BODY-CENTERED VISUALISATION FOR FREEHAND 3D ULTRASOUND

P. M. Tuomola, A. H. Gee,
R. W. Prager and L. Berman

CUED/F-INFENG/TR 362

October 1999

University of Cambridge
Department of Engineering
Trumpington Street
Cambridge CB2 1PZ
England

Email: ahg/rwp @eng.cam.ac.uk, lb@radiol.cam.ac.uk

Body-Centered Visualisation for Freehand 3D Ultrasound

Petri Tuomola, Andrew Gee, Richard Prager and Laurence Berman*
University of Cambridge

Department of Engineering
Trumpington Street
Cambridge CB2 1PZ

*Department of Radiology
Addenbrooke's Hospital
Cambridge CB2 2QQ

Abstract

3D ultrasound data is typically visualised by any-plane slicing, volume rendering or surface rendering. None of these techniques can readily convey the spatial relationship between the visualised data and the patient's body, something which is particularly important when the data is reviewed after the scan has taken place, perhaps by a remote expert who did not even perform the scan. This paper describes a facility to register the 3D ultrasound data to the patient's body and then display the data correctly superimposed on a rendered mannequin. This way, the user can appreciate the position and orientation of any visualisation with respect to the patient's body. The facility relies on efficient implementation of progressive meshes to manage the level-of-detail of the mannequin model.

1 Introduction

Conventional diagnostic ultrasound imaging is performed with a hand-held probe which transmits ultrasound pulses into the body and receives the echoes. The magnitude and timing of the echoes are used to create a 2D greyscale image (B-scan) of a cross-section of the body in the scan plane.

Using a technique called **freehand 3D ultrasound imaging** [18, 21], it is possible to construct 3D data sets from a series of 2D B-scans — see Figure 1. A conventional 3D freehand examination can be broken into three stages: scanning, reconstruction and visualisation. Before scanning, some sort of position sensor is attached to the probe. This is typically the receiver of an electromagnetic position sensor [5, 11, etc.], as illustrated in Figure 1, although alternatives include acoustic spark gaps [12], mechanical arms [13] and optical sensors [23]. Measurements from the position sensor are used to determine the positions and orientations of the B-scans with respect to a fixed datum, usually the transmitter of the electromagnetic position sensor. In the next stage, the set of acquired B-scans and their relative positions are used to fill a regular voxel array. Finally, this voxel array is visualised using, for example, any-plane slicing, volume rendering or surface rendering (after segmentation).

Recently, we have proposed an alternative approach to freehand 3D ultrasound, which bypasses the voxel array stage. In **sequential** freehand 3D ultrasound, the data is visu-

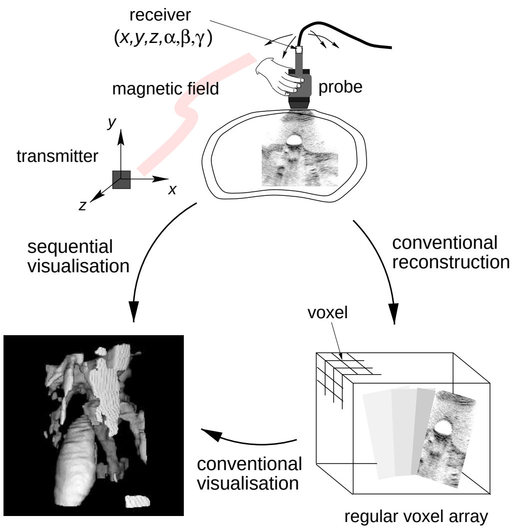


Figure 1: **Freehand 3D ultrasound imaging**. The conventional approach is a three-stage process, involving scanning, reconstruction and visualisation. The alternative, sequential approach allows visualisation directly from the raw B-scans and positions. The figure illustrates an examination of a gall bladder.

alised and analysed directly from the raw B-scans and positions — see Figure 1. Any-plane slicing, non-planar slicing, panoramic imaging, volume estimation and surface rendering can be performed without the use of an intermediate voxel representation [7, 14, 15, 16, 22]: all of these facilities are implemented in the Stradx freehand 3D ultrasound system[1]. The sequential approach offers several advantages:

- When reslicing, the data is resampled only once, from the B-scan pixels to the slice pixels. The conventional approach requires two resampling stages, from the B-scan pixels to the voxel array, then from the voxel array to the slice pixels. Since resampling usually involves data approximation, more accurate visualisation is possible by avoiding one resampling process.

- Reslicing can be performed at the full resolution of the B-scans, without the significant memory overhead of a high resolution voxel array.

- Visualisation and data analysis can be performed in real time, as the data is being acquired, since the sequential visualisation and volume measurement algorithms reference each B-scan only once, in the order in which they are acquired.

- Segmentation (for volume measurement and surface rendering) is performed on the B-scans themselves, instead of parallel slices through the voxel array. The B-scans

[1]The software can be downloaded from `http://svr-www.eng.cam.ac.uk/~rwp/stradx/`.

are high resolution and exhibit no reconstruction artifacts, making them relatively easy to interpret for manual or assisted segmentation. The same cannot be said of slices through the voxel array.

A shortcoming of both the conventional and sequential approaches is that the data is visualised with respect to an arbitrary datum: there is no way of knowing where the data came from with respect to the patient's body. For example, Figure 2 shows a typical obstetrics visualisation example using the Stradx system. The 'Review' window shows a selected B-scan, onto which the user has drawn two segmentation contours; the 'Reslice' window shows a particular planar reslice of the data set; the 'Manifold' window shows a particular non-planar reslice of the data set; while the 'Outline' window shows the outlines of all the recorded B-scans and also the positions of the segmentation contours, reslice plane and non-planar reslice surface. The user can manipulate the viewpoint of the 'Outline' window interactively, thus obtaining a better impression of the various structures' relative spatial locations. What is totally lacking is any sense of orientation with respect to the mother's body. This is particularly important in telemedicine applications, where the data may be reviewed by a remote expert who did not perform the scan. However, telemedicine is not the only area of interest. 3D ultrasound opens up a wealth of possibilities for reporting on scans *after* the patient has left the clinic (a practice which is common with other medical imaging modalities), but it would be necessary for the reporter to somehow visualise the data with respect to the patient's body.

This report describes a recent enhancement of Stradx to allow body-centered visualisation. This is achieved by superimposing the objects in the 'Outline' window on top of a rendered mannequin. This way, the user can appreciate the position and orientation of any visualisation with respect to the patient's body. The major technical challenge to be overcome concerns level-of-detail control: the mesh models of the mannequin need to be detailed enough to cope with a scan of, say, a single finger, but also compact enough to allow rendering of, say, an entire torso at interactive rates. We overcome this technical challenge using the relatively new computer graphics technique of **progressive meshes**, which is the main focus of Section 3.

The other major issue involves determining the position and orientation of the patient's body with respect to the coordinate system of the position sensor. Without this **registration** process, it would not be possible to display the mannequin in the right place with respect to the 3D ultrasound data. In Section 4, we describe a suitable registration procedure which can be performed in a few seconds shortly before or after the clinician acquires the ultrasound data.

To provide a suitable context for the subsequent details, we present a broad overview of the entire system in Section 2. The paper includes some illustrative results in Section 5 and concludes with some pointers towards future work in Section 6.

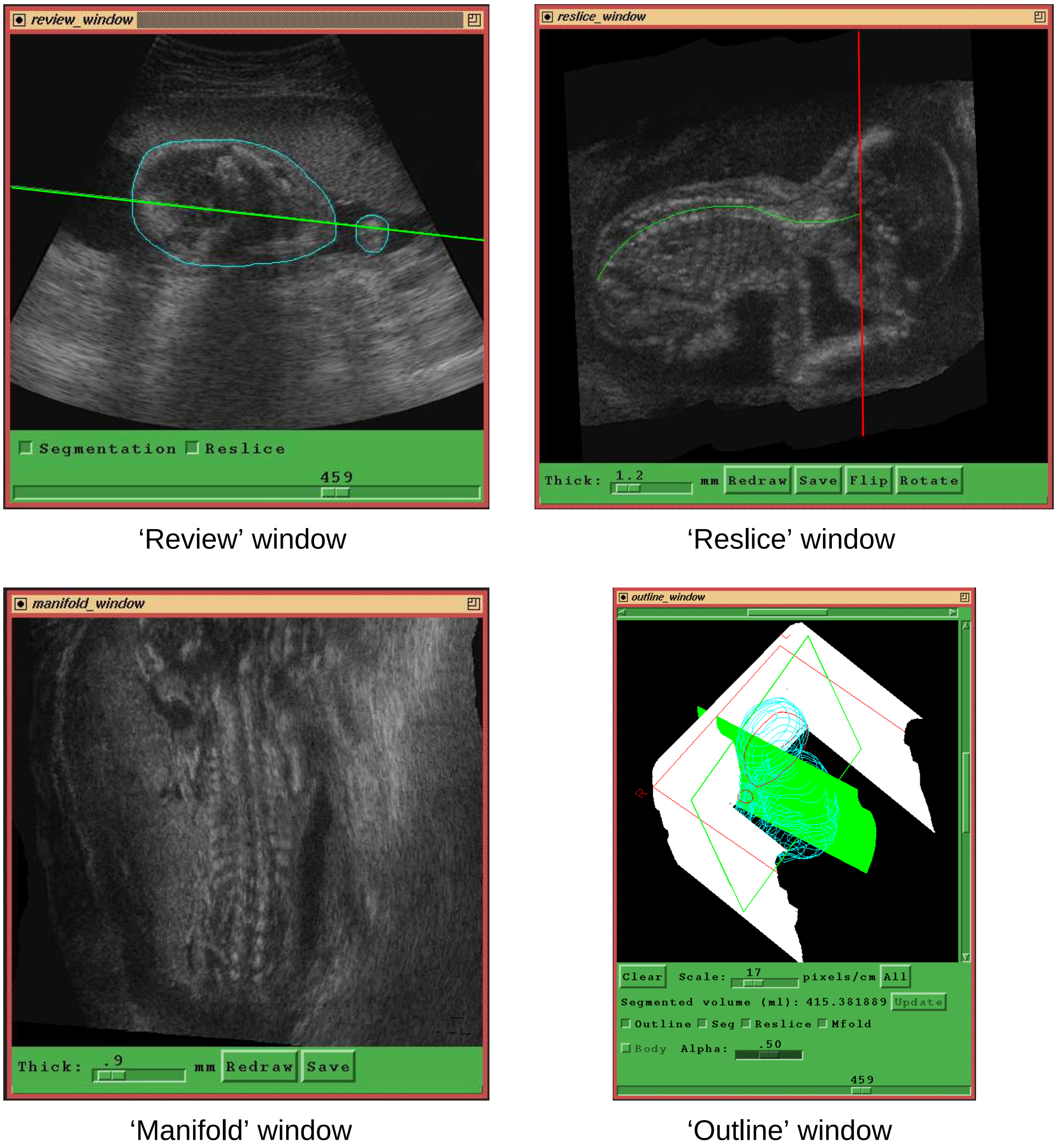


Figure 2: **3D ultrasound visualisation using Stradx**. This example shows an examination of a 22-week foetus. The 'Review' window shows a selected B-scan, onto which the user has drawn two segmentation contours. The 'Reslice' window shows a planar reslice, which intersects the selected B-scan along the straight line in the 'Review' window. The vertical line in the 'Reslice' window corresponds to the B-scan in the 'Review' window. The user has drawn a non-planar reslice contour onto the 'Reslice' window. The 'Manifold' window shows the resulting non-planar reslice, which runs along the foetus' spine. The 'Outline' window shows the outlines of all the recorded B-scans and also the positions of the segmentation contours, reslice plane and non-planar reslice surface.

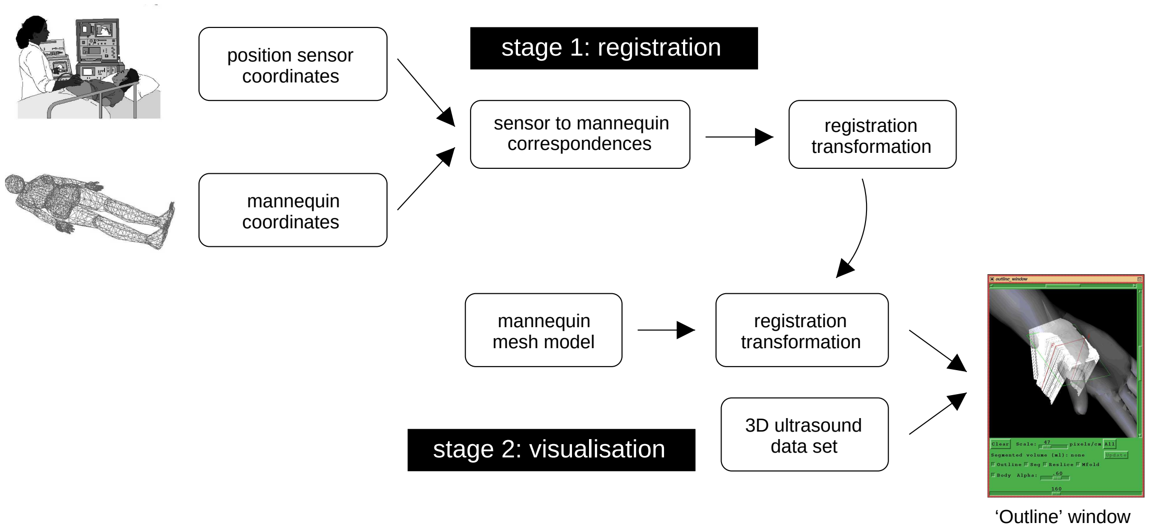


Figure 3: **System overview**. The body-centered visualisation facility involves two stages. A **registration** process determines the transformation between the mannequin model and the position sensor coordinate systems. The **visualisation** process uses this transformation to display the mannequin in the correct position relative to the 3D ultrasound data set.

2 System overview

An overview of the body-centered visualisation facility is shown in Figure 3. The registration stage proceeds as follows. With the patient in the same position as during the scan, the clinician moves the ultrasound probe to a location on the patient's body: the coordinates of the probe are recorded from the position sensor. The clinician then points out the same location on the mannequin displayed on the computer screen: the position of that point is also recorded. The process is repeated to record enough correspondences to accurately calculate the transformation between the position sensor's coordinate frame and the mannequin model's coordinate frame. Since the 3D ultrasound data is defined in position sensor coordinates, knowledge of this transformation allows the mannequin and 3D ultrasound data to be rendered in the same display.

In the subsequent visualisation stage, the registration transformation is used to display the mannequin in Stradx's 'Outline' window. Since the 'Outline' window also displays the outlines of the recorded B-scans, along with any segmentation contours, reslice planes and non-planar reslice surfaces, the position and orientation of the 3D data set is readily observed with respect to the patient's body.

3 Multi-resolution rendering of mannequin models

3D mannequin models are displayed in the facility in two places. The first is during the registration process, so that the clinician can indicate the model locations corresponding to the position sensor readings. The second is in the 'Outline' window, to show the position and orientation of the 3D ultrasound data with respect to the patient's body. In both instances, we need to be able to rotate, zoom and translate these models at interactive rates: displaying the mannequin should not slow down visualisation of the data or delay the registration process.

The wide variety of possible scans makes the display of the mannequin a difficult task. 3D ultrasound can be used to scan anything from parts of fingers to whole limbs, and the system needs to be able to display a mannequin mesh with sufficient detail whatever the application. This would not be a problem given a computing platform endowed with infinite processing power and memory. However, Stradx needs to run on standard, modestly-priced computer systems with good but limited graphics hardware: this restricts the size of the mesh models that can be rendered at interactive rates. Also, since most of the computer's memory is required for storing the ultrasound data, the amount of memory for storing and processing the model data should be reduced as much as possible.

Stradx's body-centered visualisation facility therefore employs a two-stage level-of-detail (LOD) control. First, a *discrete* LOD is applied: the user is requested to select the scanned body parts before the registration process is initiated. Secondly, a *continuous* LOD is applied to render the selected body parts at a resolution appropriate for the computing platform's graphics capabilities. The continuous LOD is achieved through the use of progressive meshes.

3.1 Selecting the scanning area

The user interface for selection of the scanning area, illustrated in Figure 4, draws on techniques popular in the implementation of HTML image maps. A map editor is used to divide a bitmap image of a human body into several active zones, each corresponding to a different body part. The user can toggle inclusion of body parts with the mouse. The shaded body parts indicate the currently selected scanning area. Any combination of body parts can be selected.

Explicit specification of the scanning area has two key benefits. First, it reduces memory and computational overheads by allowing the application to discard the model data related to unscanned body parts. Secondly, it facilitates visualisation of the mannequin model, since body parts outside the scan region would otherwise clutter the display.

3.2 Mannequin mesh models

The mannequin model used in the facility is described in the form of a triangular mesh approximation. The mesh comprises a set of vertices and a set of faces. Each vertex defines

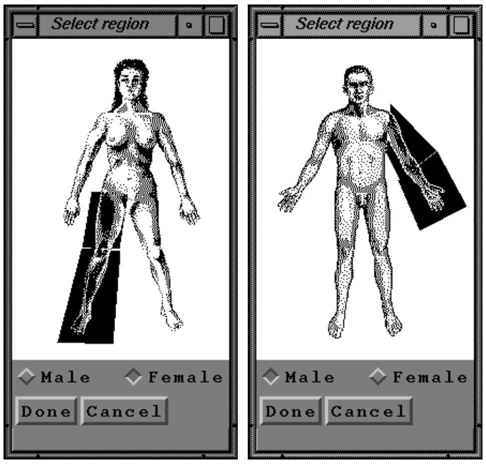


Figure 4: **User interface for selection of the scanning area.** Any combination of body parts can be selected on male or female models.

the coordinates (x, y, z) of a point in space and each face defines a triangle through ordered listing of the vertices that are its corners. A mesh can be stored efficiently as:

$$V = \{(x_1, y_1, z_1), (x_2, y_2, z_2) \ldots (x_n, y_n, z_n)\}$$
$$F = \{(i_0^0, i_1^0, i_2^0), (i_0^1, i_1^1, i_2^1) \ldots\}$$

where V is the list of vertices, F is the list of faces and i is an index to V

Obtaining suitable mannequin models is potentially problematic. Both male and female models are needed and they have to be anatomically 'neutral': not overweight or too muscular. In other words, models of 'perfectly average' humans are required. Fortunately, suitable mannequin models are available on the internet — see Figure 5.

The models are by no means perfect. They represent only one body pose and there is no facility for changing any of the body dimensions apart from the overall scale. It is therefore likely that the model does not perfectly reflect the body of the scanned patient. However, the model suffices to indicate the position and orientation of a 3D ultrasound scan with respect to the patient. The exception is when the scan spans an articulated joint, and the mannequin joint is not in the same position as the patient's. This suggests scope for further research into dynamic model generation, a topic we shall return to in Section 6.

The meshes were clipped into submeshes corresponding to the selectable body parts (Figure 4) using facilities within the Visualisation Toolkit (VTK) [19]. Care was taken to ensure that the boundaries of the partial meshes were consistent, so that two adjacent partial meshes, when displayed together, join seamlessly.

3.3 Mesh decimation

The mannequin models are highly detailed: the male model comprises 67724 triangles, while the female model is more detailed with 253241 triangles. Even clipped models for

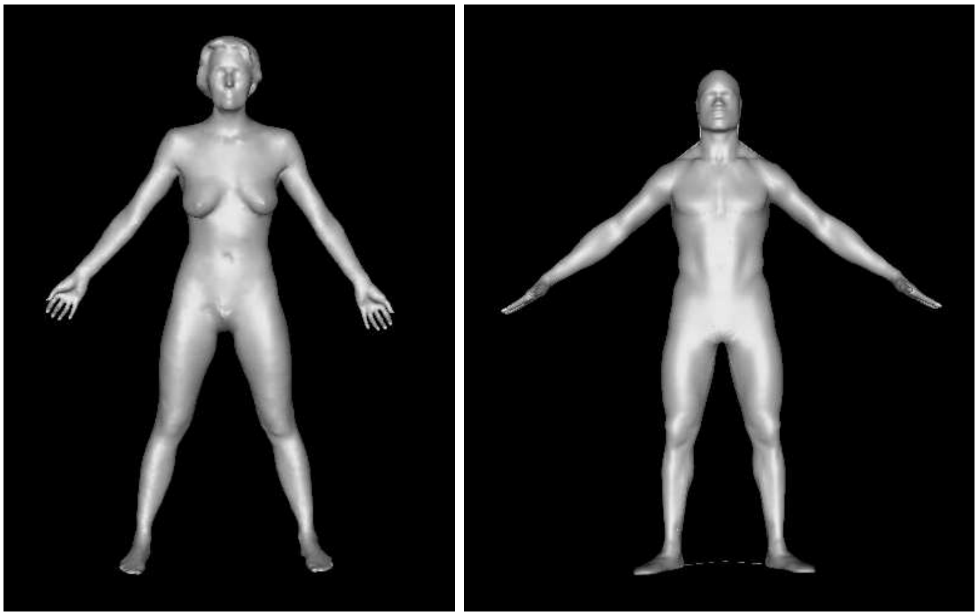


Figure 5: **Male and female mannequin models.** We are grateful to Paraform Inc., Santa Clara, California, USA for the male model, and Headus (Metamorphosis), Osborne Park, Australia for the female model.

individual body parts contain up to 10000 polygons. In general, models built from various combinations of body parts are too complex to be rendered at interactive rates on standard computer hardware, and therefore need to be simplified significantly.

Polygonal meshes can be simplified by **decimation** [3]. The aim is to produce a mesh M_d that approximates the surfaces present in the original mesh M_n, so that $f_d < f_n$, where f is the number of faces in a mesh. Most decimation algorithms [2, 4, 6, 10, 20] define an error metric E that can be used to evaluate the quality of an approximation. What varies between different algorithms is the technique used to generate the candidate approximations.

We opted for the decimation algorithm described in [20], since a public domain implementation (including the source code) is readily available as part of VTK [19]. A slightly restricted version of the algorithm can also be embedded in a progressive mesh framework for dynamic LOD control — see Section 3.4. The restricted algorithm operates as follows:

1. Each vertex of the input mesh is classified according to the edges it lies on. Edges can be:

   Feature edges: edges which border triangles whose dihedral angle is greater than a specified feature angle.

   Boundaries: edges which belong to only one triangle.

   Non-manifold edges: edges which belong to more than two triangles.

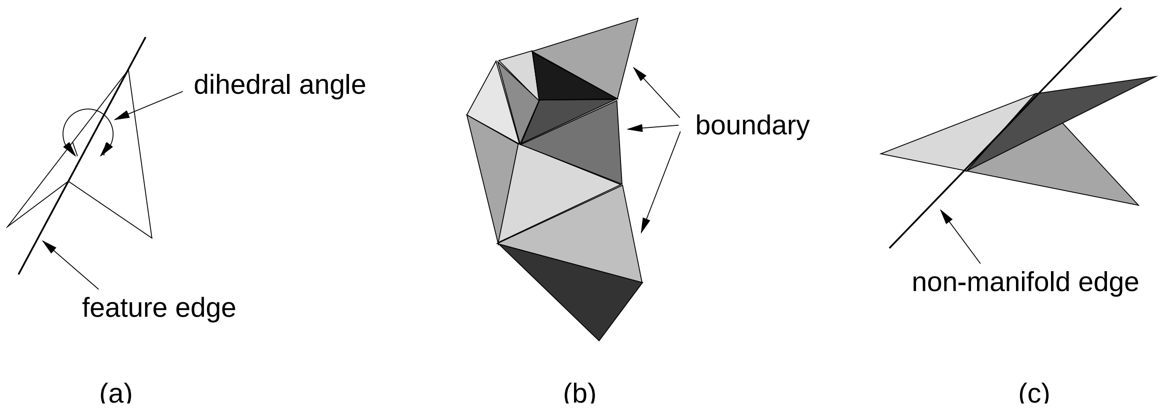


Figure 6: **Vertex classification:** (a) a feature edge, (b) boundary edges and (c) a non-manifold edge.

The different types of edges are illustrated in Figure 6.

2. Taking one vertex at a time, the error introduced by deleting each vertex is determined. Only manifold, non-boundary and non-feature vertices are considered. The error metric is a relative distance measure of the vertex to an average plane calculated from the surrounding triangles — see Figure 7(a).

3. Vertices are inserted into a priority queue[2] with their priorities dependent on the error metric calculated in step 2.

4. Once all eligible vertices have been inserted into the priority queue, the queue is processed. The vertex at the head of the queue is removed from the mesh and the hole introduced is retriangulated. Retriangulation takes place through **edge collapse** — see Figure 7(b). Step 4 is repeated until the queue is empty or the mesh has attained the desired number of faces.

The full version of the algorithm can decimate a mesh to any number of triangles by splitting the mesh along feature edges, deleting boundary vertices and modifying the mesh topology (by deleting non-manifold vertices). For the mannequin application, we opted for the restricted version, described above, for a number of reasons. First, boundary vertices need to be preserved, to ensure compatibility between the body part meshes at their boundaries. Secondly, allowing mesh splitting would complicate the implementation of progressive meshes, as described in Section 3.4. The decimation was therefore performed using edge collapse alone. Despite this restriction, the algorithm was found to be capable of reducing all the body part meshes to a sufficiently low level of detail (500–1000 triangles).

[2]A priority queue is a list in which the items are ordered according to their priority. The item with highest priority (lowest error) will always be at the head of the list.

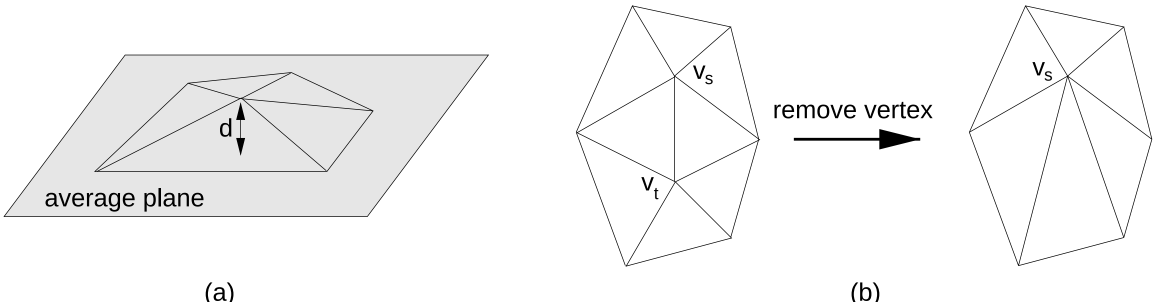


Figure 7: **Decimation:** (a) evaluation of local approximation error and (b) retriangulation through edge collapse.

3.4 Continuous LOD control

While the decimation algorithm described in Section 3.3 can produce body part meshes at almost any LOD, it remains to select the appropriate LOD for the application. The worst case scenario would involve all body parts being rendered simultaneously on a slow computing platform. One solution would be to decimate all meshes sufficiently to cope with this worst case scenario.

Clearly, this solution is not optimal. With meshes decimated so much, choosing a small scanning area (such as one finger) would lead to a body part model which is probably too coarse to be of any use. It follows that the LOD needs to dynamically reflect the viewing conditions. When the user selects a smaller scanning area, the resolution of the model should increase. In other words, *the number of triangles displayed should be kept constant as the scanning area is varied.* The appropriate constant can be tuned interactively by the user to match the computing platform's graphics capabilities.

One trivial implementation would involve decimating the model on-the-fly to the required resolution every time the scanning area is changed. However, the decimation algorithm is computationally expensive and requires sufficient memory to store the entire detailed mesh M_n. On-the-fly decimation could lead to a delay of several minutes between selection of the scanning area and display of the corresponding model. In this application, such a delay is unacceptable.

Another possibility would be to store each mesh decimated to various resolutions $M_0, M_1, M_2 \ldots M_n$. An appropriate mesh could then be chosen depending on the size of the scanning area and the speed of the graphics hardware. However, this would lead to excessive use of disk space, as several copies of the same mesh would need to be stored. Choosing the appropriate resolutions to work with would also be problematic.

Instead, we opt for a progressive mesh approach. Progressive meshes, introduced in [8], offer a technique for performing decimation very quickly and efficiently, with minimal memory and disk overhead. The underlying principle is simple. A detailed mesh M_n is stored as a much coarser mesh M_0, produced by decimating M_n, together with a sequence of n **vertex split** records. Each of these records describes an operation that adds one

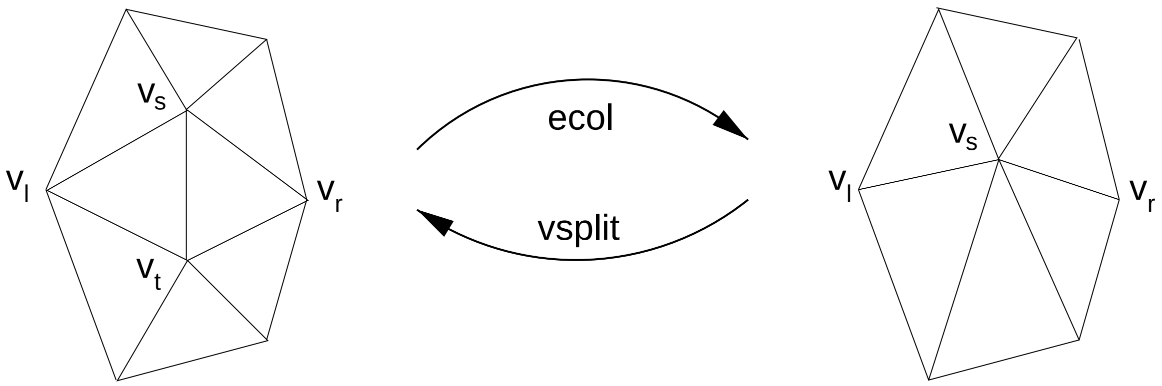


Figure 8: **Edge collapse and vertex split.** The edge collapse (ecol) operation is used by the decimation algorithm to simplify the detailed mesh. The inverse operation, vertex split (vsplit), is central to the implementation of progressive meshes.

vertex to the mesh. The coarse mesh, together with the vertex split records, defines a continuous sequence of meshes $M_0, M_1, M_2 \ldots M_n$ with increasing resolution. A mesh with any level of detail can be generated by applying r vertex splits, $0 \leq r \leq n$, to M_0. The original mesh M_n can be restored by applying all the vertex splits.

The vertex split records are generated using the edge collapse decimation algorithm described in Section 3.3. Consider decimating the mesh M_n by collapsing one edge at a time:

$$M_n \overset{\mathsf{ecol}_{n-1}}{\Rightarrow} \ldots \overset{\mathsf{ecol}_1}{\Rightarrow} M_1 \overset{\mathsf{ecol}_0}{\Rightarrow} M_0$$

$\mathsf{ecol}(s,t)$ is an edge collapse operation. Its effect is to remove vertex v_t from the mesh by collapsing the edge $v_s v_t$ to the vertex v_s — see Figure 8. As a result, two faces $\{v_t, v_s, v_l\}$ and $\{v_t, v_r, v_s\}$ vanish.

The edge collapse operation is invertible: $\mathsf{vsplit}_m = (\mathsf{ecol}_m)^{-1}$, where $\mathsf{vsplit}(l,r,s,t)$ is a vertex split operation — see Figure 8. v_s is replaced by two new points v_s and v_t. In the process, two new faces $\{v_t, v_s, v_l\}$ and $\{v_t, v_r, v_s\}$ are introduced. The mesh is restored to its state prior to the application of ecol.

Vertex split forms the basis of the progressive mesh representation:

$$M_0 \overset{\mathsf{vsplit}_0}{\Rightarrow} M_1 \overset{\mathsf{vsplit}_1}{\Rightarrow} \ldots \overset{\mathsf{vsplit}_{n-1}}{\Rightarrow} M_n$$

A progressive mesh system can be implemented by storing information about every edge collapse performed in the decimation process. When a more detailed mesh is required, the corresponding vertex splits are applied to the decimated mesh. The order of vertex splits is such that the *final* edge collapse applied in the decimation process is reversed *first* in the refinement process.

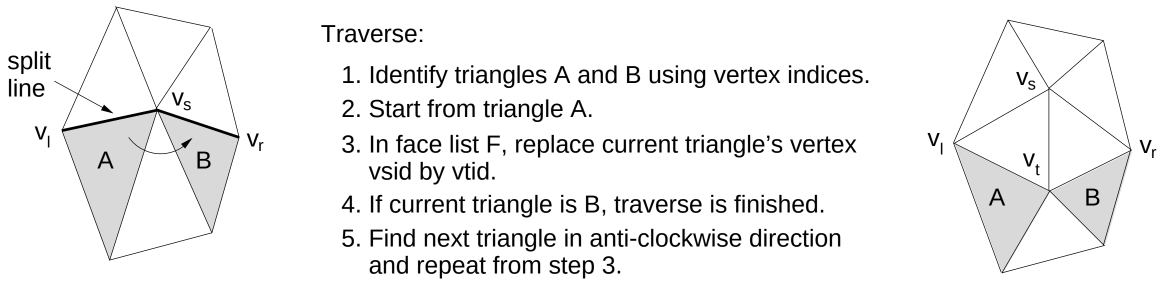


Figure 9: **Traverse stage of the vsplit operation.** The traverse finds all the triangles which need to be modified as a consequence of the insertion of vertex v_t.

3.5 Progressive mesh implementation

For clarity, we first describe a basic implementation of progressive meshes, before going on to describe several refinements which make the implementation more efficient.

Vertex split operations can be stored in the following data structure:

```
struct _vsplit {
     float vtx, vty, vtz;    /* Coordinates of the new vertex  */
     float ntx, nty, ntz;    /* Unit normal for the new vertex */
     int vtid;               /* Index of the new vertex created */
     int vsid;               /* Index of the vertex to be split */
     int vlid, vrid;         /* Points identifying the split line */
     struct _vsplit *next;   /* Pointer to next split record */
};
```

There is no requirement in the generic progressive mesh paradigm for the vertex to be split, v_s, to coincide with either end of the edge created, $v_s v_t$. However, the particular decimation algorithm employed in this work collapses the edges to one of the end points, not to an arbitrary point in the middle. This reduces the amount of information we need to store, since only the coordinates of the new vertex v_t need to be defined in the vertex split record.

One vertex split record is generated for each edge collapse performed in the decimation. These are joined together to form a linked list of vertex splits. Any number of vertex splits can then be 'played back', reversing a part of the decimation process. Applying a vertex split involves the following steps:

1. Create a new vertex with index `vtid`, coordinates `(vtx,vty,vtz)` and vertex normal `(ntx,nty,ntz)`.

2. Create two new faces $\{\texttt{vtid}, \texttt{vlid}, \texttt{vsid}\}$ and $\{\texttt{vtid}, \texttt{vsid}, \texttt{vrid}\}$.

3. Start from the triangle $\{x, \texttt{vsid}, \texttt{vlid}\}$ just below the left half of the split line (edge $v_l v_s$): this is triangle A in Figure 9. Traverse around `vsid` in an anti-clockwise

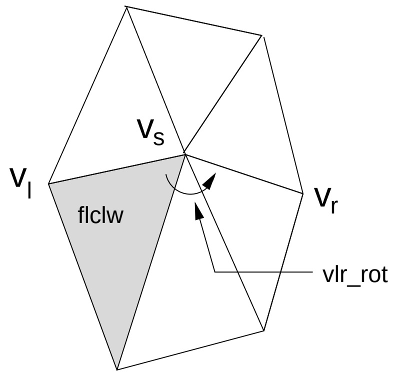


Figure 10: **Parameters of the efficient vsplit implementation.** The traverse starts from triangle `flclw` and visits `vlr_rot` adjoining triangles in an anti-clockwise direction.

> direction until the triangle $\{y, \texttt{vrid}, \texttt{vsid}\}$, just below the right half of the split line (edge $v_r v_s$), is reached: this is triangle B in Figure 9. On the way, replace all triangles $\{a, b, \texttt{vsid}\}$ with $\{a, b, \texttt{vtid}\}$.

The traverse will always be possible, since the restricted decimation algorithm does not delete boundary vertices. If v_s was a boundary vertex, one of the triangles below the split line could be missing, causing the traverse to fail.

An efficient progressive mesh implementation [9] precalculates as much information as possible, so the vertex splits can be applied rapidly. The first optimisation involves the search for a triangle's neighbours, which is required in the vertex split traverse operation and is potentially expensive. To mitigate this expense, each triangle's three neighbours (one for each edge) are precomputed and stored in the mesh file. Note that this approach disallows non-manifold constructs. In a non-manifold case, a triangle would have more than one neighbour for a given edge and could not be stored in this format. While some of the body part meshes did originally contain non-manifold edges (associated with thin protrusions like fingernails), these were easily removed by manually deleting the appropriate triangles.

The second optimisation involves storing more precomputed information in the vertex split records. Identifying the start and end points of the traverse from the vertex indices `vlid` and `vrid` is costly. Instead, we store the index `flclw` of the triangle from which the traverse should start, and the number of triangles `vlr_rot` which need traversing — see Figure 10. Also, storing the index of the new point created, `vtid`, is unnecessary. By renumbering the vertices in the coarse mesh M_0 and in the vertex split records to run from 1 to N, new vertices can be numbered in the order in which they are created. This removes the need to explicitly define `vtid`.

What emerges is an improved implementation which uses the following data structure for vertex split records:

```
struct _vsplit {
```

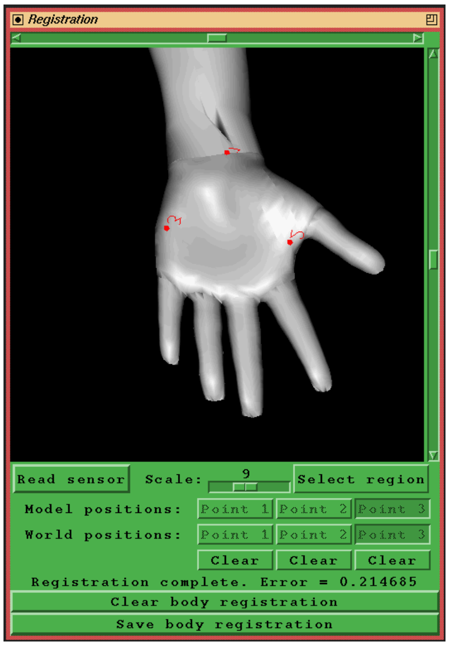


Figure 11: **User interface for the registration facility.** The selected body parts can be rotated and scaled until the required landmark is clearly visible: the user then indicates the landmark by clicking with the mouse.

```
    float vtx, vty, vtz;      /* Coordinates of the new vertex  */
    float ntx, nty, ntz;      /* Unit normal for the new vertex */
    int vsid;                 /* Index of vertex to be split */
    int flclw;                /* Index of face under split line */
    short vlr_rot;            /* Number of anti-clockwise traverses */
    struct _vsplit *next;     /* Pointer to next split record */
};
```

The representation is compact and efficient. The male and female mannequin models occupy only 2.7MBytes of disk space in total. After the user has selected the appropriate body parts, the mannequin model, decimated to the prespecified resolution, appears on the screen after an imperceptible delay, since application of the improved vertex split records is extremely rapid.

4 Registration

In the registration process, the user is prompted to indicate landmarks on the mannequin model and corresponding landmarks in the position sensor's coordinate system. The user marks points on the mannequin using the interface shown in Figure 11. To locate the corresponding points in the position sensor's coordinate system, the user simply holds the

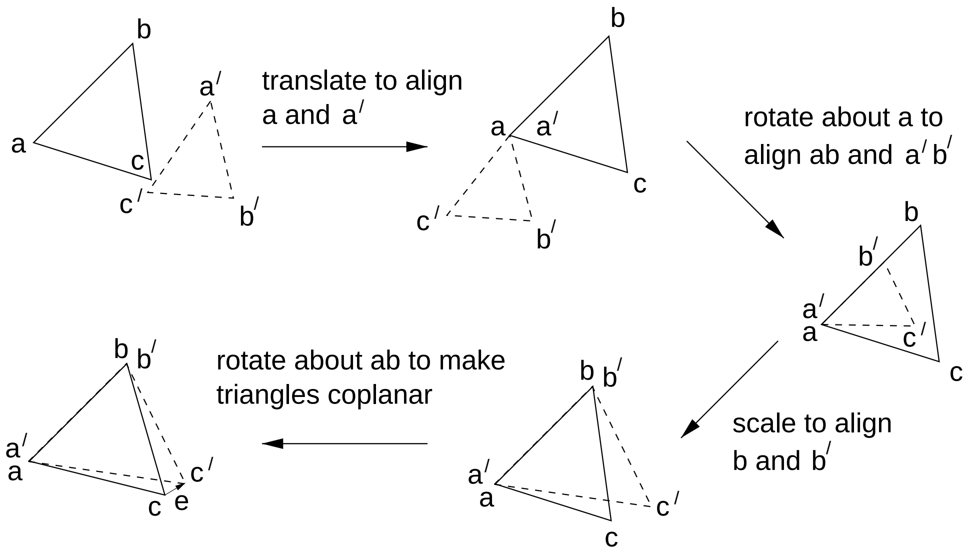


Figure 12: **Heuristic for three point registration.** Triangle abc is formed by the position sensor landmarks, while triangle $a'b'c'$ is formed by the corresponding mannequin landmarks. The heuristic is applied three times, chosing a different edge to align each time. The transformation which results in the smallest error $|e|$ is retained. For a well-conditioned solution, the points abc and $a'b'c'$ should not be nearly collinear.

ultrasound probe against the same point on the patient's body and presses a key to take a position sensor reading[3].

The transformation between the mannequin and position sensor coordinate systems has seven degrees of freedom: three for translation, three for rotation and one for scale. The transformation can therefore be estimated by aligning three pairs of corresponding points: each correspondence provides three constraints on the seven unknowns. Alternatively, the transformation can be estimated by aligning two corresponding points and one surface normal: the surface normal provides an additional two constraints on top of the six from the two pairs of points. The two point technique relies on the user holding the probe perpendicular to the patient's skin while taking the position reading: this defines the body's surface normal in position sensor coordinates, which can be aligned with the corresponding surface normal on the mannequin.

With the three point approach, we avoid unnecessary nonlinear optimisation and instead estimate the registration transformation using the simple heuristic illustrated in Figure 12. Essentially, a similarity transform is found to align two of the pairs of points and make the two triangles coplanar. This is repeated three times, aligning a different edge

[3]The sensor records the position and orientation of the electromagnetic receiver attached to the ultrasound probe. What is required, however, is the position of the centre of the ultrasound probe's face, which is in contact with the patient's skin. This position can be readily derived from the sensor reading using the standard 3D ultrasound calibration parameters [17].

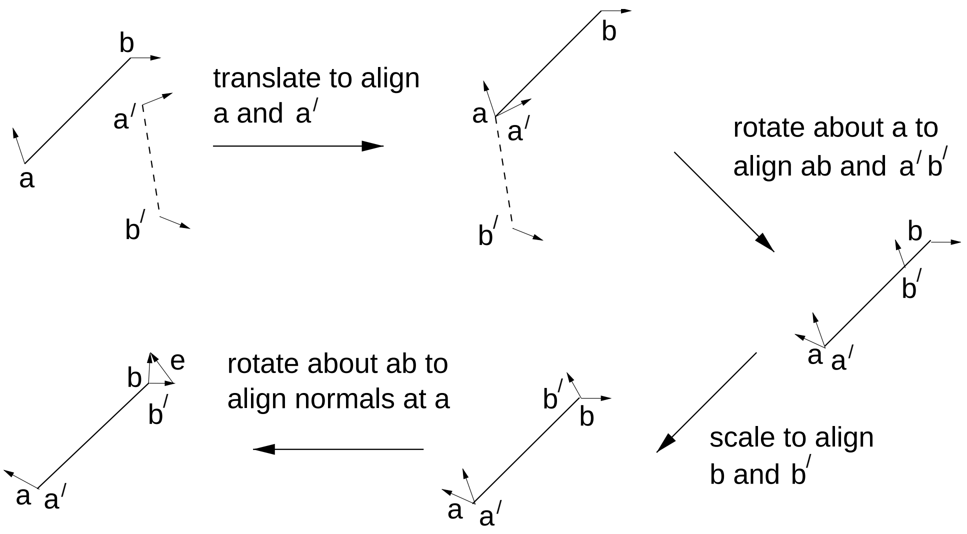


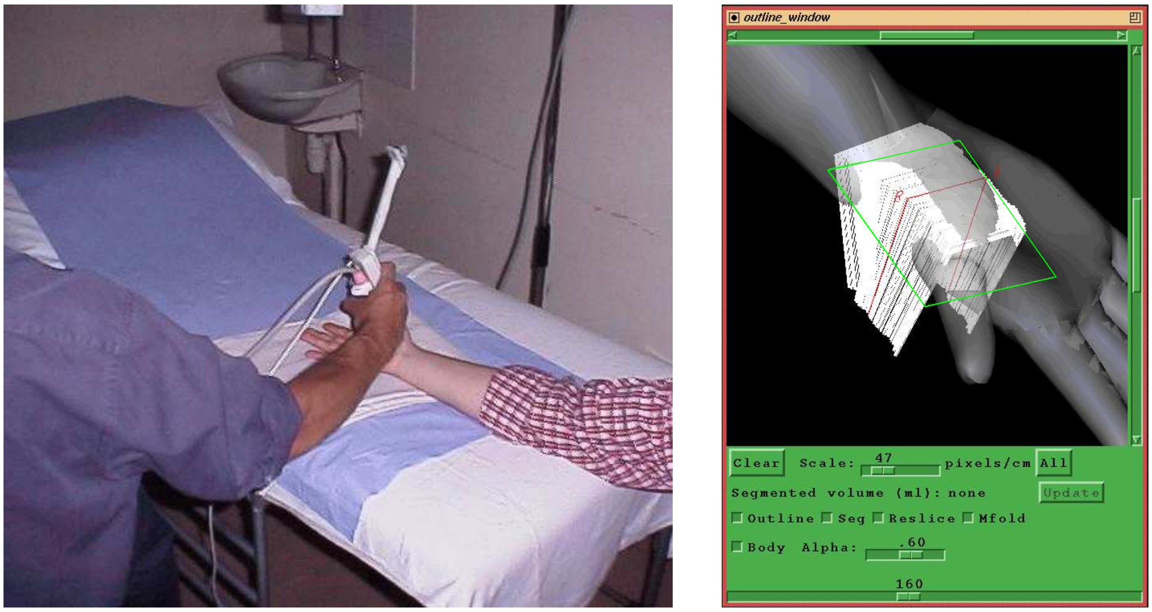
Figure 13: **Heuristic for two point registration.** Line ab is formed by the position sensor landmarks, while line $a'b'$ is formed by the corresponding mannequin landmarks.

each time, and the transformation which results in the smallest error $|e|$ is retained. The final error $|e|$ is displayed in the user interface (Figure 11): a high error indicates that the two sets of points cannot be accurately aligned, most likely because the user was careless when entering the correspondences. The user is at liberty to re-enter the corresponding landmarks until a satisfactory error $|e|$ is obtained.

With the two point approach, we employ a similar heuristic, except that surface normals are aligned in the final step instead of the triangles' planes — see Figure 13. There are two possible solutions, each matching a different surface normal, but they both result in the same error $|e|$. Since there is no meaningful way to choose between the two solutions, one of them is calculated at random.

The potential advantage of the two point approach is clear: only two points are required. However, the user must take great care to hold the probe perpendicular to the patient's skin when recording the point correspondences. With practice, however, the two point approach is the method of choice for linear scan sweeps, when there might be difficulty identifying three non-collinear landmarks for the three point technique.

The interface (Figure 11) allows the user to choose either the two or three point estimates. Once two pairs of landmarks have been entered, the system automatically calculates a two point registration and displays the error $|e|$ at the bottom of the window. The user can either accept this estimate or proceed by entering a third point, in which case the resulting three point estimate replaces the two point version already calculated.



scan in progress

'Outline' window visualisation

Figure 14: **3D ultrasound examination of the wrist.** Note the position sensor receiver mounted some distance from the ultrasound probe: the stand-off is designed to minimize any electromagnetic interference between the two devices. In the 'Outline' window, the B-scan outlines are superimposed onto the mannequin, which is rendered semi-transparently: the degree of transparency is set using the 'alpha' slider. The user can zoom and rotate the objects in the 'Outline' window at interactive rates, thanks to appropriate progressive mesh decimation of the hand model.

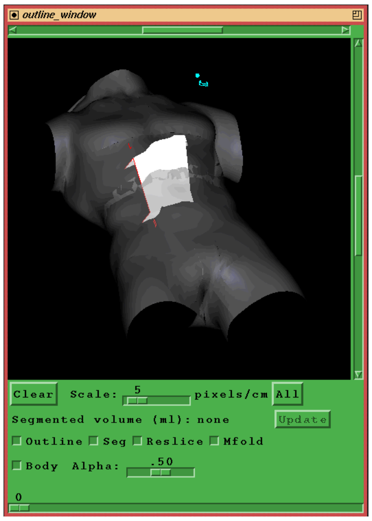
5 Illustrative results

Figures 14 and 15 show typical examples of the body-centered visualisation facility in use. Two very different types of scan are illustrated, demonstrating the system's ability to cope with different sizes of scan area through progressive meshes. In each case, the registration procedure was performed in a few seconds following the scan, using the three point method. The estimated transformation was applied to the mannequin model before displaying it in the 'Outline' window, to bring it into registration with the B-scan outlines which are rendered in the position sensor's coordinate system.

The progressive mesh implementation is transparent to the user, who simply specifies how many triangles the graphics system can cope with in Stradx's 'Setup' panel. Appropriately decimated meshes are displayed immediately following selection of the body parts, with no perceptible delay while the vertex splits are applied.

A slider in the 'Outline' window adjusts the transparency of the mannequin model, so other objects (the reslice plane in Figure 14, for example) can be seen below the surface

scan in progress



'Outline' window visualisation

Figure 15: **3D ultrasound examination of the abdomen.** Note the position sensor transmitter suspended at the end of the microphone stand above the patient. This is the origin of the position sensor coordinate system, which is also marked in the 'Outline' window display. The progressive mesh implementation ensures that there are the same number of triangles in the torso model as in the hand model in Figure 14, so interactive-rate rendering is maintained.

of the skin. The semi-transparent rendering is achieved using standard alpha-blending techniques.

While it is difficult to assess the registration accuracy on a quantitative basis, experience with the system shows that qualitatively acceptable results can be achieved with little effort. Since the 'Outline' window display is available immediately after the scan, any perceived misregistration can be corrected by repeating the brief registration procedure before the patient moves.

6 Conclusions and further work

Overall, the body-centered visualisation facility is simple to use and effective. The registration procedure is rapid and produces results which are acceptable for the application. The body part meshes are sufficiently detailed to show the necessary anatomy, yet sufficiently decimated to allow rapid rendering, regardless of the selected scanning area. The underlying progressive mesh implementation is transparent to the user.

Several minor improvements could increase the compactness of the mesh storage format. For example, the coordinates of new vertices could be given as deltas instead of absolute values. Vertex normals could be stored separately and re-used when shared by more than one vertex split record. Several other improvements are suggested in [8, 9].

The biggest single drawback of the facility concerns the inflexibility of the mannequin models. This causes problems when the scan spans an articulated joint which is not at the same angle as the corresponding joint on the mannequin. Even if the scan does not cover a joint, differences between the general shape of the patient's and mannequin's body parts can make identification of corresponding landmarks difficult in the registration stage and affect the accuracy of the visualisation in the 'Outline' window. These effects could be mitigated by utilising suitably tuned models for the particular shape and pose of the patient. Computer animation research [1] has pointed the way towards the automatic generation of parameterised human-like models, where the parameters control the general shape and pose of the model.

Acknowledgements

We would like to thank Diana Galletly and Patrick Gosling for permission to publish the data in Figure 2. The segmentation in Figure 2 was generated by Graham Treece.

References

[1] N. Badler, D. Metaxas, B. Webber, and M. Steedman. The center for human modeling and simulation. *Presence*, 4(1):81–96, 1995.

[2] A. Ciampalini, P. Cignoni, C. Montani, and R. Scopigno. Multiresolution decimation based on global error. *The Visual Computer*, 13(5):228–246, June 1997.

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

[4] J. Cohen, A. Warchney, D. Manocha, G. Turk, H. Weber, P. Agarwal, F. Brooks, and W. Wright. Simplification envelopes. In *Proceedings of SIGGRAPH'96*, ACM Computer Graphics, Annual Conference Series, pages 119–128, 1999.

[5] P. R. Detmer, G. Bashein, T. Hodges, K. W. Beach, E. P. Filer, D. H. Burns, and D.E. Strandness Jr. 3D ultrasonic image feature localization based on magnetic scanhead tracking: in vitro calibration and validation. *Ultrasound in Medicine and Biology*, 20(9):923–936, 1994.

[6] M. Garland and S. Heckbert. Surface simplification using quadric error metrics. In *Proceedings of SIGGRAPH'97*, ACM Computer Graphics, Annual Conference Series, pages 209–216, 1997.

[7] A. Gee, R. Prager, and L. Berman. Non-planar reslicing for freehand 3D ultrasound. In *Medical Image Computing and Computer-Assisted Intervention — MICCAI'99*, pages 716–725, Cambridge UK, September 1999. LNCS 1679, Springer.

[8] H. Hoppe. Progressive meshes. In *Proceedings of SIGGRAPH'96*, ACM Computer Graphics, Annual Conference Series, pages 99–108, 1996.

[9] H. Hoppe. Efficient implementation of progressive meshes. *Computers and Graphics*, 22(1):27–30, 1998.

[10] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle. Mesh optimization. In *Proceedings of SIGGRAPH'93*, ACM Computer Graphics, Annual Conference Series, pages 19–26, 1993.

[11] S. W. Hughes, T. J. D'Arcy, D. J. Maxwell, W. Chiu, A. Milner, J. E. Saunders, and R. J. Sheppard. Volume estimation from multiplanar 2D ultrasound images using a remote electromagnetic position and orientation sensor. *Ultrasound in Medicine and Biology*, 22(5):561–572, 1996.

[12] D. L. King, D. L. King Jr., and M. Y. Shao. Evaluation of in vitro measurement accuracy of a three-dimensional ultrasound scanner. *Journal of Ultrasound in Medicine*, 10:77–82, 1991.

[13] R. Ohbuchi, D. Chen, and H. Fuchs. Incremental volume reconstruction and rendering for 3D ultrasound imaging. In R. A. Robb, editor, *Proceedings of Visualization in Biomedical Computing*, SPIE 1808, pages 312–323. International Society of Optical Engineering, Bellingham, WA, USA, 1992.

[14] R. W. Prager, A. H. Gee, and L. Berman. 3D ultrasound without voxels. In *Proceedings of Medical Image Understanding and Analysis*, pages 93–96, Leeds, July 1998.

[15] R. W. Prager, A. H. Gee, and L. Berman. Real-time tools for freehand 3D ultrasound. In *Medical Image Computing and Computer-Assisted Intervention — MICCAI'98*, pages 1016–1023, Cambridge MA, October 1998. LNCS 1496, Springer.

[16] R. W. Prager, A. H. Gee, and L. Berman. Stradx: real-time acquisition and visualization of freehand three-dimensional ultrasound. *Medical Image Analysis*, 3(2):129–140, 1999.

[17] R. W. Prager, R. N. Rohling, A. H. Gee, and L. Berman. Rapid calibration for 3-D freehand ultrasound. *Ultrasound in Medicine and Biology*, 24(6):855–869, 1998.

[18] R. N. Rankin, A. Fenster, D. B. Downey, P. L. Munk, M. F. Levin, and A. D. Vellet. Three-dimensional sonographic reconstruction: techniques and diagnostic applications. *American Journal of Roentgenology*, 161(4):695–702, 1993.

[19] W. Schroeder, K. Martin, and B. Lorensen. *The Visualization Toolkit: An Object-Oriented Approach to 3D Graphics*. Prentice Hall, London, 2nd edition edition, 1998.

[20] W. Schroeder, J. Zerge, and W. Lorensen. Decimation of triangle meshes. *Computer Graphics*, 26(2):65–70, August 1992.

[21] H. Steiner, A. Staudach, D. Spitzer, and H. Schaffer. Three-dimensional ultrasound in obstetrics and gynaecology: technique, possibilities and limitations. *Human Reproduction*, 9(9):1773–1778, 1994.

[22] G. M. Treece, R. W. Prager, A. H. Gee, and L. Berman. Fast surface and volume estimation from non-parallel cross-sections, for freehand 3-D ultrasound. *Medical Image Analysis*, 3(2):141–173, 1999.

[23] J. W. Trobaugh, D. J. Trobaugh, and W. D. Richard. Three-dimensional imaging with stereotactic ultrasonography. *Computerized Medical Imaging and Graphics*, 18(5):315–323, 1994.