesting objects simulated in computer graphics, but they are also unfortunately among the most challenging to realistically model and animate. The difficulties involve both the realistic movements of such animate characters as well as complexities associated with their appearance, such as the muscle movement under the skin and the appearance of wrinkles in skin and clothes. Muscles stretch and contract across joints to cause motion, which creates significant changes in the shape of the skin as a side effect. Thus, even if a character possesses realistic motion, it can look artificial if the geometric deformations associated with muscles are not modeled. In this thesis we investigate methods of modeling realistic animated deformations based upon deformation data captured from human subjects.

1.1 Problem Description

There are many steps involved in producing computer animations. In this thesis, we focus on the modeling of a human character, including the model construction and the deformations required to animate its form. We do not discuss issues related to the dynamic animation of its skeleton, character interaction with the environment, or interaction between characters. For our purposes, movement will be defined as the changes in shape induced in the skin by the character's actions, such as bending an arm or clenching a fist.

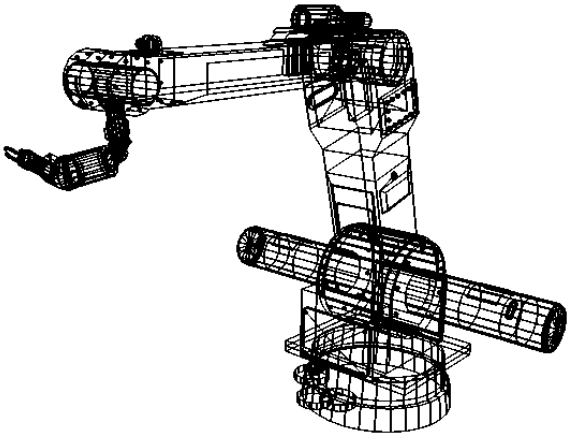

Figure 1: An articulated robot arm.

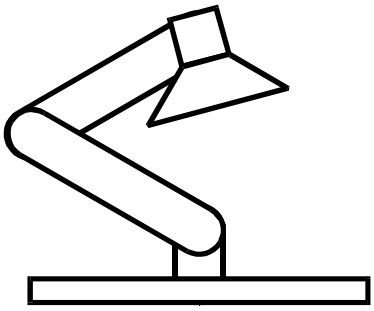

Figure 2: Luxo!, a classical animated stick figure.

In some instances, articulated computer models are composed of rigid geometric bodies that are connected with revolute joints. This is well suited for articulated figures such as insects and robots as shown in Figure 1, or simple stick figure representations, such as the animated Luxo creature, shown in Figure 2.

Other types of articulated models possess a surface or skin which is distinct from the underlying skeleton. The surface may be modeled using a polygonal mesh as shown in Figure 3 and Figure 4, spline patches, subdivision surfaces, or other boundary

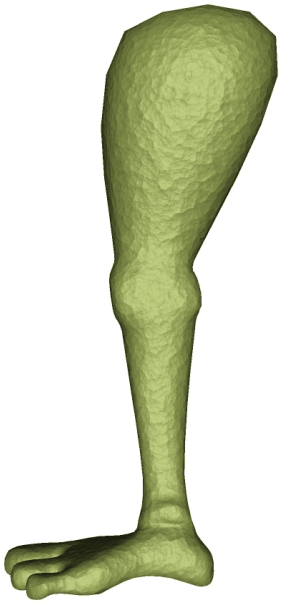

Figure 3: Mesh model of an alien leg.

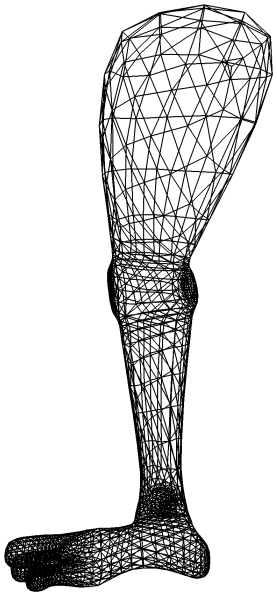

Figure 4: Wireframe rendering of the alien leg.

representations. These representations are clearly more suitable for human models, where the surface changes shape as a function of the posture[1] of the model. Because of the continuous nature of such surface models, there are no directly distinguishable links and joints. If the surfaces are to be animated, an articulated skeleton is needed to define an underlying set of links and joints which will indirectly govern the movement of the surface. The surface then needs to be bound in some fashion to the underlying skeleton. A simple way of doing this is to assign the surface control points to the closest bone. A vertex is then redefined to exist in the local frame of reference of that bone. The geometric model can either be made to rigidly follow the skeleton rigidly, or can be allowed to stretch elastically. With additional effort, features such as a parameterized muscle flexing can be introduced.

Although the model described above is very generic, allowing it to be used for any type of character, such simplified deformation models do not have any anatomical basis. Figure 5 shows an example of a simple model for animating the deformation of the thigh and calf[2]. This type of model may be adequate for a simple character, but is inadequate

Figure 5: Sequence demonstrating the effect of muscle flexing in Newtek's Lightwave 5.0.

[1] We use *posture* to refer to the position of the underlying skeleton.
[2] Parameters were deliberately set to large values in order to clearly show the effect of the deformations.

for creating realistic human models, such as those required for feature film production.

The principal advantage of the effect demonstrated in Figure 5 is that it is simple to achieve. Given a surface, the animator aligns an articulated skeleton with the model and set a few simple parameters. As the skeleton is animated, the surface follows and flexes in a qualitatively appropriate fashion.

Generally, we can express the surface *S* of a character as being a function of several parameters. These parameters might include the posture of the underlying skeleton and muscle contraction intensity. Muscles can affect more than one joint. For example, the biceps muscle controls both the elbow angle and the wrist rotation (supination). Using the above notation, the deformation of the skin due to the action of the biceps on the forearm and the upper arm would be described as:

$$S = b(P_{wrist}, P_{elbow})$$

where $P_{forearm}$ and P_{elbow} are scalars describing their respective joint angle.

This notation will be useful when describing our morphing technique in §6.2. The above definition can of course be augmented to include additional parameters, as we shall see later.

1.2 Goals

We wish to animate skin deformations based upon experimentally-obtained deformation data. The system should be as intuitive to use as the skeletons currently used in commercial packages, but should generate more realistic results.

The approach that we propose is geared towards animating human bodies, but not human faces. This is because human the limbs and torso have fewer individually controllable muscles. As well, facial animation has long been treated using techniques dedicated to that specific problem.

1.3 Our Animation System

Our solution, which we believe achieves our goals is to use range images[3] of real human bodies as keyframes and a 3-D morphing algorithm to interpolate between the keyframes. Because our system uses real surface data, there is no need for the types of parameters required with other techniques. Range scanners are accurate enough to generate extremely high quality range images – they can resolve details as minute as blood vessels on a forearm. This data can be used to generate realistic keyframes, and thus realistic deformations.

Figure 6 shows the steps in creating a deformable articulated model using our system. These steps are summarized here and explained in greater detail throughout the remainder of the thesis.

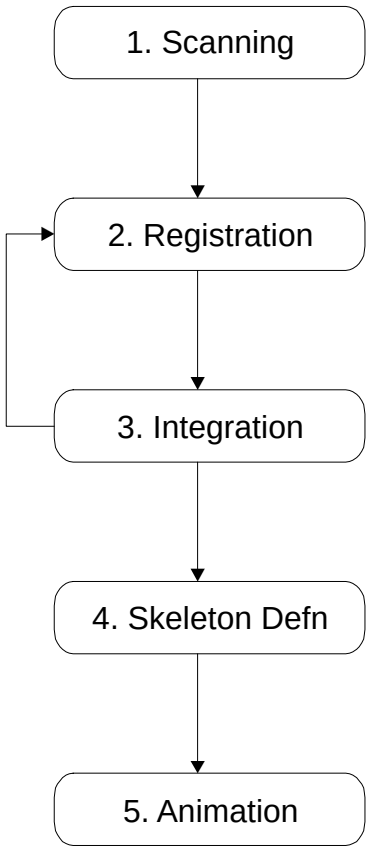

Figure 6: Steps to construct a character animation with the proposed system.

Capturing the keyframe data requires three steps, numbered one to three in Figure 6. The first step involves scanning the desired forms using a range scanner. The subject should be scanned in different postures. A consistent surface model for each posture is assembled in steps 2 and 3. This involves incrementally aligning the range images together, eliminating redundant surface areas, and merging the data sets together. Step 4 involves the construction of a skeleton. It is not yet fully automated, but there are methods (reviewed in §2.5) that would accomplish this with minimum user intervention. Once the skeleton is properly constructed and aligned with each of the postures, the polygonal models are linked to their associated skeleton. The user can now animate a skeleton and the modeled forms will deform

[3] A range image is a grid of distances (range points) that describes a surface. See §2.1.2 for a more detailed description.

accordingly.

Our primary contribution is a new way to build realistic animated models of the human form. We exploit range image technology to capture the true human form and create parameterized animated surface models based upon this data. We demonstrate results of an implementation that animates deformations of the human arm. We also improve in several ways upon the algorithms required to process the range data, as well as presenting a methodology for the required data capture, data integration and surface parameterization.

1.4 Thesis Organization

This thesis comprises 8 chapters. Chapter 2 describes data acquisition methods and previous work in range image registration, range image integration, and character animation. Chapter 3 describes the data acquisition process used to construct each static model that will make up the keyframe sequence. Chapters 4 and 5 describe how this data is aligned and assembled together. Chapter 6 describes our 3D-morphing approach to produce the skin deformation. Chapter 7 concludes the thesis and proposes future work.

2. RELATED WORK

Our work builds on results from four areas of research: range image acquisition, registration, and integration, and animation of skinned articulated characters. Work in each of these areas is reviewed in this chapter, providing a proper context for the presentation of our work in the following chapters.

2.1 Range Image Acquisition

When producing computer animations, characters are usually constructed using 3D modeling software. Even though expert modelers can achieve fantastic results, the modeled characters are not realistic enough to be mistakable for real human actors in motion pictures. This can be partly attributed to the difficulty of correctly modeling and animating skin deformations arising from joint movement and muscle contraction. An alternate approach to consider for character construction involves the use of accurate deformation measurements from real humans. This requires a mechanism for rapidly gathering accurate geometric data about the real world. We shall review several methods for capturing 3-D shape data, all of which are based on light sensing. Other modalities, such as acoustic waves or Shape Tape™ [19] are not discussed as they do not provide sufficient spatial resolution for use in modeling applications.

2.1.1 Computational Stereo

Computational stereo involves two or more cameras, which can be thought of as mimicking the stereovision ability of humans. Given two images of a scene and the relative location of the two cameras used to acquire the images, distance to the points in both images can be estimated. The distance values must be computed by detecting corresponding features in both images. A feature commonly used is the correlation between brightness patterns in both images [23]. However, correlation algorithms tend to be sensitive to different illumination in both cameras and different scale of features [33]. It should be noted that the close-up views of the human body that we require provide few

good features for the correlation algorithm to use. For example, an object such as the human arm has no detectable edge features and generally has a uniform surface appearance. Fiduciary marks could be added on the subject to help relieve this problem, as demonstrated in [27].

The main advantage of computational stereo is the data acquisition rate. In capturing the shape of the arm, for example, the subject could simply exercise their arm and have its shape captured in real time. Because of this, computational stereo vision systems could be a useful data acquisition tool, and we propose to investigate their use in future work.

2.1.2 Laser Range Scanners

Laser range scanners work by projecting a laser light beam towards the object of interest. The surface reflection of the light is captured by a detector. The detector observes a *spot* where the light beam hits the object. There are then two ways that this scheme can be used to measure distance. Active triangulation techniques exploit an accurate measurement of the location of the spot in order to compute a distance, as shown in Figure 7. Alternatively, time-of-flight techniques measure the time delay of the light in reaching the sensor.

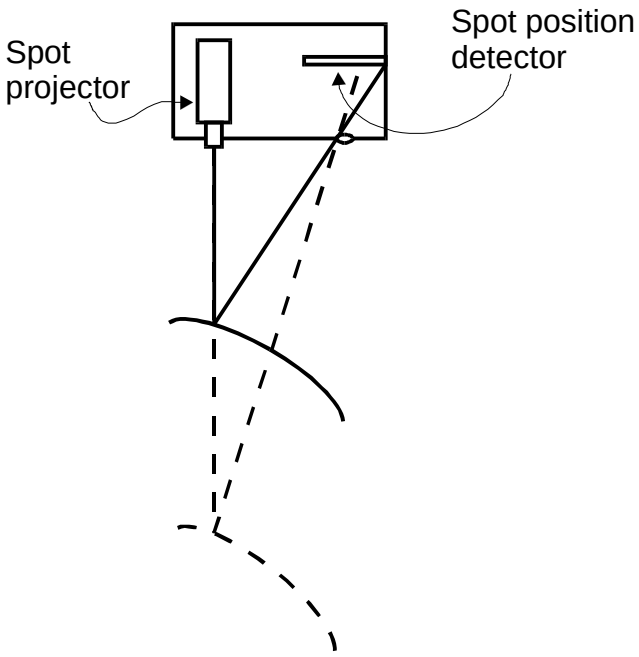

Figure 7: The operation of an active triangulation scanner

All active triangulation techniques have a range sensitivity that falls with increasing range. For ranges of ten meters and above, time-of-flight techniques have superior accuracy. Their sensitivity is essentially independent of range and is on the order of 10-20 mm. However, for our

application the scanner is always within two meters of the model. In this range, active triangulation scanners have a range accuracy of less than one millimeter.

In the configuration shown in Figure 7, we only have the distance from the range scanner to one point on the object. By driving the spot projector with a scanning pattern, much like the operation of a television screen, we can obtain the larger set of range points which comprise a range image. Scanners return a two-dimensional array of values $p_{i,j}$, where the indices correspond to the elevation and azimuth angle of the individual samples, and the values represent the distance from the scanner to the point in question. These values are then converted into three-dimensional Cartesian coordinates. The resulting set of points is ordered because of the scanning process. Furthermore, the data from any individual scan is restricted to producing a height field because only one range value can be determined for any given sampling direction.

Scanners also return the intensity of the reflected signal. This intensity is a function of the local surface reflectance, the distance between the surface and the camera, and the angle of the surface with respect to the camera projection axis [3]. If only one color of laser is used, the intensity provides a grayscale image of the scene. If three colors of laser are used, such as red, blue and green, a color image of the scene can be produced [3]. In either case, the intensity data is associated directly with the range data points.

2.1.3 Fusion of Images and Sparse Range Measurements

Range scanners can be used in conjunction with video cameras to reduce the scanning time [32]. Intensity images of a scene can be acquired much faster than range images. The camera image is segmented into regions of similar intensity and the scanner can then be directed to take measurements of a small number of points within each region. Interpolation is used where required to obtain a dense range map without holes. The main difficulty of this approach is the care required to properly register the data from both sensors. Because of the interpolation, this approach is most suitable for scenes composed mostly of flat surfaces. We would not have the desired accuracy in our application, where the scene contains almost only curved surfaces.

2.2 Triangular Mesh Construction

The data resulting from the acquisition process is an ordered set of 3-D points. Data sets from multiple viewpoints have to be aligned together in a single coordinate frame to be combined. This registration operation can be made more accurate if performed on a surface instead of a set of discrete points, because two surfaces can be properly aligned even though the representative points of the surfaces will likely not find exact matches. Building a surface thus allows for a registration error which is smaller than the distance between vertices. We therefore build a triangular mesh where the data points become the vertices. Our approach to building this mesh is similar to that of Turk and Levoy [59], which we here describe.

To construct a triangulation we exploit the ordered structure of points from the scanning process, namely that of a fixed 2D array. Every group of four points in adjacent rows and columns, such as points {a, b, d, e} and {b, c, e, f} in Figure 8 are considered for being subdivided into two triangles. The shortest diagonal arbitrates the choice of triangulation. For example, the edge {b, f} would be inserted instead of the edge {c, e} in Figure 8. If a point is missing its neighbors do not bridge the gap, leaving a "hole" in the surface. This can occur because the scanner can fail to determine a distance in particular circumstances.

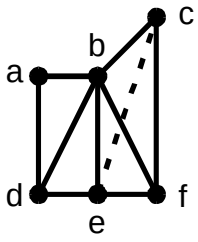

Figure 8: Constructing a mesh from range data points.

To avoid erroneously connecting points, such as across depth discontinuities as shown in Figure 9, we compare the distance *d* between each pair of vertices against a threshold. This threshold is slightly larger than the maximum known distance between sample points for a planar surface.

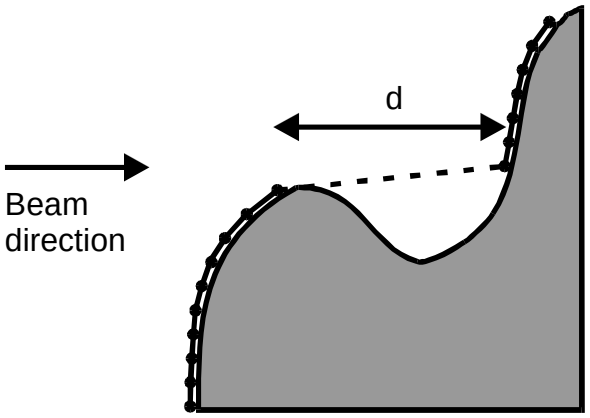

Figure 9: Depth discontinuity in range data occurring because of camera orientation.

The depth threshold criterion may falsely introduce holes, meaning that it is possible that it would fail to join points that should in fact be joined. For example, this problem would arise if a real surface actually did follow the dotted line shown in Figure 9. However, we prefer to acquire large amounts of data to ensure accuracy rather than making guesses from sparse data. As shown in Figure 10, scanning from another direction reveals details that would have been missed if the points across the depth discontinuity were erroneously connected. Our approach, as that of other work in range imaging [59], prefers the merging of multiple partially redundant sets of data.

2.3 Range Image Registration

In order to combine data from multiple viewpoints we have to *register* the different images into a single coordinate system. Because of self-occlusion, data from different viewpoints must be obtained to build a complete model of an object. This is achieved by either moving the range scanner or the object itself. In this thesis, the term registration is used to mean the alignment of two 3D data sets into a single coordinate frame. The registration process determines the relative pose of the range scanner when the two data sets were acquired by aligning the data sets' common regions. Registration is referred to

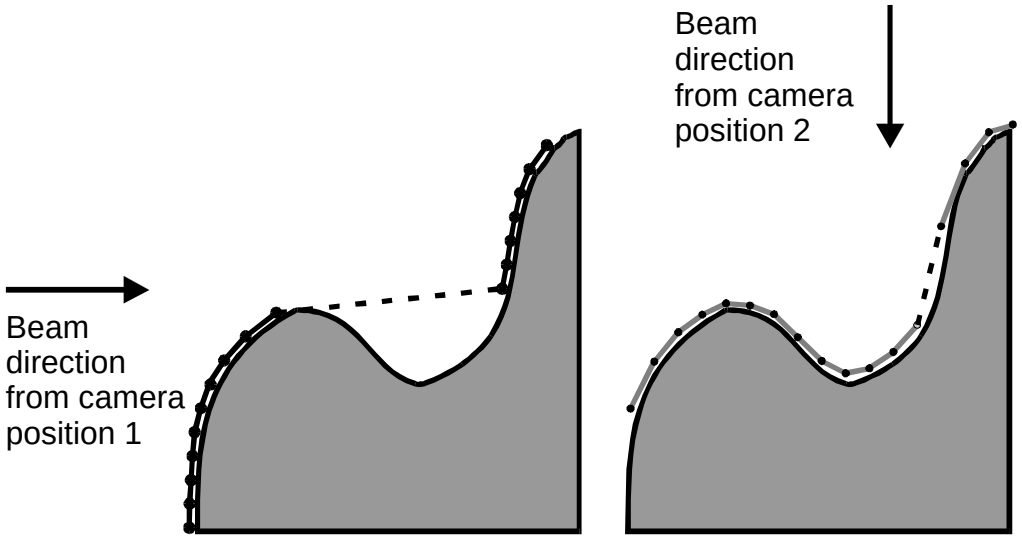

Figure 10: Range data points acquired when scanned from two perpendicular viewpoints.

as pose estimation, alignment, or motion estimation in various contexts.

One of the early applications of registration algorithms was for the construction of topographic maps. The range data was obtained from stereo aerial photographs and the data could subsequently be integrated by manually aligning landmarks. A similar problem arises in combining sonar maps of the sea floor [37], where registration of data sets is obtained by matching isocontours.

In computer vision, a similar problem exists in 3D-object recognition [22]. In this context, registration is called *pose estimation* and usually consists of extracting features such as corners and surface curvature from the data, and then solving for the 3D transformation that aligns the extracted features with their corresponding model features. The accuracy of feature-based registration directly depends on the accuracy of locating the features, which is often problematic because of noisy data. Shape inspection of industrial parts is also related to 3D-object location. Aligning 3D data with a model allows deviations from the model to be discovered.

The iterative closest point (ICP) algorithm [9] is one of the best known registration algorithms and is one we shall discuss in detail. It was motivated by applications involving shape inspection of free-form surfaces. Free-form surfaces are difficult to register using feature matching techniques because the only extractable feature is curvature. ICP does not need to look for features. Instead, it operates directly on the distances from the data points to the closest points on the target surface.

One of the most challenging applications of registration involves creating an integrated surface model from several partially overlapping views. In some applications the registration transform can be obtained directly from the measurement process, by either moving the object between the views with a high-precision robotic system, or by constraining the object motion, such as only allowing a planar rotation [61]. However, the first alternative is very expensive or impossible for several applications, and the second one is restrictive in the obtainable views.

The registration problem can be formulated as an optimization problem that finds the rotation and translation which minimizes a distance metric. The metric usually sums the squared values of one or more of the following:

- the distance and/or orientation difference of matched features;
- the curvature differences of the aligned surface locations;
- the shortest distance from a point in one data set to points on the other;
- an *approximation* of the shortest distance from a point in one data set to the points on the other.

The registration of two views of a static scene is a general problem. In the worst case, no model of the objects in the scene is available. Therefore the overlap between the data sets, which is required for registration, must be estimated by the algorithm. Furthermore, the desired transformation may include large rotations and translations from some initial configuration. In three dimensions, this makes a systematic search of the registration space too slow to be practical.

In the following discussion, we first present algorithms that align sets of corresponding features, which is the key step in solving for data registration. We then revisit other approaches to solving the registration problem. With the exception of using pre-registered data sets (described in §2.3.1), all registration methods need to compute a rigid transform that can best align two ordered sets of features where feature *i* from one set corresponds to feature *i* in the other set.

Horn [29], Faugeras and Hébert [22] and Arun et al. [2] all provide closed form solutions for the rigid transform that minimizes the sum of the squared distances between two sets of points where correspondence is established. Faugeras and Hébert's method is similar in structure to Horn's, so we shall describe the methods of Horn and Arun et al., beginning with the latter.

We denote the two data point sets to be aligned as A = $\{p_i\}$ and B = $\{p_i'\}$, where i = 1, 2, ..., N. The required correspondence can be specified as:

$$p_i = Rp_i' + T + D_i$$

where R is a rotation matrix, T a translation vector, and D_i an (unknown) noise vector. The least-squares solution to the above equation requires solving for R and T to minimize

$$e = \sum_{i=1}^{N} \left\| p_i - (Rp_i' + T) \right\|^2$$

Assuming complete overlap of the data sets and equal sampling density, it was shown that A and the transformed set B will have the same centroid [30]. Therefore, the least-squares problem can be broken down into two independent problems. First, subtract the two centroids to find the translation. Second, find the rotation that minimizes the least-squares problem.

We will now further examine two methods to find the rotation R. Let the centroids of point sets A and B be p and p', respectively. We define two new sets of points $\{q_i\}$ and $\{q_i'\}$, where $q_i = p_i - p$ and $q_i' = p_i' - p'$. We then compute the 3 x 3 matrix

$$H = \sum_{i=1}^{N} q_i' q_i^T$$

and its singular value decomposition (SVD):

$$H = U \Lambda V^T$$

where Λ is a 3x3 diagonal matrix with non-negative elements, each of these elements being the singular values. The matrix X, computed as

$$X = VU^T,$$

yields the orthonormal matrix for a least squares solution. However, orthonormal matrices can also imply reflections, and thus it remains to be verified that X is a representative rotation matrix. If the determinant of X is +1, then X is the rotation matrix R that we seek. A determinant of -1 implies that X is a reflection instead of a rotation, and needs to be handled separately: if all the noise vectors N_i are zero and the points in set A are coplanar (but are not all collinear), one of the three diagonal elements of Λ will be zero. In that case, let these elements be $\lambda_1 > \lambda_2 > \lambda_3 = 0$. Expanding the equation defining H, we obtain:

$$H = l_1 u_1 v_1^T + l_2 u_2 v_2^T + 0 \cdot u_3 v_3^T$$

where u_i and v_i are of columns of U and V. Changing the sign of u_3 or v_3 does not change H, and therefore, if $X = VU^T$ minimizes e, so will $X' = V'U^T$, where

$$V' = [v_1, v_2, -v_3]$$

Since X is a reflection, its determinant being -1, then X' is a rotation.

If the determinant of X is -1 and none of the singular values are zero, then neither the points in A and B are coplanar, and there is no rotation that yields a smaller e then the reflection represented by X. This can happen only when the noise is very large, and the algorithm can not handle this situation. Arun et al. suggest that any least-squares method is likely to be confounded by this situation and they suggest the approach of Fischler and Bolles [25].

We now describe Horn's method, which is the most widely employed, possibly because it is explained in the clearest fashion. As in Arun et al.'s method, the two sets of points are first centered by subtracting the respective center of masses from each of the two point sets. The following matrix is then computed:

$$M = \begin{bmatrix} S_{xx} & S_{xy} & S_{xz} \\ S_{yx} & S_{yy} & S_{yz} \\ S_{zx} & S_{zy} & S_{zz} \end{bmatrix}$$

where

$$S_{xx} = \sum_{i=1}^{n} x_{q,i} \cdot x_{q',i}$$

$$S_{xy} = \sum_{i=1}^{n} x_{q,i} \cdot y_{q',i}$$

$$\dots$$

and $x_{q,i}$ is the value of the x coordinate of the point i in set q.

This matrix *M* contains all the information required to solve the least-squares problem for rotation. The matrix *N*, as shown below, is then computed using the elements of *M*,

$$N = \begin{bmatrix} (S_{xx}+S_{yy}+S_{zz}) & S_{yz}-S_{zy} & S_{zx}-S_{xz} & S_{xy}-S_{yx} \\ S_{yz}-S_{zy} & (S_{xx}-S_{yy}-S_{zz}) & S_{xy}+S_{yx} & S_{zx}+S_{xz} \\ S_{zx}-S_{xz} & S_{xy}-S_{yx} & (-S_{xx}+S_{yy}-S_{zz}) & S_{yz}+S_{zy} \\ S_{xy}-S_{yx} & S_{zx}+S_{xz} & S_{yz}+S_{zy} & (-S_{xx}-S_{yy}+S_{zz}) \end{bmatrix}$$

Horn shows that the eigen vector that corresponds to the most positive eigen value of the matrix *N* is the quaternion that minimizes *e* [29]. The four eigen values of *N* can be found by solving the quartic equation

$$\det(N - lI) = 0$$

where *I* is the 4x4 identity matrix. If we let λ_m be the most positive of the four eigen values, the corresponding eigen vector $\mathbf{e_m}$ can be found by solving the homogeneous equation

$$(N - l_m I)\mathbf{e_m} = 0$$

Arun et al. show that in terms of execution speed, their method is slower than Horn's method for small sets ($n < 10$), but the methods require about the same time as the point sets grow [2]. Both methods have O(n) complexity, where n is the number of data points. Arun et al.'s method can handle reflection, but this is never needed for laser range image registration and causes problems if the reflection would reduce the error more than any rotation would. In our system, we use Horn's method.

2.3.1 Registration From Acquisition Setup

An obvious way to avoid the registration problem altogether is to rely on precise information of the range camera's position and orientation. Blais and Levine [11] refer to this approach as *open loop registration*. Vemuri and Aggarwal [61] relied on a calibrated system to obtain the registration transforms using a turntable as a base for the object to be modeled. A pattern painted on the turntable could be observed in the intensity images and

used to determine the rotation angle of the platform. This rotation angle was all the information required to register the range images together since no other movement was allowed and the sensor was fixed.

Employing artificial data, Roth-Tabak and Jain [47] simulated a system in which a moveable sensor was used in a virtual environment to build a representation of its surroundings. Because the exact pose of the sensor was known, all range images could be registered in a trivial fashion.

The major drawback of open-loop registration methods is that they suffer from relatively high inaccuracy, and the acquisition system becomes very expensive or the sensor's range of movements becomes very limited. In order to solve these problems, more sophisticated data-based registration methods need to be used. Registration from acquisition setup can be used, however, to provide a coarse estimate of the proper alignment, which is sometimes needed for more accurate approaches (see §2.3.3).

2.3.2 Registration Using Features

These methods attempt to detect features in two data sets and establish correspondences between these sets of features. The number of features can be constrained or over-constrained. If the feature-set is exactly constrained, there is the risk of establishing a wrong correspondence between features and therefore computing a wrong registration transform. If the set of features is over-constrained, a decision must be made for dealing with the possibly conflicting registration transforms.

Bolles and Horaud [13] use a model-based object recognition system where distinctive features are selected from a known model. The selection criteria for a feature include the uniqueness of the feature among all models, its expected contribution to determining the pose of the object, and the cost and likelihood of detecting it. Bhanu [10] uses an approach where generic features are selected from a simple model.

Faugeras and Hébert [22] are interested in recognizing and locating known objects from range images. The approach locates geometrical features, matches them to a model,

and finds the transformation that aligns the features. A surface representation is chosen for its robustness to occlusion and viewpoint position and because of its simple rules of transformation when rotated or translated. The use of region growing (which involves the fitting and growing of surface patches) and plane and quadrics fitting to extract the surfaces from the range data is discussed. Once the surfaces are extracted, strategies for matching surfaces from the data to surfaces from the models are explored. After rejecting relaxation matching and the Hough Transform and Clustering, a tree search strategy is proposed. The primary limitation of this approach is that registration may fail with large areas of low curvature in the data.

Schwartz and Sharir [50] use a model-based approach; from a model they find "characteristic curves" – edges that are relatively immune to noise, such as the boundary curve separating two differently colored portions of an object surface. After finding some of these characteristic curves on the range image or its associated intensity image and performing a smoothing operation on it, the two are matched together.

Bergevin et al [6][7] implemented an indirect feature-search as a preliminary step for a heuristic search in the registration search space. An initial registration transform is first computed as follows. The Delaunay triangulation of the range data in both views is computed and the resulting mesh is simplified. Corresponding regions of the two views should now be covered by triangles of similar areas, since the size of a triangle depends mainly on the local surface curvature and orientation. Pairs of neighboring triangles from one view are matched to similar pairs from the second view using criteria such as the distance between centroids of the two triangles in each pair and the angle between their normals.

Aligning the matched triangle pairs gives an initial registration of the views, but because a separate transformation is computed for each of the matched pair, there are likely to be many false matches which produce wrong registrations. The registrations are ranked according to the overlap of the registered views when projected to the same plane. The best registration is used as the initial estimate in the following step. The next step is the search in registration space, and is discussed on page 26.

Kamgar-Parsi, Jones and Rosenfeld [37] deal with maps of the sea floor, which are typically without distinctive features. Contour lines are extracted from the images and used as a matching feature. Because of acquisition errors, however, there is often single self-consistent global alignment. To alleviate this problem, non-rigid transformations are applied to bend and twist the images. The technique is not general, however, as the images are assumed to be expressible as a height field. The problem associated with grouping together several range images, however, also applies to data acquired from range scanners (see §3.1).

Johnson and Hébert present a new set of features [36] used in the context of object recognition [35]. Shown in Figure 11 is the fundamental shape element they propose: the oriented point *p*. Also shown in the figure is how an oriented point defines a local coordinate system with coordinates α, the perpendicular distance to the line λ, and β, the signed perpendicular distance to the plane π. This is conceptually similar to a cylindrical coordinate system, minus the polar angle about the line λ. Given a set of points *x*, a spin image is constructed for a point in that set by mapping the other points to the coordinate system mentioned above. The advantage of these spin-images is that they are much more discriminating features than principal curvatures and other traditional features. The registration transform computed with this method nevertheless needs to be refined with one of the free-form registration techniques. Because the time required to construct a spin-image can be substantial, we choose to have the user manually provide a coarse registration transform instead of using this method.

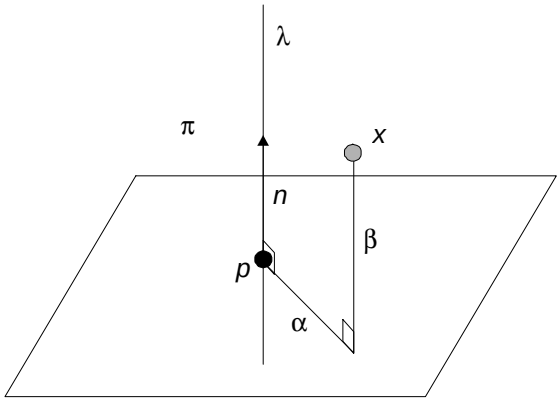

Figure 11: Oriented point basis; a spin-image for the point p is constructed from the coordinates (a, b) for each pt x in the set.

2.3.3 Registration of Free-Form Surfaces

The feature-based methods are susceptible to false matches because of the difficulty of establishing correspondences between features. Even when a correspondence is correctly established, the difficulty in accurately locating the features limits the precision of the registration transform that can be computed.

In this section we examine iterative methods that do not search for features in the data. One of the disadvantages of these methods is that they can only be used to refine an estimate of the registration transform; they require the two range views to be approximately registered before being applied. This can be accomplished through information from the acquisition system or through user interaction. One strategy would be to use a feature-based method to provide a coarse registration, and refine the alignment with a free-form registration technique. We choose not to employ this strategy because of the time that robust feature-based methods require to execute. For our system, we prefer to have the user spend on the order of thirty seconds to manually provide a suitable initial alignment of the range data sets.

Rather than attempting to match discrete features, least-squares methods minimize a distance metric evaluated over all the data points [9]. The following is a commonly-found least-squares distance metric:

$$f(T) = \sum_{i}^{N_p} \left(Tp_i - q_j \right)^2$$

where q_j is the closest point found in the data to be matched as shown in Figure 12, and T is the transform that would align P and Q (see Figure 13).

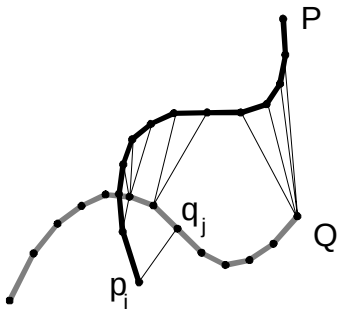

Figure 12: The closest point q_i in the set Q to point p_i in set P.

Here we are looking for a rigid 3D transformation T that minimizes the distances of the points p_i in the set P to the point q_j. This point q_i is the point in another set or surface Q that is closest to p_i. This optimization problem could be solved using a conjugate gradient method. However, such methods require computing gradients and the numerical evaluation of gradients through finite differences is time-consuming if there are many points [9].

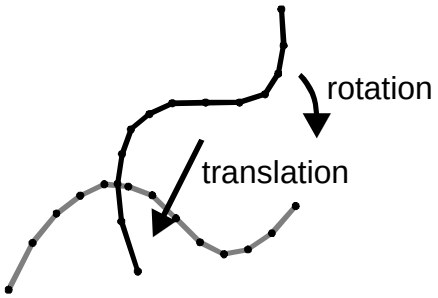

Figure 13: Transform that would align the two data sets.

Champleboux et al. [15] solved the gradient problem by pre-computing a 3D-distance map called an octree spline that allows rapid approximations of the gradients. This octree spline map is computed as follows: Begin by building a restricted octree for one of the two sets of data points [48]. A restricted octree is one in which the size of two nodes that share a common boundary in space never differs by more than a factor of two. This octree is augmented by calculating the distance to the closest data point from each of the eight corners of the cube representing each leaf in the octree [39]. The shortest distance to any data point from any point p within the space enclosed by the octree can be quickly estimated by finding the leaf in which p resides and tri-linearly interpolating the actual distance from the eight distance values stored in this leaf. Accuracy is maintained if the linear interpolation provides a good approximation of the distance.

The octree spline data structure is pre-computed for one of the range images. A least-squares minimization is then performed, applying the computed transformation to the image for which the octree spline representation was not computed. The main problem associated with this method is that the "model" must completely overlap the other range image. Points that have no correspondence must be eliminated from the range image. This is not suitable for our application, since our range images have complementary information with minimum overlap. As with other iterative methods, it is also susceptible to convergence to an erroneous local minimum if the initial alignment is poor.

A simpler approach involves iterating between pairing points and solving for the best-fit registration transformation:

$$f(T) = \sum_{i}^{N_p} \left(T_k p_i - q_{k,j} \right)^2$$

The pseudocode for the algorithm is:

```
Given the initial transform T0, let Q0 = Q
Repeat:
  Apply the transform Ti to Qi-1 to get Qi
  Determine the closest point in P for each qi
  Determine the transform Ti+1 that minimizes the distances
     between pairs of closest point
Until desired accuracy is achieved
```

Here, an initial estimate of the registration transform (T_0) is given; it is applied to the point set Q, and while holding the transformation fixed, the closest points in P for each q_i is found. A new best-fit transformation is the computed and applied, after which the process repeats, alternating between determining the closest points and computing best-fit transformations.

Besl and McKay's Iterated Closest Point (ICP) algorithm implements the above approach [9]. Horn's quaternion-based approach [29] is used to find the transform that minimizes the distances between the pairs of closest points. The algorithm achieves fast convergence in the first few iterations, but the convergence slows as the algorithm approaches the local minimum. Thirty to fifty ICP iterations are necessary for the algorithm to converge. The chosen convergence criterion is typically a registration error of approximately 0.1% of the model shape size.

The number of iterations can be approximately cut in half by using a variation of a basic line search method of multivariate unconstrained minimization, as described in Numerical Recipes [45]. As the ICP algorithm proceeds, a series of registration vectors q_i is generated. These vectors can be expressed with seven numbers – three for the translation and four for the rotation, which is expressed as a unit quaternion. These vectors represent the cumulative registration transform at step *i*. If we define

$$\Delta q_i = q_i - q_{i-1}$$

to be the transform that is applied at step i-1 to arrive at step *i*, then the angle in $\Re^7$ between the last two transforms can be defined as

$$q_i = \cos^{-1}\left(\frac{\Delta q_i \bullet \Delta q_{i-1}}{\|\Delta q_i\|\|\Delta q_{i-1}\|}\right)$$

If the angles θ_i and θ_{i-1} are smaller than a given thresholded value, then there is good alignment for the last three registration vectors and the next vector, q_{i+1}, can be extrapolated rather than computed.

Besl and McKay prove that the ICP algorithm will always converge to a local minimum [9]. The initial estimate T_0 must be close to the global minimum, otherwise ICP can reach another local minimum. This can be addressed by exploring multiple orientations of the data sets, each followed by the application of ICP. The rotations are chosen by uniformly sampling the unit sphere in $\Re^4$ to obtain quaternions. For a complex object, this may involve on the order of 300 initial orientations. A restriction of the ICP algorithm as presented in [9] is that one data set must be a proper subset of the other data set. Thus the technique cannot be directly applied to the registration of two data sets which only partially overlap.

Potmesil [44] uses an approximation to avoid searching for the closest point, especially if Q can be expressed as an analytic surface rather than a point set. In his approach, each point p_i is paired with the intersection of its normal vector and the surface Q. This method can also create better pairs in the sense that the algorithm converges faster.

All such algorithms share a common flaw: few of the closest point pairs that are assumed to correspond do actually correspond to each other. Such mutually incompatible point pairs fight against each other, which leads to slower convergence. To avoid this, Chen and Medioni [16] present a new formulation based on the work of Lowe [41]. Many unnecessary constraints are removed: the points p_i are paired with the tangent plane S_j of Q, evaluated at the locations where the normal vector n_i associated with p_i intersects Q (see Figure 14).

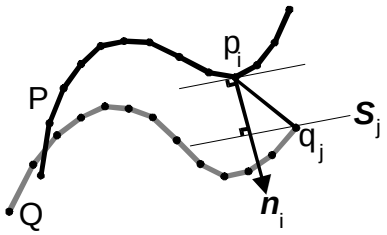

Figure 14: Matching points with tangent planes.

The algorithm remains a variant of ICP. For each iteration, an intermediate transform minimizes the distance between pairs of matching features is applied, and the process repeats until convergence. The optimization metric used at each iteration of the algorithm is the function that measures the distance of control points to the plane they are matched with. This means that no penalty is incurred when a control point "slides" along a path parallel to the plane it is matched with. The least-square minimization problem is linearized by assuming that the images are already almost registered and replacing the sine and cosine terms in the transformation matrix by linear terms.

Pulli [46] points out that the Chen-Medioni algorithm should converge much faster than the accelerated ICP because of the "free-sliding" property. However, no one has yet directly compared the two algorithms. Although it is true that the Chen-Medioni algorithm may converge in fewer iterations, the time per iteration can be potentially larger than that of ICP. Furthermore, ICP can easily be made more robust by augmenting it with heuristics, as will be presented in §4.1, while the Chen-Medioni algorithm is more difficult to improve in this fashion [46]. Finally, the Chen-Medioni algorithm probably requires a more accurate estimation of the registration transform as input because of the approximations used to linearize the least-squares problem.

Szeliski [54] describes a method for estimating motion from sparse range data. The goal was to estimate the motion of the observer between two range image frames of the

same terrain. A smoothing spline surface approximation of the range data points is first created. A conventional steepest descent algorithm is then used to rotate and translate the second data set so that it minimizes the sum of the z differences between the points and the surface. The steepest descent approach used is slower than ICP. The method is demonstrated on synthetic terrain data. It is not clear that the algorithm would generalize well for our applications.

Traditional minimization methods often limit the formulation of the objective function. For example, many optimization methods do not work unless the objective function is differentiable. As well, methods that locally refine an initial estimate are susceptible of getting caught in a local minimum, and there can be several local minima close to the global minimum. Some stochastic methods, such as simulated annealing, do not have these limitations. They simply require that the objective function can be evaluated, and employ a random element to avoid local minima. The drawback is that the evaluation of the objective function may be required tens of thousands of times.

Blais and Levine [11] use a stochastic optimization method called *very fast simulated reannealing* (VFSR) [31]. A hash table is used to map the coordinates (*x, y, z*) of a point back to the indices *i* and *j* in the array of sampled points. Although this makes finding closest point pairs an O(1) process, it assumes that the data is organized as a 2-D array. If this structure is modified, for example by integrating range images together after they have been registered, this technique could not be used.

After building the look-up tables described above, control points are chosen from one range image by uniform subsampling. Using an initial registration transform estimate obtained from the acquisition setup, the control points are transformed into the reference frame of the other range image. The cost for each point p_i is then a thresholded absolute distance to its associated point q_j. Thresholded here means that if the distance is greater than a specified threshold, the cost is set to be zero. The final objective function is the sum of the cost for each point, normalized over the number of pairs used. If the percentage of points that are used in pairs is less than a user-specified threshold (known

as the *overlap factor*), the transformation is discarded. The example given in [11] uses an initial estimate of the registration transform that requires little refinement.

The optimization problem can also be formulated as a search problem. By applying small rotations and translations to one range image in a trial-and-error fashion, one can attempt to find the optimal registration transform. Alternatively, one could try to discretize the registration space, try all the solutions, and choose the best one.

Bergevin et al [6][7] implemented a heuristic search in the registration search space. First an estimate of the registration transform is computed as described in §2.3.2. This estimate is placed at the root of a tree. A node is expanded by perturbing each of the six parameters (three for translation and three for rotation) in either direction, the amount of perturbation depending on the depth of the node in the tree. The most critical aspect of the graph search process is the generation of the successor nodes. Unfortunately, no strategy exists that would take into account the properties (positional and rotational errors) of all the nodes from the root to the parent to generate a successor. A search limited to a small number of directions, however, such as the six parameter axes might never proceed to the optimal point, given that the rotational and translational parameters are not completely independent of each other. They report that Chen and Medioni's method [16] produces better results than the search algorithm. However, the search method can handle much more complex objects than free-form registration methods. The objects considered in previous papers are compact and tend to converge towards a convex shape as the scale becomes coarser, such as the car model used by Potmesil [44], the owl statuette used by Ferrie and Levine [24], and the bust shape used by Chen and Medioni [16]. Bergevin considers more complex multi-part objects such as a pencil sharpener. The higher difficulty results from the more dramatic and less predictable self-occlusions occurring in most views.

Taubin [55] presents an exploration of implicit algebraic non-planar 3-D curve and surface estimation, with applications to position estimation without feature extraction. A method of approximating data with implicit algebraic forms and an approximate distance metric is described. Shapes can be identified based on generalized eigen-values and the

registration transform can be recovered directly. The computed representation often weakly resembles the object, but it provides a sufficient spatial description of the object at different levels of approximation for the purposes of position estimation. The method presented is shown to be useful for planar curves and space curves, but it is unclear that the effectiveness generalizes well to more complicated surfaces such as terrain data. The technique is demonstrated on some relatively simple shapes such as the sketch of a wrench and a gear-like disc. Taubin states that the numerical methods of the approximate distance fit tend to break down on polynomials of degree ten or above because of the wide variation of the terms in polynomials of such high degree.

2.3.4 Conclusion

Our application involves the registration of views of the human arm, and as such does not likely provide enough features for feature-based registration methods. We therefore choose to implement one of the free-form approaches. We adopt ICP because it is well documented and is the most widely used. It runs quickly, especially when combined with the acceleration step, as described by Besl and McKay [9]. Finally, it is guaranteed to converge to a local minimum. If it converges to an erroneous minimum, this is generally noticeable and thus ICP is manually restarted from another initial configuration. We let the user provide an initial registration transform. Given the proper interface, this can be done in a matter of seconds.

2.4 Range Image Integration

Range image integration refers to the process of creating a single representation from the sample points of two or more range images. This step is performed after the range images have been aligned together with a registration procedure. In general, the following steps are performed by an integration procedure:

- Elimination of overlapping surfaces. Because registration procedures require overlapping surfaces, there is a significant amount of redundant information.

- Warping of range images to eliminate any ‘step’ at the junction of two range images. This step occurs because of imperfect registration or because of errors in the data acquisition process (see §3.1).
- Merging the data sets together. No holes should remain where two range images meet.

We present two approaches from the literature for range image integration.

2.4.1 Unstructured Integration

Unstructured integration methods seek to integrate range images from an arbitrary collection of points in $\Re^3$ assuming no connectivity information. All the range points from multiple scans are gathered together and presented to this routine. This is typically an unnecessarily difficult problem for range image integration, as the acquisition process returns a set of ordered points. Even after multiple range images are integrated together, connectivity information typically remains available.

Boissonnat [12] proposes the Delaunay triangulation of a set of points in 3-space as the basis of such a reconstruction. Alpha shapes are introduced for points in the plane in [20], and Edelsbrunner and Mücke [21] extend it to arbitrary dimension and to weighted sets of points. Alpha shapes are a generalization of the convex hull of a point set. The approach requires a parameter, α, which is the size of the largest simplex that is allowed in the final shape. There is currently no automatic way to select this parameter, thus requiring user intervention.

Hoppe et al. [28] use graph traversal techniques to help construct a signed distance function from a collection of unorganized points. An isosurface extraction produces a polygon mesh from this distance function.

Amenta, Bern and Kamvysselis [1] describe a Voronoi-based algorithm that is proven to produce a topologically correct surface. The guarantee is based on the local sampling density, and intuitively captures the notion that featureless areas can be reconstructed with fewer samples.

Although these algorithms are widely applicable to a broad class of 3D point sets, they discard reliable information from range scanners. As a result, they may be well behaved in smooth regions of surfaces, but they are not always robust in regions of high curvature and in the presence of systematic range distortions and outliers.

2.4.2 Structured Integration

Structured integration methods make use of the structure of the data acquisition process in order to integrate the range images. Information that is readily available from range scanners includes adjacency information between points, error bounds on a point's position, and the viewing direction of the range scanner.

Soucy and Laurendeau [52] use a surface-based approach to combine multiple range images. Points that are part of overlapping surfaces are identified and used to create triangulations that are stitched together by a constrained Delaunay triangulation.

Turk and Levoy [59] propose a "zippering" scheme to merge range images; this is an incremental algorithm that erodes redundant geometry, followed by merging along the remaining boundaries. This approach will be described more in depth later in this thesis, as we base our integration scheme on similar ideas.

Curless and Levoy [17] present a robust, incremental volumetric integration method. The volume encompassing all the pre-aligned range images is first voxelized. Each range image is then converted to a signed distance function and a weight function, sampled at each voxel. For each range image i, the signed distance $d_{i,j}$ from each voxel j to the closest surface point along the line of sight of the sensor for that particular image is determined. A weight $w_{i,j}$ is also computed according to the difference between the local surface orientation and the viewing direction. For each voxel j, the distance and weight functions from each range image i is combined according to the following rules:

$$D_j = \frac{\sum_i w_{i,j} \cdot d_{i,j}}{\sum_i w_{i,j}} \qquad W_j = \sum_i w_{i,j}$$

If an isosurface extraction was to be performed at the zero-crossing of the combined signed distance function at this time, the resulting mesh would have holes at unseen portions of the surface. In order to fill those holes a *space-carving* step is performed while constructing the distance function. After updating the voxels near the surface, as described above, the line of sight is followed back from the observed surface and the voxels traversed are marked as “empty”. All the voxels are initially marked “unseen”. After this procedure voxels that contain no surface can be categorized depending on whether they were actually empty or simply not observed by the scanner. When extracting the isosurface, a surface is also extracted between regions seen to be empty and regions that are still unseen. The method is robust and can handle a large number of images and has been tested with models requiring the integration of up to 70 images. However, it is relatively slow, requiring on the order of four to six hours on a MIPS R4400 processor for a final model containing 1.8 million polygons, and must be made less accurate than other surface-based approach (cf. [59]) to speed it up. Like most other integration techniques, it has difficulty dealing with sharp corners and thin surfaces.

2.4.3 Conclusion

Because we use structured range data, we choose to apply one of the structured methods. We want highly detailed models and therefore prefer Turk and Levoy’s zippering algorithm to volumetric methods. The latter tend to be limited in the level of detail they can support because of the necessity to voxelize the workspace.

2.5 Animation of Polygonal Models

To the best of our knowledge, no work has been published on modeling the deformation of the human shape using range data. Nevertheless, some related work is briefly discussed here.

Turner and Thalmann [60] describe a layered construction technique called the *elastic surface layer model*, where an elastically deformable surface is wrapped around a

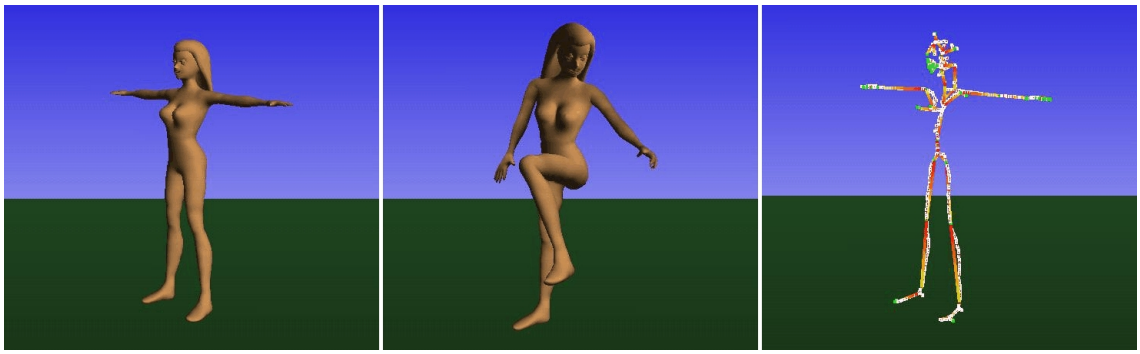

Figure 15: Animation generated by Teichmann and Teller.

kinematic articulated figure. The skin is free to slide along the underlying surface layers (such as fat and muscles) following a physically modeled spring network. Other effects, such as skin deformation at the joints and follow-through of the skin can be obtained by tuning the parameters of the physically-based model. Skin deformations due to muscle contractions are not modeled.

Teichmann and Teller [57] present an algorithm for establishing an articulated skeleton from a closed polygonal model. Given a 3D polygonal mesh representing an articulated figure, they create an articulated skeleton from the 3D Voronoi diagram of the mesh vertices. User interaction is required to specify which points on the skeleton should be articulated. There is currently no joint-based deformation implemented as part of the system. A spring network animates the skin. Figure 15 shows two keyframes produced by their system, and the associated I-K skeleton.

Yasumuro, Chen, and Chihara [63] model the human hand using range image data and an underlying skeleton. The skeleton is constrained to help produce a natural hand posture. The range data was generated by scanning the plaster model of a human hand. Each skin point is associated with an underlying bone. When a bone moves relative to its neighbors the associated skin points move rigidly with it, as shown in Figure 16.

Figure 16: Hand animation generated by Yasumuro, Chen and Chihara.

Wilhelms and Van Gelder propose a specific type of anatomically based modeling of animals [62]. Bones, muscles and other underlying tissues are the key components of the model. Muscles are modeled as deformable cylinders lying between fixed origins and with insertions on specific bones. Muscles change shape in a parameterized fashion as the joint moves. Skin is generated by voxelizing the underlying components and extracting the isosurface. All components are parameterized and can be reused on similar bodies with non-uniformly scaled parts. The results are good, but knowledge of anatomy is mandatory in order to model the muscles. A trial-and-error process is needed to customize difficult muscles. Figure 17 shows the underlying muscle models that was used in one of their earliest implementations. Figure 18 shows how the skin mesh adapts to the various postures of the *proto-human*.

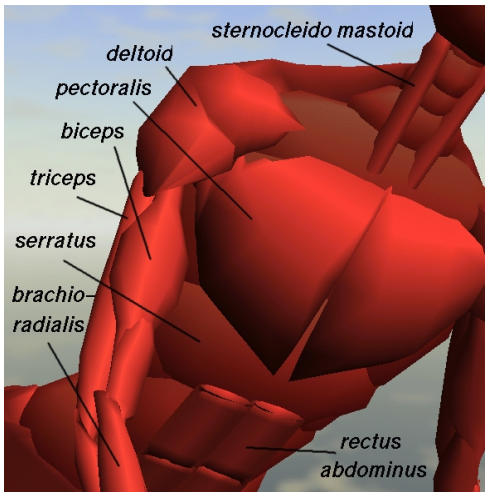

Figure 17: Underlying muscles as modeled by Wilhelms and Van Gelder [62].

Figure 18: Collage showing how skin adapts to changes in underlying muscle shape [62].

Scheepers et al. [49] follow an artistic anatomy approach to model the human musculature. Muscle and skeleton models are implemented using a procedural language. Muscles automatically deform according to the posture of the skeleton. An extra parameter is used to specify muscle tension. Figure 19 shows the underlying models comprising the torso animation that was implemented. Figure 20 shows one of their early implementation of a skin and fatty tissue model on top of the muscle and bone models.

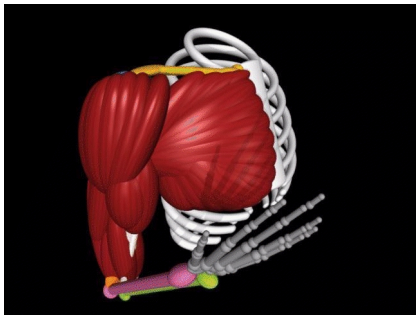

Figure 19: Muscle and bone models implemented by Scheepers et al. [49].

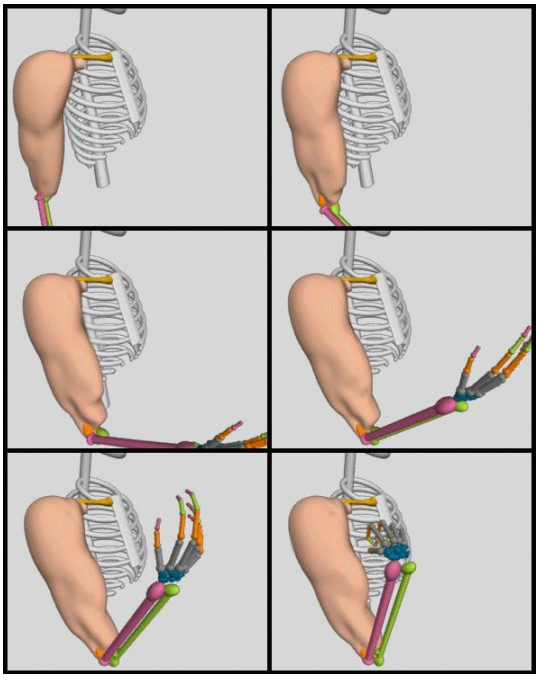

Figure 20: Scheepers et al.'s model after application of a skin and fatty tissue model [49].

3. DATA ACQUISITION

3.1 Scanner description

In this chapter we describe the particular scanner that is used in our work. The general principle of range scanning were described in §2.1.2. Our range scanner was developed by the National Research Council of Canada (NRCC). In 1993, SPAR Aerospace acquired a license to apply the technology to telerobotics.

The scanner uses active triangulation to determine range. However, rather than mechanically pointing the device in order to scan the scene, the projected beam is redirected using oscillating mirrors. In this NRCC patented technique, known as *synchronized scanning*, both the projection axis and the detection axis are scanned over the scene in synchrony. In other approaches, only the projection axis is scanned, while the detector axis remains fixed. With such an arrangement, the spot image position depends on both the range and instantaneous scan angle. Synchronous scanning practically eliminates the scan angle dependence.

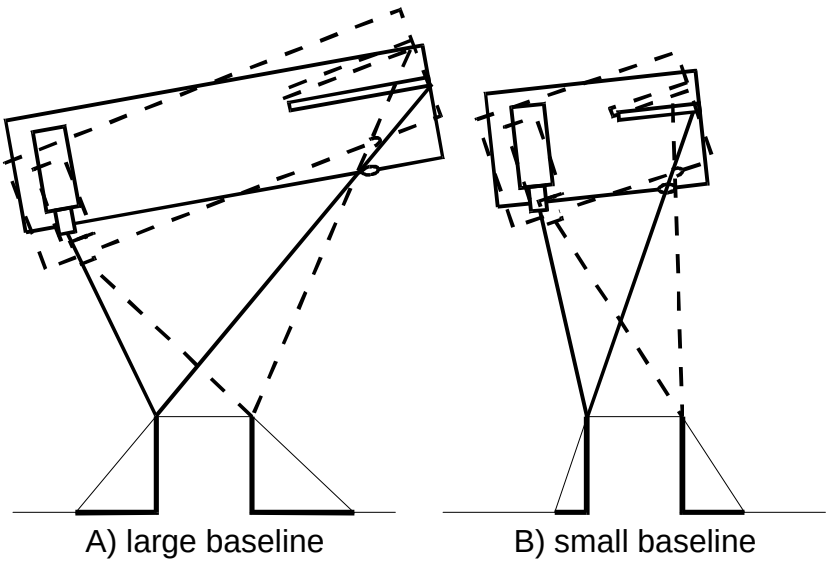

Figure 21: Shadowing effect for different baselines.

As a result of synchronized scanning, it now becomes possible to slant the detector array with respect to the plane of the imaging lens in such a way that the spot image is focused over a large span of triangulated ranges[4]. This results in a significant improvement in the depth of field. Our scanner has a depth of field ranging from .6 m to 15 m. An additional advantage is that the detector does not integrate ambient light over the complete field of view as a fixed detector device would. Hence, the SPAR scanner is relatively insensitive to ambient light. Finally, finite detector length is not wasted on accommodating scan angle. This allows for a smaller triangulation baseline[5] for a given range of accuracy. The smaller baseline minimizes the shadowing effect that plagues standard fixed-detector systems[6] as shown in Figure 21. This also allows for the SPAR scanner to be a relatively small device, as shown in Figure 22.

Figure 22: The SPAR laser range scanner

[4] This is also known as Scheimpflug Condition [14].
[5] The baseline is defined as the distance between the detector and the projector.
[6] Note that time-of-flight laser scanners may not have this shadowing problem, as the paths of the transmitted and returned signals can be almost the same [33].

For our application the scanner was always within two meters of the model. In this range, the SPAR range camera has a range accuracy on the order of .5 mm. The time to acquire a single scan at a resolution of 64 x 64 is approximately 2 seconds.

3.2 Acquisition Setup

The data that range scanners produce can be remarkably accurate under ideal conditions. However, the ideal conditions are sometimes difficult to achieve, particularly when scanning human subjects. We now discuss the acquisition setup that was used, and describe the pitfalls that were encountered. Our interest is in capturing the human form; our experiments capture the varying shape of the arm, as shown in Figure 23. This involves the subject standing still while scans are taken.

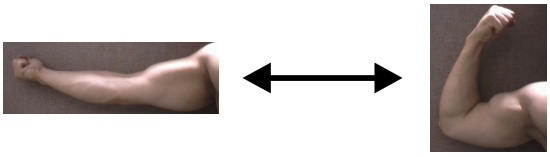

Figure 23: Change in shape of the arm as a function of elbow flexion.

In order to build a complete model of the subject's arm, we need a method of acquiring multiple views. Ideally those multiple views would be captured simultaneously. However, given this is an impossibility with the resources available to us, we chose to obtain multiple views by moving the scanner. Because we want our subject to hold their pose, we prefer moving the scanner to moving the model. Mounting the scanner on a track or robot arm would likely speed the acquisition process.

The range scanner is aimed by using a video camera mounted beside the scanner on the tripod. This is the black protrusion above the hand in Figure 22. The camera provides real-time feedback of the scanner field of view and thereby greatly reduces the setup time. The video camera can also be used to fuse video data with range images [34], although we have not made use of this feature in our implementation.

Multiple scans of the subject are required for two reasons. First, because of self-occlusion, it is not always possible to capture the entire geometry of even one side of single limb. Registration procedures also require significant overlap between the data

sets. Second, a realistic model requires dense sampling. This means that the scanner should be relatively close to the subject. The subject was required to stay immobile for several minutes as a result of this requirement. We use braces to help stabilize the model, but these braces also contribute to unwanted occlusion. In order to obtain more accurate measurements in the future, we suggest providing a more comfortable, non-intrusive way for supporting the subject's arm, or to correct for model sway with a data post-processing step. Multiple scanners could also be used to acquire multiple scans at once. Finally, other shape capture equipment that can gather data at a faster rate could prove useful. Multiple synchronize stereo camera systems could acquire data at rates of thirty frames per second. This means the subject would simply have to exercise his joints through their range of motion without having to maintain a fixed pose for a long duration.

3.3 Characteristics of the Acquired Data

Our laser scanner returns a two-dimensional array of values, as discussed in §2.1.2. Computing Cartesian coordinates for the points involves the use of a calibrated model, which corrects for errors and distortions introduced by the collection optics and the scanning mechanism [5]. There also exist several systemic sources of error that cannot be easily corrected for. These errors may differ for each point, although they can be quantified to a limited extent by assigning a weight representative of uncertainty to each data point.

A first source of uncertainty arises when the projected light beam strikes an object at a grazing angle. In such cases, the spot detector sees a less intense and warped version of the projected spot. This makes it difficult to find the center of the pattern, and therefore adds uncertainty to the position of the range points. The degree of uncertainty increases with the difference between the viewing direction of the scanner and the surface normal. A secondary source of inaccuracy occurs at edges, where only a portion of the beam hits the object. This results in a false estimation of the range since the peak detection algorithm for the spot detector assumes that the entire spot landed on the object. This error can be taken into account by assigning a high degree of uncertainty to points found

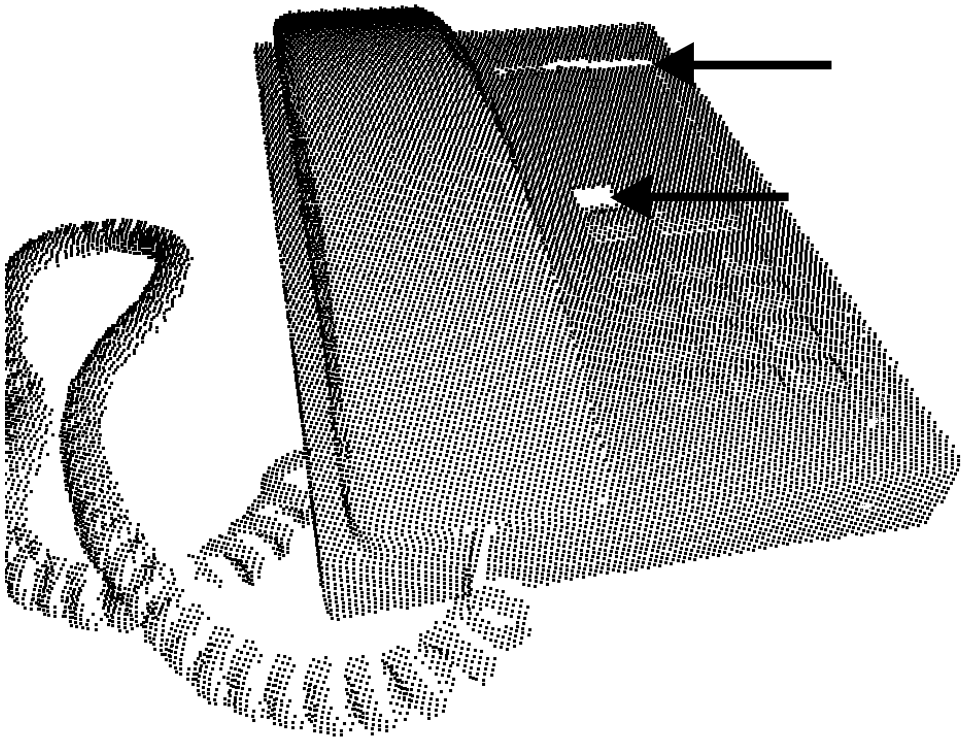

Figure 24: Range data of a telephone set.

to be adjacent to object edges. Both sources of errors are eventually taken into consideration by the surface reconstruction algorithm, as described in §2.2.

Figure 24 shows the range data from a single scan of a telephone set. The intensity data is not shown. A photograph of the telephone set is shown in Figure 25. The range data in Figure 24 was sampled at a resolution of 256 x 256. The two regions with no data, as indicated by arrows, are parts of the phone that do not reflect the projected beam back to the spot detector. The data is rendered from a viewpoint similar to the position of the scanner when the data was

Figure 25: Photograph of the telephone set.

Figure 26: The telephone set after triangulation.

acquired. Note that the telephone model has no bottom half; scans from multiple directions are required to build a complete model.

Figure 26 shows a rendering of the telephone after performing the meshing operation described in §2.2. The intensity data has been mapped onto the data points. Figure 27 shows the "edge effect" due to having only part of the projected beam hit the object. The warping of the image due to the beam hitting an object at a grazing angle only becomes apparent when aligning two range images taken from different viewpoints.

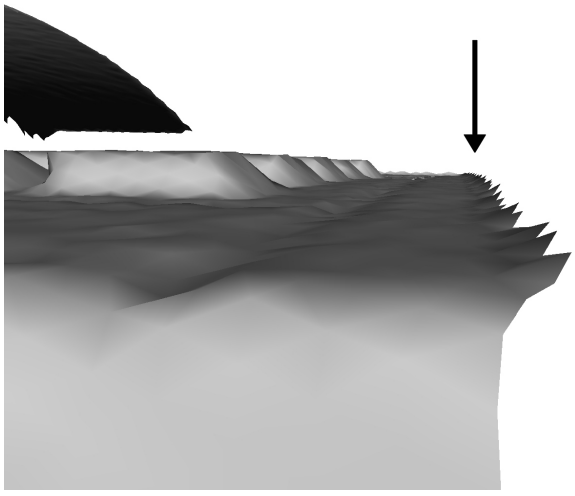

Figure 27: Edge effect of telephone range image: edges curl toward the scanner when only part of the beam strikes the object. The viewpoint is that shown by the arrow in Figure 26.

We shall make use of a hierarchy of meshes to speed up the registration process. The time complexity of our registration method is proportional to the number of points in the meshes. Thus, we first align low-detail meshes together before proceeding to the next level of detail to further refine the alignment, and repeat this process until the highest resolution meshes are aligned. Building a hierarchy of meshes with decreasing level of detail is a simple process. We resample every other point to compute a mesh of lower resolution in the hierarchy. Figure 28 shows the telephone with four levels of detail. Note that even at the coarsest level we have enough detail to align two range images of the telephone together. The isolated triangles shown in the lower-resolution meshes do not cause registration problems, because their vertices are on the mesh boundary and therefore are assigned a low weight, as discussed on page 38.

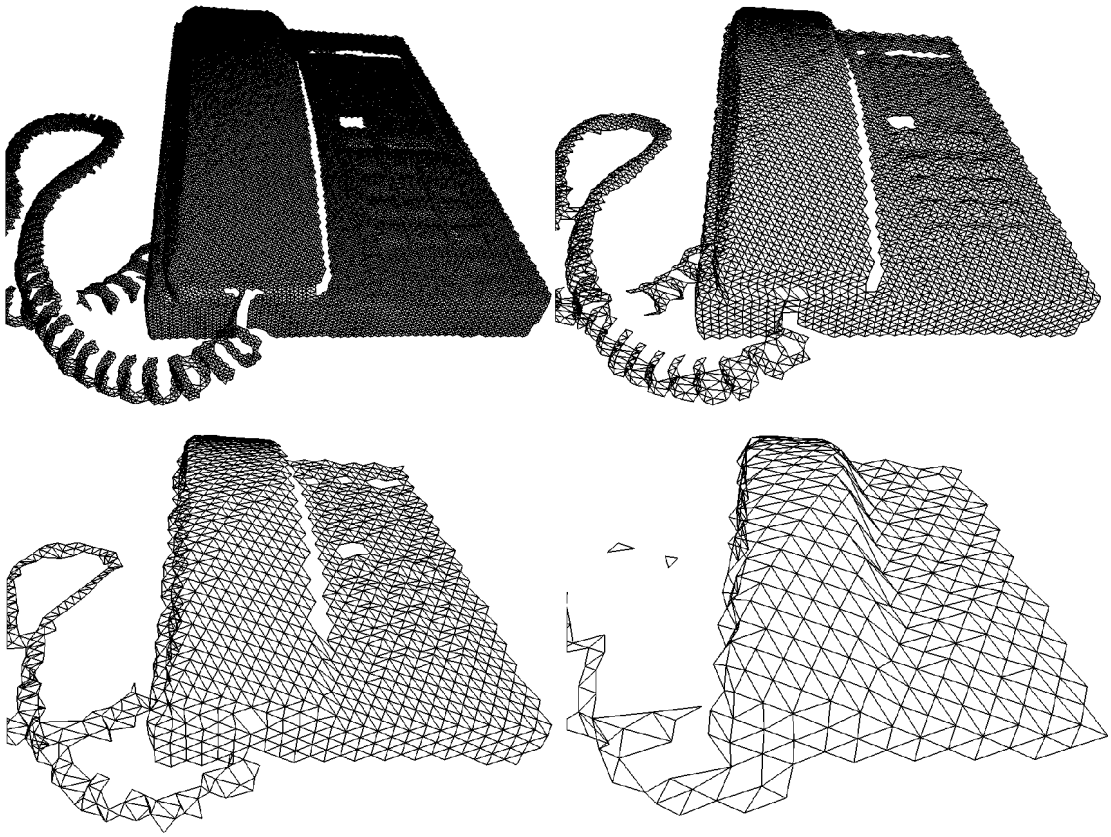

Figure 28: Hierarchy of meshes created by subsampling the original set of vertices.

4. RANGE IMAGE REGISTRATION

After acquiring multiple range images of an object using the setup described in the previous chapter, we need to bring corresponding portions of these range images into alignment with one another. In our case we have no information regarding the exact location and orientation of the scanner, and thus a data-driven registration step is required. It is also worthwhile noting that even a robotic mount for the scanner would still necessitate a registration step in order to ensure a very accurate registration.

In this chapter we describe the modified ICP algorithm that we employ for range image registration. We then elaborate on the need for an initial registration estimate and explain how we efficiently perform the closest-point searches which are part of the ICP algorithm. We conclude with the experimental results achieved after applying the ICP algorithm to the shape data acquired from the arm of our subject.

4.1 Iterated Closest-Point Algorithm

We use the algorithm presented by Turk and Levoy [59], which is a modified iterated closest-point (ICP) algorithm. ICP was first proposed by Besl and McKay [9] as described in §2.3.3. This original ICP algorithm cannot be used to register range images that are only partially overlapping because it requires establishing correspondences for all points on both surfaces. In our case, however, we are using multiple range images which provide complementary coverage of the scanned object. The modified ICP algorithm is shown below; a description of each step follows.

```
For increasing levels of detail
     While not converged
          Find the closest point on mesh A for each vertex of mesh B
          Discard poor correspondences
          Eliminate pairs in which either points is on a mesh boundary
          Compute a best-fit rigid transformation
```

To illustrate how the algorithm works, we shall consider the example shown in Figure 10 and reproduced in Figure 29. The two meshes are shown after aligning their respective center of masses. We shall call A the registered data set (R) since we shall consider it to be a fixed data set. We call B the unregistered data set (UR) since it will be aligned to A. Either of these data sets could be the result of integrating previous scans. The algorithm begins by choosing an appropriate resolution for each data set. This consists of using a simplified UR data set as will be elaborated in §4.3. The initial relative positions and orientations of the two data sets are assumed to be arbitrary. Therefore, before using ICP to align these data sets, we need to apply an initial (coarse) registration transform to one of the data sets. This is a user-controlled step. Figure 30 shows a hypothetical configuration after this initial manual registration step.

The data sets are now ready for the application of ICP.

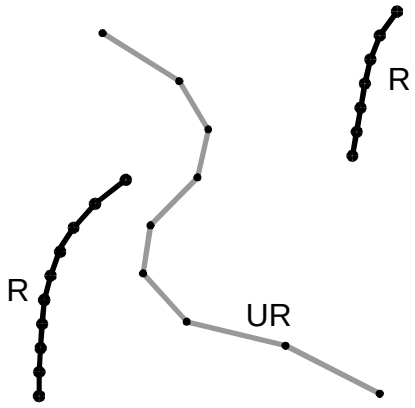

Figure 29: The two data sets from Figure 10 used in the example registration problem.

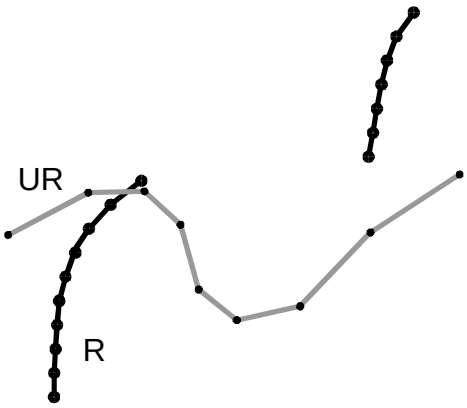

Figure 30: The two data sets after applying a rotation of sixty degrees to data set B.

4.1.1 Find the nearest position on mesh A to each vertex of mesh B

The first step in ICP is to create pairs of corresponding points. Each such correspondence consists of one original data point from the UR data set and its corresponding closest point on the surface of the R data set. Note that the closest point can be anywhere on the surface of R and does not necessarily have to be a vertex. This is illustrated in Figure 31. An efficient method for finding the closest surface point to a given data point will be elaborated in §4.3.

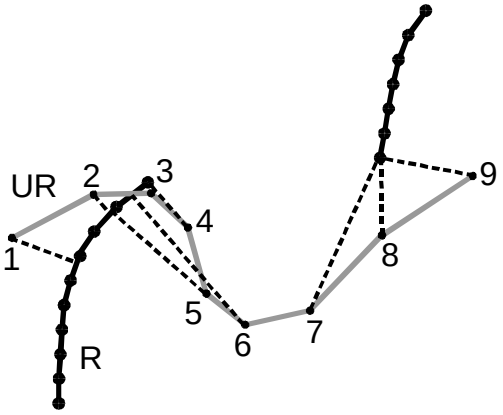

Figure 31: Example registration problem after establishing point-to-surface correspondence.

4.1.2 Discard poor correspondences

A poor correspondence is defined as a pair for which the distance exceeds a fixed threshold. This is one of the modifications that Turk and Levoy [59] propose. In our example, we get rid of three pairs of points, points 5, 6 and 7 as shown in Figure 32. In this particular example, these poor correspondences are in fact points that were occluded in the scan comprising the R data set. The distance threshold is chosen using a value proportional to the spacing between the data points.

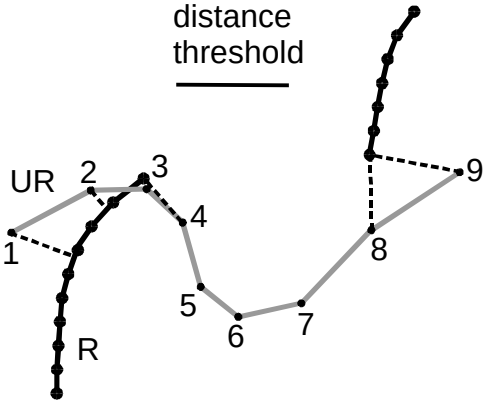

Figure 32: Example registration problem after discarding poor correspondences.

4.1.3 Eliminate pairs in which either points is on a mesh boundary

This is another modification suggested by Turk and Levoy [59]. In the case of our simple example it is not necessarily an improvement for points 1 and 3 because these had correspondences which were approximately correct. It also appears that it does not work well for points 8 and 9. However, the match between points 8 and 9 and their closest point can be seen to be poor with respect to the final (correct) registration.

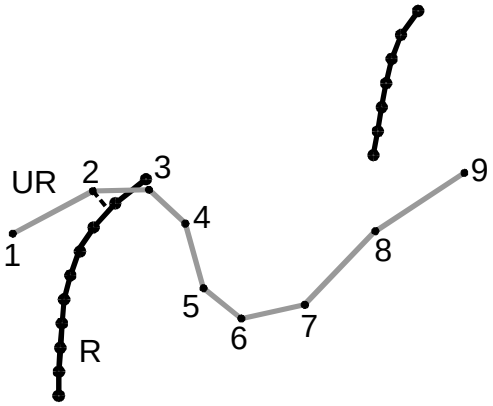

Figure 33: Example registration problem after eliminating boundary correspondences.

Finding the boundary of a mesh is a simple procedure that involves processing all edges and counting the number of triangles to which each edge belongs. If an edge is used only once, then it is a boundary edge.

4.1.4 Compute a best-fit rigid transformation

The best-fit criterion that we use is the minimization of the least-squares distance between pairs of points. We use Horn's method, as discussed in §2.3, to compute the rigid transform that will best align the remaining correspondences. In our example, this is not difficult since there is only one point left. We simply translate the UR data set so that vertex 2 coincides with its associated closest point on the R data set.

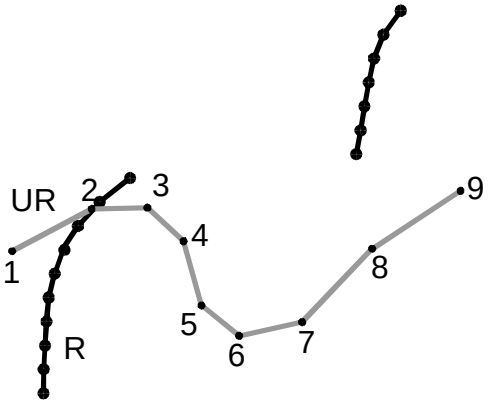

Figure 34: Example registration problem after applying a best-fit transformation.

4.1.5 Iterate until convergence

The above four steps are repeated until a convergence criterion is met. In our example, the process converges after one step because no new valid correspondences are obtained.

4.1.6 Repeat ICP with a more detailed mesh

In this step, we further refine the registration transform by considering a more detailed mesh representation. Since the computed transform depends on the data, increasing the level of detail will increase the accuracy of the registration. In Figure 35 we show the situation after increasing the level of detail of the UR data set. When steps 1 to 3 are repeated, points 2, 3, and 4 will contribute to the registration transform (all the other points will be rejected by steps 2 and 3). These will be aligned in the next computation of the best rigid transform, as shown in Figure 37. In the next iteration, point 15 will also begin to participate. Eventually ICP will converge to the

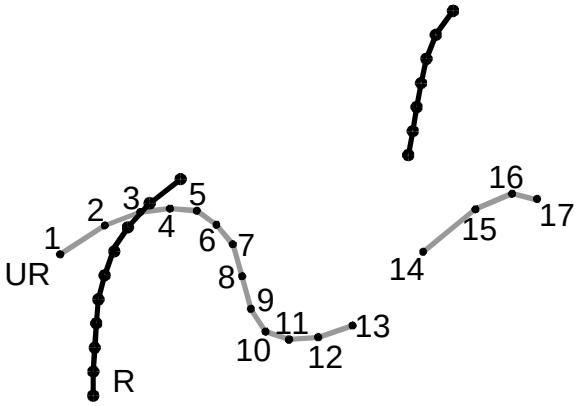

Figure 35: Example registration problem after increasing the level of detail of the UR data set.

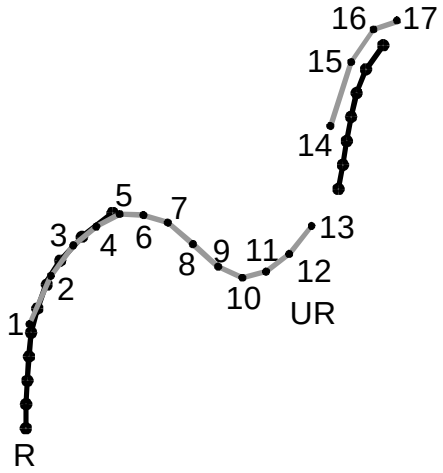

Figure 37: Example registration problem after the second ICP iteration.

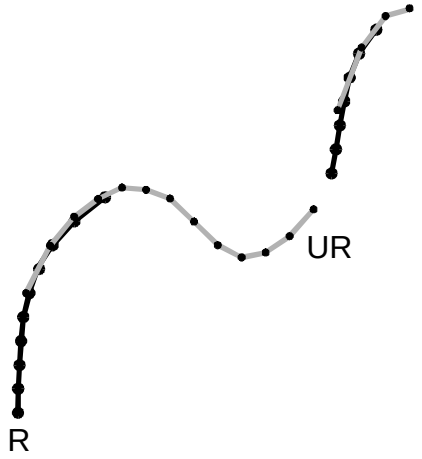

Figure 36: Alignment achieved in example registration problem after ICP converged.

alignment shown in Figure 36. As mentioned in §2.2, vertices from the overlapping portions of the data sets do not always coincide together.

4.2 Coarse Alignment

Figure 38 illustrates an initial alignment where the ICP algorithm as described thus far converges to an erroneous solution. Besl and McKay [9] proved the ICP algorithm converges monotonically to a local minimum, but not necessarily the right one. It is difficult to precisely characterize the partitioning of the registration state space into local minimum wells because this partitioning is a function of the particular data sets. Besl and McKay propose using a set of initial rotations which uniformly samples the configuration space of initial orientations. In our system, we prefer to involve the user in aligning the two data sets using an interactive interface. This typically only requires several seconds.

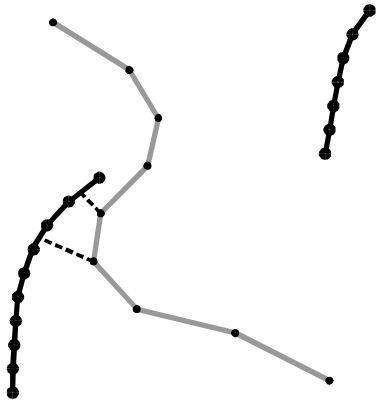

Figure 38: An example of ICP converging to wrong local minimum.

4.3 Efficiently Computing the Closest Surface Point

ICP is efficient in practice because it requires only one execution of the "closest point" routine per point per iteration, and requires few iterations to converge. Any optimization method that does not use explicit vector gradient estimates, such as simulated annealing, can require thousands of iterations [9]. Optimization methods that do use explicit gradient computations, such as the conjugate gradient method, will require at least seven closest point evaluations for each gradient evaluation computed using finite differences. Such a method would be required to converge in less than ten iterations to be competitive with ICP. They usually require well over one hundred iterations [9].

Finding the nearest position on a mesh to each vertex of the other mesh is where the ICP algorithm spends most of its time. This is also the part of the algorithm which has at least O(n) complexity, where n is the number of points being considered in the unregistered data set. It therefore needs to be made as efficient as possible. It should preferably be a constant-time operation, independent of the number of points in the registered data sets. Turk and Levoy divided this search in two steps. First, the closest mesh vertex is found. Second, all edges and triangles connected to the vertex are searched for a point that is possibly closer.

Turk and Levoy's method of finding the closest vertex imposes a spatially uniform 3D grid where each cell contains a list of all data points found within. When searching for the closest vertex to a point p, the cell containing p as well as its neighbors are searched for the closest vertex. Because the length of segments comprising the mesh triangles created during the mesh building step is restricted, it can be guaranteed that no triangles that lie within a certain distance threshold will be missed if the size of the grid cells is appropriately chosen.

We choose to implement a search algorithm using a more adaptive data structure. Turk and Levoy's approach may be valid when registering individual range images, but after integrating some range images together and registering data to these new data sets, their search criterion may no longer hold. We implement a *k-d tree* search algorithm [26]. The k-d tree is a strictly binary tree in which each node represents a group of data points and a partitioning of those points. The root of the tree represents the entire data set. The leaf nodes represent mutually exclusive small subsets of the points, which collectively form a partition of the data set. Any one of the (x, y, z) coordinates can serve as a discriminator, or key, to partition the data for a node's successors. It is shown in [26] that the best discriminator is the one with the largest spread of values. The median value in the selected discriminator is chosen for partitioning the data. The search for the closest data point to a given point p happens as follows. Starting at the root, the algorithm proceeds down the tree to find the leaf node that has geometric boundaries closest to p. This leaf node does not necessarily contain p, as p can be located anywhere in $\Re^3$ while the k-d tree spans a finite region of $\Re^3$. Since a leaf node may contain more than one

point, all points in the leaf node are examined to find the point *c* that is closest to *p*. Let *d* be the distance separating *p* from *c*. The algorithm tests whether the sphere of radius *d* fits within the cell of the current leaf node. If so, no point from any of the surrounding tree nodes can be closer than *c*, and the algorithm returns *c*. Otherwise, the algorithm reascends the k-d tree and searches the neighboring branches whose geometric boundaries overlap the sphere with radius *d*. Eventually the sphere of radius *d* will either fit inside the geometric boundaries of a leaf node, or it will fit inside the geometric boundaries of a non-leaf node for which both successor nodes were already searched.

The time complexity of searching for the closest point in a k-d tree is proportional to *log N* [26], where *N* is the number of points in the search tree. In the ICP process, we need to search the tree containing the vertices of the registered (R) data set for each vertex of the unregistered (UR) data set. If R contains *N* vertices and UR contains *M* vertices, each ICP step requires *M log N* time to complete.

As discussed in §3.3, we use a hierarchy of meshes to speed up the ICP registration. We simplify the UR since it contributes the most to the time spent searching. We choose not to exploit the 2D grid-like structure of the range images for the same reasons that we choose not to use a spatial subdivision scheme. Once two range images are integrated together, this property is lost.

In order to compare the k-d tree to the uniform grid approach, we performed several experiments using a range image available from Stanford[7]. To create our test data we found the bounding box of the range data and uniformly sampled that box in all three dimensions with 132 561 points (51 points in each of the three dimensions). We performed a closest-point query from each of these points. The hash table search employed by Turk and Levoy [59] took 45 seconds to complete while our k-d tree search took 55 seconds. However, the hash table could not find 8985 points. This occurs when the cells neighboring *p* do not contain any data.

[7] The image that was used for running our test can be acquired from ftp:// www-graphics.stanford.edu/pub/zippack/data/phone/phone.1.tar.Z

The k-d tree could possibly be updated to search for the nearest point on a triangle instead of simply searching the space for a closest vertex, by relaxing the requirement that each leaf node must contain mutually exclusive data. We did not pursue this possibility, however.

4.4 Registration Results

The previously described registration algorithm is used to register the multiple scans taken of our subject's arm. Figure 39 shows two range images of the same arm scanned from two different viewpoints. In Figure 40 we see the meshes after registration. The mesh that is on the right in Figure 39 is drawn as a solid mesh. In this particular case there was no need for the user to supply an initial registration transform; ICP could simply be applied from the initial configuration shown in Figure 39. Because the viewpoint did not change dramatically from one scan to another we found that there was often no need to supply an initial registration, depending on the data set. The mesh on the right has 4961 vertices, and the mesh on the left has 5707 vertices. The registration time was 45 seconds[8]. There were 2775 pairs of corresponding points established, because the

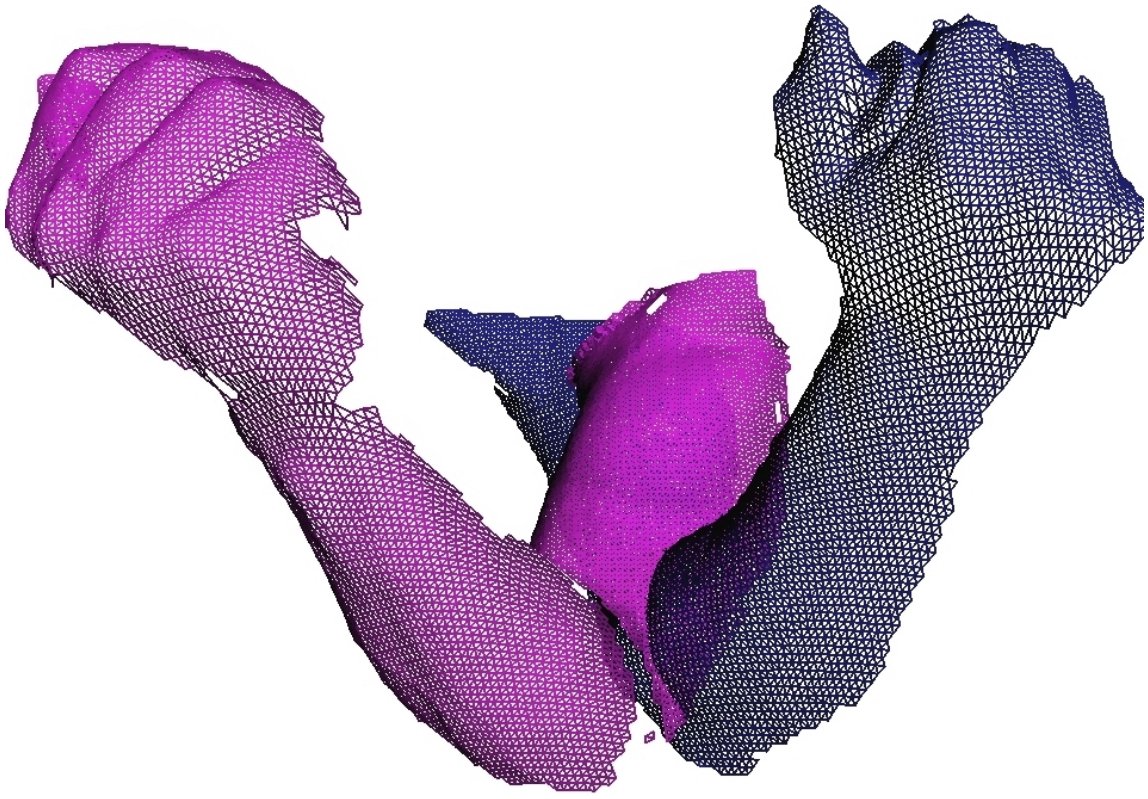

Figure 39: Two range images of an arm, scanned from different viewpoints.

[8] The registration algorithm was executed on an SGI Indigo2 (R4400 processor), with 192 MB of RAM.

range images contain complementary data. The registration error, computed as the root-mean square distance between the point pairs, was 0.4% of model size. The registration error may be due to movement in the subject's arm, as well as acquisition errors. This is corrected for in the subsequent integration step, as will be seen in chapter 5.

To process a group comprising more than two range images, we align and integrate two images together, then treat this integrated image as one range image. Another image is aligned and integrated and so on until all the images in the group are integrated. Such an example is shown when we discuss our animation results in chapter 6.

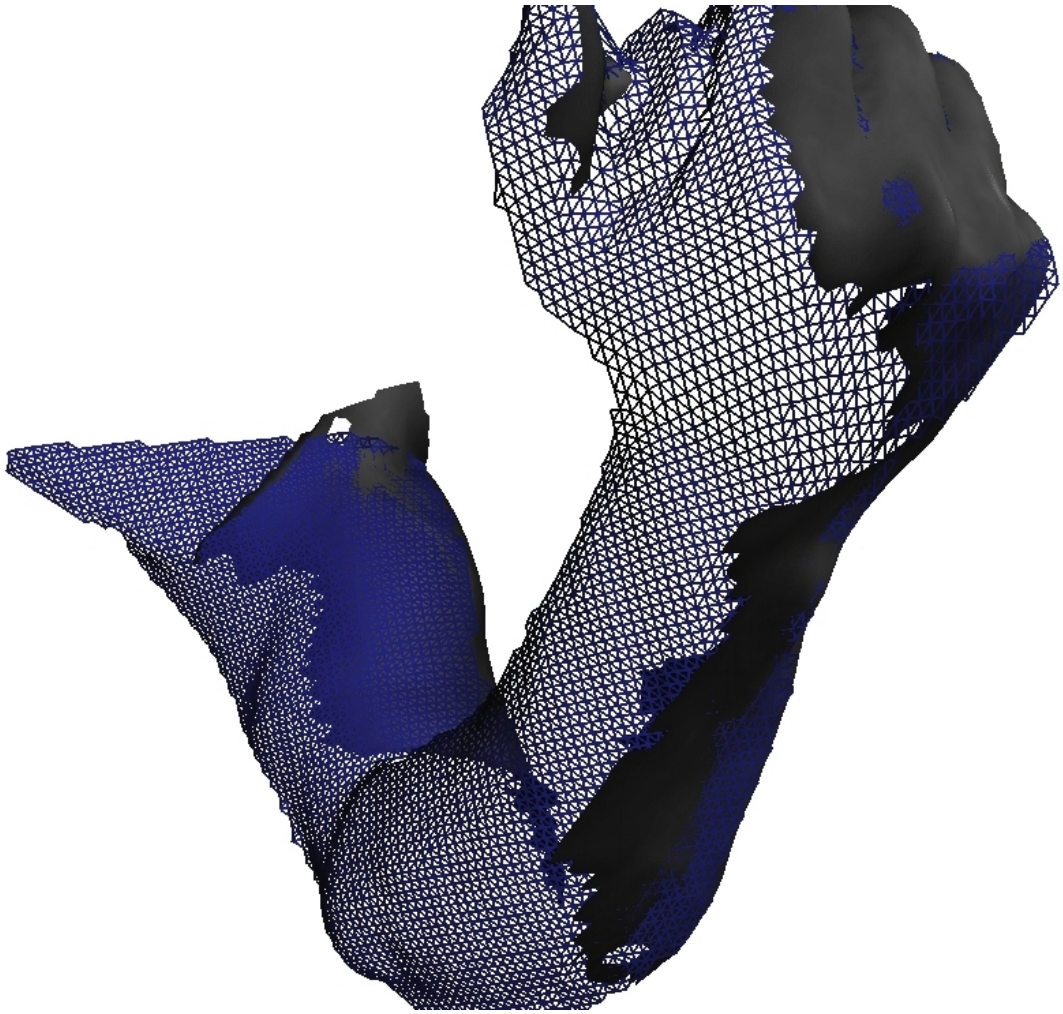

Figure 40: Two meshes of an arm aligned using ICP. One mesh is drawn as a surface.

5. RANGE IMAGE INTEGRATION

The registration process itself does not produce a usable surface model. Rather, it produces a patchwork of overlapping and intersecting surfaces. The integration process is responsible for cleaning up this representation and producing a single integrated surface model. We use a modified version of Turk and Levoy's zippering algorithm [59] to perform this integration. The integration process involves three steps: correcting for any remaining registration error, eliminating redundant surfaces, and bridging the gap between mesh boundaries.

5.1 Position averaging

A variety of sources contribute to local surface alignment errors which remain even

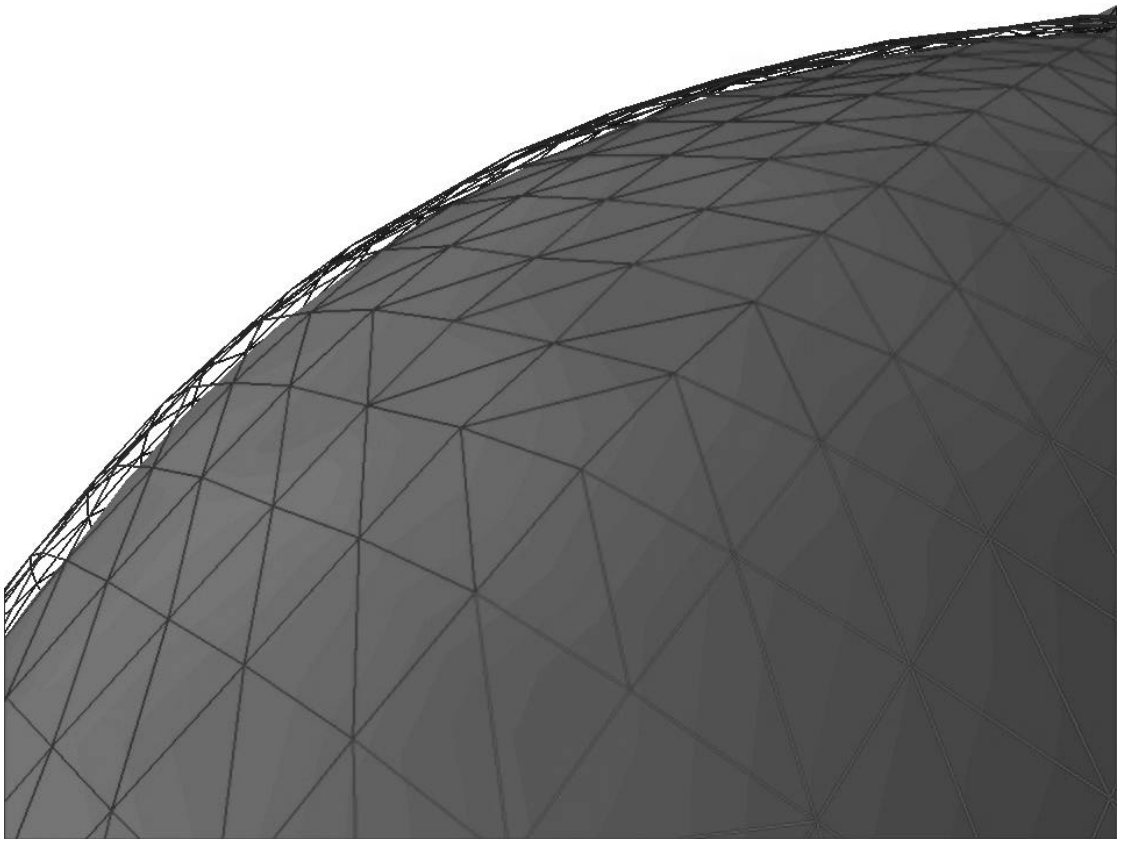

Figure 41: A detail view of the bicep showing a remaining gap between surfaces after ICP has converged.

after the registration algorithm has converged. These include data acquisition errors, as described in §3.3, and subject sway during the acquisition. These errors can cause some warping between data taken from different viewpoints. Because of this, even when the registration algorithm has converged, there is likely to be a remaining gap between overlapping portions of range images, as illustrated in see Figure 41. To avoid seeing a "scar" where two range images meet, we need to apply local corrections to the surfaces to obtain better alignment.

To avoid unnecessary blurring of surface detail, we move vertices along the local approximation of their normal. The warping algorithm can be described as follows:

```
Repeat until convergence:
     for each mesh
          for each vertex v_i
               find its closest point p_i on the other surface
               if p_i is further than a certain distance, proceed
                    to the next v_i;
               project the vector v_i p_i  on the local normal
                    approximation of v_i;
               translate v_i  by a fraction k of the projection
                    found above;
```

In practice we choose k to be 0.5. If k is too small it takes unnecessarily long for the warping procedure to converge, and if it is too large, the procedure may fail to converge.

Note that each vertex is treated independently. It would also be interesting to experiment with a deformable model approach [58]. In this approach, vertices affect neighboring vertices. This would help avoid the situation where a local surface aberration can be created by one vertex being too far to be considered for displacement by the other surface, while all its neighbors do get displaced.

Soucy and Laurendeau [52] perform this warping step before merging the meshes together. Turk and Levoy perform warping after merging the meshes together using the original vertex positions. We implemented and tested both methods and obtained better results when performing the warping before the merging. It makes finding the overlap between two meshes and filling the gap between two meshes more reliable. In fact, the

warping could be effected both before and after merging, but in our experiments we found that repeating it after the integration made no significant difference.

5.2 Removal of Redundant Surface Areas

Following the warping, the redundant surfaces need to be removed. Surface areas are considered redundant if they overlap, as shown in Figure 1. Although overlap is well defined in 2D, it is ill-defined in 3D. We therefore use the following scheme.

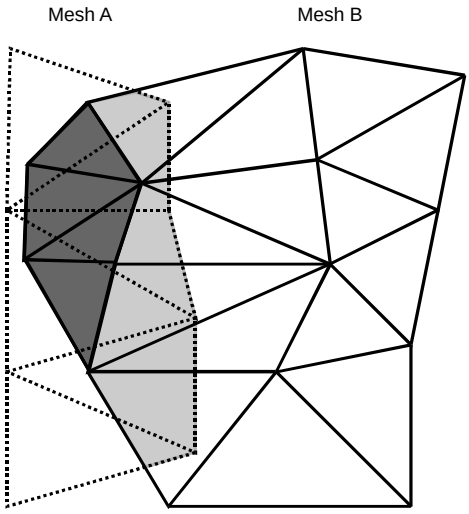

Figure 42: Example of overlapping surfaces in 2D.

We can find out if parts of mesh *B* are redundant with respect to mesh *A* by examining each triangle in mesh *B* to see if it is redundant. A triangle *T* in mesh *B* can be considered *completely redundant* with respect to mesh *A* if the closest surface point in mesh *A* to each of *T*'s vertices lies strictly in the interior of mesh *A*. The dark shaded triangles in Figure 42 are completely redundant. A triangle *T* in mesh *B* can be considered *partially redundant* with respect to mesh *A* if *T* has at least one vertex for which the closest point in mesh *A* lies strictly in the interior of mesh *A* and at least one vertex for which the closest point in mesh *A* lies on the boundary of mesh *A*. The lightly shaded areas in Figure 42 represent partially redundant areas of mesh *B*'s triangles. A distance threshold *d* is also used to compare to the distance separating a triangle vertex to its closest point in the other surface. This is to avoid classifying a triangle as redundant if it is too far from the other surface. *d* is chosen as a function of the maximum triangle edge length.

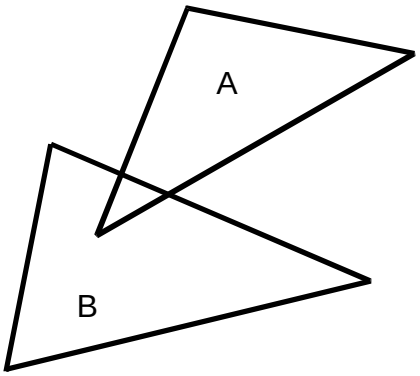

Figure 43: Partially redundant triangles. Triangle *A* would be detected properly, but not triangle *B*.

The above scheme would classify triangle *A* in Figure 43 as being (partially) redundant with respect to *B*, but triangle *B* is not redundant with respect to *A*. This is not a concern in our approach since we alternate between finding the redundant triangles on one mesh and the other, as will be described shortly.

Once the redundancies have been identified, they need to be acted upon. At least two strategies are possible. Turk and Levoy approach the merging problem by only getting rid of completely overlapping triangles, and then clipping together all remaining partly overlapping triangles. Our approach eliminates all overlapping triangles, whether they are partly or completely overlapping, and then merges the gap between adjacent boundaries. In our experience, this makes the merging process simpler to implement, as it requires fewer heuristics.

5.3 Merging meshes

To illustrate the merging process, we employ the 2D example shown in Figure 44. Figure 44 (a) shows the two overlapping meshes of Figure 42 after the removal of the completely overlapping triangles. Figure 44 (b, d) shows the results of the method of Turk and Levoy. All the partially-redundant triangles of mesh *A* are truncated, and split to

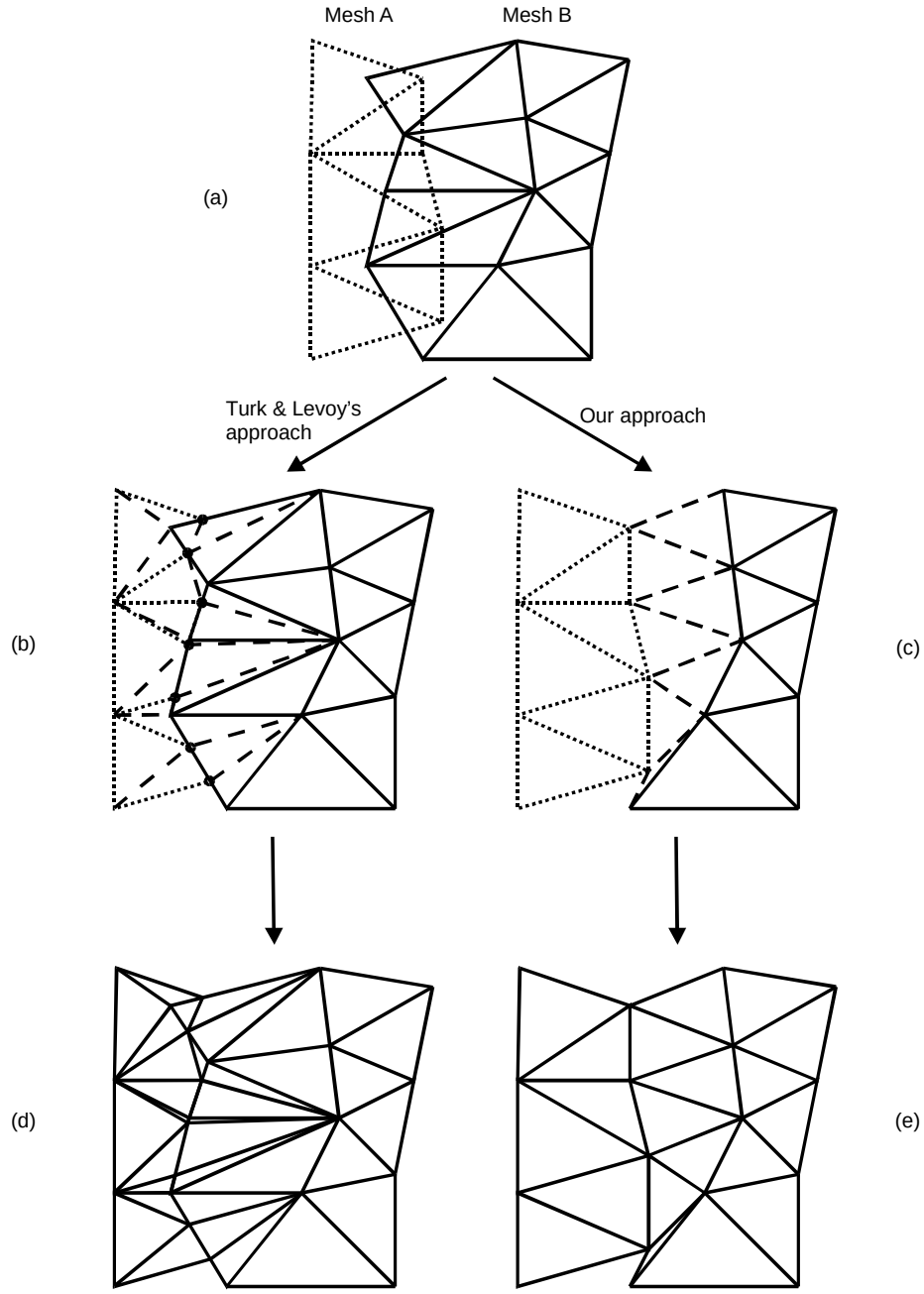

Figure 44: Comparison of two different approaches to merging meshes.

remove all T-junctions. The next step removes the small triangles generated by the above procedure.

Our approach to integration can be described as follows. Figure 44 (e) shows the resulting mesh. As discussed in §5.2, we begin by removing redundant triangles from either mesh. In the example of Figure 44, we remove them from mesh B. This is followed by a constrained triangulation [8], which connects the meshes by stepping along the mesh boundaries. Our approach, compared to that of Turk and Levoy, tends to produce a smaller number of triangles and fewer *small* triangles, and thus avoids any small-triangle-elimination step. This reduces the number of heuristics employed in the merging step.

5.4 Integration Results

Figure 45 shows the meshes from Figure 40 after they have been merged. For this data set, the warping step took 7 seconds to execute, the redundancy elimination step took 8 seconds, and the merging step took 5 seconds[9]. The merged data set has 8238 vertices and 15 920 triangles.

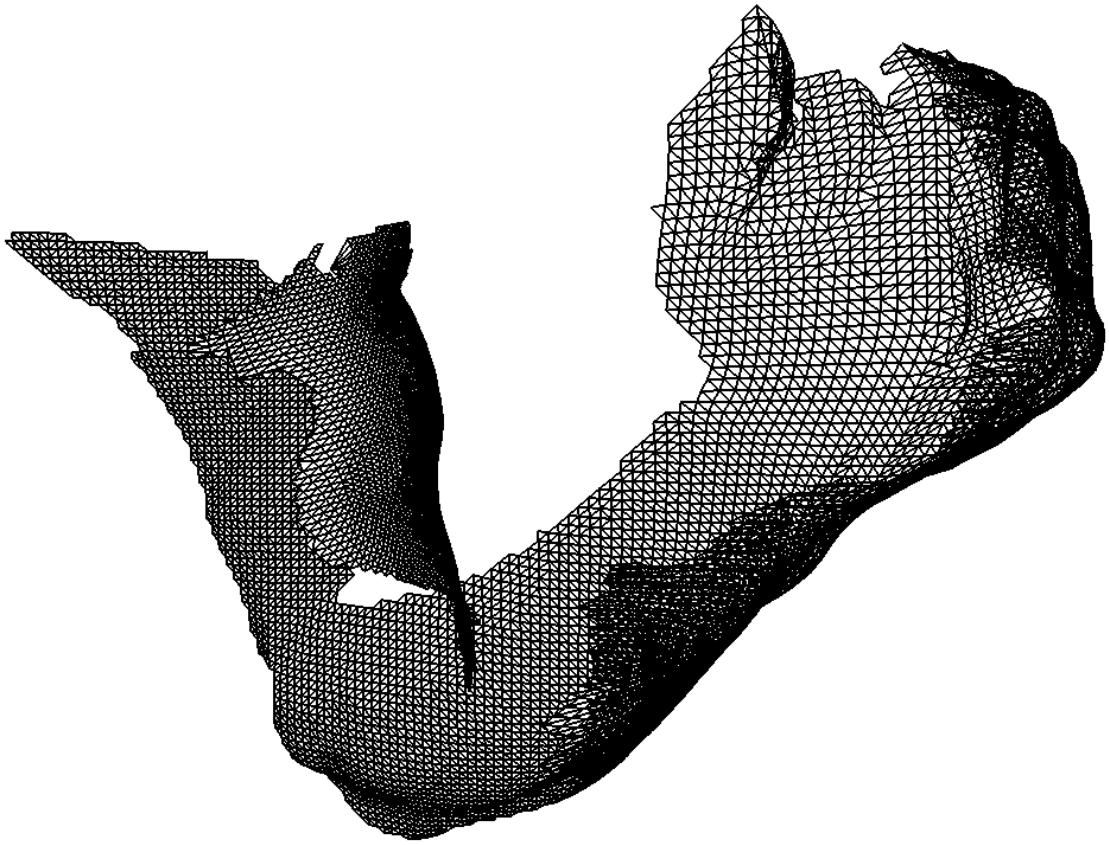

Figure 45: A human arm model created from merging two meshes.

[9] Timings are from execution on an SGI Indigo2 (R4400 processor), with 192 MB of RAM.

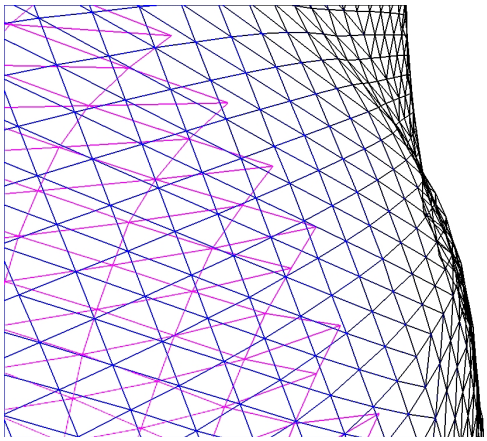

Figure 46: A detailed view of the mesh before merging.

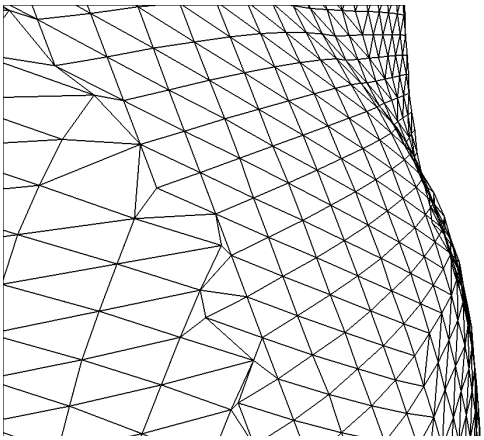

Figure 47: A Detailed view of the mesh after merging.

Figure 46 shows a particular area of the model before merging. Figure 47 shows the resulting integrated mesh. Merging the meshes produces a visual boundary where the two meshes meet, due to the different densities and orientations of the scan data. The boundary is much less visible when rendering the mesh as a solid, as seen in Figure 48.

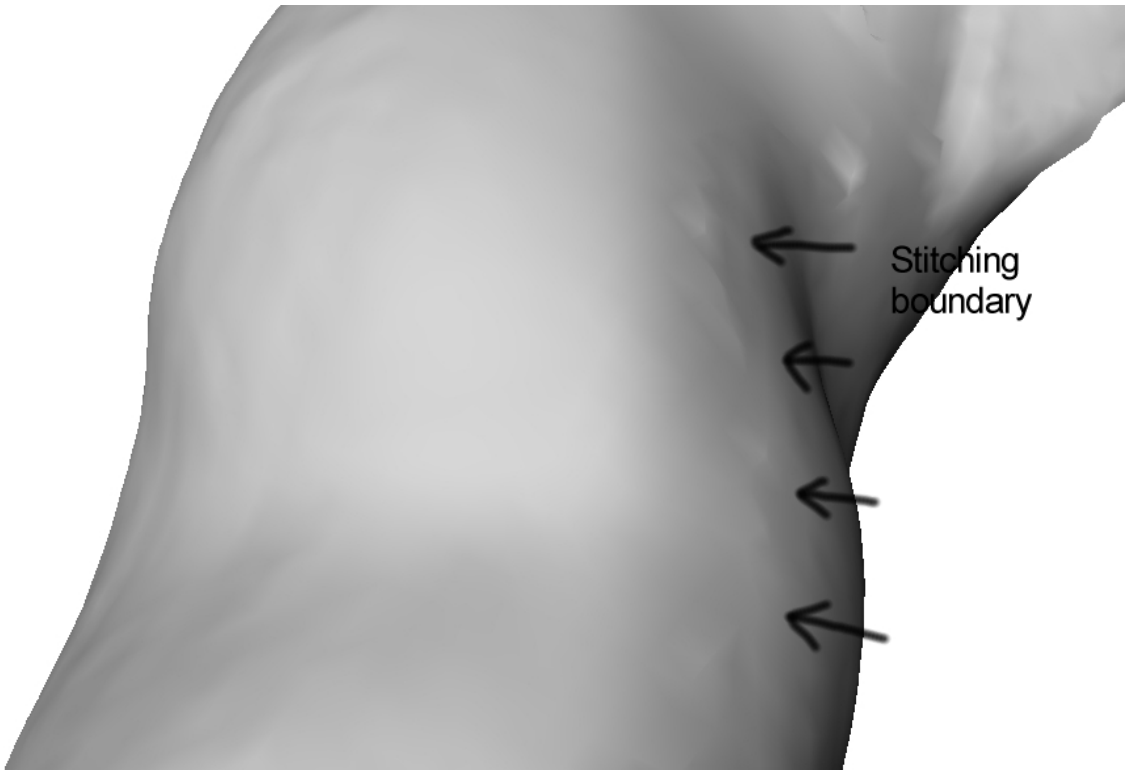

Figure 48: The stitching boundary that remains visible after merging.

After merging, the result can be used as a single data set or surface model to which other range images can be registered and integrated.

In this chapter we have presented a method for integrating aligned data sets together. This is necessary to produce a single surface representation of a model. In the context of our animation system, the registration and integration steps are repeated until each keyframe model is completed. These models are then ready to be augmented with an underlying skeleton, as will be presented in the next chapter.

6. CONSTRUCTING PARAMETERIZED DEFORMATIONS

In this chapter we describe how to combine and utilize the information that we have from each keyframe. We begin by describing the choice of parameters on which a deformation model depends. This is particularly important because the number of parameters ultimately dictates the minimum number of keyframes required. Once the parameterization has been chosen, we discuss the interpolation method employed between keyframes. The skeleton that is used as part of the interpolation process is described in detail. We conclude by showing the results of our implementation, including frames from the animation of an arm.

6.1 Parameter Selection

The shape of the arm is dependent primarily on the position of the shoulder, elbow, and wrist, as well as the activation intensity (relaxed or tensed) of the muscle groups associated with the arm. A brute force approach to modeling arm deformations would capture the shape of the arm for every possible setting of this large set of parameters. A simpler model can be constructed, however, if we can assume we can localize the effect of any of these parameters. For example, flexion of the wrist will not greatly affect the shape of the upper arm.

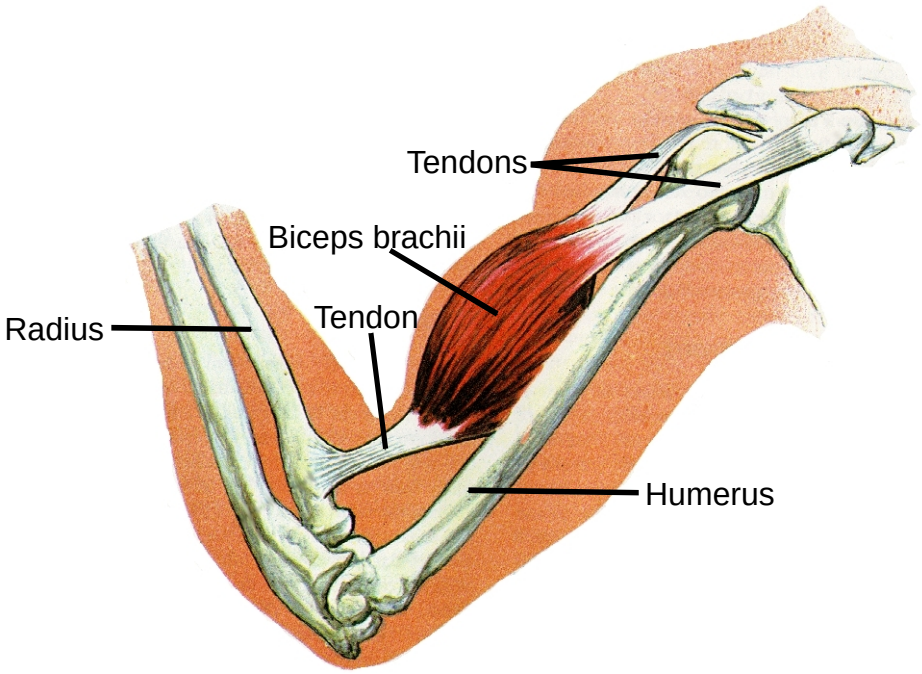

Figure 49: Anatomy of the biceps brachii.

Figure reproduced from p. 67 of [42]

In order to better understand the relationship between the shape of the arm and these parameters, we shall focus on the influence of the biceps. Figure 49 shows the anatomy of the biceps muscle *(biceps brachii)*. We can see that it crosses both the elbow joint and the shoulder joint; it will therefore exert some control on both of these joints. Its main functions are the flexion of the forearm, as shown in Figure 50, and supination of the hand, as shown in Figure 51. It is also a weak flexor of the upper arm at the shoulder joint [53].

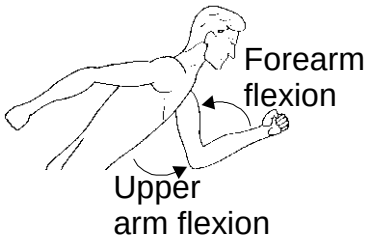

Figure 50: Upper arm and forearm flexion.

Figure reproduced from p. 21 of [53]

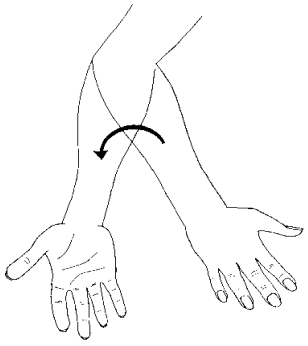

Figure 51: Supination of the hand.

Figure reproduced from p. 23 of [53]

Figure 52 shows the flexion of the elbow joint at the two extremes of the range of motion. In both poses the biceps muscle is relaxed. Using the notation introduced in §1.2, we can define the deformation of the skin induced by elbow flexion as:

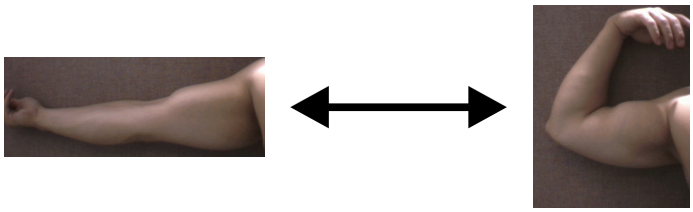

Figure 52: Flexion of the elbow, with biceps relaxed.

$$S_K = b(P_{upperarm})$$

where $P_{upperarm}$ represents the degree of elbow flexion.

The intensity of the biceps muscle contraction also influences the skin deformation. This is related to how much force the biceps muscle is exercising. As seen in Figure 53, there is a significant difference of shape as a function of biceps contraction when the forearm is flexed. The skin deformation function now depends on two parameters:

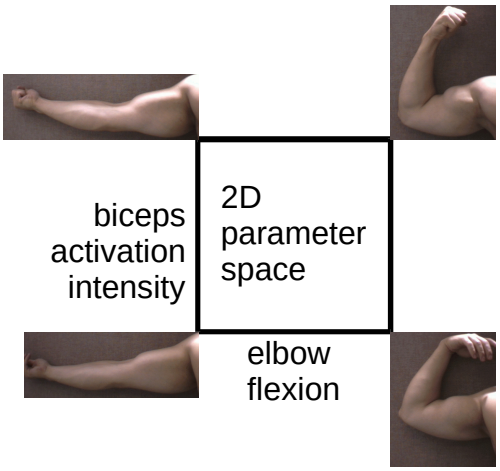

Figure 53: A comparison of elbow flexion with relaxed and tensed biceps.

$$S_K = b(P_{upperarm}, F_{biceps})$$

As discussed previously, however, the biceps muscle also supinates the hand and flexes the upper arm. Adding these parameters to our skin deformation function gives us the more complex function:

$$S_K = b(P_{shoulder}, P_{upperarm}, P_{forearm}, F_{biceps})$$

These are most of the parameters that would be required to properly model the deformation of the skin induced by the biceps.

Must all of these parameters be used in the model? This depends on the required accuracy of the model and the range of motion that needs to be modeled. For example, if

the character animation involves no shoulder flexion, then the dependence on this parameter can be eliminated. Also, some dependencies are less important than others. Shoulder flexion, for example, induces less skin deformation than forearm flexion. Dependency on this parameter can therefore be eliminated without much impact on the quality of the deformation model. On the other hand, if we wish to create a realistic character that can strike any pose, all parameters would have to be used.

Two additional questions need answering before we can make use of the above model. How many samples (keyframes) are needed for each of the parameters P and F? And, what morphing function b should be used to interpolate between the keyframes? The answers to these two questions are somewhat interdependent and depend in part on finding an interpolant which is well suited to the data in question. The simplest choice involves using linear interpolation with many keyframes. Our approach uses a skeleton to induce a local coordinate frame for mesh points as well as a morphing (interpolation) method with is applied in this local coordinate frame.

6.2 The Deformation Model

In this section we review the 3D morphing problem. We then describe our skin model, and discuss our interpolant.

6.2.1 3D Morphing

The traditional 3D morphing problem can be stated as follows. Given two objects S and T, henceforth called the *source* and *target* objects respectively, we must produce a sequence of intermediate objects, the *morphs*, meeting the following two conditions:

1) Realism: the morphs should be realistic objects, providing that the source and target are themselves realistic. Essential features of the source and target should be retained.
2) Smoothness: the morphs must depict a smooth transition from the source to the target.

These conditions are met by warping the source object into the target object. This warping process is feature-based; corresponding features in the source and target objects are identified and an interpolation scheme creates the morphs. The major challenge in designing a 3D-morphing system is automating the feature recognition and matching process [40].

In our case, we can make use of the underlying skeleton to simplify the problem. The skeleton helps avoid the typical shortening of rigid links that can occur when they are rotated, as shown in Figure 54. It also alleviates the feature matching problem.

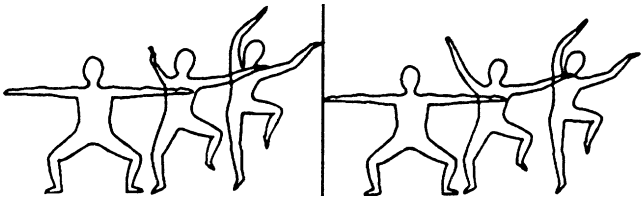

Figure 54: Sequence showing the shortening of the arm due to the linear interpolation used in the vertex path calculation (left); the sequence on the right shows the correct interpolation.

Figure reproduced from [51]

6.2.2 Overview of the Surface Model

Our approach to create the character animation involves four steps.

1) Build a skeleton for the arm. The skeleton will be used to guide the interpolation between poses. Then, position the skeleton appropriately for each pose to best fit the skeleton. This is currently a manual, interactive process.

2) Designate a particular pose as being a "master pose" for the surface triangulation of the model. All the other poses will be eventually converted to this triangulation.

3) For each of the mesh vertices in the master pose, assign ownership to the closest skeleton bone. Redefine the coordinates of each vertex in terms of the local coordinate frame of the bone.

4) Convert the non-master poses to the same representation as the master pose, as follows. Using the skeleton, bring the master pose surface in alignment with the target pose. Then, determine the closest surface point on the target pose for each vertex from the master mesh. This point is then used as the corresponding position of the given master vertex for the given target pose, and is once again stored in the local coordinate frame of one associated bone.

When the skeleton's posture is changed, the surface motion follows the skeleton as well as locally morphs between the two closest keyframes to simulate muscle deformation, as shown in Figure 55.

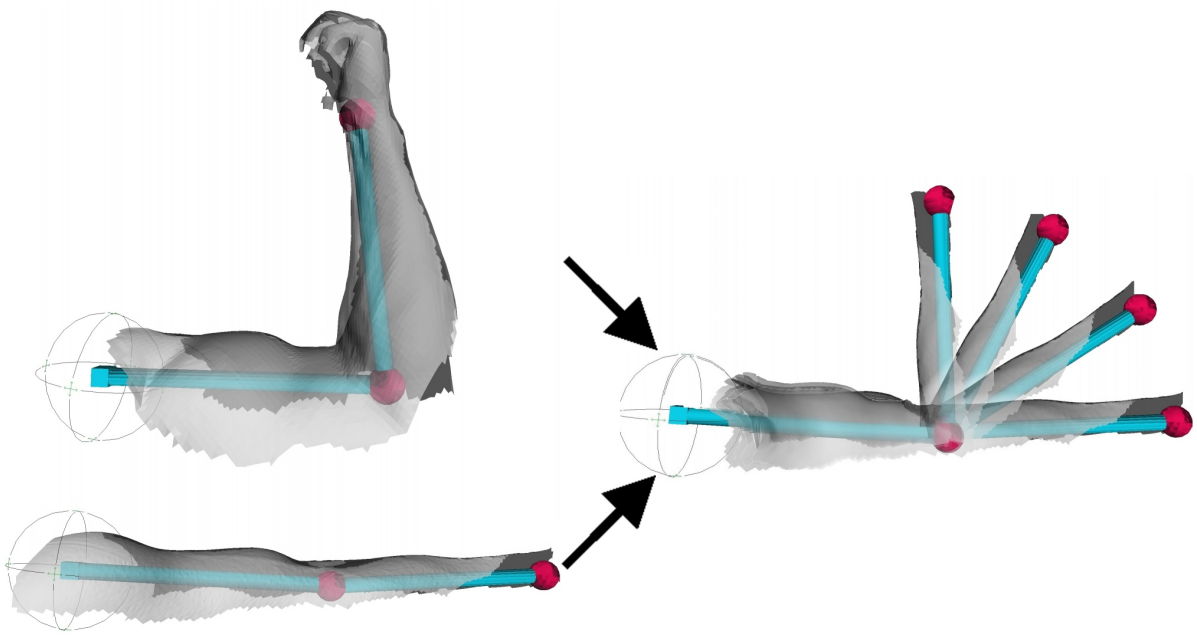

Figure 55: Using a skeleton and morphing to animate a human arm.

6.2.3 The Skeleton

Boundary representations, such as created by the integration step, have no distinct links and joints. If such surfaces are to be animated, an articulated skeleton is needed to define an underlying set of links and joints. This skeleton is not meant to be anatomically correct, but rather provide a frame of reference with which parts of the boundary

representation can be associated. Providing a skeleton for a surface model consists of two steps. First, designing the skeleton. Second, positioning the skeleton with the skin model.

The second step, alignment of the skeleton, is currently performed manually using a graphical interface. Figure 56 shows the skeleton used for our implementation. It has manipulation handles for the shoulder, elbow and wrist. The shoulder handle has six degrees of freedom which translate and rotate the skeleton arm as a whole. The elbow controller has two (translational) degrees of freedom. It moves in the plane defined by the shoulder, the elbow and the wrist. Translation of the elbow results in bones *A* and *B* changing length. The wrist handle can also be translated in the same plane as the elbow handle. Moving it causes bones *A* and *B* to rotate while their length remains constant.

While this particular interface is just one of many possible designs, we have found that it works well in practice and allows the rapid creation of an appropriate skeleton that underlies the surface model. One skeleton instance is created for each of the surface models in different postures. All skeleton instances are linked together so that the bone *A* length and the bone *B* length remain the same between the different skeletons. Their respective elbow angles remain constant when the bone lengths are adjusted. This is to allow alignment of the skeleton in different postures.

Steps one and two could be combined if we used an algorithm as proposed in [57] to generate the skeletons automatically.

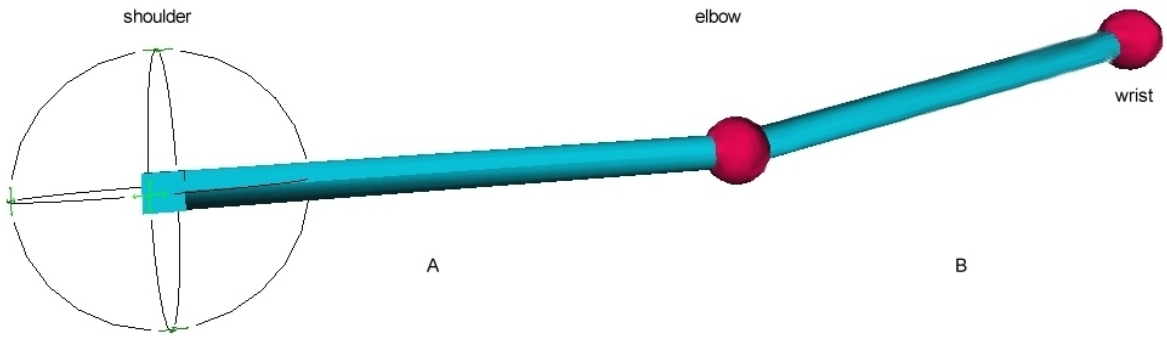

Figure 56: IK skeleton of an arm.

Figure 57 shows the surface models of an arm in two different poses, with their underlying skeletons. We used two keyframes in our example, but could have used more to provide a more realistic model.

6.2.4 Choosing a Master Pose

We choose to have all the meshes converted to one triangulation. This means that the morphs created will never be exactly the same to any pose besides the master pose. However, because we assume dense range data was used to construct each keyframe, the morphs will be a good approximation to each keyframes.

6.2.5 Assigning mesh vertices to a bone

The binding of the surface to the skeleton, as mentioned in step 3, creates a set of rigid links from the mesh vertices similar to the robot of Figure 1. Discontinuities in the

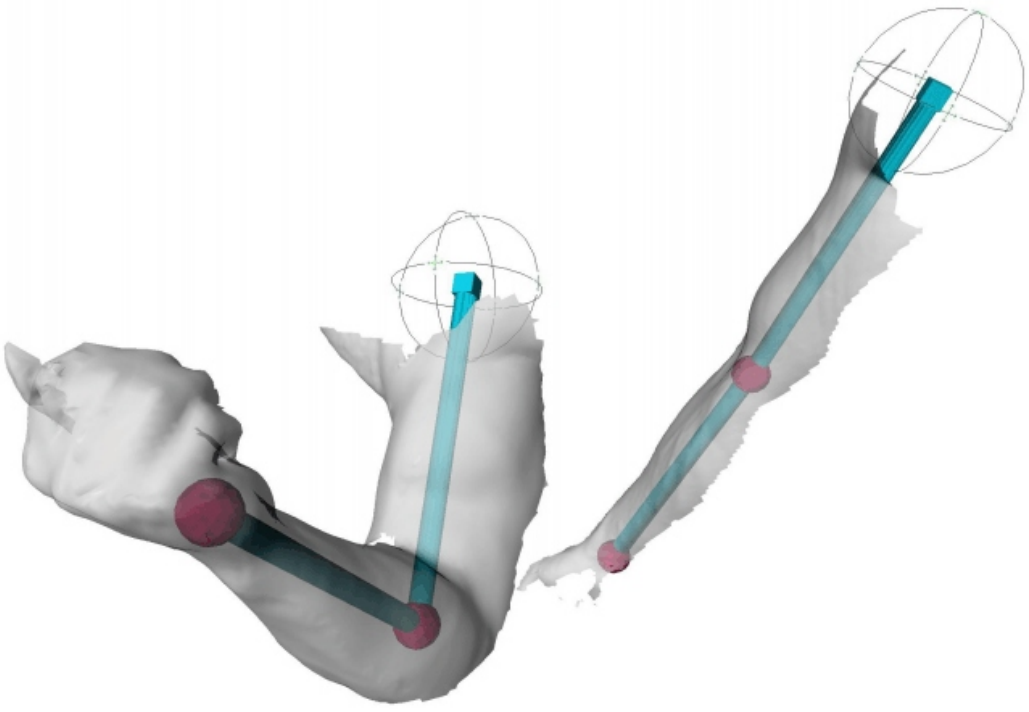

Figure 57: Two poses for a human arm, shown together with their underlying skeleton.

surface will appear when bones are rotated.

A more sophisticated binding scheme can be used to avoid this discontinuity problem. Vertices can be associated with their two closest bones. The position of a vertex in the global reference frame would be a weighted average of the position resulting from the frames local to each bone. The weighting function could be a function of the distance from each bone, with a shorter distance resulting in a stronger influence of a bone on a vertex position.

Alternatively, a spring network can be used to connect the surface vertices to the underlying bones [38] and between the surface vertices themselves [57]. This approach has the vertices following the bones as above and then using the spring network to fine-tune the vertex positions. This creates a flexible skin-like surface, and has been widely used in applications such as plastic surgery simulation [38].

Our approach is more concerned with the surface deformation that occurs due to muscle contraction than avoiding discontinuities; we therefore implement the ‘rigid link’ approach as mentioned in step 3. Furthermore, the availability of keyframe data means that the role of vertex ownership is less important than in previous work in this area. Our technique could nevertheless be augmented with any of the techniques mentioned above in order to fine-tune the vertex positions after the (larger scale) deformations have been applied.

6.2.6 Establishing Correspondence Between Surface Vertices

We do not attempt to establish direct correspondences between the vertices of the integrated meshes which have been obtained for each posture. We have found that simply using the closest surface points to establish correspondence between two keyframes produces very good morphs. We assume that little information is lost in this change of representation because of the dense range data used to create our keyframes, and because the skin deformations induced by muscle contractions are relatively small and vary smoothly.

6.2.7 Interpolation

To animate the skin deformations as a function of the selected parameters, each vertex is translated to create a smooth transition of the skin between keyframes. Because the closest surface point is used to establish correspondence between the source and target surface, each vertex follows a linear path in space from the source position to the target position.

A key consideration in choosing a suitable interpolant is the number of keyframes which participate in producing an interpolated shape. In the case of a first-order interpolant, we need at least 2^n keyframes to interpolate in the n-dimensional space of possible shapes determined by n parameters. A muscle group like the biceps requires up to four parameters. Another important consideration is the result that the interpolant produces.

A linear interpolation between 2 keyframes for one parameter was implemented as a proof of concept for our system. Higher order interpolants are possible, but are not implemented in our work. The results produced by this interpolant are visually satisfying.

6.3 Animation Results

Figure 59 shows the animation generated from the keyframes presented in Figure 57. Generating the interpolating model took 6 seconds[10]. The resulting arm model is interactive and can be made to adopt any posture between the two keyframes. The correct correspondence between surface points is established everywhere but near the elbow joint. This is because the vertices near the elbow joint in the keyframe representing the bent arm were invisible to the range scanner. A more sophisticated skin model is required to

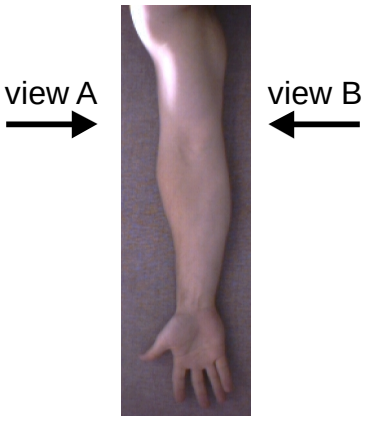

Figure 58: The two viewpoints of the arm animation shown in Figure 59 and Figure 60.

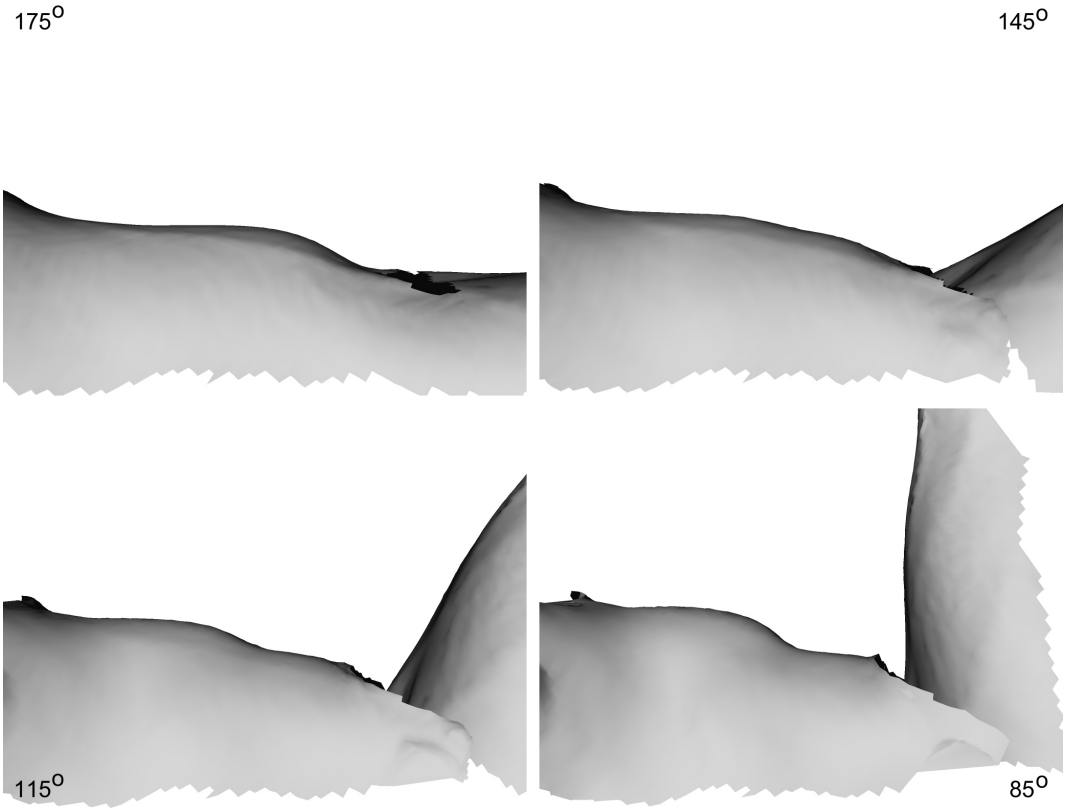

Figure 59: Animation of a human arm (view A).

[10] Timings are from execution on an SGI Indigo2 (R4400 processor), with 192 MB of RAM.

compensate for this deficiency in the data. As well, a better surface model could eliminate the discontinuity that appears in the surface as the elbow flexion occurs. Figure 60 shows the same arm animation as seen from the other side.

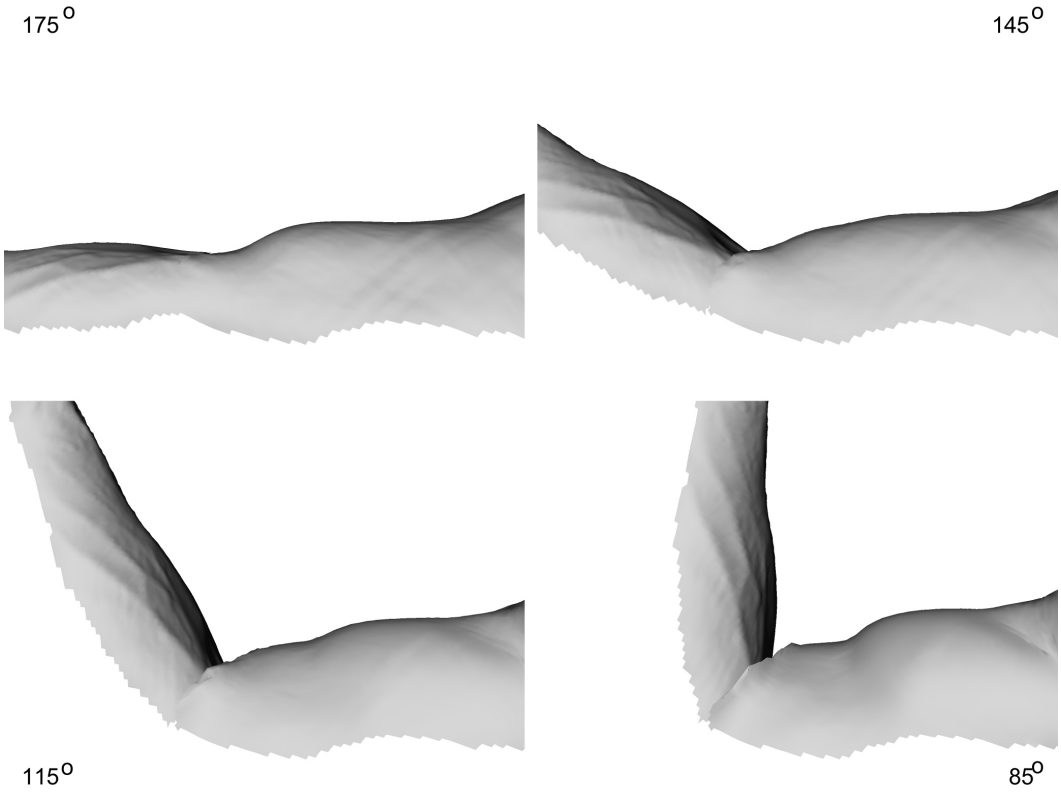

Figure 60: Animation of a human arm (view B).

7. CONCLUSION

The goal of this thesis was to investigate a method for creating accurate animations of the human form using range data. The method should be as intuitive to use as techniques currently available in commercial packages, yet produce deformations of higher quality. The system that we proposed and implemented satisfies these goals by using experimentally obtained deformation data to create realistic models with realistic deformations for a human arm.

The main contribution of this thesis has been a methodology for building a realistic animated model of the human arm using range data. The secondary contributions of this thesis are the use of k-d trees for an efficient closest-point search during the ICP step of range image registrations, and the novel approach to zippering meshes together. A proof-of-concept of the complete system was also implemented.

7.1 Future work

An obvious direction to pursue for this work involves modeling an arm with more parameters. It will be very interesting to model the potentially complex interaction of multiple joints and multiple muscle groups. For example, the biceps muscle controls the supination of the wrist. How many different positions (wrist rotation and elbow flexion) are required to be able to accurately interpolate the correct shape of the arm is an open question. There are also more complex joints with even more parameters, such as the shoulder joint. The skin will also need to be implemented as a continuous surface, perhaps employing technique such as proposed in [60]. Another obvious direction would be to model a complete human being.

Another interesting question is at what point anatomically-based models might become a better representation than simply working directly with the surface shape alone. The trick in this type of scheme would be that of solving an estimation problem, namely that of computing estimates of the many anatomical parameters based upon the captured shapes.

The system could be extended to automatically model humans with different proportions. Different body types would need to be scanned to build a library of models that could be used to create any type of body. This presents several new challenges: How many different bodies would be needed in that library to model predominant body types? How would the parameters that let one select a particular body type be presented to the user without requiring the user to have a degree in anatomy? Perhaps an interface with increasingly detailed parameters could be used. At the simplest level, the user only decides the very basic traits, such as the sex of the character. In the next level, the user decides the character's height, percentage body fat, overall muscular development. At the most detailed level, specific parameters of individual muscle groups are under direct user control.

Although our animation system is currently keyframe-based, it could easily be interfaced through a physics-based animation system. In the implementation presented the degree of skin deformation is only dependent on the joint angle, but it could also be made to depend on the force exerted by the muscles. This information would be available from the physics-based animation system. Our system could also provide information for the physical simulation, such as placing a limit on the amount of force an individual with certain muscular development can generate. More research is needed as to how much data is required to ensure that deformations arising from muscle contraction intensity are accurately modeled.

The above suggestions imply that more data must be acquired. The amount of data required can potentially become very large. As a result, the data acquisition process needs to be improved. Even though performing one scan takes on the order of 2 seconds, adjusting the scanner for different poses is time consuming and it is difficult for the model to remain immobile for extended periods of time. It is preferable to avoid using plaster models in the interest of keeping the creation of models a quick and simple process, and because of the potential loss of detail that would result.

Faster acquisition systems, such as the Cyberware WB4 stripe scanner can take 22 seconds for a whole body scan, and even this is too long for a human to stay immobile.

The Cyberware scanner is in use in less rigorous applications such as apparel design [43], and model swaying is still a problem. Possible solutions are to help the model remain immobile by building some supports, to let him sway slightly and correct the data with post processing, or to rapidly detect whether the model moved [18] and to repeat the scanning until the subject is immobile. If supports are constructed, they should be covered with material that absorbs the projected laser light. They would still cast a shadow and should therefore be made as small as possible, but at least they would not have to be manually eliminated from the data. Shape Tape™ [19] could be used to help correct for sway as a data post-processing step.

Using a faster data acquisition system, such as stereo cameras, could eliminate problems associated with subject movement. The subject would only have to exercise his joints and the system could capture a large number of keyframes. Using multiple stereo vision systems to capture data simultaneously from different viewpoints, along with Shape Tape™ [19] to automatically generate the underlying skeleton could greatly speed the entire process.

More detail could perhaps be preserved in the morphs if vertices were added or subtracted from the source surface to create the same surface as the target surface. When morphing from the source keyframe to the target keyframe, we translate the source surface's vertices. This means that the source will never look exactly like the target since it won't have the same vertices in the same location. Intensity data (or even color data, if available) could be used to establish better correspondence in the morphing step.

The rendering of dynamic surfaces creates potential problems because of the large number of model points which require updating. This results in poor cache performance, and there may therefore be interesting methods of avoiding this potential rendering penalty.

7.2 Summarizing Remarks

Current skin deformation models are typically over-simplified, sacrificing results for ease of use, or require too much anatomical knowledge, sacrificing ease of use for results. We proposed an approach that potentially offers the best of both worlds. The system in its current implementation still requires significant work to make it practical for use, but an implementation has demonstrated the viability of this approach.

8. REFERENCES

[1] Amenta, N., M. Bern & M. Kamvysselis, "A New Voronoi-Based Surface Reconstruction Algorithm", Proceedings of Siggraph 1998, Orlando, Florida, pp. 415-421.

[2] Arun, K. S., T. S. Huang & S. D. Blostein, "Least-Squares Fitting of Two 3-D Point Sets", IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 9, no. 5, 1987, pp. 698-700.

[3] Ballantyne, J. "Range Imaging Development: 1995 Summary Report", Technical Report SPAR-R.1313, SPAR Aerospace, Brampton, Ontario, May 1996.

[4] Ballantyne, J., "VERO Final Report", Technical Report SPAR-R.1388, SPAR Aerospace, Brampton, Ontario, August 1997.

[5] Beraldin, J. A. & M. Rioux, "Calibration of an Auto-synchronized Range Camera with Oblique Planes and Collinearity Equation Fitting", NRCC Report ERB-1041, November 1994.

[6] Bergevin, R., D. Laurendeau, & D. Poussart, "Estimating the 3D Rigid Transformation Between Two Range Views of a Complex Object", Proceedings of the 11th International Conference on Pattern Recognition, The Hague, The Netherlands, 1992, pp. 478-482.

[7] Bergevin, R., D. Laurendeau & D. Poussart, "Registering Range Views of Multipart Objects", Computer and Vision Image Understanding, vol. 61, no. 1, 1995, pp. 1-16.

[8] Bern, M. & D. Eppstein, "Mesh Generation and Optimal Triangulation", Technical Report P92-00047, Xerox Palo Alto Research Center, March 1992.

[9] Besl, P. & N. McKay, "A Method for Registration of 3-D Shapes", IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 14, no. 2, February 1992, pp. 239-256.

[10] Bhanu, B., "Representation and Shape Matching of 3-D Objects", IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. PAMI-6, no. 3, May 1984, pp. 340-351.

[11] Blais, G. & M. D. Levine, "Registering Multiview Range Data to Create 3D Computer Objects", Technical Report TR-CIM-93-16, Centre for Intelligent Machines, McGill University, 1993. Available: ftp://ftp.cim.mcgill.ca/pub/techrep/1993/CIM-93-16.ps.Z

[12] Boissonnat, J. D., "Representing 2-D and 3-D Shapes with the Delaunay Triangulation", Proceedings of the 7th International Conference on Pattern Recognition, 1984, pp. 745-748.

[13] Bolles, R. & P. Horaud, "3DPO: A Three-Dimensional Part Orientation System", International Journal of Robotic Research, vol. 5, no. 3, fall 1986, pp. 3-26.

[14] Burke, M. W., "Image Acquisition", Chapman & Hall, London, UK, 1996, 917 pp.

[15] Champleboux, G., S. Lavallée, R. Szeliski & L. Brunie, "From Accurate Range Imaging Sensor Calibration to Accurate Model-Based 3-D Object Localization", Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Champaign, Illinois, June 15-20, 1992, pp. 83-89.

[16] Chen, Y. & G. Medioni, "Object Modeling by Registration of Multiple Range Images", Proceedings of the 1991 IEEE International Conference on Robotics and Automation, April 1991, pp. 2724-2729.

[17] Curless, B. & M. Levoy, "A Volumetric Method for Building Complex Models from Range Images", Proceedings of Siggraph 1996, New Orleans, Louisiana, pp. 303-311.

[18] Daanen H. A. M., M. A. Brunsman & K. M. Robinette, "Reducing Movement Artifacts in Whole Body Scanning", Proceedings of the International Conference on Recent Advances in 3-D Digital Imaging and Modeling, Ottawa, Canada, May 1997, pp. 262-265.

[19] Danisch, L. A., "Bend-enhanced fiber optic sensors", Fiber Optic and Laser Sensors X, SPIE Vol. 1795, 1992, pp. 204-214.

[20] Edelsbrunner, H., D. Kirkpatrick & R. Seidel, "On the Shape of a Set of Points in the Plane", IEEE Transactions on Information Theory, vol. 29, no. 4, 1983, pp. 551-559.

[21] Edelsbrunner, H. & E. P. Mucke, "Three-dimensional Alpha Shapes", Proceedings of the 1992 Workshop on Volume Visualization, Boston, October 19-20, 1992, pp. 75-82.

[22] Faugeras, O. D. & M. Hébert, "The Representation, Recognition and Locating of 3-D Objects", International Journal of Robotic Research, vol. 5, no. 3, fall 1986, pp. 27-52.

[23] Faugeras, O. D., "Real time correlation-based stereo: algorithm, implementation and applications", Report INRIA 2013, 1993.

[24] Ferrie, F. P. & M. Levine, "Integrating Information from Multiple Views", Proceedings of the IEEE Computer Society Workshop on Computer Vision, December 1987, pp. 117-122.

[25] Fischler, M. & R. C. Bolles, "Random Sample Consensus: a Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography", Communications of the ACM, vol. 24, no. 6, 1981, pp. 3-26.

[26] Friedman, J. H., J. L. Bentley & R. A. Finkel, "An Algorithm for Finding Best Matches in Logarithmic Expected Time", ACM Transactions on Mathematical Software, vol. 3, no. 3, Sept. 1977, pp. 209-226.

[27] Guenter, B., C. Grimm, D. Wood, H. Malvar & F. Pighin, "Making Faces", Proceedings of Siggraph 98, pp. 55-66.

[28] Hoppe, H., T. DeRose, T. Duchamp, J. McDonald, & W. Stuetzle, "Surface Reconstruction from Unorganized Points", Proceedings of Siggraph 1992, pp.71-78.

[29] Horn, B. K. P., "Closed-form Solution of Absolute Orientation Using Unit Quaternions", Journal of Optical Society of America A, vol. 4, no. 4, April 1987, pp. 629-642.

[30] Huang, T. S., S. D. Blostein & E. A. Margerum, "Least-squares Estimation of Motion Parameters from 3-D Point Correspondences", Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Miami Beach, Florida, June 24-26, 1986.

[31] Ingber, L., "Very Fast Simulated Reannealing (VFSR)", Mathematical and Computer Modeling, vol. 12, no. 8, 1989, pp. 967-973.

[32] Jasiobedzki, P., "Detecting Driveable Floor Regions", Intelligent Robots and Systems IROS95, pp. 264-270.

[33] Jasiobedzki, P., "Stereo Vision for Range Measurements", Technical Report SPAR-R.1312, SPAR Aerospace, Brampton, Ontario, December 1995.

[34] Jasiobedzki, P., "Fusing and Guiding Range Measurements with Color Video Images", Proceedings of the International Conference on Recent Advances in 3-D Digital Imaging and Modeling, Ottawa, Canada, May 12-15, 1997, pp. 339-344.

[35] Johnson, A. E. & M. Hébert, "Recognizing Objects by Matching Oriented Points", Technical Report CMU-RI-TR-96-04, Carnegie Mellon University, May 1996, 36 pp.

[36] Johnson, A. E. & M. Hébert, "Surface Registration by Matching Oriented Points", Proceedings of the International Conference on Recent Advances in 3-D Digital Imaging and modeling, Ottawa, Ontario, May 12-15, 1997, pp. 121-128.

[37] Kamgar-Parsi, B., J. Jones, & A. Rosenfeld, "Registration of Multiple Overlapping Range Images: Scenes without Distinctive Features", Proceedings of the IEEE Computer Vision and Pattern Recognition Conference, June 1989, pp. 282-290.

[38] Koch, R. M., M. H. Gross, F. R. Carls, D. F. von Büren, G. Fankhauser and Y. I. H. Parish, "Simulating Facial Surgery Using Finite Element Models", Proceedings of Siggraph 1996, New Orleans, Louisiana, pp. 421-428.

[39] Lavallée, S., R. Szeliski, and L. Brunie, "Matching 3-D Smooth Surfaces with their 2-D Projections Using 3-D Distance Maps", SPIE vol. 1570 Geometric Methods in Computer Vision, San Diego, July 1991, pp. 322-336.

[40] Lerios, A., C. D. Garfinkle and M. Levoy, "Feature-Based Volume Metamorphosis", Proceedings of Siggraph 1995, Orlando, Florida, pp. 448-456.

[41] Lowe, D. G., "Perceptual Organization and Visual Recognition", Kluwer Academic Publishers, 1985.

[42] Memmler, R. L. & D. L. Wood, "Structure and Function of the Human Body", J.B. Lippincott Company, 1987, 272 pp.

[43] Paquette, S. P., "3D Scanning in Apparel Design and Human Engineering", IEEE Computer Graphics and Applications, 1996, pp. 11-15.

[44] Potmesil, M. "Generating Three-Dimensional Surface Models of Solid Objects from Multiple Projections", Ph.D. Thesis, Rensselaer Polytechnic Institute, 1982, 239 pp.

[45] Press, W. H., B. P. Flannery, S. A. Teukolsky & W. T. Vetterling, "Numerical Recipes in C", Cambridge, UK, Cambridge University Press, 1988, 994 pp.

[46] Pulli, K., "Rigid Registration of 3D Data", General Examination Report, University of Washington, February 1996. Available: http://www.cs.washington.edu/homes/kapu/generals/paper.ps

[47] Roth-Tabak, Y. & R. Jain, "Building an Environment Model Using Depth Information", Technical Report CSE-TR-07-88, University of Michigan, September 1988

[48] Sameth, H., "The Design and Analysis of Spatial Data Structures", Addison-Wesley, Reading, Massachusetts, 1989.

[49] Scheepers, F., R. E. Parent, W. E. Carlson & S. F. May, " Anatomy-Based Modeling of the Human Musculature", Proceedings of Siggraph 1997, Los Angeles, California, pp. 163-172.

[50] Schwartz, J. & M. Sharir, "Identification of Partially Obscured Objects in Two and Three Dimensions by Matching Noisy Characteristic Curves", International Journal of Robotics Res., vol. 6, no. 2, 1987, pp. 29-44.

[51] Sederberg, T. W., P. Gao, G. Wang & H. Mu, "2-D Shape Blending: An Intrinsic Solution to the Vertex Path Problem", Proceedings of Siggraph 1993, Anaheim, California, pp. 15-18.

[52] Soucy, M. & D. Laurendeau, "Multi-Resolution Surface Modeling from Multiple Range Views", Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Champaign, Illinois, June 15-20, 1992, pp. 348-353.

[53] Stone, R. J. & J. A. Stone, "Atlas of the Skeletal Muscles", WCB Publishers, Dubuque, IA, 1990, 210 pp.

[54] Szeliski, R., "Estimating Motion from Sparse Range Data Without Correspondence", 2nd International Conference on Computer Vision, Dec. 5-8, 1988, pp. 207-216.

[55] Taubin, G., "Algebraic Nonplanar Curve and Surface Estimation in 3-Space with applications to Position Estimation", Technical Report LEMS-43, Div. Eng., Brown University, Providence, RI, 1988, 25 pp.

[56] Taubin, G., "About Shape Descriptors and Shape Matching", Technical Report LEMS-57, Div. Eng., Brown University, Providence, RI, 1989, 24 pp.

[57] Teichmann, M. & S. Teller, "Assisted Articulation of Closed Polygonal Models". Available: http://graphics.lcs.mit.edu/~marekt/anim/

[58] Terzopoulos, D. and K. Fleisher, "Deformable models", The Visual Computer, vol. 4, no. 6, 1988, pp. 306-331.

[59] Turk, G. & M. Levoy, "Zippered Polygon Meshes from Range Images", Proceedings of Siggraph 1994, pp. 311-318.

[60] Turner, R. & D. Thalmann, "The Elastic Surface Layer Model for Animated Character Construction", in Proceedings of Computer Graphics International '93, Lausanne, Switzerland, pp. 399-412.

[61] Vemuri, B. & J. Aggarwal, "3-D Model Construction from Multiple Views using Range and Intensity Data", Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, June 1986, pp. 435-438.

[62] Wilhelms, J. & A. Van Gelder, "Anatomically Based Modeling", Proceedings of Siggraph 1997, Los Angeles, California, pp. 173-180.

[63] Yasumuro, Y., Q. Chen & K. Chihara, "3D Modeling of Human Hand with Motion Constraint", proceedings of the International Conference on Recent Advances in 3-D Digital Imaging and Modeling, May 12-15, 1997, Ottawa, Canada, pp. 275-282.