2009

A Self-Configuring 3-D Body Scanner

David James Holtkamp
Wright State University

A SELF-CONFIGURING 3-D BODY SCANNER

A Thesis submitted in partial fulfillment

of the requirements for the degree of

Master of Science in Computer Engineering

By

David Holtkamp

B.S.C.E., Wright State University, Dayton, Ohio — 2008

2009

Wright State University

WRIGHT STATE UNIVERSITY

SCHOOL OF GRADUATE STUDIES

June 3, 2009

I HEREBY RECOMMEND THAT THIS THESIS PREPARED UNDER MY SUPERVISION BY David Holtkamp ENTITLED A Self-Configurable 3-D Body Scanner BE ACCEPTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF Master of Science in Computer Engineering.

_____
Arthur A. Goshtasby, Ph.D.

Thesis Director

_____
Thomas Sudkamp, Ph.D.

Department Chair

Committee on
Final Examination

_____
Arthur A. Goshtasby, Ph.D.

_____
Bin Wang, Ph.D.

_____
Thomas Wischgoll, Ph.D.

_____
Joseph F. Thomas, Jr., Ph.D.
Dean, School of Graduate Studies

ABSTRACT

Holtkamp, David. M.S., Department of Computer Science and Engineering, Wright State University, 2009. A Self-Configuring 3-D Body Scanner.

A flexible, self-configuring body scanner is described that is capable of capturing and merging range data from multiple views into a single coordinate system without the use of registration. The scanner uses two disconnected frames with embedded lights to merge the coordinate systems of multiple cameras. The frame also serves in finding the laser plane as the lasers are swept over the surface from multiple locations. Both hardware and software details are presented as well as techniques for automating most aspects of the scanner. A new implicit surface implementation is also described for processing and triangulating the resulting point clouds along with the design and use of measurement tools for analyzing the completed scan.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF FIGURES (CONTINUED)

LIST OF FIGURES (CONTINUED)

LIST OF FIGURES (CONTINUED)

LIST OF TABLES

ACKNOWLEDGEMENTS

I would like to express my appreciation to all the professors in the Department of Computer Science and Engineering at Wright State whose dedication, patience, and genuine interest in their students' knowledge and understanding of subject matters have helped bring me to where I am today.

I would especially like to convey my sincere gratitude to Professor Arthur Goshtasby for his instruction and guidance throughout the research and writing of this thesis as well as throughout all of my graduate and undergraduate studies. His insights and direction have been a source of continual learning. I am also thankful of Drs. Bin Wang and Thomas Wischgoll for serving on my thesis committee.

I also want to express gratitude to my parents for always being an encouragement and an inspiration throughout my studies and my entire life and to my wife for her support during my studies.

Chapter 1
INTRODUCTION AND BACKGROUND

3-D body scanners have a growing number of applications in today's society, creating a marketplace with a large variety of scanners and technologies. The apparel industry is one of the most notable users of this new and forthcoming technology. These scanners provide the ability to produce tailor-fit clothes without the time or labor to perform traditional measurements. For instance, in the future, this technology has the potential to enable a customer to "try on" virtual clothing to determine the suit with the best fit [1].

Along with creating the potential for easily obtaining custom-fit clothing, 3-D body scanners may also change the way clothing is mass produced. By using large amounts of anthropometric data gathered from the 3-D scans, the clothing industry will be able to determine ways to make clothing that fits more people, leading to a larger portion of their customer base liking the way their clothes look on them. Other industries have also found important uses for body scanners. The airline, tractor, and automobile industries are using 3-D scans to create better seating that sit pilots and drivers more comfortably [1].

3-D scanners can be classified into two types: active and passive [2]. Passive methods, such as binocular stereo and depth from defocus, are not commonly used in applications such as 3-D body scanning due to the density and accuracy of the data required. Instead, active methods, such as structured light [3] and time of flight [4], are used.

The most common type of structured light scanner is the laser scanner, which typically consists of four or eight cameras and four lasers that move synchronously together on four poles in such a way that creates a single laser plane. With the positions of both the lasers and cameras known, the scanner uses triangulation to extract 3-D information from the scene. Figure 1.1 shows an example of a 4-pole laser range scanner. White light body scanners are also available which cast a series of light stripes on the body from which measurements are extracted. Sixteen non-moving white light scanner heads are used [5].

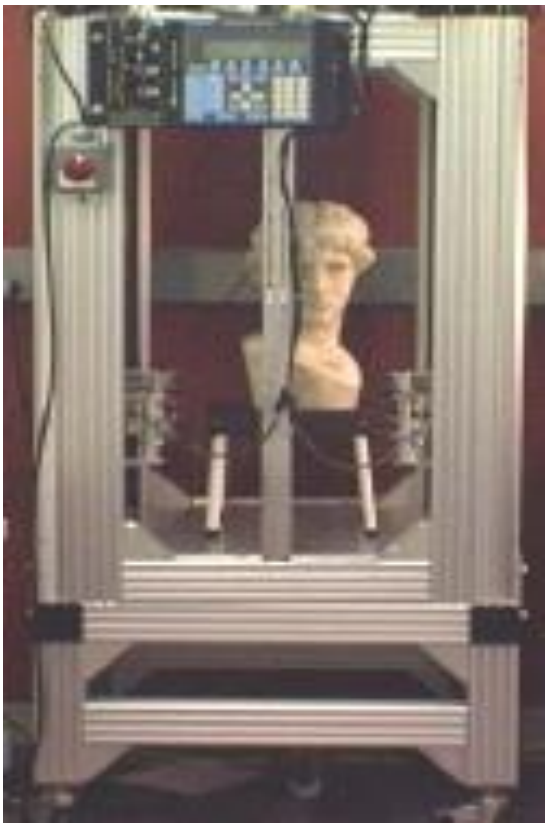

Figure 1.1: A 4-pole upper-body scanner utilizing 4 cameras and 4 lasers.

More recently, a new type of scanner has emerged that utilizes RF waves in the millimeter bandwidth and time of flight to scan subjects. These scanners are being used by the Transportation Security Administration in some airports as an alternative to pat downs. These scanners use two arrays of transmitters and receivers that rotate around the user and determine body dimensions. This type of wavelength is well suited for finding things such as hidden weapons and explosives due to its ability to see directly through clothing [6]. This same property, however, also makes it a privacy concern and a subject of active debate [7].

Two issues which have helped hinder 3-D body scanners from being used by the general public are their cost and their portability. Almost all scanners on the market today use expensive automation hardware to keep the scanner heads synchronized and determine their positions with respect to each other. In the body scanner developed in this work, both of these problems are addressed simultaneously. By using a configuration frame, the cameras and lasers may be casually placed around the scanning area, removing the need for expensive automation hardware and also eliminating the need for any type of image registration. The proposed design makes the scanner flexible in usage, cheap to produce, and also easy to transport and reconfigure in a new location.

In the following sections, hardware and software organization of the scanner are detailed, including an implicit surface for creating a triangulated model of the scan as well as useful measurement tools. Next, experimental results are shown, and the results are analyzed and concluding remarks are made. Finally, ideas and suggestions for future improvements to the system are presented.

Chapter 2
HARDWARE ORGANIZATION

2.1 Frame

The frame of the scanner consists of two rigid white rectangular structures made out of square hollow tubing. On each side of both of the frames are four circular lights that can be turned on and off automatically by the software. These lights are used as marker points to associate points on the frame with points on the image plane of the cameras. The lights are embedded within the frame and covered with a semi-transparent covering that both diffuses the light coming out and diffuses the laser light when it strikes it on the outside of the frame. One of the two rectangular frames is placed flat on the floor and the other is either hung from the ceiling or set on a stand directly above the other at a height around seven to eight feet. Figure 2.1 shows an illustration of the reference frames.

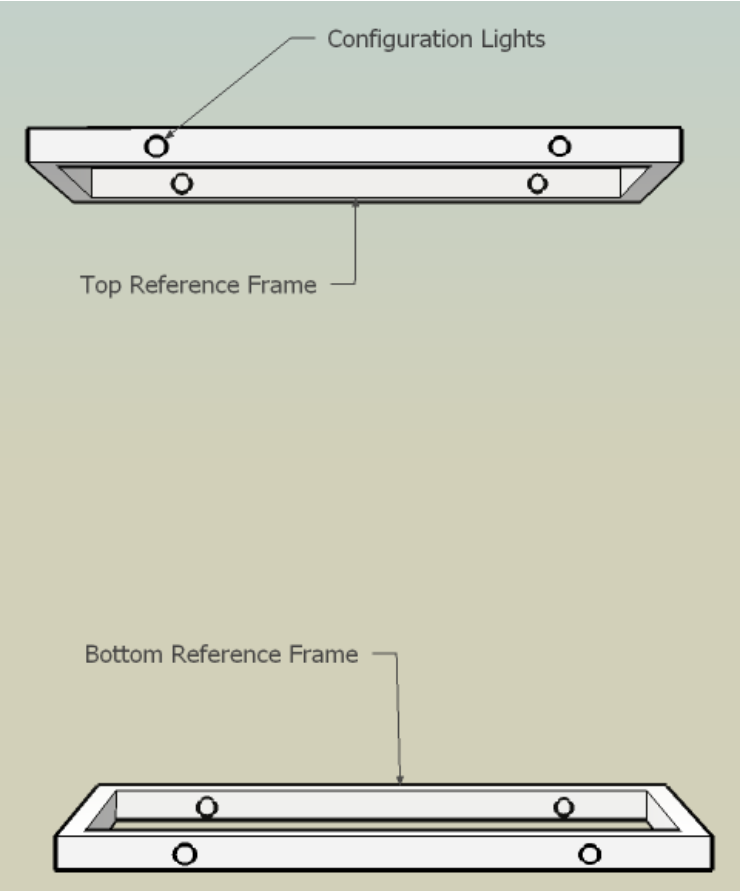

Figure 2.1: Top and bottom frame layout.

2.2 Cameras

The cameras are placed around the reference frames facing inward, with all eight light points from either the front or the back of the frames visible within the image plane of each individual camera. Any number of cameras may be used, but using fewer than three cameras will lead to a considerable amount of occlusion and loss of data from the resulting scan. An optimal setup will have the

largest angle possible between all the cameras while taking into account both the symmetry of the body and the symmetry of the frame.

The resolution of the camera will directly correlate to the accuracy of the scanner. For our implementation, the scanner used three IEEE 1394 monochrome machine vision cameras with 1280 x 1024 pixels CCD. One factor to consider when selecting the resolution of the camera is the resulting frame rate due to bandwidth constraints. A slow frame rate will result in a longer time period to scan a subject, which will raise the likelihood of the subject moving during the scan.

The type of lenses used on the cameras should also be taken into consideration. In some circumstances, a scanner may be required to fit within a smaller area; for this application, the use of wide angle lenses would be desirable. Although distortions caused by wide angle lenses can be corrected beforehand using training data, it should be taken into consideration that the accuracy of the scan will suffer to some extent. This is especially true for data captured by areas of the CCD far from the center of the image where there are far fewer pixels representing each pixel in the final corrected image. In addition to this, placing the cameras closer to the subject during a scan will increase the amount of occlusion.

2.3 Lasers

The lasers for the scanner are line lasers mounted on software-controlled, high-precision step motors with the planes aligned vertically so as to intersect both the top and bottom frame simultaneously. The lasers may be placed anywhere around the frame, but the accuracy of the scanner will decrease as the angle between the lasers and the cameras decreases. Conversely, placing the laser at a far distance from the camera will result in higher amounts of occultation; therefore, a balance must be reached between the accuracy and the amount of data gathered. This choice will be dependent on the number of lasers and cameras used within the particular scanner configuration. Another factor to consider is that if two lasers lie on opposite sides of a camera or multiple cameras, then both of the lasers may be run simultaneously as long as the laser planes do not intersect within the bounding area of the frame and the person being scanned is within this bounding area as well. Being able to use multiple lasers in this manner can cut the scan time by a factor of two. See Figure 2.2 for a layout using three cameras and four lasers, wherein two lasers are run simultaneously on their respective sides.

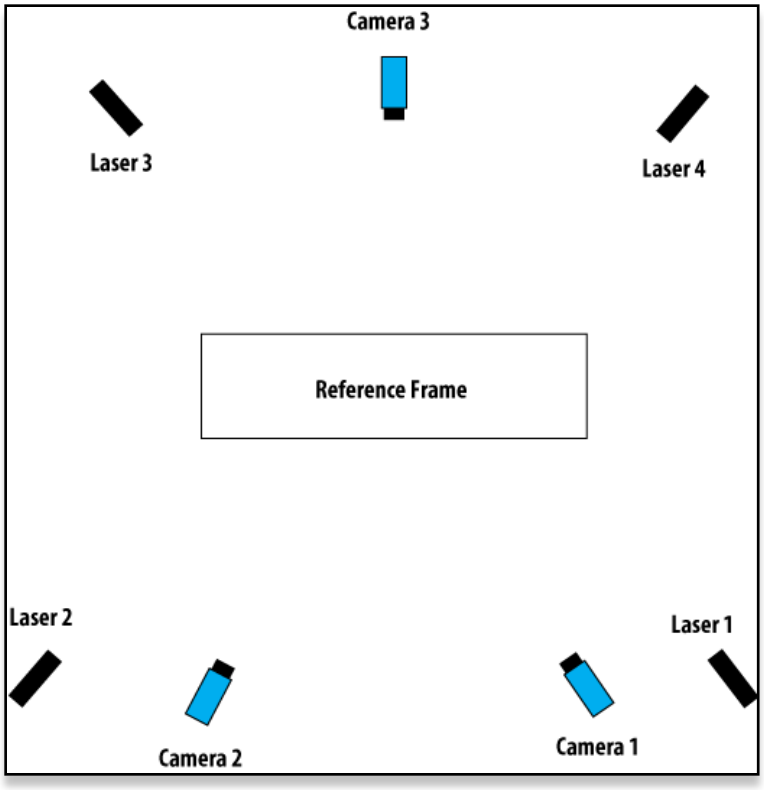

Figure 2.2: The 4-camera 3-laser layout from top view.

2.4 Automation Hardware

In order for the software to control the room lights, frame lights, and lasers, additional hardware is required. In our implementation of the scanner, three stepper boards from Peter Norberg Consulting, Inc., are used. Although there are a wide variety of stepper board controllers and automation products available, these boards were selected based on their low cost, high performance, and extensive options.

The primary board is a SS0705UCR with the routing firmware installed and a USB interface. This is the only board that interfaces with the PC running the scanner software. The other two boards are model BS0710 and SS0705CR and are both loaded with the step motor firmware. Both of these boards are wired directly into the first board. In order to communicate with these two stepper boards, the PC routes commands through the primary board.

Both boards with the steeper board firmware control two of the step motors. Along with routing commands to the other two boards, the board with the routing firmware has the ability to drive 6 lines with an additional three-volt power supply. Four of these lines are used to turn on and off the lasers. The remaining two lines are routed to an external box containing two solid state relays, which turn on when the board drives their respective line voltage high. One of the lines controls a group of lights placed around the room used for ambient light, and the other line controls the lights within the frame, which are used for calibration.

Chapter 3
SOFTWARE ORGANIZATION

3.1 Scanner Software

3.1.1 Camera Calibration

The first step in calibrating the scanner system is to remove geometric distortions from lens nonlinearities. These distortions are even greater in wide angle and aspheric lenses, which may be desirable in some configurations due to their ability to reduce the area needed to set up the scanner

To remove these distortions, a grid of evenly spaced circular spots is used. Each camera is placed in front of the grid and aimed such that the optical axis is normal to the grid. The goal of this process is to find a transformation function that will change the distorted grid in the captured image into the ideal grid.

To find this transformation for each camera, the following steps are followed:

1. Capture an image of the configuration grid with the camera that is to be configured.
2. Segment the image by local thresholding to find the spots.

3. For each spot:
    a. Find the centroid of the segmented area to approximate the center of the circular spot.
    b. Determine its ideal position within the image based on its position within the grid.
4. Knowing the ideal and obtained spot centers, determine two thin-plate spline mapping functions that map the acquired grid of spots into the ideal grid. Then, for each pixel in the distorted image, determine the coordinates of the pixel in the ideal image and save this mapping in two arrays to be used as lookup tables (See Figure 3.1).

Once the lookup tables for correcting the distorted image are created for a camera, they can be used for any subsequently acquired image from that camera as long as the lens' focus, zoom, and position with respect to the frame are not changed. For this reason, lenses that have locking screws are suggested.

$$f(x,y) = a_0 + a_1 x + a_2 y + \sum_{i=1}^{n} F_i \, r_i^2 \ln r_i^2$$

Figure 3.1: The equation for a thin-plate spline where $r_i^2 = (x - x_i)^2 + (y - y_i)^2$ *and the a's and F's are the parameters of the spline to be determine by using a number of corresponding points in the ideal and captured grids [8].*

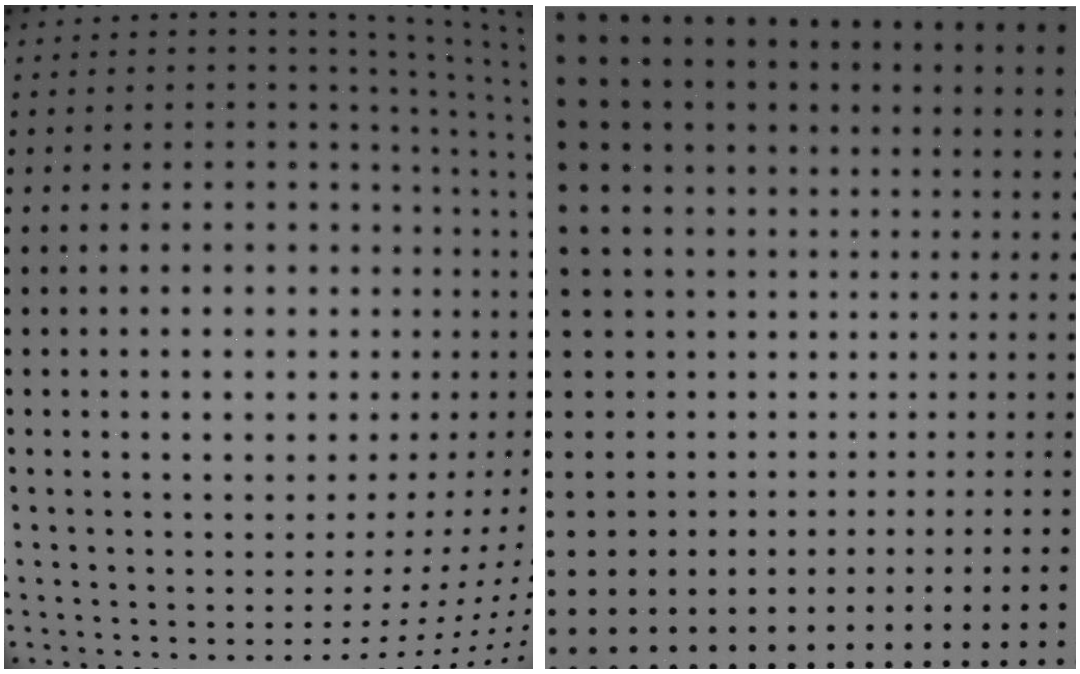

Figure 3.2: The configuration grid before and after correction for lens distortion.

3.1.2 Determining the Relation Between Image and Frame Coordinates

In order to determine the 3-D position of a laser stripe during a scan, it is first necessary to find the transformation between the image planes of each camera and the 3-D planes created by the top and bottom reference frames of the scanner. To achieve this automatically, the software-controlled lights within the scanner are used as reference points. In order to get an accurate transformation from these marker points, the location of the top frame with respect to the bottom frame should be carefully measured.

To find the frame markers, an image is first taken of the frame with the frame lights turned off. This image serves as the base image. Next, the software turns on the frame lights and takes another image. This image is subtracted from

the base image in order to remove everything except the lights. All values below a threshold value are then set to 0 so that only areas where the light points are located will be non-zero. This threshold value is found experimentally.

After the image has been cleared of noisy regions, a search is made for the non-zero regions. Once a region is located, its centroid is calculated. The centroid is then used as the center point for the light marker on the frame. Once this process is completed, eight individual light points are identified, four belonging to the top frame and four belonging to the bottom frame.

With the geometry of the frame known, it is trivial to determine the correspondence between points in the 2-D image and the corresponding points in 3-D. The only information that requires manual entry is which side of the frame the camera is placed, but even this step could be removed with an additional distinguishing marker on one side of the frame. Four sets of correspondences between image coordinates and 3-D coordinates are used to determine the following projective transformations:

$$\begin{bmatrix} a1 & a2 & a3 \\ a4 & a5 & a6 \\ a7 & a8 & a9 \\ a10 & a11 & 1 \end{bmatrix} * \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} X \\ Y \\ Z \\ w \end{bmatrix} \qquad (3.3)$$

for the bottom frame and

$$\begin{bmatrix} b1 & b2 & b3 \\ b4 & b5 & b6 \\ b7 & b8 & b9 \\ b10 & b11 & 1 \end{bmatrix} * \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} X \\ Y \\ Z \\ w \end{bmatrix} \quad (3.4)$$

for the top frame, where (x, y) is the 2-D coordinate of the point in the image plane and (X/w, Y/w, Z/w) is its corresponding 3-D point lying in the plane defined by the top or bottom reference frame. This will produce twelve equations and eleven unknowns for each frame, which can be solved using least squares to determine the best parameters of the projective transformations.

The above process is repeated on each camera for both the top and bottom frame so that each camera has transformation parameters (a1 – a11) and (b1 – b11), relating bottom and top frame coordinates to the image coordinates. Once the transformation parameters are determined, knowing the coordinates of image points representing the intersection of a laser plane with the two frames, from Equations 3.3 and 3.4, we can determine the 3-D coordinates of the corresponding frame points.

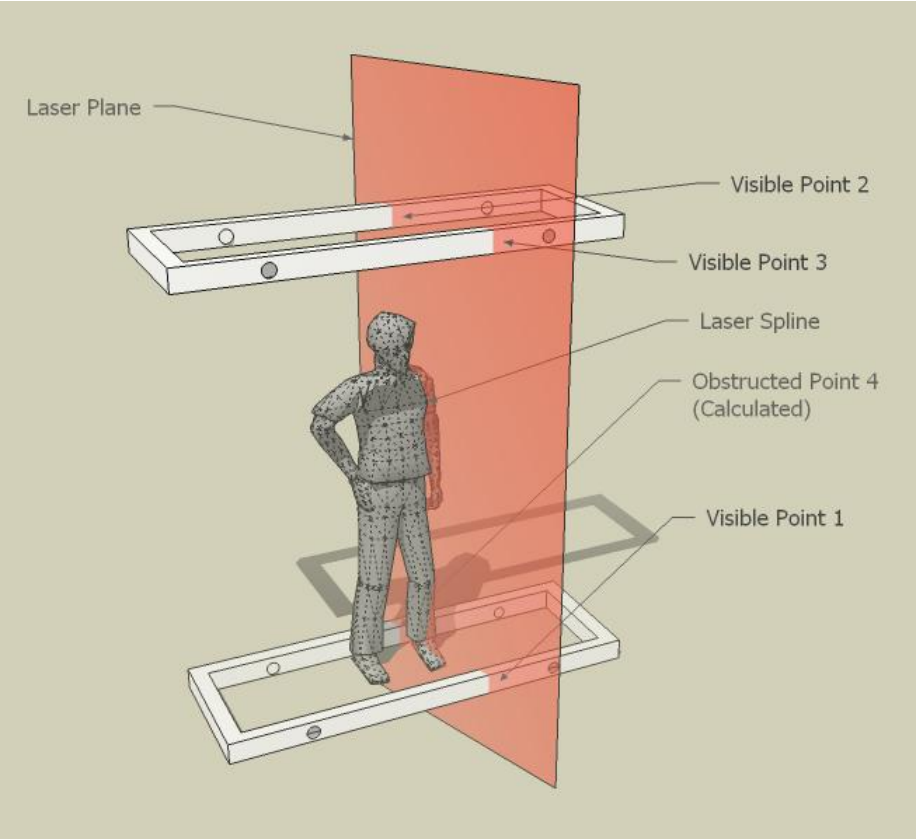

Figure 3.5: Calculating the intersections of the laser plane with the frame poles in 3-D.

3.1.3 Determining the 3-D Equation of the Laser Plane

Once the cameras are configured and the top and bottom frame transformation matrices are calculated, the location of the laser stripe forming on an object during a scan needs to be determined. To find this location, another projective transformation that relates the image coordinates to coordinates in the laser plane needs to be found, which will be denoted as c1 – c11.

$$\begin{bmatrix} c1 & c2 & c3 \\ c4 & c5 & c6 \\ c7 & c8 & c9 \\ c10 & c11 & 1 \end{bmatrix} * \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} X \\ Y \\ Z \\ w \end{bmatrix} \quad (3.6)$$

To find three of the four needed corresponding point sets to calculate the transformation, the configuration frame can be used. During the scan, the current laser plane should intersect both the top and bottom frame. Visibility of both the front pole and back pole on the top frame is ensured, but on the bottom frame the view of the far pole may be obstructed by the subject who is being scanned. To find the three visible point correspondences, the (x, y) positions of the frame markers, which were found previously, are used. Each set of two corresponding points on the same pole are used to determine the equation of the line that best fits the location of the pole on the image plane. This line is then traced along on all three visible poles to find the (x, y) location of the laser intersection using the same technique described in Section 3.1.4 in order to determine its sub-pixel position. The two top frame points are then transformed using matrix (a1 – a11) to find their corresponding (X, Y, Z) coordinates, and the bottom frame coordinates (x, y) are transformed with (b1 – b11) to find their corresponding (X, Y, Z) coordinates. Three correspondences are now known yielding nine equations, meaning additional information must still be found before the transformation for the laser plane can be determined. To obtain this information, the three points already found will be used along with the inverse projection of matrix (b1 – b11), which can be expressed as:

$$\begin{bmatrix} ib1 & ib2 & ib3 \\ ib4 & ib5 & ib6 \\ ib7 & ib8 & 1 \end{bmatrix} * \begin{bmatrix} X \\ Z \\ 1 \end{bmatrix} = \begin{bmatrix} x \\ y \\ w \end{bmatrix} \quad (3.7)$$

where (X, Z) are the respective components of the 3-D position in the plane defined by the bottom frame, and (x/w, y/w) are the corresponding coordinates in the camera's image plane. This transformation will stay constant as long as the camera and frame are not moved and, therefore, can be calculated once per camera like the transformation with parameters (a1 – a11) and (b1 – b11). It is important not to use the (X, Y) coordinates instead of (X, Z) as this will yield an under-determined system of equations if the plane lies directly in an (X, Z) plane, or a loss of accuracy if it lies very close to an (X, Z) plane.

The next step in finding the fourth corresponding point set is to estimate its 3-D position. The 3-D plane of the laser can be found using the three known points and the plane can be expressed by the equation Ax +By + Cz + D = 0. The estimated position of the obstructed 3-D point would then be at the intersection of the laser plane and the line defined by the obstructed pole. The equation of this line can be found by the location of the two configuration points already known, which will be denoted as *P1* and *P2*. The intersection can be found at point *P* with

$$P = P1 + u\,(P2 - P1) \quad (3.8)$$

where *u* is equal to:

$$u = \frac{A\,x1 + By1 + C\,z1 + D}{A\,(x1 - x2) + B\,(y1 - y2) + C\,(z1 - z2)} \quad (3.9)$$

with (x1, y1, z1) being the coordinates of *P1* and (x2, y2, z2) being the coordinates of *P2* [9].

With the 3-D coordinates of the fourth corresponding point set known, the (x, y) camera plane position can be solved for using the inverse *b* projection found earlier. This will yield the fourth set of corresponding points. With four sets of corresponding 2-D and 3-D points, the projective transform for the laser plane (c1 – c11) can now be solved with least squares using the same method as was used for the top and bottom frames' projection matrices. See Figure 3.5 for an illustration of this process.

3.1.4 Determining the 3-D Point Cloud

Now that a transformation has been determined between the image plane coordinates and the plane of the laser, all that remains is to obtain the 3-D coordinates of laser points by accurately determining the (x, y) image coordinates of the laser stripe in the image plane. This process is simplified by the fact that our laser is mounted with the projected plane approximately vertical, allowing the stripe to only intersect each row of an image a single time.

Although most laser-plane generators can be focused to produce very thin stripes, both the material on which they strike and the angle at which they strike can make the true center of a stripe difficult to determine. Some materials may

diffuse a significant amount of the laser light. The stripe may appear very wide due to the projection angle or may even appear to be projected onto two non-adjacent surfaces. To help alleviate these problems and to extract some level of sub-pixel information from the image, the following steps are followed:

Given an image frame during an image scan and a frame (base image) immediately prior to the laser becoming active:

1. From the current image, subtract the base image from before the laser was turned on to remove any other light sources in the room from the image.
2. For each row in the new image:
    a. Find the pixel with the highest intensity.
    b. If this pixel is greater than the set threshold, then continue to step c. If it is not, then go to the next scanline and back to step a.
    c. At the location of the highest intensity, find the weighted average of the values, using the pixel intensities as the weight

$$Px = \frac{\sum_{x=XMax-r}^{XMax+r} x*i(x)}{\sum_{x=XMax-r}^{XMax+r} i(x)} \qquad (3.10)$$

    where *x* is the position in the image, *i(x)* is the intensity at this position in the image, *Px* is the final sub-pixel position of the stripe for the current scanline, and *r* is the radius of the search, which should be set based on the camera resolution and the laser stripe width.
    d. Transform the (x, y) location, which was found to be (*Px,* row), with the transformation (c1 – c11) to find the (X, Y, Z) coordinate of the laser stripe at the current scanline.

e. If additional scanlines exist, go to the next scanline and go back to step a. If this was the last scanline, then stop.

3.2 Surface Generation Software

3.2.1 Problem Statement

In order to render the acquired points as a surface on graphics hardware, it is desirable to represent the surface as a set of joining triangles that accurately approximate the surface represented by the point cloud. Due to both the noise present in the scan and the varying density of data gathered by a laser scanner, it is necessary to pre-process the point cloud instead of using direct triangulation to connect the data points onto a joining surface. In addition, the surface implementation must be able to join two adjacent surfaces that lay close and run parallel to each other, which could be the result of a small error in configuration of the scanner or the movement of the subject during the scan. In order to meet the aforementioned criteria, an implicit surface has been formulated with the surface being defined as:

$$f(x, y, z) = \text{c} \tag{3.11}$$

Similar methods with implicit surfaces have been used in the past. Blinn [10] and Muraki [11] both placed a decreasing exponential on the field for each point in the point cloud in order to approximate the surface. Nishimura et

al. [12] and Wyvill et al. [13] got the same results but faster by using a polynomial function instead of exponentials. Although these methods are simple to implement, they only perform well when the point cloud used is evenly spaced.

3.2.2 Approach

3.2.2.1 Normal Estimation

Before finding the implicit surface that fits the underlying point cloud, surface normals must be estimated. In order to estimate the normal of the surface at each point in the scanned point cloud, the following covariance matrix is created using the point in question and a set of its nearest neighbors.

$$M = \begin{bmatrix} I_{xx} & I_{xy} & I_{xz} \\ I_{yx} & I_{yy} & I_{yz} \\ I_{zx} & I_{zy} & I_{zz} \end{bmatrix} \tag{3.12}$$

$$I_{xy} = \sum_{j=1}^{n} |x_j - \bar{x}| \, |y_j - \bar{y}| \tag{3.13}$$

The eigenvalues and eigenvectors are then calculated for the matrix. The eigenvector corresponding to the smallest eigenvalue is the approximation of the normal for the underlying surface and the other two eigenvectors are used as the orthogonal basis of the tangent plane. [14]

Before proceeding, it is important to make sure that all of the normal vectors are pointing in a uniform direction. In order to automate this process, a least spanning tree can be used. Each node in the tree represents a point in the cloud and each edge has the cost of the dot product between the two vectors in

question. Edges should be created between each node and its nearest *n* neighbors. The tree should be traversed and each child should have its sign changed if it would reduce the angle between itself and its parent. [14]

3.2.2.2 Implicit Surface

First, the field which will contain our implicit surface, denoted by *F*, is initialized to 0. For each point in the point cloud, a 3-D Gaussian is added to the field at the location corresponding to the point. In the two directions that are tangent to the estimated underlying surface, a Gaussian function is applied. In the direction of the surface normal, the first derivative of the Gaussian function is applied. This results in the field values within the model having one sign and the field values outside the model being the opposite sign. This can be written as follows:

$$f(\mathrm{t}_1,\mathrm{t}_2,n) = \sum_{i=0}^{n} G(t_1) * G(t_2) * G'(n) \qquad (3.14)$$

$$G(x) = e^{\frac{-x^2}{2\sigma^2}} \qquad (3.15)$$

where (t_1, t_2, n) is the position of the point in the coordinate system relative to the normal and the orthogonal basis of the tangent plane.

See Figure 3.16 for a 2-D representation of this field. Parameters σ_{t1} and σ_{t2} control the smoothing of the surface and parameter σ_n must be adjusted based on the noise in the point cloud in the normal direction. For example, if the

subject being scanned shifts during the scan, a higher σ_n will allow the two surfaces to be better merged into a single surface, but it will also have some smoothing effect upon the model, resulting in the loss of details.

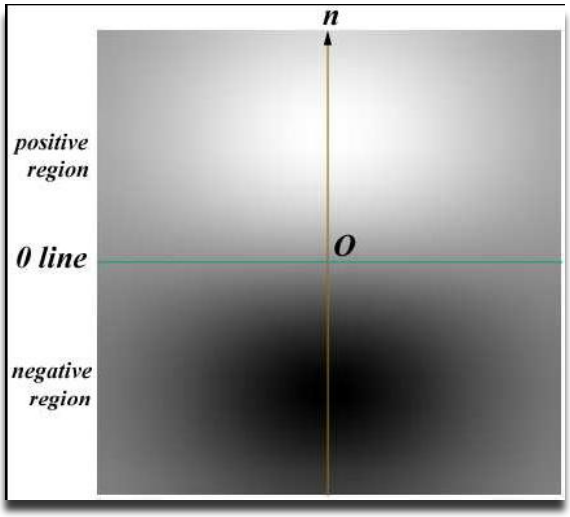

Figure 3.16: A 2-D cross section of function f(x, y, z) = G(x)G(y')G(z).

3.2.2.3 Density Adaptation

With the current formula for the implicit surface, a change in point density could result in the loss of detail in the surface represented by the point cloud. In order to accurately render an area with sparse points within a model, a high σ would be required. If in another part of the same model, the density of the points increases, it will result in smoothing of details, which could have been present in the underlying surface. Situations such as this could occur when rendering a model that has had the data points decreased in areas of low curvature. This could also happen in a scanner such as the one described in this document when the object being scanned has overlapping areas from two different

cameras or lasers, or if the surface being scanned is at a large or small viewing angle from the camera.

In order to compensate for this problem and preserve the details of the model, the σ_{t1} and σ_{t2} are scaled based on the distance of the point to the *n* points nearest to it:

$$\sigma_{Density} = \frac{\sum_{i=0}^{n} Dist[i]}{n} \quad (3.17)$$

$$\sigma_{t1} = \sigma_{t2} = \sigma_{Density} * \sigma_{User\ Selected} \quad (3.18)$$

where *n* is the number of nearest neighbors and *Dist[i]* is the distance of the point in question to its *i*'th nearest neighbor. A larger *n* in this equation will cause a slower change in the way density affects the model, which could still over-smooth fine details. On the other hand, a small *n* will cause rapid changes in σ and that could cause a small group of points that just happen to lay next each other in a sparse area to cause an artifact on the surface. For this paper, the number of nearest neighbors used in the implementation of the surface was 10, which was found after testing to be the best number in most circumstances.

This refinement to the formula also removes any issue of model scale when rendering the implicit surface from a point cloud. Regardless of whether the model is in inches or millimeters, the *σ* will never need to be adjusted by the user to compensate for scale.

3.2.2.4 Curvature Adaptation

In order to allow for sharp corners to be represented correctly in the implicit surface, an additional factor is added. Because the Gaussian field being added to our primary field *F* can cause sharp curves and corners to expand, it cannot accurately represent the underlying surface. To alleviate this flaw, σ is lowered in areas of high curvature in the direction of the curvature. This is done by approximating the curvature of the surface at each point in the direction of both of the principle axes of the tangent plane as determined by the covariance matrix. The curvature can be approximated from the point cloud in a variety of different ways. The most straightforward approach would be to fit a cubic surface, oriented with respect to the two tangent vectors, to the surrounding point cloud using least squares. This process is computationally expensive, and, if there is significant noise in the model, it may require repeating the process in order to remove any outlier points, which may cause large errors in the calculation when using least squares.

In order to lower the computation time, two other methods were also tested. One method fits circles to the neighboring points. Because a circle is a two-dimensional shape, all of the points cannot be used simultaneously. Instead, to find the estimated curvature, points that lie near the planes defined by the normal vector and each tangent are projected onto each plane. A circle is then fitted to the points in each plane using the Levenberg-Marquardt algorithm [15]. The estimated curvature around each tangent vector is the reciprocal of the

circle's radius fitted in the plane defined by the opposite tangent and the normal vector.

Similar to surface fitting, the circle fitting is very vulnerable to outliers due to the use of least squares and may require use of the algorithm multiple times to remove the outliers. The other method addresses both the stability and speed issue while sacrificing some accuracy by only taking into account the angle between the normal vectors of the point in question and its neighbors and distances between the point in question and its neighbors.

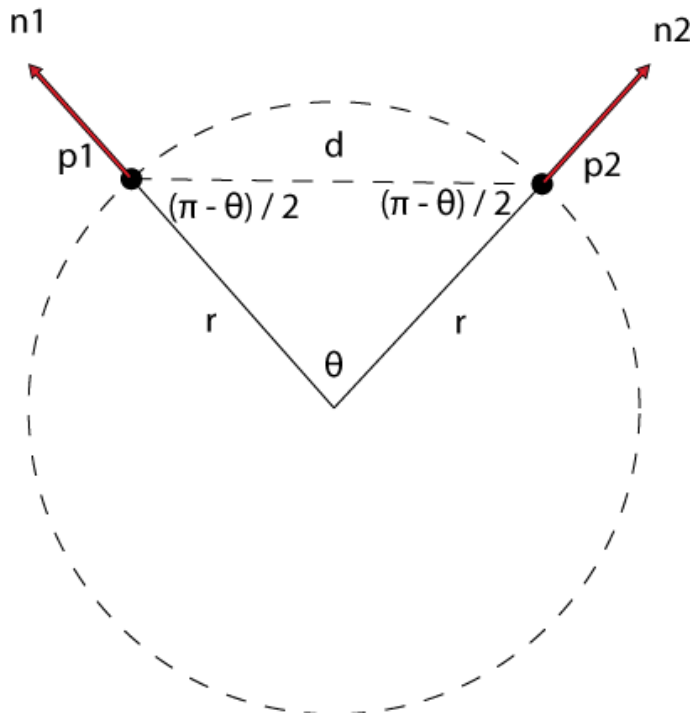

Figure 3.19: Finding the estimated curvature of the point cloud.

For this method, we assume that the local curvature approximates a sphere. For the purpose of illustration, this will be shown in 2-D, but it is actually done in three dimensions. Given two points, *p1* and *p2*, the distance *d* between

them, and the angle θ between the two normals, the estimated curvature is equal to the inverse of *r*, where *r* is the radius of the circle, which can be determined using the law of sines.

$$\frac{r}{\sin\left(\frac{\pi-\theta}{2}\right)} = \frac{d}{\sin\left(\theta\right)} \tag{3.20}$$

Solving for the inverse of *r* yields the curvature approximation:

$$c = \frac{1}{r} = \frac{D*\sin\left(\theta\right)}{\sin\left(\frac{\pi-\theta}{2}\right)} \tag{3.21}$$

Or,

$$c = \frac{2*\sin\left(\frac{\theta}{2}\right)}{D} \tag{3.22}$$

Finally, we expand this formula and use it to find the curvature at point *p* with normal *n*, its *i*th nearest neighbor NN_i, its normal n_{NNi} , and the distance between *p and* NN_i being *Dist[i].* The formula can either be applied directly to

$$c = \frac{\sum_{i=0}^{Neighbors} 2*\sin\left(\frac{\cos^{-1}\left(n * n_{NNi}\right)}{2}\right)}{Neighbors} \tag{3.23}$$

the local point cloud to estimate the overall curvature or, similar to the previous method, applied to only the points near the planes defined by the vectors composing the orthogonal basis of the tangent plane in order to get the curvature in both directions individually.

The final implementation in this project uses a combination of the last two methods given to estimate the curvature with the actual curvature determined by fitting circles and using the last more stable method to see whether the circle fitting has become unstable. If this is found to be the case, the curvature value is set to the magnitude of the curvature value found by the last method.

After estimating the curvature using one of the aforementioned methods, these values are used to adjust the σ's in their corresponding direction. To allow the user to select the desired surface sharpening, an additional parameter s is added. When s is set to 0, the curvature will have no additional effect on the model.

$$\sigma_{t1} = \sigma_{Density} * \sigma_{User\ Selected} - \frac{s * c_{t1}}{\sigma_{Density}} \qquad (3.24)$$

$$\sigma_{t2} = \sigma_{Density} * \sigma_{User\ Selected} - \frac{s * c_{t2}}{\sigma_{Density}} \qquad (3.25)$$

The curvature term is divided by $\sigma_{Density}$ in order take local density into account. Without this term, a model at a smaller scale, which has a small curvature relative to its local density, might erroneously receive a low σ.

3.2.3 Implementation

In order to efficiently implement the implicit surface, the surface is calculated discretely by using a voxel grid to represent the field at discrete locations. Initially all field values are set to zero. For each point in the scanned

point cloud, the Gaussian function is calculated using its corresponding σ values. This field is then rotated to correspond with the normals and other two axes found from the inertia matrix's eigenvectors and then added to the primary field. Once all points in the point cloud have their fields summed together, the marching cubes algorithm is used to generate triangles from the field's zero surface [16].

Because a Gaussian approaches 0 exponentially, it is not necessary to calculate the values of the individual point fields for every position in the final model field. Instead, an error tolerance can be decided upon and then a block size determined based on the σ of the Gaussian.

$$e^{\frac{-\frac{D}{2}^2}{2\sigma^2}} < \epsilon \tag{3.26}$$

$$D = 2\sigma\sqrt{-2\ln(\epsilon)} \tag{3.27}$$

where D is the size of the block in one dimension, and ε is the error tolerance. Calculating the field in this way allows for a significant reduction in computation time with a negligible loss in accuracy.

Depending on the hardware being used and the resolution of the desired surface, it may not be possible to allocate a single contiguous block of memory large enough to represent the overall field. To address this issue, sub-fields are used. Each subfield must take into account all points lying within its field as well as all points lying within $D / 2$ voxels in order to assure all points that may affect the field are taken into account, where D is the size of the largest block as

defined by Equation 3.27. With this stipulation in place, all of the subfields may be rendered individually and the triangles resulting from each sub-field may be merged into a single model with no artifacts present on the interface between adjacent fields. See Figure 3.28 for a depiction of this process.

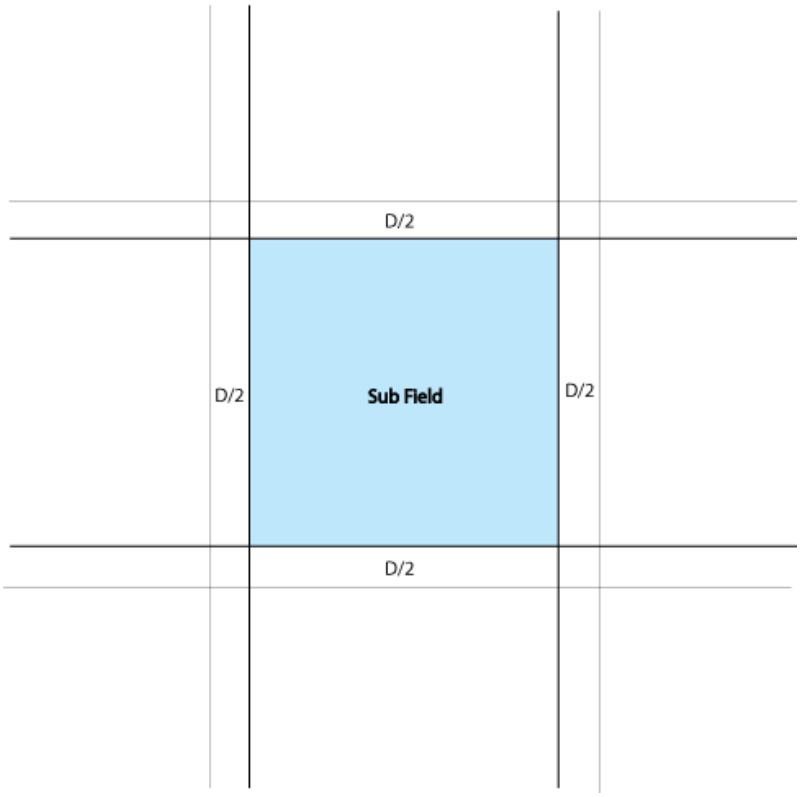

Figure 3.28: Breaking the model field into subfields.

3.2.4 Examples

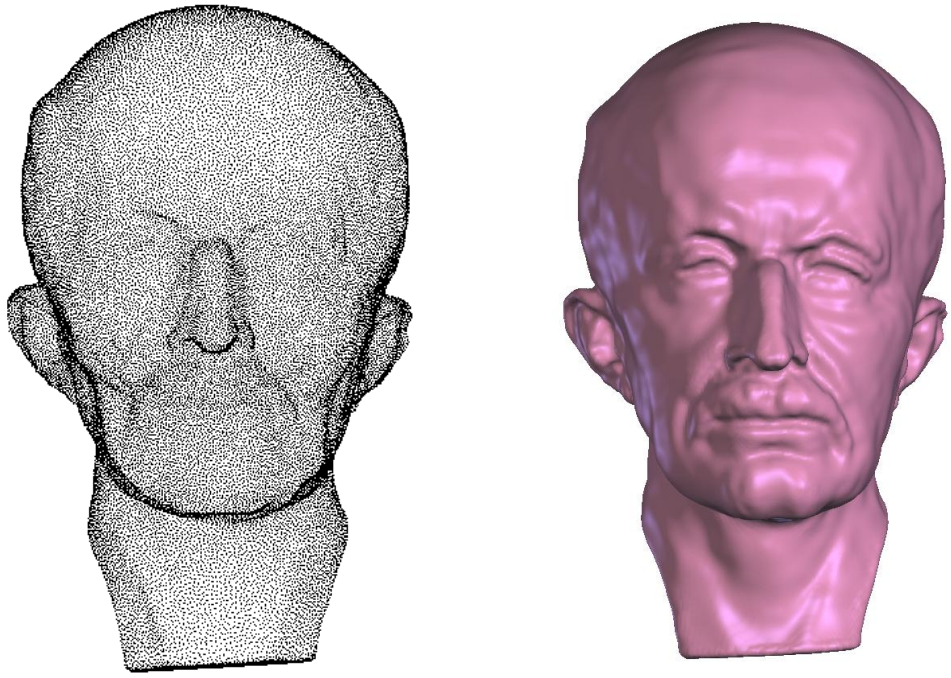

Figure 3.29: Point cloud and final model of the Max Planck bust model from MPI.

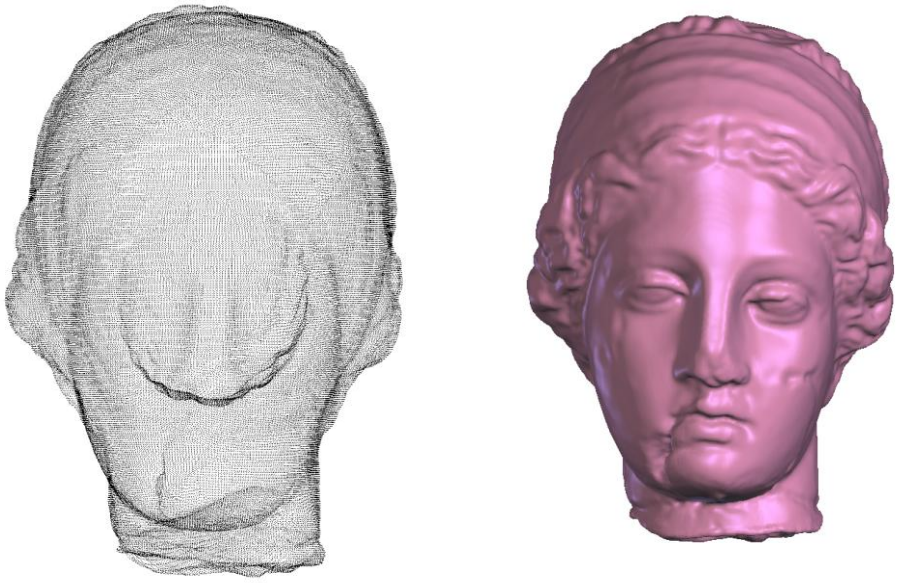

Figure 3.30: Point cloud and final model of the Igea artifact model from Cyberware.

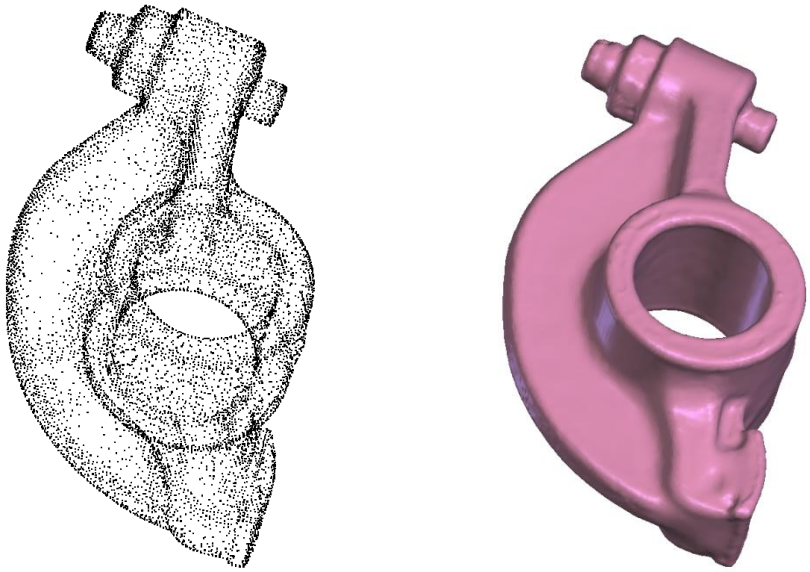

Figure 3.31: Point cloud and rendered rocker arm, which is a good example of varying point density.

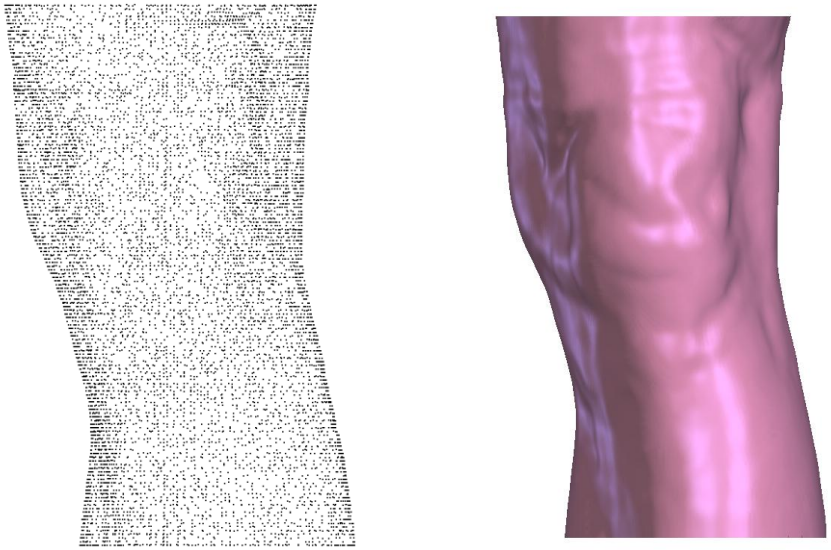

Figure 3.32: Point cloud and final model of a knee, which is a good example of an open surface.

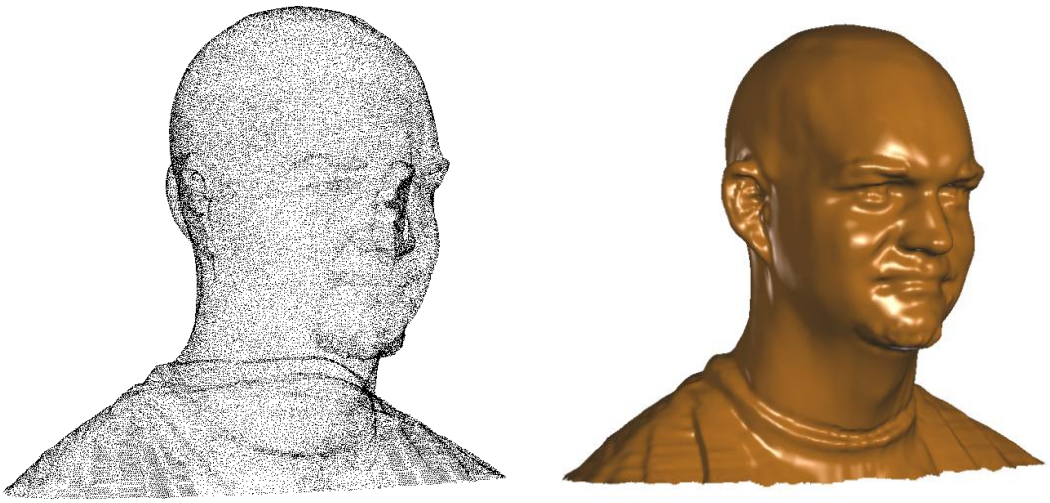

Figure 3.33: A normalized point cloud with no Gaussian noise applied and its corresponding triangulated model.

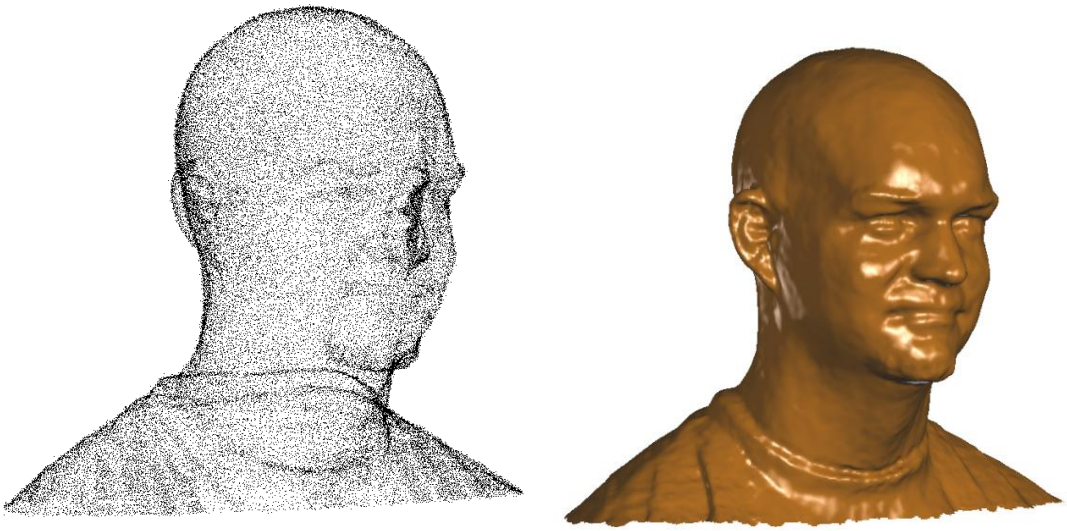

Figure 3.34: A normalized point cloud with .005 Gaussian noise applied and its corresponding triangulated model.

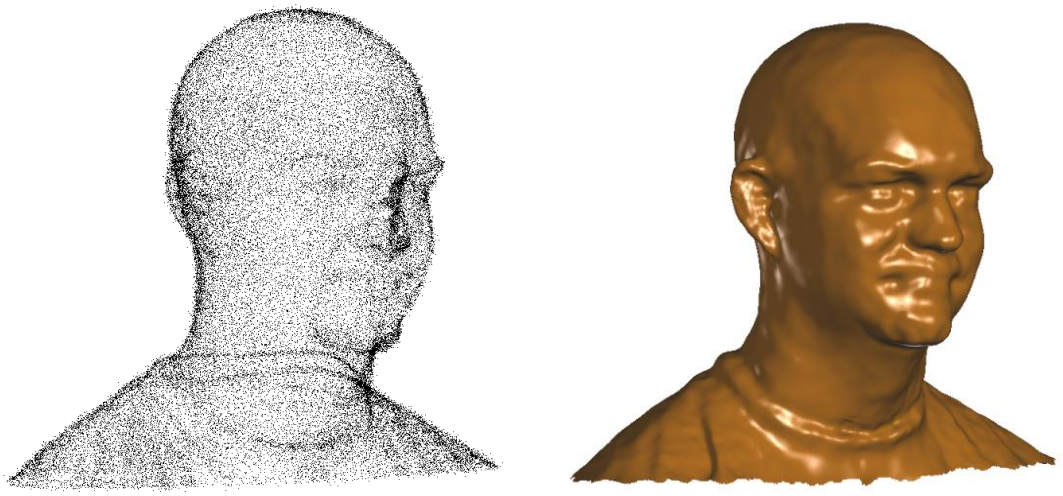

Figure 3.35: A normalized point cloud with .01 Gaussian noise applied and its corresponding triangulated model.

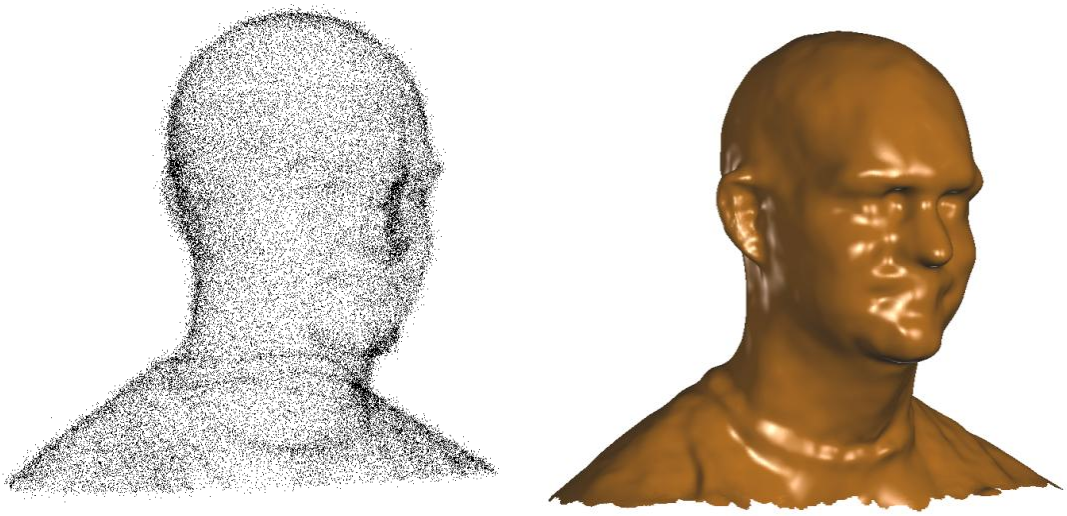

Figure 3.36: A normalized point cloud with .02 Gaussian noise applied and its corresponding triangulated model.

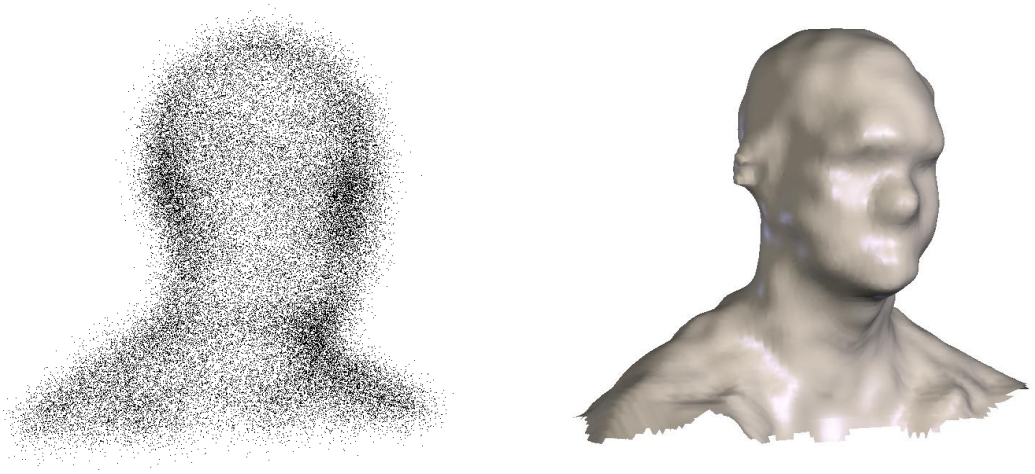

Figure 3.37: A normalized point cloud (same model as the previous figures) with .05 Gaussian noise applied and its corresponding triangulated model.

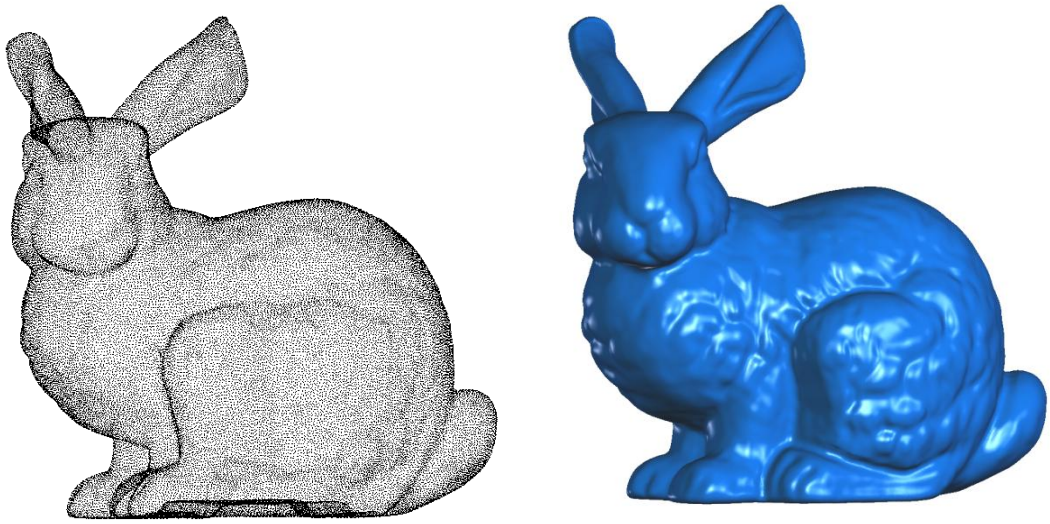

Figure 3.38: The Stanford Bunny and its corresponding triangulated model.

3.3 Measurement Tools

3.3.1 User Interaction

A number of measurement tools were developed for the 3-D body scanner to allow the user to perform measurements similar to those that a tailor would perform. These include the distance between two points along the surface, the distance around the surface, and the direct distance from point to point. The measurement tools work in an interactive manner, allowing the user to rotate the model in 3-D space. To find the distance along a curve, the user clicks on two to three points on the curve. Similarly, to find the direct distance between two points, the user clicks at two different points in the point cloud. Finally, in order to find the distance around a specific slice of the model, such as the neck or waist, the user would rotate the model to the desired angle, and then draw a line across the model where the cross section should be taken. The length around the model contained within the cross section is then calculated and displayed.

3.3.2 Implementation

To implement the aforementioned tools, either the raw point cloud from the scan is used or the vertices from the triangulation of the implicit field are used as input to the measurement software. The model is first displayed on screen using an orthogonal projection matrix, allowing the user to rotate the model. When the tool is selected for finding the distance between two points, the program waits for the user to click at a point. Once a point is clicked on screen, a

copy of the point cloud is made and rotated to coincide with the current rotation of the model on screen. The relation between the (x, y) screen coordinates and the (X, Y) model coordinates is then found using the orthogonal projection. All points within a certain distance threshold from the clicked (X, Y) point are found, and the point closest to the viewer is found, which is equivalent to the point with the smallest Z value. This is done for two points and then the Euclidian distance is calculated and displayed.

A similar method is used to find the distance around the object. Once a line is drawn on screen, a copy of the point cloud is once again created and then rotated according to the orientation of the model. All points that fall within a threshold (X, Y) distance of the line drawn are found. This collection of 3-D points is then rotated so that the plane created by the line projected through the model is aligned with the (X, Y) plane. The points in the slice can then be treated as 2-D points on a plane. To find the distance around the cloud, the convex hull is calculated using Graham's Scan algorithm, and the sum of all the segment lengths creating the convex hull is taken as the distance around the object [17]. The purpose of the convex hull is to make the measuring tool work like a measuring tape. When measuring for tailoring, it would not be desirable to measure concave areas of the body because clothes will not stick to these areas. In addition, measuring the concave areas would cause small amounts of noise in the model to create a large error in the final measurements. Using the convex hull of the object solves both problems simultaneously.

To find the distance along a curve, the program uses a combination of the previous two methods. First, the user selects two points from the cloud with which to define the curve. Note that the user may rotate the model between selecting these two points. The two points are remembered and then a line is drawn between them. This line is used to find the convex hull in the exact manner as was outlined in the previous method. Once the convex hull is found, the two points that were selected to define the curve are used to segment the convex hull into those segments lying on the curve and those lying outside the curve. The sum of all segments lying on the curve is then calculated and used as the length of the curve. An example is shown in Figure 3.39 through Figure 3.43.

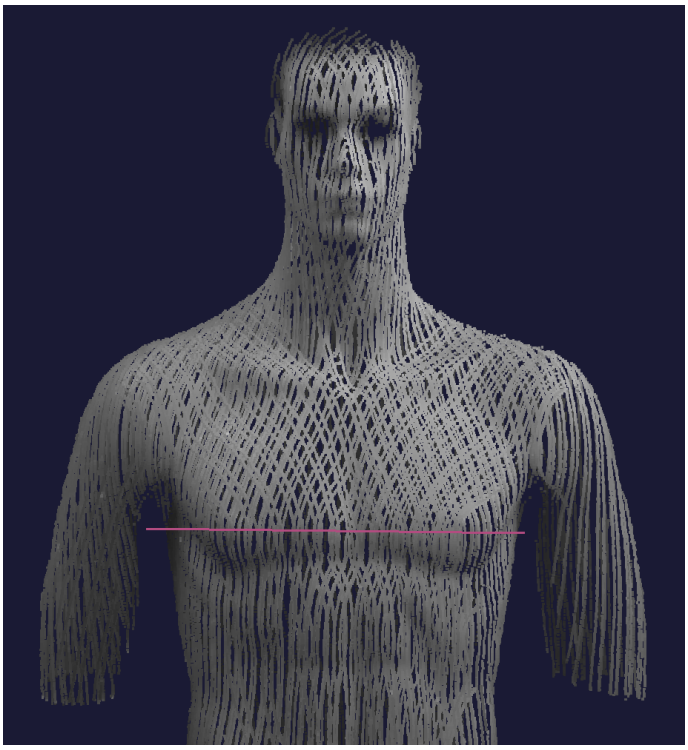

Figure 3.39: Selecting a cross section of a scan.

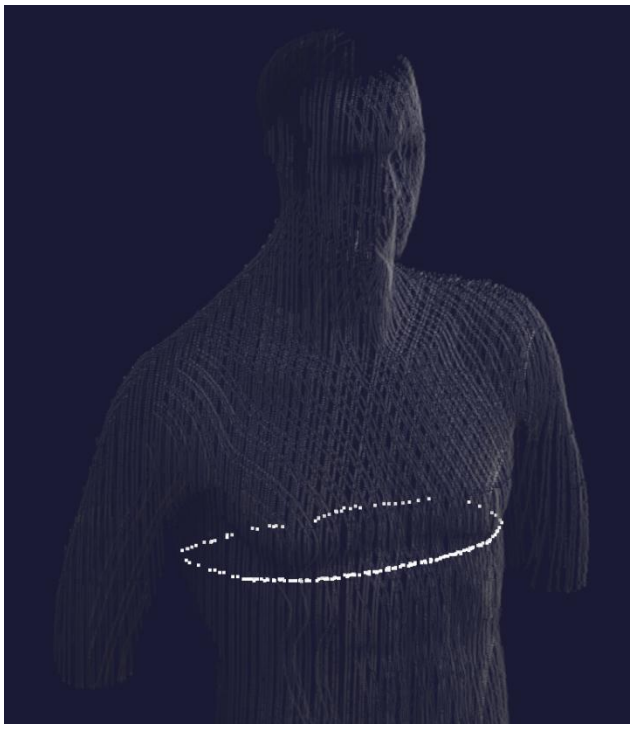

Figure 3.40: Selected points near the cross section.

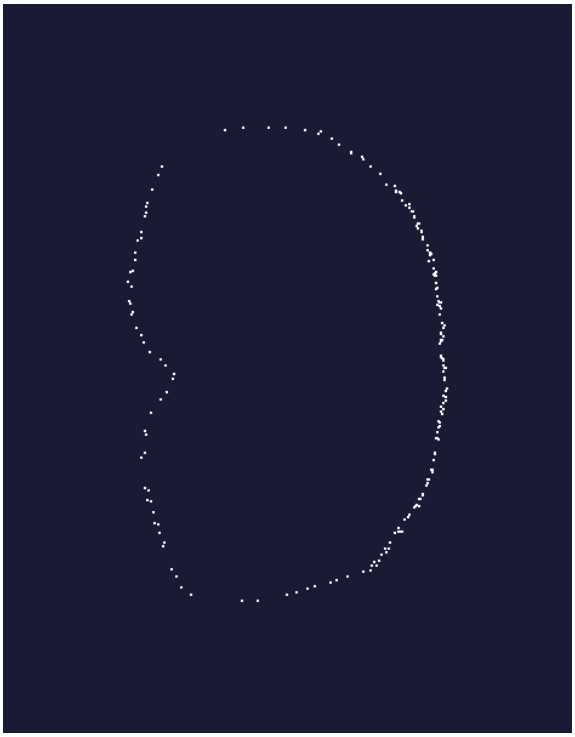

Figure 3.41: Selected points rotated in the XY plane.

Figure 3.42: A convex hull is found from the points in the XY plane.

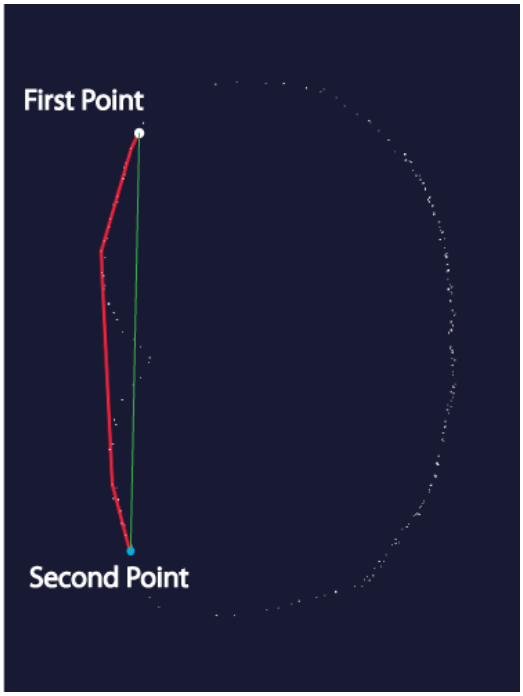

Figure 3.43: When finding the distance around a subsection, the convex hull is then restricted to the curve between the two selected points.

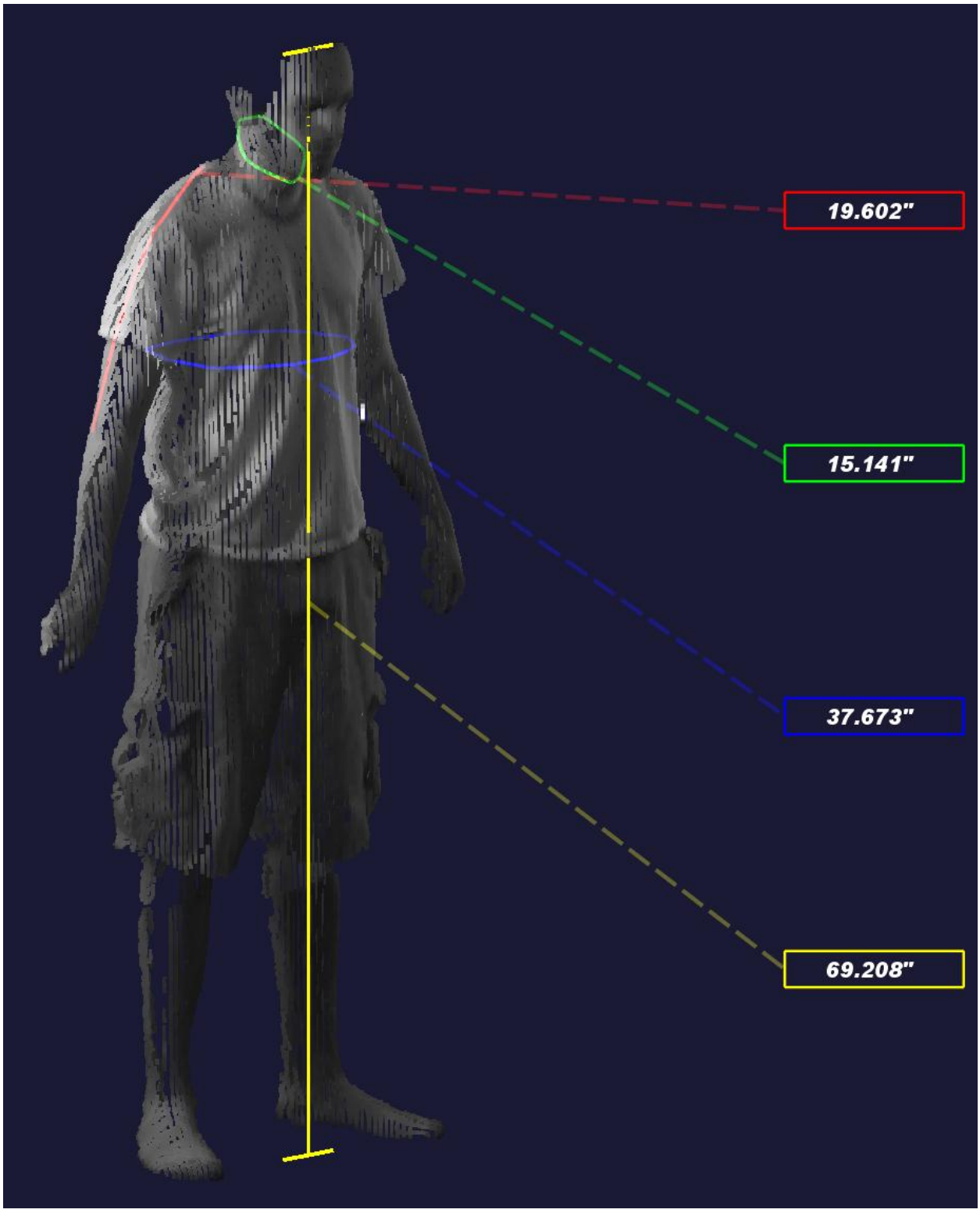

Figure 3.44: A screen shot created by the scanner's measurement tools that shows a variety of different measurements on the raw point cloud of a subject.

In summary, the software system can be broken down into four components: system calibration, scan process, surface creation, and measurement tools. The first step in calibration involves correcting lens distortions. This is done by mapping the distorted surface to the ideal surface using a grid of evenly spaced spots. The mapping is done using two thin-plate spline surfaces that represent the X offset and Y offset of the distorted surface with respect to the ideal surface. Once both surfaces are found, these offsets are used in all subsequent frames as long as the zoom, focus, and lens position do not change.

Once the cameras are configured, transformations from the image plane of each camera and the 3-D planes that are defined by the top and bottom frame are found. These transformations are determined using the light points on the frame and their corresponding positions in the image planes.

After calibration, the scanned images can be converted into a point cloud. For each frame of the scan, the location of the laser stripe is found on both poles of the top frame and the front pole of the bottom frame. These three points, which can be converted into 3-D points using the transformations found previously, along with a fourth estimated point, are used to find the projective transformation from the image plane to the current 3-D plane of the laser. Each frame is then examined along the x-axis in order to determine the current location of the laser plane, if present, in each row of the image. If the laser stripe is found, it is then converted into a 3-D point by using the previously found transformation.

Once the 3-D point cloud has been created, a surface can be obtained using a new implicit surface formulation, which was developed to help deal with some of the less desirable features of the raw point clouds: varying point density, noise, and misalignment of the scans. To create the surface, a discrete field is created and initialized to zero. At each point, a special Gaussian function is placed, oriented towards the normal, and scaled based on curvature and local point density. Once this process is completed for each point, the zero surface is extracted using marching cubes.

Finally, the built-in measurement tools can be used to determine common measurements of the scan model, including point to point, distance around a curve, and distance around a cross section. These operations can either be performed directly on the raw point cloud, or they can be performed on the vertices obtained from the marching cubes algorithm after the implicit surface is found.

Chapter 4
RESULTS

Presented below are the results obtained from the complete 3-D body scanner. First, the accuracy of the implicit surface implementation is gauged by using spheres of varying point density to measure the RMS error of the final surface. This test is done both for evenly spaced point clouds and unevenly spaced point clouds, as well as with varying levels of Gaussian noise.

Sample images are then shown to visually represent the image correction, which transforms the distorted images gathered by the cameras to the final ideal images used for processing the raw point cloud. Two scans are then presented, one of a manikin and the other of a human subject. Four images are shown for each scan to show the different steps of the scan processing procedure. First, an intensity-mapped point cloud of the subject is presented, followed by a picture of the normals found for each point. Next, an intensity texture-mapped representation of the curvature at each point is presented. Finally, the triangulated model from the implicit surface is shown. Following the visual results, each scan has two tables where different measurements taken with a tape measure are compared to the computer measurements using both the raw

point cloud and the point cloud of vertices obtained from the implicit surface and marching cubes.

4.1 Implicit Surface Results

4.1.1 Evenly Spaced Sphere

Table 4.1: RMS error of the vertices of the implicit surface of a unit sphere at varying levels of point density (y-axis) and noise added (x-axis in standard deviations). For this table, the points on the sphere were evenly spaced around the sphere as shown in Figure 4.1. σ_{xy} and σ_z were adjusted so as to get the minimum RMS for each measurement.

16384	0.00168	0.003702	0.006593	0.007977	0.008106	0.010805	0.023484
4096	0.002356	0.005056	0.008021	0.010576	0.012244	0.014794	0.02999
1024	0.004395	0.006308	0.010053	0.012786	0.01579	0.01947	0.034901
256	0.011795	0.013112	0.015654	0.021765	0.024259	0.02649	0.044915
128	0.029443	0.030343	0.03398	0.040067	0.037699	0.045917	0.064353
	0	0.01	0.02	0.03	0.04	0.05	0.1

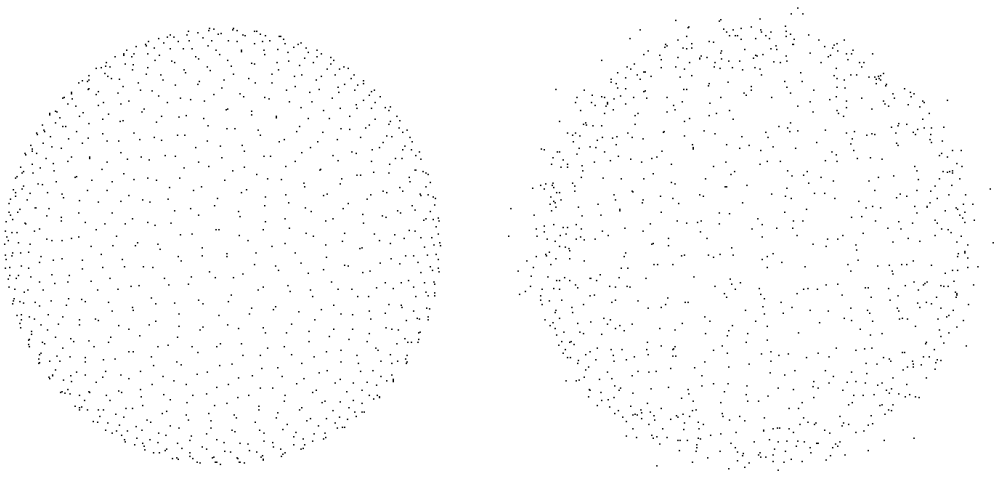

Figure 4.1: Visual representation of the sphere structure with 1024 uniformly spaced points before and after added Gaussian noise.

4.1.2 Nonuniformly Spaced Sphere

Table 4.2: RMS error of final vertices of the implicit surface of a unit sphere at varying levels of point density (y-axis) and noise added (x-axis in standard deviations). For this table, the points on the sphere were nonuniformly spaced around the sphere as shown in Figure 4.2. σ_{xy} and σ_z were adjusted so as to get the minimum RMS for each case.

16384	0.00168	0.003702	0.006593	0.007977	0.008106	0.010805	0.023484
4096	0.002356	0.005056	0.008021	0.010576	0.012244	0.014794	0.02999
1024	0.004395	0.006308	0.010053	0.012786	0.01579	0.01947	0.034901
256	0.011795	0.013112	0.015654	0.021765	0.024259	0.02649	0.044915
128	0.029443	0.030343	0.03398	0.040067	0.037699	0.045917	0.064353
	0	0.01	0.02	0.03	0.04	0.05	0.1

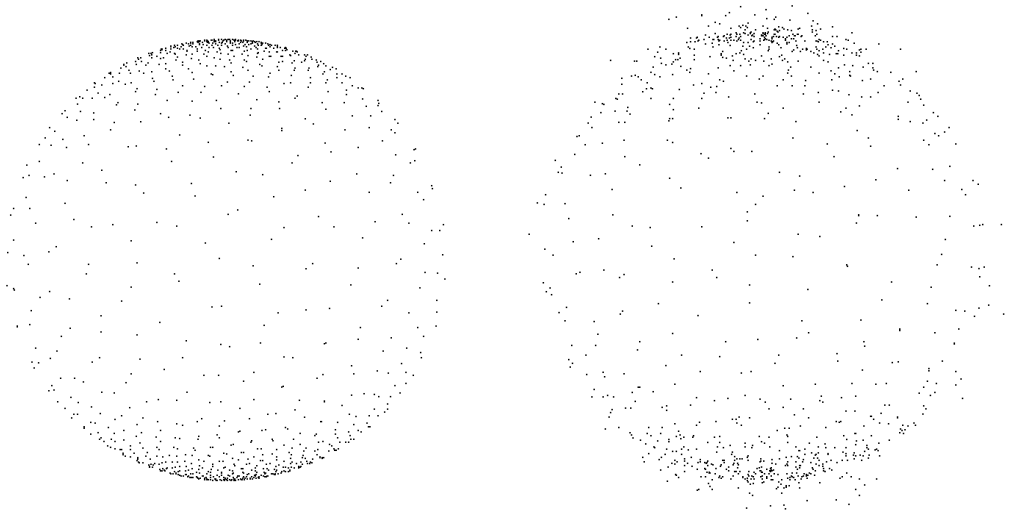

Figure 4.2: Visual representation of the sphere structure with 1024 nonuniformly spaced points before and after Gaussian noise is applied.

4.2 Image Correction Results

An example showing image correction for lens distortion is given in Figure 4.3.

Figure 4.3: An example image before and after lens distortion correction.

4.3 Examples of Actual Scans

An example of an actual scan by the developed scanner is given below.

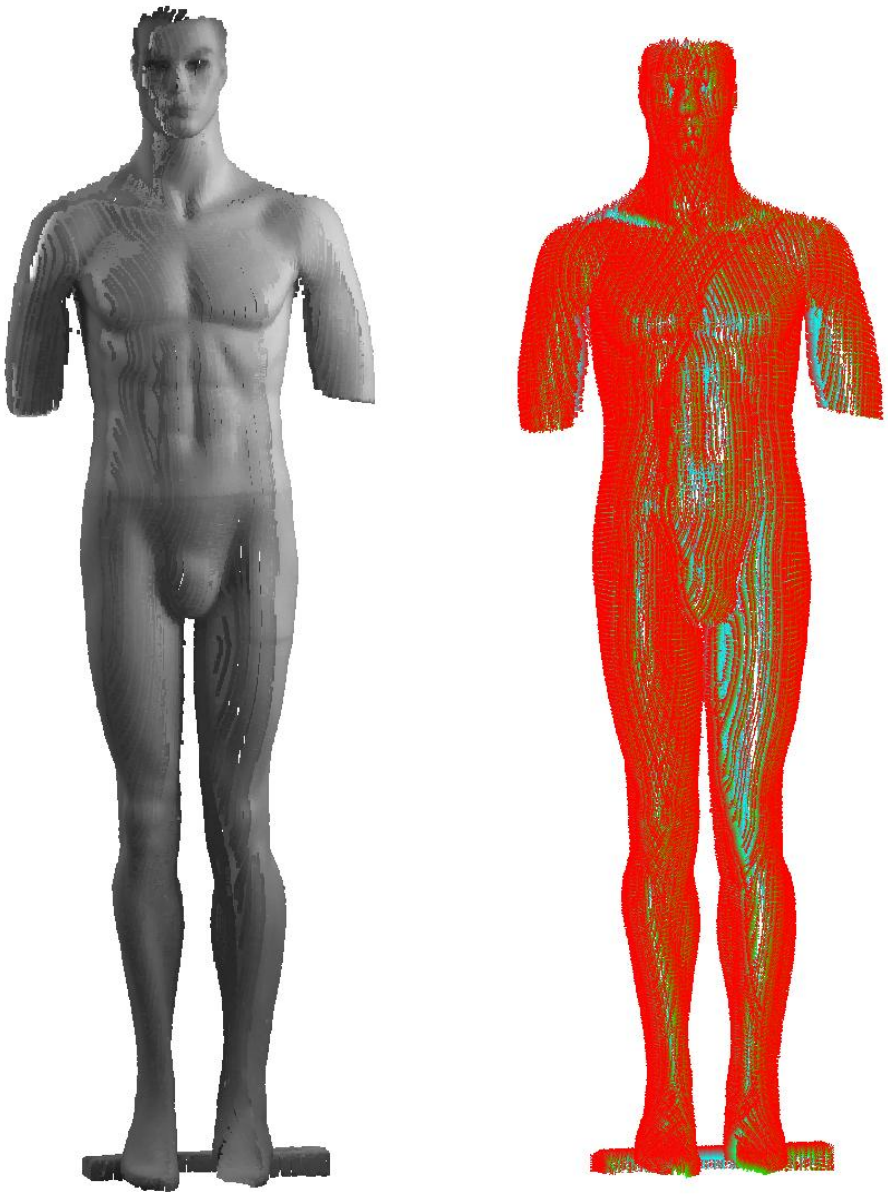

Figure 4.4: Left: Point cloud drawn with the captured intensities. Right: The points drawn as normal vectors, with the red end of the vectors facing outside. These normal vectors are used when orienting the Gaussian fields at each point during implicit surface creation.

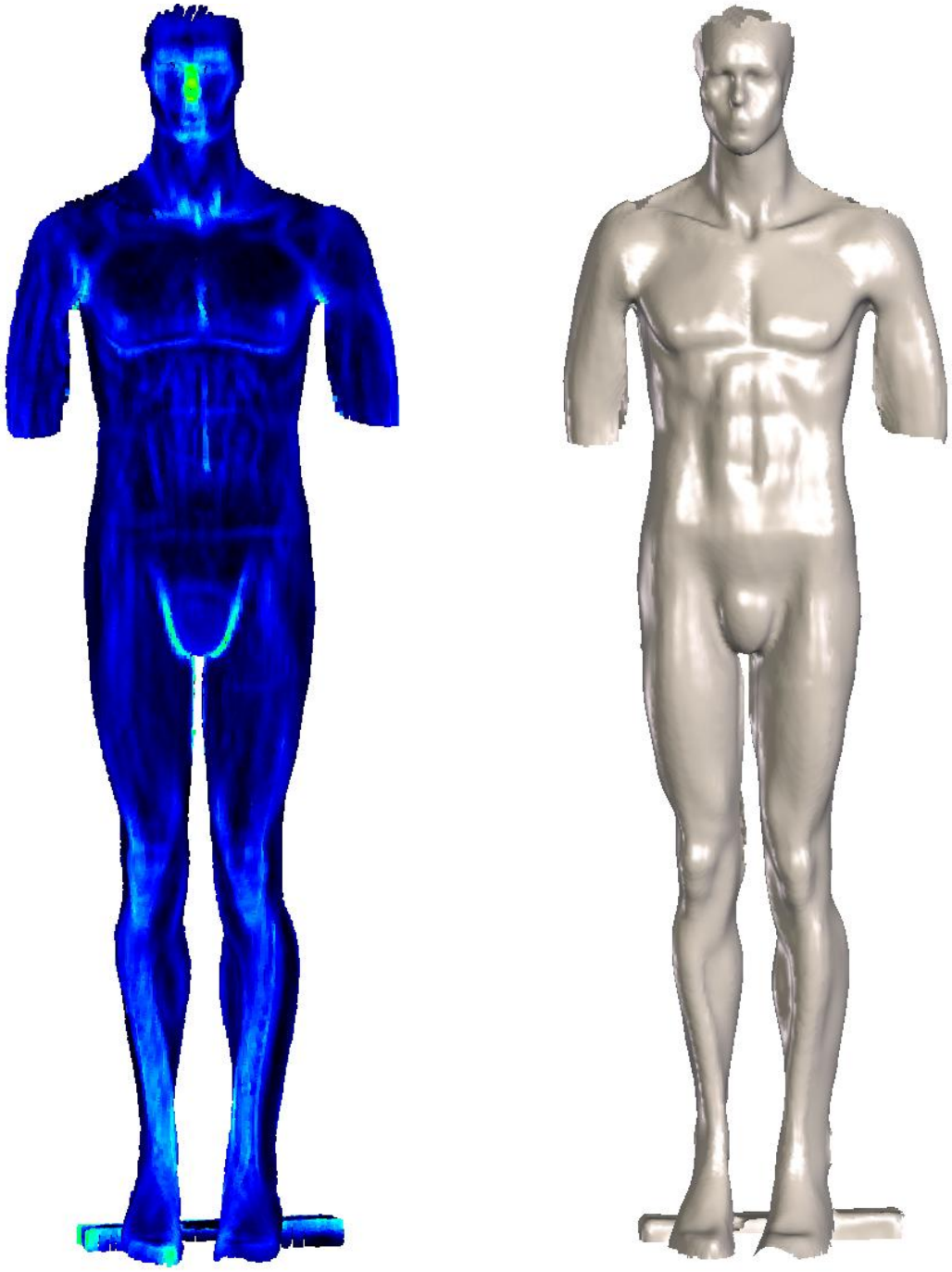

Figure 4.5: Left: The original point cloud with the point colors representing the estimated curvatures. Brighter points show points with higher curvatures. σxy is reduced in areas of high curvature in order to preserve surface details and minimize surface expansion. Right: The final rendered model.

The accuracy of the measurements taken by the measuring tools is summarized in Table 4.3 and 4.4.

Table 4.3: Measurements obtained by hand measurement versus the measurement tools when using the raw point cloud. Each hand and computer measurement was taken three times and the average was tabulated.

Measurement Location	*Hand Measurement*	*Computer Measurement*	*Difference*
Waist Circumference	*35.13”*	*34.92”*	*-.21”*
Neck Circumference	*15.03”*	*15.07”*	*+.04”*
Toe to Nose	*68.03”*	*67.96”*	*-.07”*
Support Board Depth	*3.38”*	*3.42”*	*+.04”*
Upper Leg Circumference	*21.45”*	*21.17”*	*-.28”*
Toe to Groin	*35.08”*	*35.21”*	*+.13”*

Table 4.4: Measurements obtained by hand measurement versus the measurement tools when using the processed point cloud. Each hand and computer measurement was taken three times and the average was tabulated.

Measurement Location	*Hand Measurement*	*Computer Measurement*	*Difference*
Waist Circumference	*35.13”*	*34.41”*	*-.72”*
Neck Circumference	*15.03”*	*14.95”*	*-.08”*
Toe to Nose	*68.03”*	*68.18”*	*+.15”*
Support Board Depth	*3.38”*	*3.40”*	*+.02”*
Upper Leg Circumference	*21.45”*	*21.15”*	*-.30”*
Toe to Groin	*35.08”*	*35.32”*	*+.24”*

4.4 Human Scan Results

An example scan of an actual person is given below.

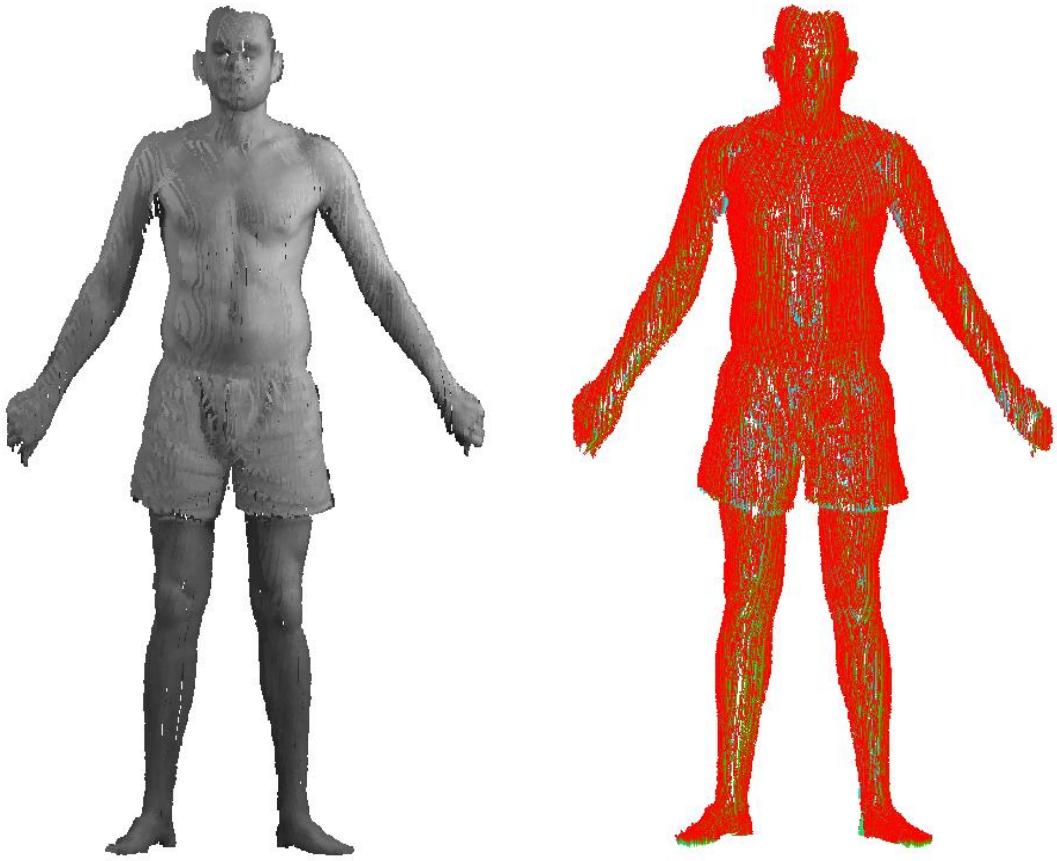

Figure 4.6: Left: Point cloud drawn with the captured intensities. Right: The points drawn as normal vectors, with the red end of the vectors facing outside. These normal vectors are used when orienting the Gaussian fields at each point during implicit surface creation.

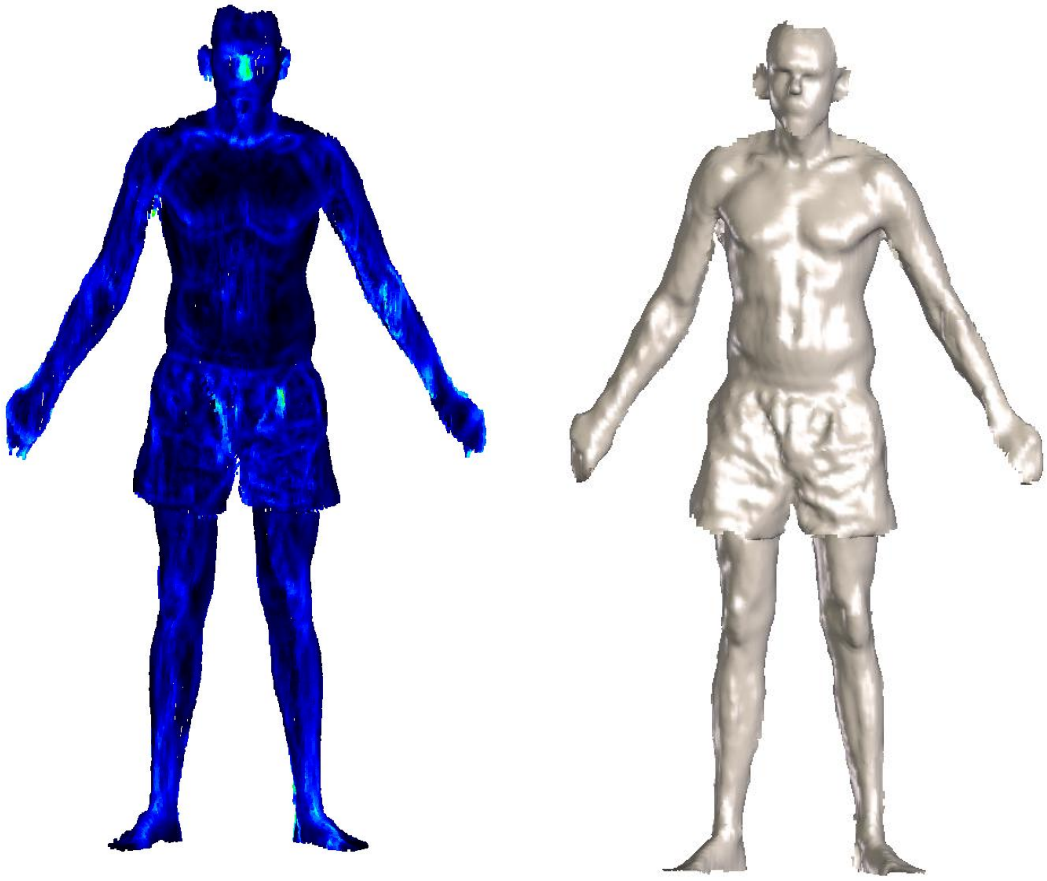

Figure 4.7: Left: The original point cloud with the point colors representing the estimated curvatures. Brighter points show points with higher curvatures. σxy is reduced in areas of high curvature in order to preserve surface details and minimize surface expansion. Right: The final rendered model.

The accuracy of the measurements taken by the measuring tools is summarized in Table 4.5 and 4.6.

Table 4.5: Measurements obtained by hand measurement versus the measurement tools when using the raw point cloud. Each hand and computer measurement was taken three times and the average was tabulated.

Measurement Location	*Hand Measurement*	*Computer Measurement*	*Difference*
Chest Circumference	*39.25”*	*38.55”*	*-.7”*
Neck Circumference	*14.77”*	*14.42”*	*-.35”*
Waist Circumference	*34.48”*	*34.86”*	*+.38”*
Neck to Waist	*18.41”*	*18.29”*	*-.12”*
Height to Nose	*65.58”*	*65.45”*	*-.13”*

Table 4.6: Measurements obtained by hand measurement versus the measurement tools when using the processed point cloud. Each hand and computer measurement was taken three times and the average was tabulated.

Measurement Location	*Hand Measurement*	*Computer Measurement*	*Difference*
Chest Circumference	*39.25”*	*38.52”*	*-.73”*
Neck Circumference	*14.77”*	*14.35”*	*-.42”*
Waist Circumference	*34.48”*	*34.69”*	*.21”*
Neck to Waist	*18.41”*	*18.09”*	*-.32”*
Height to Nose	*65.58”*	*65.676”*	*.096”*

Chapter 5
CONCLUSIONS

From the preceding results of the sphere (Tables 4.1 and 4.2), important properties of the implicit surface can be deduced. The first noticeable feature is that as noise increases, the surface expands. This can be attributed to the use of a larger σ_{xy} trying to smooth the noise. This expansion is due to the zero surface of the Gaussian applied at each point on a plane. As the σ_{xy} gets larger, this plane has more effect on its local neighborhood and draws the surface away from where the true surface should be.

Observing the tables, it can also be seen that as the point density decreases, the surface expands. This is also due to the local planar nature of the Gaussian affecting its neighbors. It should be noted that in the case of the sphere, lower density is analogous to higher curvature.

The result of measurements on the manikin shows a varying level of accuracy depending on the location and type of measurement preformed (Tables 4.3 and 4.4). When doing a point to point measurement or measuring along an ark using the raw point cloud, a high level of accuracy is achieved with measurement variations from -.07” to +.13” when compared to the hand

measurements. The measurements for the distance around a cross section varied more with variations ranging from -.28” to +.04”. Most of this variation is likely due to occlusions on the sides of the manikin and in-between the legs. This would also explain the reason for the large differences being negative.

When measurements are performed using the implicit surface obtained from the point cloud, all measurements, with the exception of the support board depth, became less accurate. This could be due to a few different reasons: First, as discussed earlier, the implicit surface can cause some expansion of the model on curved surfaces. Similarly, when the implicit surface fills the holes, it will do so with a flat plane, causing curved surfaces that abruptly end on a curved surface to expand greatly.

A similar set of measurements was also preformed on a human subject using both the raw point cloud and the implicit surface points. Similar to the previous results, the point-to-point and arc-length results were very accurate with a maximum disparity of only -.13”. The circumference of the waist and chest also followed the same pattern varying from -.70” to +.38”. This is a wider variance than was seen on the manikin, which could be caused by a few factors. First, human subjects breathe during the scan, causing both the chest and waist measurements to change. Similarly, they may also sway between front and back scans, which will have the effect of making the measurements larger or smaller. Lastly, unlike the manikin, the tape measure causes skin, fat, and muscle tissue to compress slightly. This, along with occlusion, is likely the reason for the chest

circumference being smaller when measured by hand than when measured by the computer.

From the preceding data, it can be concluded that use of implicit surface for measuring is unnecessary and in some cases may actually impair correct measurements. When the scanner is not configured correctly or the subject being scanned moves, the implicit surface may help to bring together false double surfaces and improve measurements results. Regardless of its use in measuring the subject, the implicit surface is a powerful tool for visual inspection of the scanned result and for cleaning and compressing the model for storage.

Chapter 6
FUTURE WORK

Using the design described in this thesis, it would be possible to build a large variety of scanners by changing the size and shape of the reference frames. These frames could be used interchangeably and manufactured at a relatively low cost. Some frame shapes, such as a triangle or square, have the potential for better scans of the whole body due to the equal angles and side lengths. This would allow for better coverage on the edges and, unlike the rectangle frame, would allow more even spacing of the cameras and the lasers around the frame and a more consistent accuracy of gathered range points. See Figures 6.1 and 6.2 for an example layout.

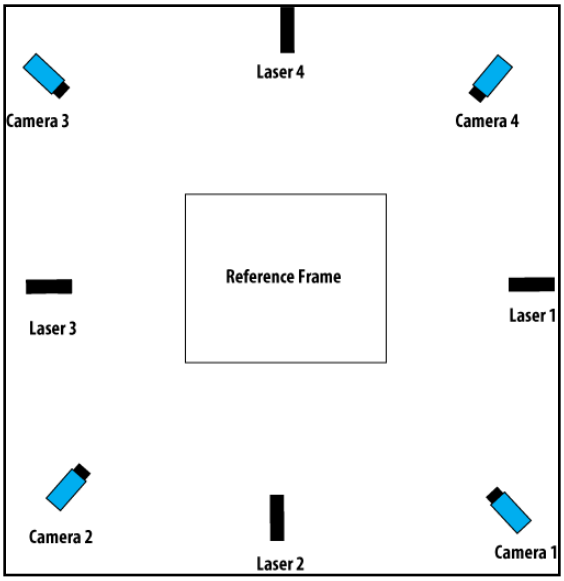

Figure 6.1: A top view of a layout of a four-camera and four-laser scanner.

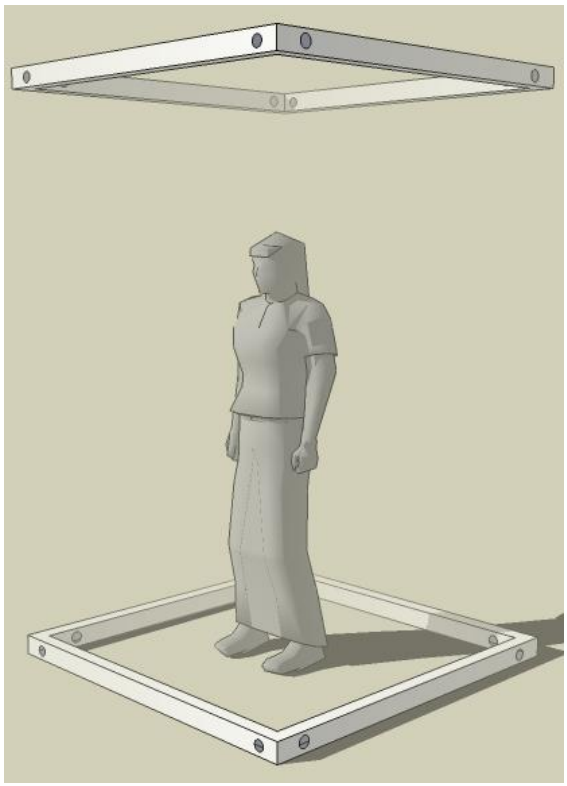

Figure 6.2: A 3-D rendering of the above layout.

To yield a better final model, the implicit surface formulation could also be altered to take into account different measures of confidence from the scan. For instance, once a scan is complete and the normals have been calculated, it would be possible to approximate the angle at which the laser struck the surface at that point. As this angle increases, the accuracy would decrease and, therefore, the weight of the point should be decreased when applying the Gaussian to the implicit field. The end result would be that lower-confidence points could still be taken into account, but they would have far less effect and introduce far less distortion when neighbored by high-confidence points.

Along with adjustments in the frame configuration and surface formulation, another area of the scanner that could be changed to yield substantial improvements is the cameras. The current implementation of the scanner uses Prosilica IEEE 1394a cameras with a maximum resolution of 1024x1280. The overall accuracy of the scanner is limited by the resolution of the camera used, and moving to a higher resolution would certainly raise the accuracy of the scanner. One important point to note, however, is that although there are higher resolution cameras available with the common IEEE 1394a interface, the bandwidth constraints keep them from being a viable option for use in a human body scanner as the frame rate of the cameras would be too low, resulting in a long scan where the subject would be likely to move. To solve this issue, faster technologies, which require special hardware such as Camera-Link, could be used or cheaper and more readily available standards IEEE 1394b or Gig-E could be used. In the near future, USB 3.0 will likely provide a cheaper, faster,

and more readily available solution for computer to camera interface than what is currently available on the market today.

Bibliography

1. *About the Body Scanner.* Cornell University College of Human Ecology. [Online] 2006. [Cited: May 2, 2009.] http://www.bodyscan.human.cornell.edu/scene60df.html.

2. Zagorchev, Lyubomir and Goshtasby, Ardeshir. A paintbrush laser range scanner. *Computer Vision and Image Understanding,* 2006, Vol. 101, No. 2.

3. Besl, P. J. Active optical range imaging sensors. *Machine Vision and Applications,* 1988, Vol. 1, No. 2, pp. 127-152.

4. Ullrich, A., et al. Long-range high-performance time-of-flight-based 3-D imaging sensors. *3-D Data Processing Visualization and Transmission,* 2002, pp. 852-855.

5. Automatic body measurement tracking for apparel, athletics, health/fitness, and medical applications. TC2. [Online] [Cited: May 7, 2009.] http://www.tc2.com/products/body_scanner.html.

6. Millimeter wave whole body imaging. Tranportation Security Administration. [Online] [Cited: May 3, 2009.] http://www.tsa.gov/approach/tech/mwave.shtm.

7. O'Neill, S. Ban "whole-body" TSA scanners? *newsweek.com.* [Online] May 18, 2009. [Cited: May 14, 2009.] http://current.newsweek.com/budgettravel/2009/05/activists_are_calling_to_ban_w.html.

8. Goshtasby, A. Registration of images with geometric distortions. *IEEE Transactions,* 1988, Vol. 26, pp. 60-64.

9. Bourke, P. Intersection of a plane and a line. [Online] August 1991. [Cited: May 10, 2009.] http://local.wasp.uwa.edu.au/~pbourke/geometry/planeline/.

10. Blinn, J. F. A generalization of algebraic surface drawing. ACM Transactions on Graphics, 1982, Vol. 1, no. 3, pp. 235–256.

11. Muraki, S. Volumetric shape description of range data using Blobby Models. Proc. SIGGRAPH, 1991, pp. 227–235.

12. Nishimura, H., et al. Object modeling by distribution function and a method of image generation. Trans. Inst. Elect. Commun. Eng. Japan, 1985, vol. 4, pp. 718–725.

13. Wyvill, G., et al. Data structures for soft objects. The Visual Computer, 2000, vol. 80, pp. 295–314.

14. Hoppe, H., et al. Surface reconstruction from unorganized points. *Computer Graphics,* 1992, Vol. 26, pp. 71-78.

15. Marquardt, D. An algorithm for least-squares estimation of nonlinear parameters. *SIAM Journal on Applied Mathematics,* 1963, Vol. 11, pp. 431-441.

16. Cline, H. E. and Lorenson, W. E. Marching cubes: A high resolution 3-D surface construction algorithm. *Computer Graphics,* 1987, Vol. 21, No. 4, pp. 132-133.

17. Grahm, R. L. An efficient algorithm for determining the convex hull of a finite planar set. *Information Processing Letters,* 1972, Vol. 1, pp. 132-133.