Matching 3-D Anatomical Surfaces with Non-Rigid Deformations using Octree-Splines

Richard Szeliski†, Stéphane Lavallée‡,

†Digital Equipment Corporation, Cambridge Research Lab
One Kendall Square, Bldg. 700, Cambridge, MA 02139

‡TIMB - TIMC - IMAG
Faculté de Médecine de Grenoble, 38 700 La Tronche, France

Abstract

This paper presents a new method for determining the minimal non-rigid deformation between two 3-D surfaces, such as those which describe anatomical structures in 3-D medical images. Although we match surfaces, we represent the deformation as a volumetric transformation. Our method performs a least squares minimization of the distance between the two surfaces of interest. To quickly and accurately compute distances between points on the two surfaces, we use a precomputed distance map represented using an *octree spline* whose resolution increases near the surface. To quickly and robustly compute the deformation, we use a second octree spline to model the deformation function. The coarsest level of the deformation encodes the global (e.g., affine) transformation between the two surfaces, while finer levels encode smooth local displacements which bring the two surfaces into closer registration. We present experimental results on both synthetic and real 3-D surfaces.

1. Introduction

The matching of 3-D anatomical surfaces between anatomical atlases and patient data, or between different patient data sets, is an important element of 3-D medical image analysis and quantification. Matching between atlases and patient data enables more accurate and reliable segmentation and the functional labeling of medical images, as well as multimodality data registration and integration. In computer vision, this problem corresponds to finding the non-rigid deformation between two surfaces, with applications to model-based object recognition and deformable object tracking.

In previous work [21, 11], we developed a fast and accurate technique for determining the rigid transformation between two surfaces, and also between a 3-D surface and its 2-D projections. In this paper, we extend our technique to recover smooth non-rigid deformations between 3-D surfaces. Our approach is based on describing the deformation as a warping of the space containing one of the surfaces. In particular, we use a multiresolution warp or displacement field based on concepts from free-form deformations [31], octree splines [21], and hierarchical basis functions [33]. Our approach enables us to locally adapt the resolution of the deformation field to bring the two surfaces into registration, while maintaining smoothness and avoiding unnecessary computation. The result is a rapid and efficient registration algorithm which does not require the extraction (manual or automatic) of features on the two surfaces, and which can work on arbitrarily shaped surfaces with highly complicated deformations.

The main application of our technique is model-based segmentation of 3-D medical images. Segmenting medical structures is important both in medical diagnosis and as a component of computer assisted medical interventions [19, 20]. While unsupervised segmentation based on purely local operators can be very difficult, the *a priori* knowledge contained in anatomical models can make the segmentation process easier and more robust. Such anatomical models can either be digitized from textbooks, or they can be built by elastically registering data sets from many patients.

A second application of our technique is in quantification of normal anatomical structures and deviations from normal (*morphometrics* [6]), e.g., studies of brain asymmetry, and in deviation from self over time, e.g., changes in liver tumors. In some cases, a volume registration between a patient data set (e.g., a 3D MRI data set) and a model (e.g., a brain atlas) is made possible by the existence of some common reference surfaces in both data sets (e.g., the surface of brain ventricles). This registration can be used to *infer* the location of specific features (e.g., thalamus brain nuclei) on the patient data set. Such an inference would not be possible if the elastic registration was a surface deformation instead of a volume deformation.

Another application of our technique is the calibration of non-linear distortion in medical imaging devices such as MRI through elastic registration with CT images which do not have such distortions. Our technique can also be used in other applications of deformable model-based vision, e.g., in the tracking of deformable objects such as the beating heart [16] or animate motion [35]. Another novel application could be in performing *morphing* [3] between 3-D data sets for computer animation effects.

We begin the paper in section 2 with a review of related work in the fields of medical image registration, computer vision, and computer graphics. In section 3, we present our formulation of the problem. In section 4, we present the Levenberg-Marquardt iterative nonlinear least-squares algorithm which we use to compute the deformation estimate. Section 5 presents the *octree spline* 3-D distance map we use to rapidly compute minimal distances between points on the two surfaces. Section 6 describes in more detail the hierarchical octree spline used to represent the multiresolution non-rigid deformation. Section 7 presents some experimental results, with both synthetic data and real data. Finally, the algorithm, extensions of the method, and future improvements are discussed in section 8.

2. Previous work

The registration of 3-D and 2-D images has been widely studied in both medical image processing and computer vision. In medical imaging, the problem of image registration has been usually solved using external fiducial markers placed on the body of the patient [18] or by interactively selecting pairs of matching points [30]. An automated algorithm for matching 3-D surfaces with other 3-D surfaces (such as the head skin surface) has been developed by Pelizzari [27]. Our previous algorithm solved this problem by minimizing the sum of squared distances between the transformed points on one surface and a stationary description of the other surface [21, 11] (see also [4]). It also solved the more difficult problem of registering a 3-D surface with its 2-D projections [21, 8]. Methods based on the registration of 3-D curves (*ridges*) have been proposed in [15, 24].

The registration of 3-D medical images under non-rigid deformations has been studied by Bajcsy and Kovacic [1] using volumetric deformations based on physical properties. Another approach from Evans et al. [13] is based on approximating a 3D warping function by 3D thin-plate splines fit to manually matched reference points. In computer graphics, non-rigid deformations are widely used for modeling and animation purposes [2, 38]. In computer vision, non-rigid deformations have been used for fitting flexible models to both image and range data [9, 36, 35], and for fitting models to 3D volumetric data [16, 23]. The approach we use in this paper is based on *free-form deformations* [31], which use volumetric, 3-D tensor-product splines to describe the warping or displacement of points embedded in the space. For increased efficiency, we represent the deformation spline using a multiresolution representation [14, 37], based on the concept of hierarchical basis functions [39, 33]. Deformable models have also been used for the segmentation of 3-D medical data. Leitner and Cinquin [22] use a tensor-product spline surface model to segment complicated structures such as vertebra.

3. Problem formulation

We formulate our deformable matching problem as follows. Given a *model object* in a coordinate system $\text{Ref}_{\text{model}}$ and *sensed data* in a coordinate system $\text{Ref}_{\text{sensor}}$, estimate the transformation $\mathbf{T}$ parameterized by a parameter vector $\mathbf{p}$ which relates $\text{Ref}_{\text{sensor}}$ to $\text{Ref}_{\text{model}}$. More specifically, we assume for now that both data sets are surfaces, with the sensed data represented as a collection of points $\{\mathbf{q}_i, i = 1 \ldots N\}$, and the model surface S represented in some arbitrary way. The estimation task is then to find a geometric transformation $\mathbf{T}$ such that the transformed coordinates $\mathbf{r}_i = \mathbf{T}(\mathbf{q}_i; \mathbf{p})$ all lie on the surface S.

In practice, due to noise and the inability to perfectly register two surfaces, this condition will never be satisfied. Instead, we pose the problem as a minimization of the cost function

$$C(\mathbf{p}) = \sum_{i=1}^{N} \frac{1}{\sigma_i^2}[d(\mathbf{r}_i, S)]^2 = \sum_{i=1}^{N} \frac{1}{\sigma_i^2}[d(\mathbf{T}(\mathbf{q}_i; \mathbf{p}), S)]^2, \tag{1}$$

where $d(\mathbf{r}_i, S) = \min_{\mathbf{s} \in S} ||\mathbf{r}_i - \mathbf{s}||$ is the minimum Euclidean distance from the point $\mathbf{r}_i$ to S and σ_i^2 is the variance associated with point i [21, 4].

To solve the minimization problem, we require three components: an iterative minimization algorithm (section 4), an efficient method for computing $d(\mathbf{r}, S)$ along with its gradient (section 5), and a suitable representation for the geometric transformation $\mathbf{T}(\mathbf{q}_i; \mathbf{p})$, which we discuss below.

3.1. Global polynomial deformations

The simplest representation for $\mathbf{T}(\mathbf{q}_i; \mathbf{p})$ is a rigid body transformation which can be parameterized by 6 degrees of freedom (3 for the translation, and 3 for rotation.) In our previous work [21], we used the Euler angle representation for the rotation

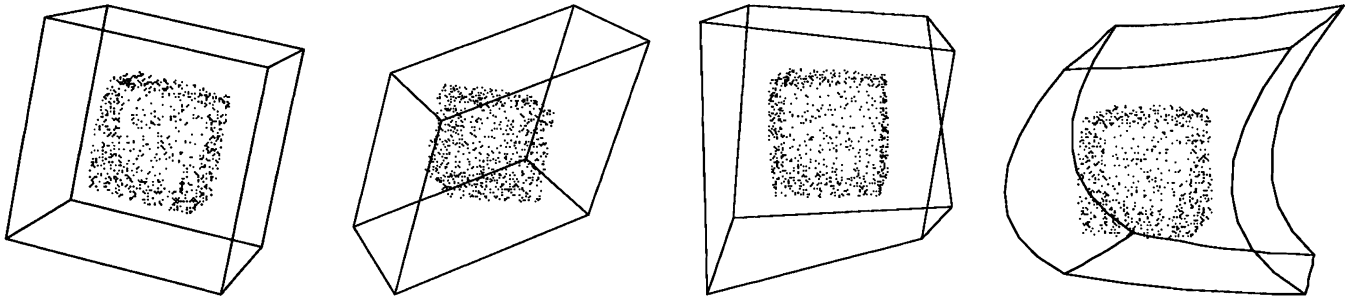

Figure 1: Sample global deformations: (a) rigid, (b) affine, (c) trilinear and (d) quadratic.

matrix $\mathbf{R}$. The rigid body representation is appropriate when working with rigid anatomical structures where only the *pose* of the patient (e.g., on the operating table or in the scanner) is unknown.

A more general class of transformations are the affine transforms, which can be parameterized with 12 degrees of freedom

$$\mathbf{T}(\mathbf{q}_i;\mathbf{p}) = \begin{bmatrix} p_{00} & p_{01} & p_{02} & p_{03} \\ p_{10} & p_{11} & p_{12} & p_{13} \\ p_{20} & p_{21} & p_{22} & p_{23} \end{bmatrix} \begin{bmatrix} 1 \\ x_i \\ y_i \\ z_i \end{bmatrix} \tag{2}$$

where $\mathbf{q}_i = (x_i, y_i, z_i)$ are the coordinates of point i in $\text{Ref}_{\text{sensor}}$. Affine transforms have scaling along each dimension as well as shearing. The trilinear class of transformations adds another 12 degrees of freedom for a total of 24,

$$\mathbf{T}(\mathbf{q}_i;\mathbf{p}) = \begin{bmatrix} p_{00} & p_{01} & & p_{07} \\ p_{10} & p_{11} & \cdots & p_{17} \\ p_{20} & p_{21} & & p_{27} \end{bmatrix} \begin{bmatrix} 1 & x_i & y_i & z_i & x_iy_i & y_iz_i & z_ix_i & x_iy_iz_i \end{bmatrix}^T. \tag{3}$$

The trilinear transformation can also be parameterized by the 3-D positions of the 8 corners of a bounding cube, but the equations are somewhat more complicated. Finally, the 30-parameter quadratic family of transformations can be similarly formed by a matrix product with the vector $\begin{bmatrix} 1 & x_i & y_i & z_i & x_iy_i & y_iz_i & z_ix_i & x_i^2 & y_i^2 & z_i^2 \end{bmatrix}^T$. Figure 1 shows an illustration of the above four global transforms where a unit cube has been deformed by a representative sample from each transformation class.

Each of these global transformations is more general than the previous one, which means that it can model a wider range of deformations, but at the expense of having more parameters whose values may be difficult to estimate reliably. All of these transformations share the property that the position of the transformed point $\mathbf{r}_i = \mathbf{T}(\mathbf{q}_i;\mathbf{p})$ is a *linear* function of the parameters in $\mathbf{p}$, i.e., the transformation can be written as $\mathbf{r}_i = \mathbf{M}_i\mathbf{p}$ [38]. For example, the affine deformation can be represented with

$$\mathbf{M}_i = \mathbf{M}(\mathbf{q}_i) = \begin{bmatrix} 1 & 0 & 0 & x_i & 0 & 0 & y_i & 0 & 0 & z_i & 0 & 0 \\ 0 & 1 & 0 & 0 & x_i & 0 & 0 & y_i & 0 & 0 & z_i & 0 \\ 0 & 0 & 1 & 0 & 0 & x_i & 0 & 0 & y_i & 0 & 0 & z_i \end{bmatrix}$$

where we have ordered the parameter vector as $\mathbf{p} = \begin{bmatrix} p_{00} & p_{10} & \cdots & p_{23} \end{bmatrix}^T$.

3.2. Local spline deformations

To obtain an even wider range of more flexible deformations, we could continue increasing the order of the polynomial. However, this suffers from the same problems as high degree polynomial fitting, e.g., instability and the presence of ringing. Instead, we model the deformation using a family of volumetric tensor product splines,

$$\mathbf{T}(\mathbf{q}_i;\mathbf{p}) = \sum_{j,k,l} \mathbf{d}_{jkl} B_j(x_i) B_k(y_i) B_l(z_i), \tag{4}$$

where the $\mathbf{d}_{jkl}$ are the spline coefficients which comprise the parameter vector $\mathbf{p}$, and B_j, B_k, and B_l are B-spline basis functions [31]. Each basis function (a piecewise polynomial function) only has a limited range of support, i.e., it is non-zero only in the interval $x_{j-o} \le x \le x_{j+o}$, where the x_j, $j = 0 \ldots M_x$ form a subdivision of each coordinate axis in $\text{Ref}_{\text{sensor}}$. This implies that only a small number of the $\mathbf{d}_{jkl}$ will contribute to the value of $\mathbf{r}_i$, or, equivalently, that the matrices $\mathbf{M}_i$ in this

representation will be extremely sparse (only a few non-zero entries). In general, this means that the deformations required to bring a local area of two surfaces into registration will not affect the registration at far-away portions of the surface. The $\mathbf{d}_{jkl}$ can be thought of as local estimates of the displacements required to register the two surfaces, which are smoothed and interpolated through the action of the spline functions.

For our initial experiments, we first find a set of global transformations, starting with a rigid transformation and then fitting an affine transformation. We then estimate a local trilinear spline deformation. To better decouple local *elastic* deformations from global effects such as scale change, we use both transformations in series, i.e., first use (4) to transform from $\text{Ref}_{\text{sensor}}$ to an intermediate frame Ref_{int}, and then a global transformation to transform from Ref_{int} to $\text{Ref}_{\text{model}}$,

$$\mathbf{q}_i' = [\mathbf{M}_1(\mathbf{q}_i)]\mathbf{p}_1 \quad \text{and} \quad \mathbf{r}_i = [\mathbf{M}_2(\mathbf{q}_i')]\mathbf{p}_2, \tag{5}$$

where $\mathbf{p}_1$ and $\mathbf{p}_2$ are the local and global deformation parameters.

4. Least squares minimization

To perform the nonlinear least squares minimization, we use the Levenberg-Marquardt algorithm because of its good convergence properties [28]. Least squares techniques work well when we have many uncorrelated noisy measurements with a normal (Gaussian) distribution.[1]

In order to update the current estimate of the parameters $\mathbf{p}^{(k)}$, Levenberg-Marquardt requires the evaluation of the distance function $d(\mathbf{r}_i, S)$ along with its derivative with respect to all of the unknown parameters. Efficient techniques for computing the distance function d, as well as its spatial gradient $\mathbf{g} = \nabla_{\mathbf{r}} d$, are presented in the next section. The evaluation of the derivative involves a straightforward application of the chain rule,

$$\frac{\partial d_i}{\partial \mathbf{p}} = \mathbf{g}_i \frac{\partial \mathbf{r}_i}{\partial \mathbf{p}} = \mathbf{g}_i \mathbf{M}_i \tag{6}$$

in the case when the transformation can be written as $\mathbf{r}_i = \mathbf{M}_i \mathbf{p}$. For the more complicated local/global transformation (5), we compute the derivative as

$$\frac{\partial d_i}{\partial \mathbf{p}_2} = \mathbf{g}_i \mathbf{M}_2(\mathbf{q}_i') \quad \text{and} \quad \frac{\partial d_i}{\partial \mathbf{p}_1} = \mathbf{g}_i [\frac{\partial \mathbf{M}_1}{\partial \mathbf{q}_i'} \mathbf{p}_2] \mathbf{M}_2(\mathbf{q}_i). \tag{7}$$

Once the distance samples d_i and their derivatives $\partial d_i / \partial \mathbf{p}$ have been computed, the Levenberg-Marquardt algorithm forms the approximate Hessian matrix $\mathbf{A}$ and the weighted error gradient vector $\mathbf{b}$,

$$\mathbf{A} = \sum_{i=1}^{N} \frac{1}{\sigma_i^2} \frac{\partial d_i}{\partial \mathbf{p}} \left(\frac{\partial d_i}{\partial \mathbf{p}} \right)^T \quad \text{and} \quad \mathbf{b} = -\sum_{i=1}^{N} \frac{1}{\sigma_i^2} d_i \frac{\partial d_i}{\partial \mathbf{p}} \tag{8}$$

and then computes an increment $\delta \mathbf{p}$ towards the local minimum by solving

$$(\mathbf{A} + \lambda \mathbf{I}) \delta \mathbf{p}^{(k)} = \mathbf{b}, \tag{9}$$

where λ is a stabilizing factor which varies over time [28]. After setting $\mathbf{p}^{(k+1)} = \mathbf{p}^{(k)} + \Delta \mathbf{p}^{(k)}$, this process is repeated until $C(\mathbf{p})$ is below a fixed threshold, the difference between parameters $|\mathbf{p}^{(k)} - \mathbf{p}^{(k-1)}|$ at two successive iterations is below a fixed threshold, or a maximum number of iterations is reached.

The solution of the system of equations (9) is straightforward when the number of parameters being estimated is reasonably small (as in global deformations). We currently use the Gauss-Jordan elimination algorithm suggested by [28]. For systems with more parameters, such as local deformations, it becomes impractical to explicitly compute the Hessian $\mathbf{A}$, which takes $O(M^2)$ space, where M is the number of parameters (say $16^3 \ldots 64^3$), or to directly solve (9), which takes $O(M^3)$ operations. Instead, we use an iterative sparse matrix solver, namely conjugate gradient descent [33].

Once the Levenberg-Marquart algorithm has converged, we compute a robust estimate of the parameter $\mathbf{p}$ by throwing out the measurements where $d_i^2 \gg \sigma_i^2$ and performing some more iterations [17]. This process removes the influence of the *outliers* which are likely to occur when only part of the two data sets is in proper registration. We can also estimate the uncertainty in the parameters $\mathbf{p}$ by computing the covariance matrix $\text{Cov}(\mathbf{p})$ as the inverse of the final Hessian matrix and performing eigenvalue analysis of $\text{Cov}(\mathbf{p})$ [28].

When using a gradient descent technique such as Levenberg-Marquardt, there is a possibility that the minimization might fail because of local minima in the high-dimensional parameter space. We try to avoid this by first estimating the simplest possible

[1] Under these assumptions, the least squares criterion is equivalent to maximum likelihood estimation.

transformation (a rigid matching), and then estimating successively more complex deformations. To prevent underconstrained parameters from acquiring bad estimates, we also add a *prior constraint* on their values in the form of an additional cost function

$$C'(\mathbf{p}) = \sigma_p^{-2}||\mathbf{p}||^2. \tag{10}$$

5. Fast distance computation using octree splines

The method described in the previous section relies on the fast computation of the distance $d(\mathbf{r}, S)$ and its gradient. To speed up this computation, we precompute a 3-D *distance map*, which is a function that gives the minimum distance to S from any point $\mathbf{r}$ inside a bounding volume V that encloses S [12, 7, 26]. In looking for an improved trade-off between memory space, accuracy, speed of computation, and speed of construction, we developed a new kind of distance map which we call the *octree spline* [21, 8, 11]. The intuitive idea behind this geometrical representation is to have more detailed information (i.e., more accuracy) near the surface than far away from it. We start with the classical octree representation associated with the surface S [29] and then extend it to represent a continuous 3-D function that approximates the Euclidean distance to the surface. This representation combines advantages of adaptive spline functions and hierarchical data structures.

As with the previously described distance map, the input to the octree spline construction algorithm is a set of n points $\mathbf{s}_i$ regularly spread on the surface S. The algorithm performs the following steps (see [21] for details):

1. **Surface point octree construction:** First, the octree associated to the set of points $\mathbf{s}_i$ is built according to classic octree subdivision [29]. Starting from the initial cube V, each node that contains points (grey node) is recursively split into 8 sub-cubes until it contains no points (white node) or it has the maximal selected resolution (black node).
2. **Subdivision (refinement):** The octree previously computed may have large empty nodes near the surface, because no rules about subdivision near the surface have been introduced. To overcome this problem, we perform a further subdivision of the octree to ensure that two nodes which are neighbors along a face, edge, or corner differ in size by at most a factor of k_S (in practice, we choose $k_S = 2$).
3. **Corner distance computation:** For each corner $\mathbf{c}$ of each terminal node (white or black), the distance $d(\mathbf{c}, S)$ is computed. The spatial organization of surface points created by the octree makes this process much faster since we can use best-first search to find this minimum distance quickly [21].
4. **Crack elimination (continuity enforcement):** Because the distance is computed at any point $\mathbf{r}$ by an interpolation based on the 8 corner values of the terminal node that contains $\mathbf{r}$ (see below), discontinuities or *cracks* can appear if we simply interpolate the true corner distance values. To avoid this, if a corner $\mathbf{c}$ of a node N_1 of size s_1 lies on a face or an edge of another node N_2 of size $s_2 > s_1$, then the distance value of the corner $\mathbf{c}$ is simply replaced by the distance computed at $\mathbf{c}$ by interpolation inside N_2.

After the previous steps have been performed, $d(\mathbf{r}, S)$ can be computed for any point $\mathbf{r}$ by first finding the terminal node N that contains the point $\mathbf{r}$ (using classical binary search) and then using a trilinear interpolation of the 8 corner values d_{ijk}. We can compute the gradient $\mathbf{g} = \nabla d(\mathbf{r})$ of the distance function by simply differentiating the trilinear interpolant with respect to u, v, and w. Because d is only C^0, $\mathbf{g}(\mathbf{r})$ is discontinuous on cube faces. However, these gradient discontinuities are relatively small and do not seem to affect the convergence of our iterative minimization algorithm.

6. Hierarchical octree spline deformations

The benefits of using a hierarchical data structure for representing the distance map can be extended to the representation used for the tensor-product (local deformation) displacement spline. In this case, the coefficients associated with the spline are 3-D vectors representing the displacement between the model and sensor reference frames. As in the distance octree spline, we use the octree to determine at what resolution the spline coefficients are interpolated. To ensure continuity, we must enforce "crack filling". In this case, however, since the coefficients in the octree spline are unknown parameters being estimated, this is equivalent to controlling which coefficients are free variables and which are determined by their parents' values.

As mentioned in section 4, the inner update loop of the Levenberg-Marquardt requires the solution of a large system of linear equations. Because direct methods are prohibitive, we use the iterative conjugate gradient descent algorithm. To accelerate its convergence, we wish to use a *hierarchical basis* representation of the octree spline [39, 33]. In this representation, displacement values at finer levels are added to the displacements interpolated from the parents, making this a *relative* or *offset representation* [14, 34, 32, 37]. In a true hierarchical basis representation, only nodes not coincident with their parents in location have a non-zero offset, which keeps the number of variables in the hierarchical and regular (*nodal*) representations the same [39].

In our implementation, we keep both the hierarchical and nodal representations, and map between the two as required. For accumulating the distances and gradients required in (8), we compute the interpolated displacements and the derivatives with

respect to the parameters in the nodal basis. We then use the hierarchical basis to *smooth* the residual vector $\mathbf{b}$ before selecting a new conjugate direction and computing the optimal step size (for details, see [33]). The hierarchical basis also makes it easier to understand the influence that the octree has on which displacement parameters are free to change. In the hierarchical basis, a node is free to change (has non-zero value), if all of the cubes within its support region have been subdivided to at least its level. In other words, a basis function associated with a non-zero node cannot extend into a larger cube where its influence would not be accounted for.

To use the hierarchical octree spline for modeling deformations, we still have to devise a strategy for deciding how to adaptively subdivide the octree spline. The heuristic we use is to measure the average squared distance for all of the points inside a given octree cube, and to subdivide those cubes which exceed a threshold. This in itself will often not suffice, since a node is not freed up to take on a non-zero value unless all of the cubes it influences have been subdivided. We therefore mark all neighboring cubes for subdivision as well (the extent of the neighborhood depends on the order of the spline). We proceed in a top down fashion, subdividing all cubes within a given level before increasing the finest resolution level.

The hierarchical octree spline favors smoother displacement solutions because it tries to account for deformations at the coarsest (most global) level possible. In many situations, however, we may still wish to add an *elastic* smoothness constraint on the deformation spline [1]. This can be done in one of two ways: either adding a regular regularization term on the interpolated displacement field [1, 33], or by penalizing each node in a hierarchical basis independently as in (10) [34]. We are currently investigating the tradeoffs between these two approaches.

7. Experimental results

To determine the reliability of our global and local deformation estimates, we first performed a series of experiments on both real and synthetic surfaces under simulated (known) motion. For a given synthetic surface (e.g., the cube in Figure 2), we first compute the octree distance map. Next, we select a subset of the surface points (typically 5%) and transform these through the inverse of the deformation we are simulating.[2] Figures 2a, 2d, and 2g show the original and subsampled data sets. We then initialize the Levenberg-Marquardt algorithm with some initial rigid pose estimate (about 40° away from the true solution for the examples in Figure 2) and set the non-rigid parameter estimates to 0. Finally, we run the Levenberg-Marquardt algorithm in stages, first computing the best rigid match, then the best global non-rigid match, and finally the best local displacement warp. For the examples in Figure 2, we perform 5 iterations of rigid motion estimation, followed by 5 iterations of global non-rigid motion. To stabilize the motion estimates, we use a small prior penalty term of $\sigma_p = 1$ (compared to $\sigma_i = 0.5$). The middle column shows the shape of the estimated deformation represented as a warped bounding cube. The right hand column shows the error between the estimated and true (simulated) deformations by mapping the bounding cube through both the estimated deformation and the inverse simulated deformation. As we can see, the estimated parameters are quite close to the true parameters, as evidenced by the close fits between the two bounding cubes.

To demonstrate the local non-rigid matching, we use two different sets of range data acquired with a Cyberware laser range scanner (Figures 3a and 3b). In their initial positions, the data sets overlap by about 50% and differ in orientation by about 10° (Figure 3c). Here, the octree spline distance map is computed on the larger of the two data sets (`georgel` in Figure 3a), and the smaller of the two data sets is deformed (`heidi` in Figure 3b). After 8 iterations of rigid matching and 8 iterations of non-rigid affine matching, the registered data sets appear as in Figure 3d. We then perform 8 iterations at each level of the local displacement spline for 1, 2, 3, and 4 levels. The finest octree spline level has $(2^4 + 1)^3 \approx 5000$ nodes for a total of about 15000 degrees of freedom.[3] Even with this large number of parameters, the algorithm converges very quickly, because it is always in the vicinity of a good solution (a typical iteration at the finest level takes about 2 seconds). From Figure 3f, we see that the two data sets are registered well, except for the eyebrows, which would require a more detailed deformation. We also note that the deformed face of `heidi` (Figure 3e) resembles that of `georgel` (Figure 3a) more than its former (undeformed) self (Figure 3b).

As a final example of our algorithm, we matched the surface of a real patient vertebra to the surface of a plastic "phantom" vertebra (both 3-D images sets were acquired with a CT scanner). Figure 4 shows the result of our matching. After affine registration, a fair amount of discrepancy remains. After the local spline registration, most of the patient vertebra (contour lines) matches the phantom model (cloud of dots), except for the tips of the vertebra which have not been pulled into registration (this may be an artifact of the asymmetrical nature of the cost function).

[2] For affine transforms, this inverse has a closed form solution. For other non-linear transforms, we use an iterative inversion technique.

[3] In our current implementation, the octree is uniformly subdivided throughout.

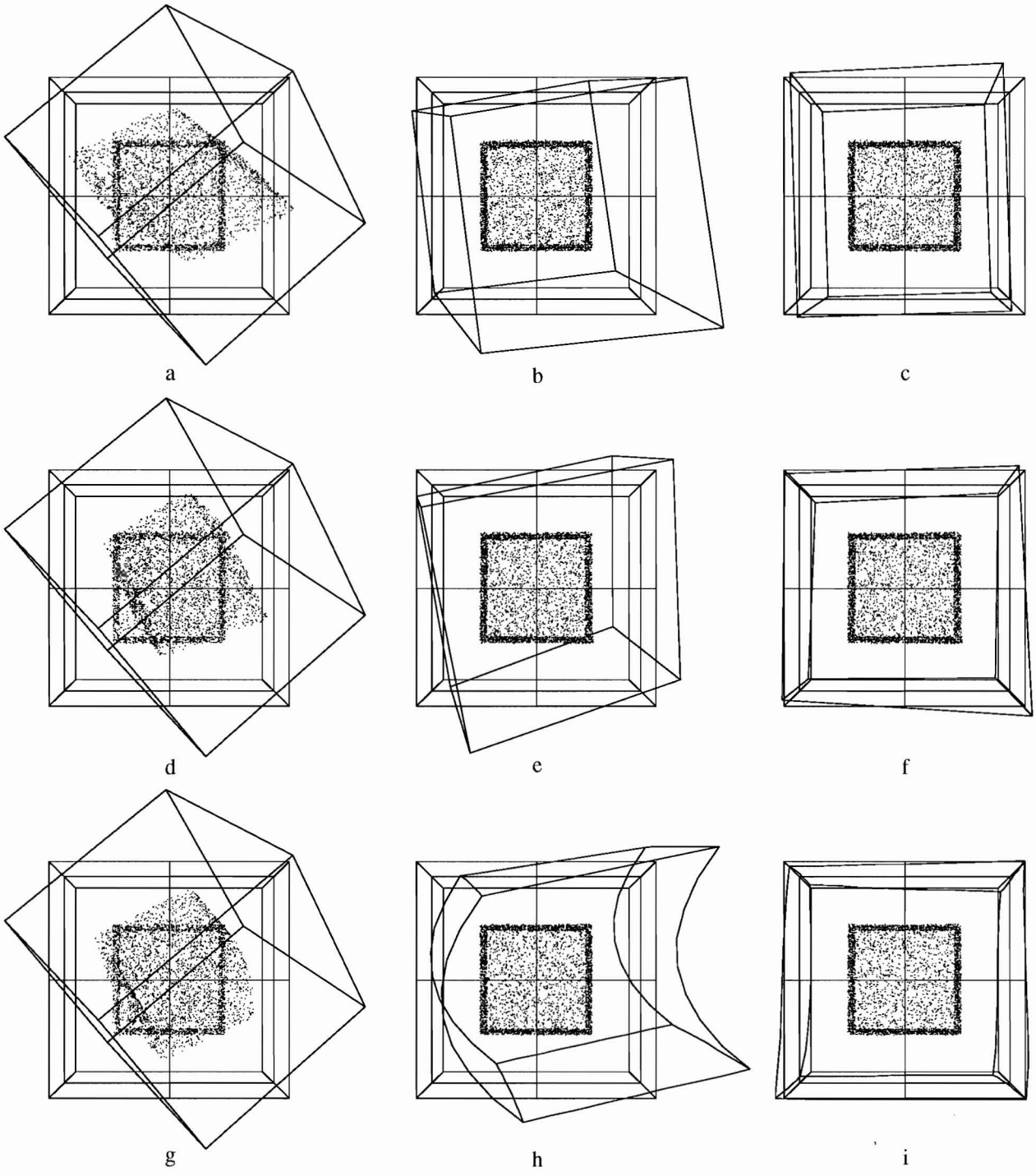

Figure 2: Convergence on synthetic cube surface:
(a–c) affine deformation, (d–f) trilinear deformation, (g–i) quadratic deformation. The first column shows the data before registration. The second column shows the registered data and the shape of the deformation. The third column shows the error in the deformation estimate (bounding cube transformed by estimated deformation and inverse simulated deformation).

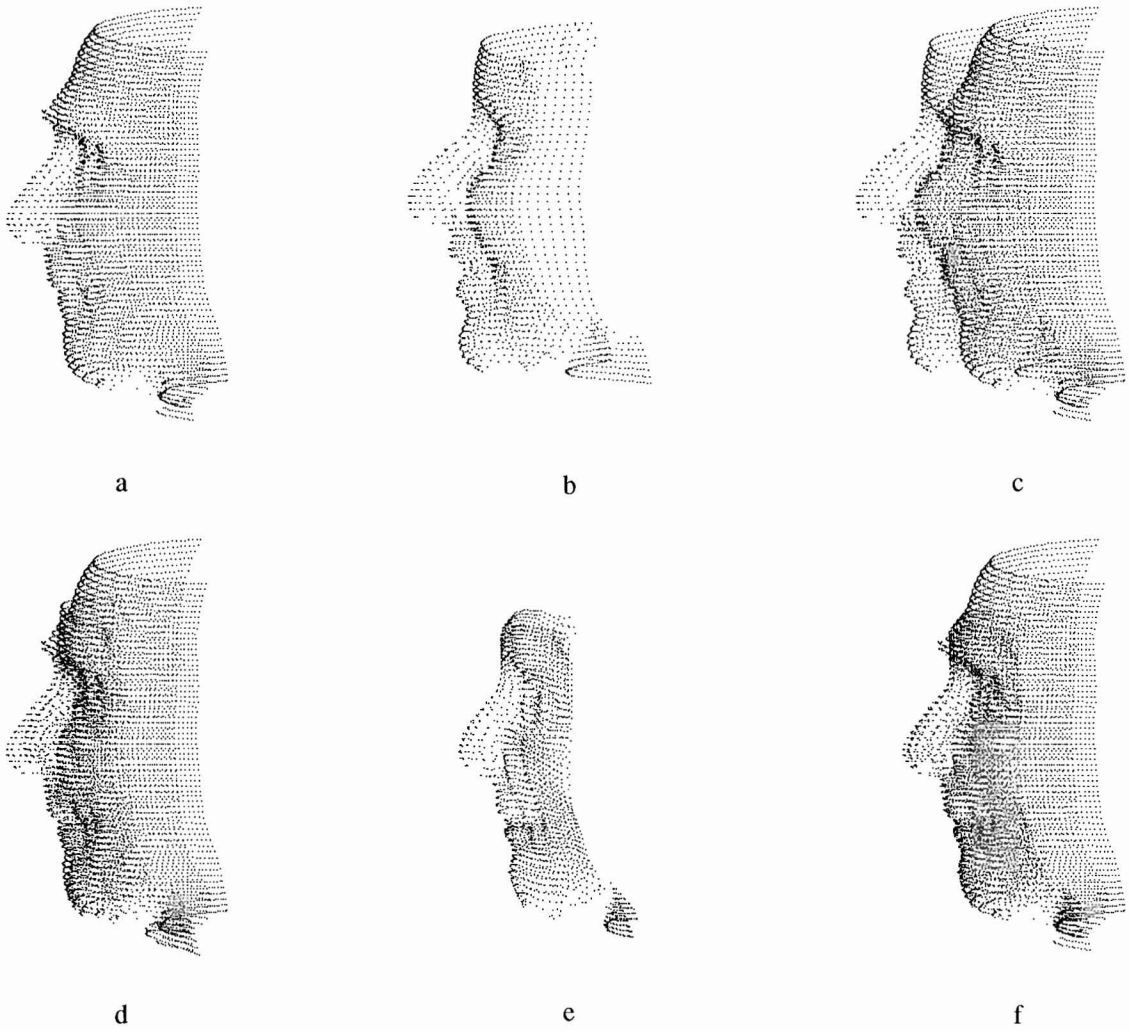

Figure 3: Registration of two face data sets:
(a) model data set (`georgel`) (b) sensor data set (`heidi`) (c) both data sets overlayed (initial position) (d) after affine registration (e) final deformed sensor data (f) final registered data sets

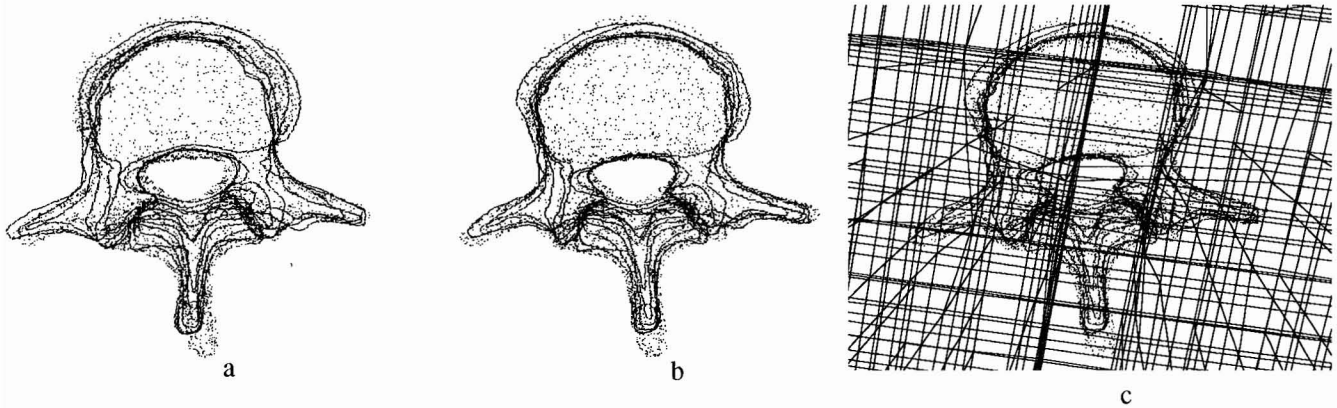

Figure 4: Registration of patient vertebra with plastic "phantom":
(a) after affine registration (b) after final local spline registration (c) selected lines in deformation spline

8. Discussion and Conclusions

In this paper, we have developed a technique for registering 3-D surfaces which are related by non-rigid deformations. Our approach represents the deformation using a combination of global polynomial deformations and local displacement splines (free-form deformations). We use an algorithm which directly minimizes the squared distances between the two surfaces, rather than identifying sparse features (e.g., ridge lines or feature points [15, 24]) and then trying to match them.

We believe that the direct matching of surfaces has better accuracy and removes the need for a feature detection stage, which may not always operate reliably. Two arguments which favored feature-based approaches in the past were computational complexity and global correspondence search. The computational complexity argument assumed that if a small set of features was found and matched (say dozens of discrete features or hundreds of ridge points, as opposed to tens of thousands of surface points), the overall complexity of the algorithm would be drastically lower. However, by using the octree spline distance map, the complexity of each iterative adjustment step in our algorithm is linear in the number of sensed surface points.

The second argument in favor of discrete features is related to the first, and examines the combinatorial complexity of certain discrete matching (correspondence) algorithms. In our approach, we use a modified gradient descent which avoids combinatorial search but only finds locally optimal matches. In practice, we have found that false local minima are usually far away from the true solution. In medical applications, *a priori* knowledge about position and shape is usually available, and manual intervention or guidance is often not a problem (e.g., in interactive analysis or surgical planning), so that only small displacements and local deformations have to be estimated.

Our approach embeds one of the surfaces being matched into a deformable space, rather than equipping it directly with elastic properties (as in [35, 16, 23]). While the latter approach may be more physically realistic if an anatomically and biomechanically correct model is constructed, our approach enables us to deal with arbitrary surfaces which may not even have a smooth connected representation. Volumetric deformations also yield auxilliary information about the motion of nearby structures which did not participate in the matching, e.g., registrations performed on certain easily identifiable brain structures can be used to estimate the registration for the whole brain. This suggests an incremental, model-driven segmentation process.

In theory, volumetric deformation models are of a higher complexity than elastic surface models, i.e., they represent the deformations over a dense region of 3-D, rather than over a 2-D manifold. However, the use of a multiresolution, coarse-to-fine strategy results in a very efficient matching algorithm. Furthermore, the use a adaptive spatial subdivision can make a volumetric representation such as the octree competitive with a surface representation for sufficiently smooth surfaces [29].

Our experiments to date have been performed on pre-segmented 3-D surfaces. We are planning to extend our algorithm to work directly with unsegmented 3-D medical images. Since we are matching a known anatomical model with patient data, one of the data sets will still be a segmented surface. Our choice is then to use gradients in the medical image to attract a deformable surface model (as in *snakes* [10]), or to run an edge operator over the medical image [25] and use this as the (noisy) surface to be deformed. In the latter case, our algorithm must be sufficiently robust to reject large number of outliers. We are currently studying such an extension to our basic algorithm [5]. Other areas of future investigation include an adaptive algorithm for deciding how to refine the local octree deformation spline, the choice of the order of the octree spline, and the choice of various smoothness constraints (regularizers).

To summarize, the technique we have developed allows us to rapidly register two segmented 3-D surfaces which are related by an unknown non-rigid deformation. Our use of octree splines makes this algorithm efficient, and our choice of weighted non-linear least squares makes it statistically optimal. Our algorithm has immediate applications to the registration of 3-D medical surfaces with anatomical models and to change detection in medical imagery. We are extending the algorithm to support the automatic model-based segmentation of 3-D medical images, and we expect this to further widen the applicability and utility of our approach.

References

[1] R. Bajcsy and S. Kovacic. Multiresolution elastic matching. *Computer Vision, Graphics, and Image Processing*, 46:1–21, 1989.

[2] Alan H. Barr. Global and local deformations of solid primitives. *Computer Graphics (SIGGRAPH'84)*, 18(3):21–30, July 1984.

[3] T. Beier and S. Neely. Feature-based image metamorphosis. *Computer Graphics (SIGGRAPH'92)*, 26(2):35–42, July 1992.

[4] P.J. Besl and N.D. McKay. A method for registration of 3-D shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2):239–256, 1992.

[5] E. Bittar, S. Lavallée, and R. Szeliski. A method for registering overlapping range images of arbitrarily shaped surfaces. In *SPIE Vol. 2059, Sensor Fusion VI*, Boston, September 1993. Society of Photo-Optical Instrumentation Engineers.

[6] F. L. Bookstein. Principal warps: Thin-plate splines and the decomposition of deformations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(6):567–585, June 1989.

[7] G. Borgefors. Distance transformations in digital images. *Computer Vision, Graphics, and Image Processing*, 34:344–371, 1986.

[8] L. Brunie, S. Lavallée, and Szeliski R. Using forces fields derived from 3D distance maps for inferring the attitude of a 3D rigid object. In G. Sandini, editor, *Second European Conference on Computer Vision (ECCV'92)*, pages 670–675, Santa Margherita, Italy, May 1992. Springer Verlag.

[9] D. J. Burt. A dynamic model for image registration. *Computer Graphics and Image Processing*, 15:102–112, 1981.

[10] I. Carlbom, D. Terzopoulos, and K. M. Harris. Reconstructing and visualizing models of neuronal dendrites. In N. M. Patrikalakis, editor, *Scientific Visualization of Physical Phenomena*, pages 623–638. Springer-Verlag, New York, 1991.

[11] G. Champleboux, S. Lavallée, R. Szeliski, and L. Brunie. From accurate range imaging sensor calibration to accurate model-based 3-D object localization. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'92)*, Champaign, IL, June 1992.

[12] P.-E. Danielson. Euclidean distance mapping. *Computer Graphics and Image Processing*, 14:227–248, 1980.

[13] A. C. Evans, W. Dai, L. Collins, P. Neelin, and S. Marett. Warping of a computerized 3-d atlas to match brain image volumes for quantitative neuroanatomical and functional analysis. In *SPIE Vol. 1445, Medical Imaging V*, pages 236–247, 1991.

[14] D. R. Forsey and R. H. Bartels. Hierarchical B-spline refinement. *Computer Graphics (SIGGRAPH'88)*, 22(4):205–212, August 1988.

[15] A. Guéziec and N. Ayache. Smoothing and matching of 3-D space curves. In G. Sandini, editor, *Second European Conference on Computer Vision (ECCV'92)*, pages 620–629, Santa Margherita, Italy, May 1992. Springer Verlag, LNCS Series Vol. 588.

[16] B. Horowitz and A. Pentland. Recovery of non-rigid motion and structure. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'91)*, pages 325–330, Maui, Hawaii, June 1991.

[17] P. J. Huber. *Robust Statistics*. John Wiley & Sons, New York, New York, 1981.

[18] B.A. Kall, P.J. Kelly, and S.J. Goerss. Comprehensive computer-assisted data collection treatment planning and interactive surgery. In *SPIE, Medical Imaging, Vol. 767*, pages 27–35, 1987.

[19] S. Lavallée. *Geste Medico-Chirurgicaux Assistes par Ordinateur : Application a la Neurochirurgie Stereotaxique*. PhD thesis, Grenoble University, France, December 1989.

[20] S. Lavallée, L. Brunie, B. Mazier, and P. Cinquin. Matching of medical images for computer and robot assisted surgery. In *IEEE EMBS Conference*, pages 39–40, Orlando, Florida, November 1991.

[21] S. Lavallée, R. Szeliski, and L. Brunie. Matching 3-D smooth surfaces with their 2-D projections using 3-D distance maps. In *SPIE Vol. 1570 Geometric Methods in Computer Vision*, pages 322–336, San Diego, CA, July 1991.

[22] S. Leitner, I. Marque, S. Lavallée, and P. Cinquin. Dynamic segmentation : finding the edge with spline snakes. In P.J. Laurent, editor, *International Conference on Curves and Surfaces*, Chamonix, 1991. Academic Press.

[23] T. McInerney and D. Terzopoulos. A finite element model for 3D shape reconstruction and nonrigid motion tracking. In *Fourth International Conference on Computer Vision (ICCV'93)*, Berlin, Germany, May 1993.

[24] O. Monga, S. Benayoun, and N. Ayache. From partials derivatives of 3D density images to ridge lines. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'92)*, pages 354–359, Champaign, IL, June 1992.

[25] O. Monga, R. Deriche, G. Malandrain, and J. P. Cocquerez. Recursive filtering and edge closing : Two primary tools for 3D edge detection. In *First European Conference on Computer Vision (ECCV'90)*, pages 56–65, Antibes, France, April 1990. Springer-Verlag.

[26] D. W. Paglieroni. Distance transforms: Properties and machine vision applications. *CVGIP: Graphical Models and Image Processing*, 54(1):56–74, January 1992.

[27] C.A. Pelizzari, G.T.Y. Chen, D.R. Spelbring, R.R. Weichselbaum, and C-T. Chen. Accurate 3-D registration of CT, PET, and-or MR images of the brain. *J. Computer Assisted Tomography*, 13(1):20–26, 1989.

[28] W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling. *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge University Press, Cambridge, England, second edition, 1992.

[29] H. Samet. *The Design and Analysis of Spatial Data Structures*. Addison-Wesley, Reading, MA, 1989.

[30] C. Schiers, U. Tiede, and K.H. Hohne. Interactive 3D registration of image volumes from different sources. In H.U. Lemke, editor, *Computer Assisted Radiology, CAR 89*, pages 667–669, Berlin, June 1989. Springer-Verlag.

[31] T. W. Sederberg and S. R. Parry. Free-form deformations of solid geometric models. *Computer Graphics (SIGGRAPH'86)*, 20(4):151–160, August 1986.

[32] R. Szeliski. *Bayesian Modeling of Uncertainty in Low-Level Vision*. Kluwer Academic Publishers, Boston, MA, 1989.

[33] R. Szeliski. Fast surface interpolation using hierarchical basis functions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(6):513–528, June 1990.

[34] R. Szeliski and D. Terzopoulos. Parallel multigrid algorithms and computer vision applications. In *Fourth Copper Mountain Conference on Multigrid Methods*, pages 383–398, Copper Mountain, Colorado, April 1989. SIAM.

[35] D. Terzopoulos and D. Metaxas. Dynamic 3D models with local and global deformations: Deformable superquadrics. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(7):703–714, July 1991.

[36] D. Terzopoulos, A. Witkin, and M. Kass. Constraints on deformable models: Recovering 3D shape and nonrigid motion. *Artificial Intelligence*, 36:91–123, 1988.

[37] W. Welch and A. Witkin. Variational surface modeling. *Computer Graphics (SIGGRAPH'92)*, 26(2):157–166, July 1992.

[38] A. Witkin and W. Welch. Fast animation and control of nonrigid structures. *Computer Graphics (SIGGRAPH'90)*, 24(4):243–252, August 1990.

[39] H. Yserentant. On the multi-level splitting of finite element spaces. *Numerische Mathematik*, 49:379–412, 1986.