The Visual Computer

Image interpolation using unions of spheres

Roger C. Tam, Alain Fournier

Department of Computer Science,
University of British Columbia, 2366 Main Mall,
Vancouver, British Columbia, Canada, V6T 1Z4
E-mail: {rtam,fournier}@cs.ubc.ca

Numerous algorithms for two-dimensional image interpolation have been developed. Many such techniques are unstable with respect to changes in the input data, or they rely on the user to specify object features and even manually determine matches between features. We present a novel algorithm for image interpolation aimed at addressing these issues. We use pixel intensity as the third dimension to form a *union of spheres* volumetric model of an image. The technique requires little preprocessing and is not labour-intensive. It allows the user a reasonable degree of control of the matching process. Our method easily facilitates intuitive manual specification of features where desired. The results of our tests are very positive. The problems encountered in using our algorithm are largely predictable and suggest definite directions for future work.

Key words: Feature representation – Feature matching – Image interpolation – Shape interpolation – Union of spheres

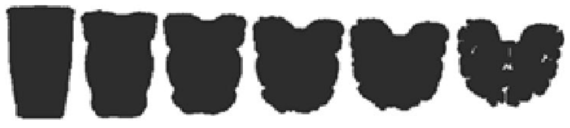
Correspondence to: R.C. Tam

1 Introduction

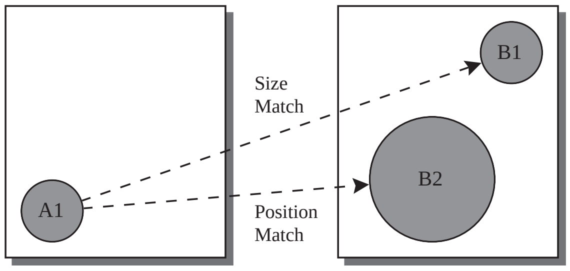
The general objective of two-dimensional (2D) image interpolation is to derive a number of intermediate images between a beginning image and an end image. There are numerous applications for image interpolation, ranging from aesthetic purposes (e.g. morphing) to scientific visualization (e.g. 3D volumetric reconstruction from 2D slices). Although the requirements for an interpolation algorithm vary somewhat with the application (for example, in medical imaging, accuracy is more important than aesthetic appeal), there are a number of elements considered desirable in most algorithms.

The interpolation of images requires a matching of features that can humanly be identified as having a certain degree of similarity. Most people would consider the shapes of objects to be one of the most important criteria for feature matching. In order for an algorithm to generate "good" interpolations, it must take into account similarities in shape. Although shape is a very intuitive and commonly used property, it is also very difficult to define. Because the definition of shape itself tends to be imprecise, it is difficult to explicate what constitutes a good interpolation between two objects. For example, most people would agree that the series of images in Fig. 1 (Ranjan and Fournier 1996), showing a vase morphing to a cross-section of the brain, represents a good interpolation, but few can describe in exact terms the properties that make it so. Usually, shape is given as an important factor, but rarely with a precise definition of the term.

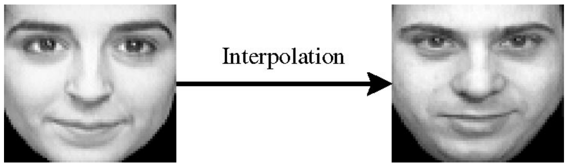
Although it is difficult to describe exactly what good shape interpolation is, there are a number of properties that are generally considered desirable. Because the quality of an interpolated image is judged by comparison with the start and end images, a particularly important property is one that concerns the preservation of similarities between objects. In an interpolation between two objects, the intermediate forms should preserve as many of the similarities between the original objects as possible. For example, if the two objects have approximately the same area, the interpolated forms should not be significantly larger or smaller. The similarity property also applies to the image as a whole. For example, similarities in the positioning and orientation of objects relative to each other should be preserved as much as possible.



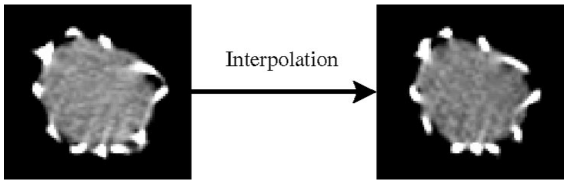
1



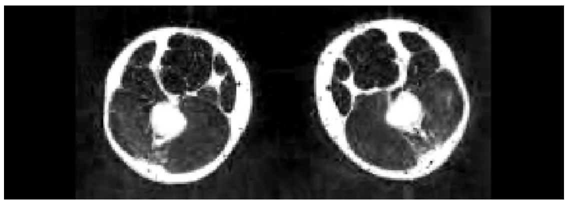
2a 2b



3a



3b



3c

Fig. 1. An example of good shape interpolation: morphing from a vase to a cross-section of the brain

Fig. 2a, b. An example showing the need for user control in the matching

Fig. 3a–c. Test images: **a** faces; **b** CT slices; **c** Visible Man legs

Partly because of their importance to image interpolation, the representation and matching of shapes have been the focal points of intense study. Many successful algorithms have been developed to interpolate between objects, but many of these methods rely on the user to specify the features and often the correspondences between them. Manual specification of features can be a labour-intensive task, especially when there are many objects or very complex objects in an image. In contrast, there are algorithms that attempt to automatically extract all of the matchable features from images. Some commonly used features include points, edges, corners and skeletons. These algorithms try to extract object features either directly from the image data or from object boundaries extracted from the image by a segmentation algorithm. A common problem with feature extraction methods is instability with respect to changes in the input data. For example, rescaling the intensities in a greyscale image can cause an edge detection algorithm to output a different set of edges. Both object segmentation and feature extraction can have stability problems. Ideally, an algorithm should require little or no user assistance in forming a representation of the image features, but should have enough stability to be able to handle reasonably large variations in input data.

Once the computer representation of the image features is formed, the matching can begin. As mentioned, the requirements for a good interpolation are application- and user-dependent, so an algorithm should allow users flexible control of the matching process. For example, object A1 in Fig. 2a should be matched to object B1 in Fig. 2b if the main criterion for matching is object size, but the same object in Fig. 2a should be matched to object B2 in Fig. 2b if the main criterion is object position.

In this paper, we present a new approach to image interpolation that focusses on good shape interpolation, as well as stability and flexible control of feature matching. The central idea is to represent each image by a *union of spheres* (UoS) volumetric model. After a UoS is created, a simplification process (*clustering)* is used to reduce the number of spheres in the model. The spheres from the two simplified UoSs are then matched, constrained by a number of user-controlled parameters. The matches are used to produce interpolated UoSs, from which the intermediate images are derived.

There are a number of main advantages to using this approach. All of the image features are "captured" in the UoS representation without having to be explicitly extracted from the image. The positions and sizes of the spheres and the relationships between the spheres and their neighbours represent very well the shapes of objects in the image, as well as representing the relationships between objects. Thus, for many applications, our algorithm does not require any user input for the specification of image features. In such cases, even the selection of an isovalue (i.e. thresholding), a task required in many algorithms for generating boundary points, is not necessary. Although our method is designed to be largely automatic, for some applications, such as aesthetic morphing, augmenting the algorithm with some user specification of features may enhance the results. In such cases, manual specification is done in a preprocessing step by rescaling the pixel intensities of the features that need to be highlighted. An example of how this is done is given in Sect. 3.

Ranjan and Fournier (1994) show that UoS models are stable with respect to changes in the input data. No algorithm is immune to very large changes – features can actually appear or disappear from the image as a result of such changes. However, in most cases that we have studied, the UoS representation allows for the reliable and intuitive prediction of how the model will change in response. This allows the user to compensate by adjusting the parameters in the simplification and matching processes. In cases where user specification of features is used, the stability provided by the UoS representation allows the algorithm to be reasonably forgiving of human error.

Our algorithm allows for user control at two very important stages. First, the user can specify the degree of simplification to be done in the clustering process. This corresponds to specifying the level of detail to be used in the feature matching. Second, the user can adjust several other parameters to tune the matching process to suit the application at hand. These parameters are explained in detail in Sect. 2.

Figure 3 shows the images used as test cases in this paper. For simplicity, we are only dealing with greyscale images for this paper. We have chosen these particular test cases to demonstrate the wide range of input data that our algorithm can effectively handle. Our data set consists of images from three imaging modalities, and gives the reader a good indication of the potential applicability of our method.

Figure 3a shows two images of faces; the left image (face 1) will undergo a morph to become the right image (face 2). We use faces because morphing faces is generally considered a difficult task – even minor artifacts in the interpolated images are easily noticed by a human observer. In addition, lighting effects such as shadows cause problems in many automatic interpolation methods. In this test case, some effects typical of photographic images, such as darker patches in certain areas of each face, are clearly visible. These effects are a good test of the robustness of our algorithm, in particular the stability of the UoS model. After illustrating the basic technique, we use the face data to show how a small amount of user feature specification can be used to overcome some of the inherent limitations of automatic methods.

Figure 3b shows two consecutive slices from a computed tomography (CT) data set. Each slice shows a cross-section of an aorta (the largest artery in the human body) surrounded by the wire supports of a stented graft implant. These supports are visible as small, bright white patches around the circumference of the blood vessel. Comparing the second slice to the first, we can see that some of the wires move closer together, while others move farther apart. The shapes of some of the wires also change. We use the CT slices as an example of data that is particularly well suited for use in our algorithm. Lighting effects are not a concern. The features are fairly well defined, but it would still be very time-consuming if the user had to specify the boundary of each small wire support. This is especially true when some of the wires are partially embedded in the aorta, making the definition of boundaries between these wires and the aorta difficult. The problem is further compounded by the fact that if a data set has many slices, it would be difficult to maintain consistency in the segmentation between slices. We show that our algorithm performs well in interpolating between the CT images with no user specification of features and very little user input overall. The intermediate slices generated by our algorithm can be used for volumetric reconstruction, as shown in Sect. 3.

Fig. 3c shows a cross-sectional view of a man's upper thighs. This image, from the Visible Man data set, is a photographed image of a physical cross-section of a cadaver. To demonstrate the UoS feature matching capabilities, we take the mirror image of the left leg and allow the algorithm to automatically register it with the right leg. We use this image because there are enough similarities between the two legs for a human observer to be able to qualitatively judge the results of the registration procedure. At the same time, the two legs are different enough to pose an interesting challenge for any algorithm.

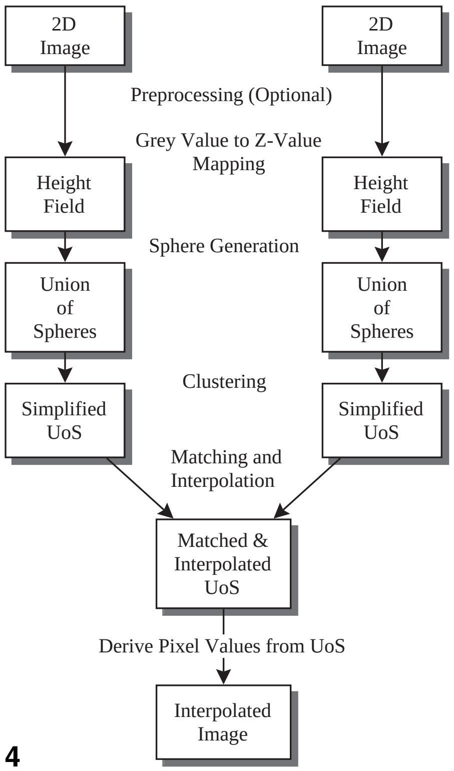
1.1 Related work

There is a large amount of literature that is related to the work described in this paper, particularly in the areas of feature representation and shape interpolation. Therefore, only a brief summary of the major techniques is given here.
In general, there are two classes of algorithms that perform shape interpolation. Some algorithms first establish a correspondence between primitives in the two objects, then interpolate between the matched primitives; our algorithm falls into this category. Others do not match explicitly, but use a global transformation that acts on all of the primitives simultaneously. Many algorithms in the first class have the problem of being unstable with respect to the input data. The main problem associated with the second class is that it is very difficult to control the intermediate shapes. Our algorithm is designed to provide stability without sacrificing control.
A number of successful shape interpolation algorithms that work in object space have been developed. Object space algorithms work on models of objects (e.g. polygons, boundary points) and are not concerned with the often difficult task of extracting the models from images. An example of an object-space morphing algorithm is presented by Carmel and Cohen-Or (1997). Their algorithm requires a user to match individual points manually. An automatic object-space method for feature representation and shape interpolation using unions of circles is presented by Ranjan and Fournier (1996). There are some similarities to our new algorithm, but the Ranjan and Fournier (1996) algorithm works completely within the 2D domain. Our paper presents a method that starts in image space, automatically forms 3D models to represent the images in object space, and uses the models for interpolation. The UoS model used in this paper for the representation of volumetric data was developed by Ranjan and Fournier (1994).
The most popular methods of image interpolation use specialized matching primitives drawn by the user, such as strokes, skeletons or line segments. Examples of this type of algorithm include work by Beier and Neely (1992) and Lee et al. (1995). In the case of Lee et al. (1995), energy-minimizing splines are used to assist the user in specifying image features. These algorithms are very effective in dealing with object features that are well defined and relatively straightforward to specify interactively. The facial images in Fig. 3a are examples of images with such features. The CT images in Fig. 3b are examples of images for which interactive specification of features can be problematic and labour intensive because some of the wires have fuzzy boundaries.
Many techniques have been developed for the automatic extraction of image features. These methods range from relatively simple ones, such as thresholding and edge detection, to more complex ones, such as skeleton extraction (e.g. Attali and Montanvert 1997; Brandt and Algazi 1992). As evidenced by the large number of techniques available, automatic feature extraction is a difficult task and often a source of instability. Our use of the UoS model allows image features to be represented without being explicitly extracted.
There is a significant overlap in the objectives and techniques for image interpolation and image registration. The reader is referred to the paper by Brown (1992) for a good survey of registration methods.

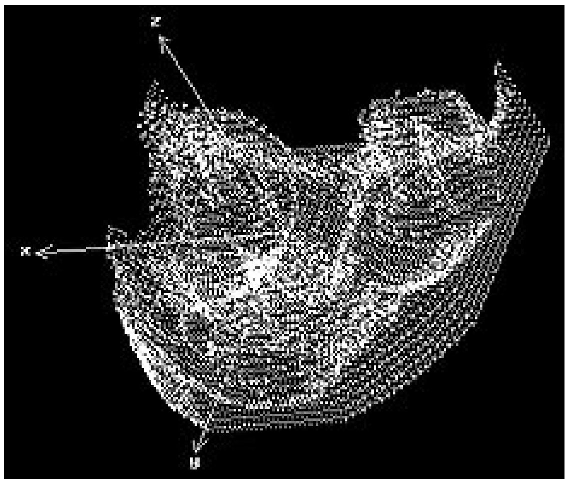
2 Methodology

This section describes the main steps of our interpolation algorithm. Steps 3 to 5 are explained in more detail in two papers by Ranjan and Fournier (1994, 1996), so they are only summarized here. As illustrated in Fig. 4, the main algorithm steps are:

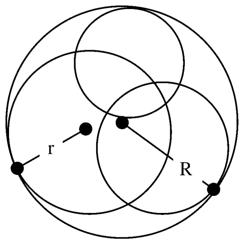
1. (Optional). Preprocess each image for input into the interpolation algorithm (e.g. user feature specification, scale pixel intensities, noise removal, etc.).



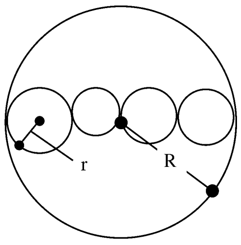
4



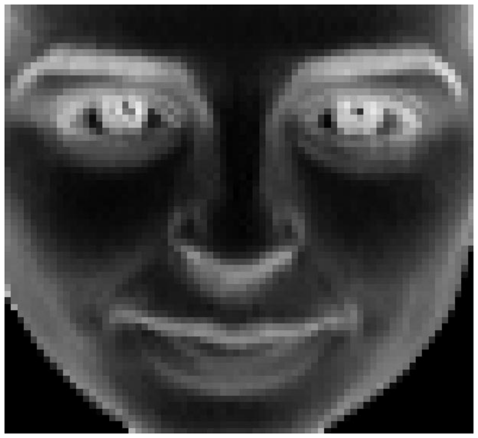
6



7a



7b



5

Fig. 4. The main steps of the algorithm for image interpolation using unions of spheres (UoSs)

Fig. 5. Face 1 with pixel intensities inverted

Fig. 6. Boundary points of the face 1 volume

Fig. 7. Definition of sphericity for $(\frac{r}{R})$ clusters of circles: **a** large sphericity; **b** small sphericity

2. Generate the height field for each image from pixel intensity data, and use the resulting point set as boundary points for volume.
3. Generate UoS models from the two boundary point sets.
4. Simplify the UoS models by clustering.
5. Match the spheres between the two simplified UoS models.
6. Generate intermediate UoS models (one for each intermediate frame in the interpolation).
7. Generate image data from each intermediate UoS model.

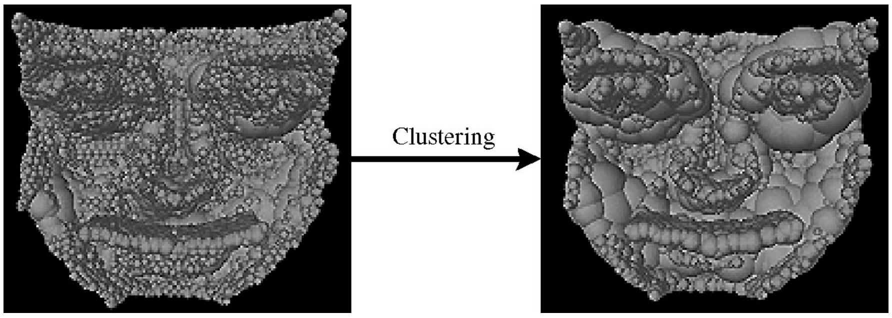
The amount of preprocessing required is dependent on the application and the original image characteristics. For example, in the facial images shown in Fig. 3a we notice that most of the important features (e.g. eyes, eyebrows, outline of the nose, mouth, etc.) are lower in pixel intensity than the rest of the face. To ensure that these features are well represented in the UoS models, we invert the pixel intensities, so that the darker features become the lighter ones. Figure 5 shows face 1 from Fig. 3a with the pixel intensities inverted. The height field generated from this image would have peaks where the important features are. In contrast, the CT slices in Fig. 3b do not need to be preprocessed at all for input into our algorithm. In this case, the most important features to be matched are the graft wire supports that appear as bright white spots, which naturally become peaks in the height field.

The generation of the height field is relatively straight-forward. All that is required is a mapping from the intensity value at each pixel to a z-value in the height field. A simple linear mapping works well in most cases. Scaling the z-values is often useful for controlling the number of spheres in the derived UoS. The points in the height field are then used as the boundary points of a volume. This volume is bounded by the image plane. Figure 6 shows the boundary points generated from the face 1 image. The important facial features already mentioned are visible as peaks in the height field. (In this case, the positive z direction is perpendicular to the face and points in the same direction as the nose.)

After the boundary points are generated, the UoS model can be formed. There are three basic steps to forming a UoS from a set of boundary points. The first step is to compute the Delaunay tetrahedralization of the point set. The second step is to compute the circumscribing sphere of each tetrahedron. The last step is to discard all spheres that are "outside" the object. The remaining spheres form the UoS. The Delaunay tetrahedralization is used because of its empty sphere property, which guarantees that each sphere contains exactly four boundary points, all of which lie on the surface of the sphere. Since a sphere cannot contain any other boundary points, all voxels contained within a sphere must be either inside the object or outside the object. In other words, no sphere contains both inside and outside voxels. Using this property, we have a quick test for determining whether a sphere is inside or outside the object. For each sphere, we only need to test one of its voxels to decide whether the sphere is outside and should be discarded.

The next main step in our algorithm is a simplification process aimed at reducing the number of spheres while preserving the features as much as possible. The degree of simplification corresponds to the level of detail in the UoS model. Reducing the number of spheres has the advantage of increased speed in the matching and visualization processes. The simplification algorithm replaces clusters of spheres within the UoS with larger encompassing spheres. Hence the process is called clustering. The degree of simplification is controlled by a user set parameter called *sphericity*, which is a measure of how well a set of spheres can be modelled by a single sphere. Mathematically, the sphericity of a cluster of spheres is defined as the ratio of the radius of the largest sphere in the cluster to the radius of the smallest sphere enclosing all spheres in the cluster. Figure 7 shows how sphericity is defined for circles; the extension to spheres is trivial.

The clustering algorithm processes the spheres in order of decreasing size. In each iteration, the algorithm takes the largest unprocessed sphere **a**, and calculates the smallest sphere encompassing **a** and as many other unprocessed spheres as possible, under the constraint that the cluster must have a sphericity greater than or equal to the user-chosen threshold. The cluster is then replaced by the newly formed encompassing sphere. Ranjan (1996) shows that the distance between the surface of a simplified UoS and the original point set is bounded; therefore, clustering is guaranteed not to distort the original image features beyond what is expected at a given sphericity. In addition, the clustering process is very effective at greatly simplifying areas of low detail while preserving high detail where required. For example, Fig. 8 shows the unclustered and clustered UoSs of face 1. The unclustered version has about 7000 spheres, whereas the clustered one has about 700. Even with this great reduction in the number of spheres, the important facial features are still well represented in the simplified UoS. Notice that, while most of the smaller spheres in the unsimplified UoS are clustered and replaced by larger spheres in the simplification process, some small spheres still appear in the clustered UoS, es-



8

Fig. 8. Clustering of the face 1 UoS model, reduction from ~7000 to ~700 spheres

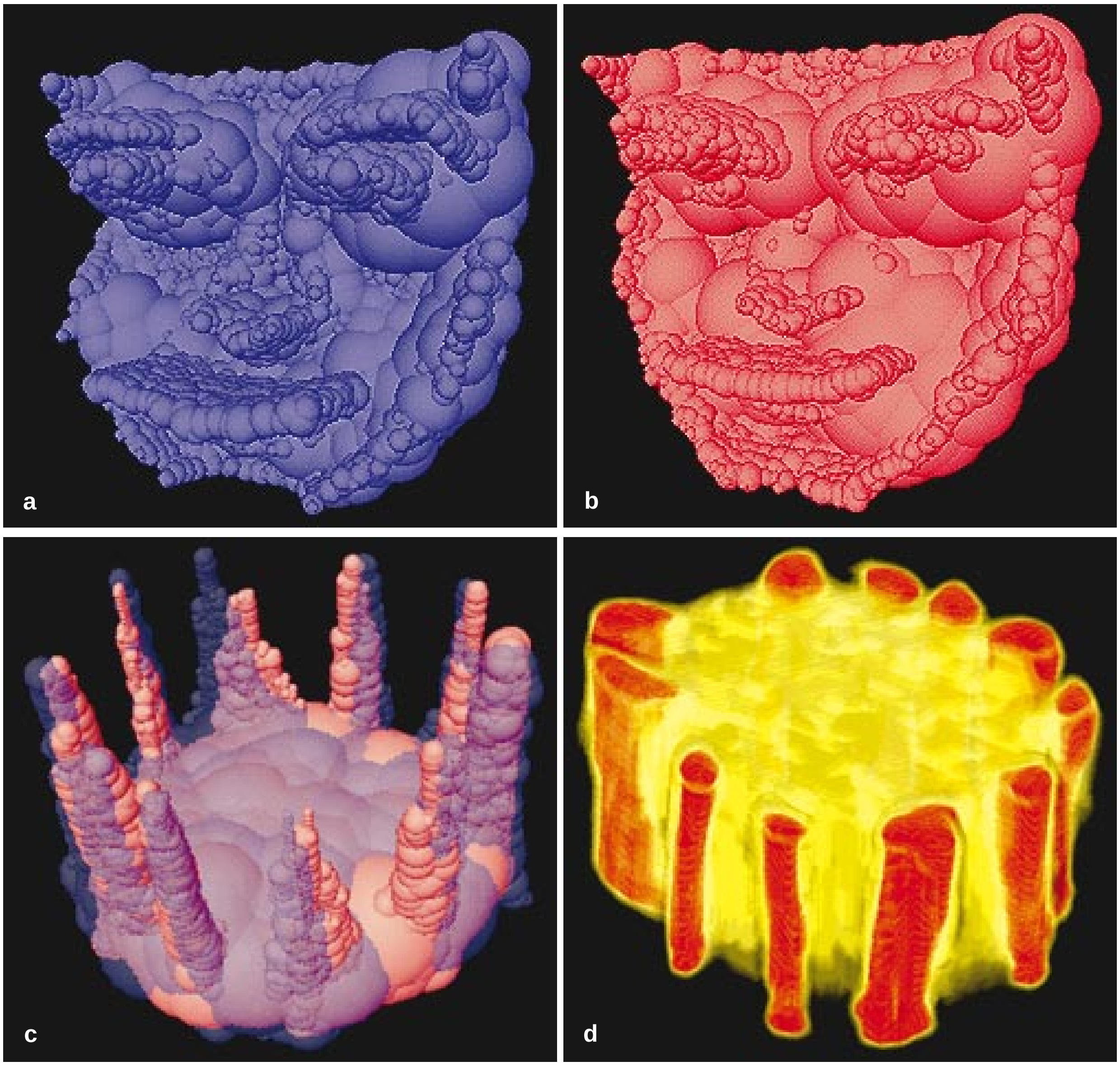


Fig. 9. **a** Union of spheres (UoS) model of face 1 – the facial features are well represented and clearly visible; **b** UoS model of face 2; **c** UoS models of two CT slices superimposed to show their differences; **d** volume rendering of interpolated CT slices

pecially in the areas with greater detail in the original image, such as the eyebrows, eyes and mouth.

Visualization of the UoS after simplification (using Open Inventor) to assess qualitatively how well the image features are being represented can help the user determine if the current sphericity value is appropriate. For example, Fig. 9a, b show the simplified UoSs of face 1 and face 2, respectively. The sphericity used in this case (0.65) seems to result in a suitable simplification of both volumes; the number of spheres (approximately 700 in both cases) is quite manageable, but the features of the two faces are present in enough detail that the user can see clearly which groups of spheres from face 1 should be matched to which in face 2.

Once the simplification process is complete, the matching process between the two UoSs can begin. The first step in the matching process is the calculation of the distances $d(a, b)$ between every a and b, where a is a sphere in the first UoS, and b is a sphere in the second. We define the distance between two spheres as a function of the differences in their locations, sizes and characteristic *features*. The definition of a feature in this case is a mathematical relationship between a sphere and its four largest neighbours. We use four neighbours because, in an unsimplified model, each sphere has a maximum of four neighbours. In a simplified model this choice is somewhat arbitrary. Between a sphere and each neighbour, we take the gradient $\frac{dR}{dD}$, where dR is the signed difference between the radius of the sphere and the radius of the neighbour, and dD is the unsigned distance between the centres of the spheres. Figure 10 shows the 2D analog using circles. The gradients in the directions of the four largest neighbours of a sphere form the feature of that sphere. If a sphere has less than four neighbours, the value for each missing neighbour is set to $-\infty$ because in this direction the neighbouring sphere shrinks to 0 for any distance moved. The $\frac{dR}{dD}$ value is then mapped to the range [0, 2], where $-\infty$ is mapped to 0, 0 to 1, and $+\infty$ is mapped to 2.

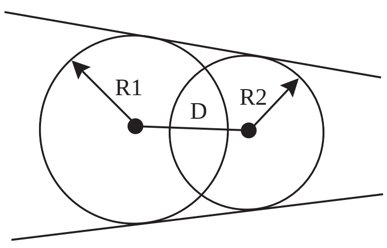
The distance between the features of two spheres can best be explained by a physical analogy. If the two features have a common centre and are free to rotate around it, and if, between the extremities of each pair of "branches" (in the directions of neighbours), there is a spring that has a pulling force proportional to its length, then the system will be at rest when the potential energy is at a minimum. The sum of the residual distances between the branch extremities in this minimum energy state is taken to be the feature distance between the two spheres. Fig. 11 illustrates the feature distance between two circles, where the feature of a circle is defined by its relationship with its three largest neighbours.

The overall distance between two spheres a, b is given by:

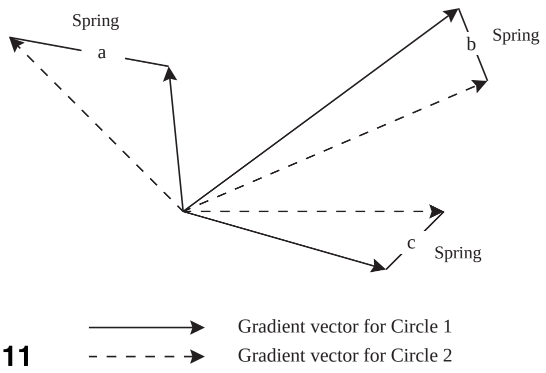
$$d(a, b)=w_p\, d_p(a, b)+w_s\, d_s(a, b)+w_f\, d_f(a, b)$$

where $d_p(a, b)=(x_a-x_b)^2+(y_a-y_b)^2+w_z(z_a-z_b)^2$, $d_s(a, b)=(r_a-r_b)^2$, and $d_f(a, b)$ is the feature distance between a and b. The user controls the matching process by setting values of w_p, w_s, w_f and w_z; w_p is the position weight, w_s is the size weight, w_f is the feature weight, and w_z is the z-value weight, normally only used when there is a large difference in the z-value ranges of the two UoSs. For example, to match only by sphere size, the user would set $w_s>0$, but all other weights to 0. Appropriate scaling of the height fields before UoS generation makes w_z somewhat redundant. The primary reason for not setting w_z to 1 permanently is that trying different scale factors for the height fields can be time consuming because a new UoS needs to be generated for each new scale factor. With w_z, the user needs only to estimate the scale factors roughly and can simply use w_z to compensate for differences in the resulting height ranges.

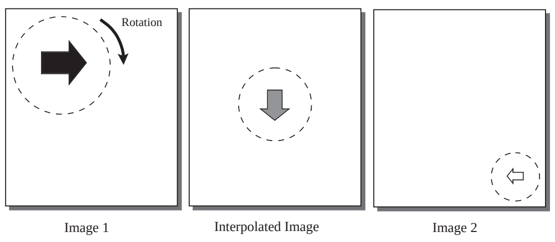
The user selects the weights w_p, w_s, w_f and w_z by experimentation and visualizing the UoSs. For example, Fig. 9c shows the UoSs computed from the CT data shown in Fig. 3b. In this visualization, the two UoSs are superimposed, with the transparency of one set to 50% to give the user a good idea of how the spheres should be matched. In this case, w_p, w_s and w_z should be relatively large (0.90 to 1.0), with w_f somewhat smaller (0.25 to 0.50) because we can see that the spheres that should be matched are quite close in position and size. In contrast, getting the desired matches between the two UoSs shown in Fig. 9a, b would require a larger feature weight because the neighbourhood context of the spheres is more important for matching in this case. After matching with some initial values for the weights, a number of interpolated UoSs can be formed and visualized; the user can then make adjustments if necessary. We find that ani-



10



11



12

Fig. 10. The gradient for circle 1 in the direction for circle 2

Fig. 11. The feature distance between two circles (a+b+c)

Fig. 12. Computing an image from the interpolated union of speres (UoS)

mating the interpolation is especially useful in helping the user find appropriate values. Normally, close to optimal values are attained within several iterations.

After all the distances between spheres have been calculated, a bipartite graph is built in which the nodes correspond to the spheres and the weights on the edges are the distances between them. A maximum match is computed so that the sum of the distances between all matched pairs is a minimum. If the number of spheres in the two UoSs are not the same, there will be a number of unmatched spheres on one side. These spheres can be dealt with in a number of ways. For example, they can simply be matched to their nearest neighbours. In other cases, the number and/or locations of the spheres may be such that they do not affect the appearance of the derived image. In such cases, the spheres can be discarded.

The interpolation step comes next in our algorithm. For each pair of matched spheres a and b, a number (the number of frames required in the interpolation) of intermediate spheres are produced. The position and size of each intermediate sphere are obtained by linear interpolation from the matched spheres. In addition, because the features of a and b have specific orientations, the intermediate sphere should be rotated to reflect the change in orientation. The degree of rotation is also linearly interpolated. For most spheres on the positive z surface, the axis of rotation is likely to be close to the z-axis (i.e. within 30 degrees) because the features of these spheres represent 2D image features.

The final step in the algorithm is the generation of the images from the interpolated UoSs. The basic idea is to associate a pixel with a particular location on a sphere, and track the movement of that pixel as the sphere moves, scales and rotates across frames in the interpolation. This is done by projecting the image onto the upper surface of the UoS. For each pixel in the interpolated image, the algorithm finds the corresponding pixels in the original two images by comparing the locations, sizes and orientations of the associated spheres. The final value for that pixel is linearly interpolated from the two values in the original images. Figure 12 shows a simple example of how this process works. Consider the arrow-shaped pattern in the centre of the interpolated image. The 2D projection of the surface sphere at that location is shown as a dotted circle. The 2D projections of the matched spheres are shown in Images 1 and 2. The sphere from Image 1 moves to the right and down, gets smaller in size, and rotates about the z-axis as we move through the frames of the interpolation. The pixel values of the arrow in the centre image are interpolated from the arrows in Images 1 and 2.

Some problems may occur if there are areas where spheres have an axis of rotation that is far from the z-axis because, in such cases, parts of spheres not initially associated with any pixels may appear at the surface of an interpolated UoS. This may cause artifacts in the image. So far, our tests have not had these problems to any significant extent. This is due to the fact that, for any given sphere in our UoS model, its four largest neighbours are likely to have very similar z-values. However, we intend to investigate the possibilities of the occurrence of this type of artifact further, as well as means of "filling in" such areas.

3 Results

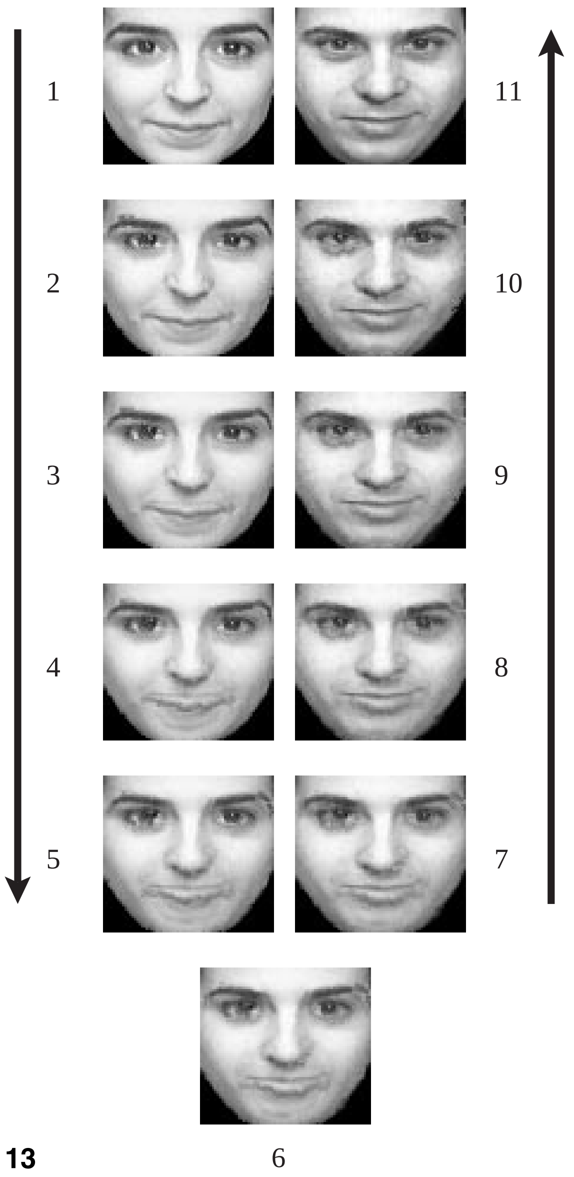
As shown in Fig. 3, the test images we use for this paper are a pair of faces, two consecutive CT slices and a cross-section of a man's legs. These test cases are used to demonstrate the capabilities and limitations of the method, as well as the wide range of potential applications of our technique. This section describes the results of applying our algorithm to the test images. We believe the results presented in this paper show great potential for practical applications of the method.

Figure 13 shows the facial images produced with our interpolation algorithm without any user specification of features. Frame 1 in the sequence is face 1, and frame 11 is face 2. The most obvious observation is that all of the intermediate frames look like human faces. A human viewer normally focusses on areas such as the eyes, eyebrows, nose, mouth and the curvature of the face, all of which are reasonably well interpolated, as shown in the intermediate frames (2–10). For example, the nose gets larger gradually, the eyes get smaller, the eyebrows change shape and move towards the eyes in a smooth manner, and the face gets thinner without getting jagged. This is a good result, especially considering that no user specification of features is used.

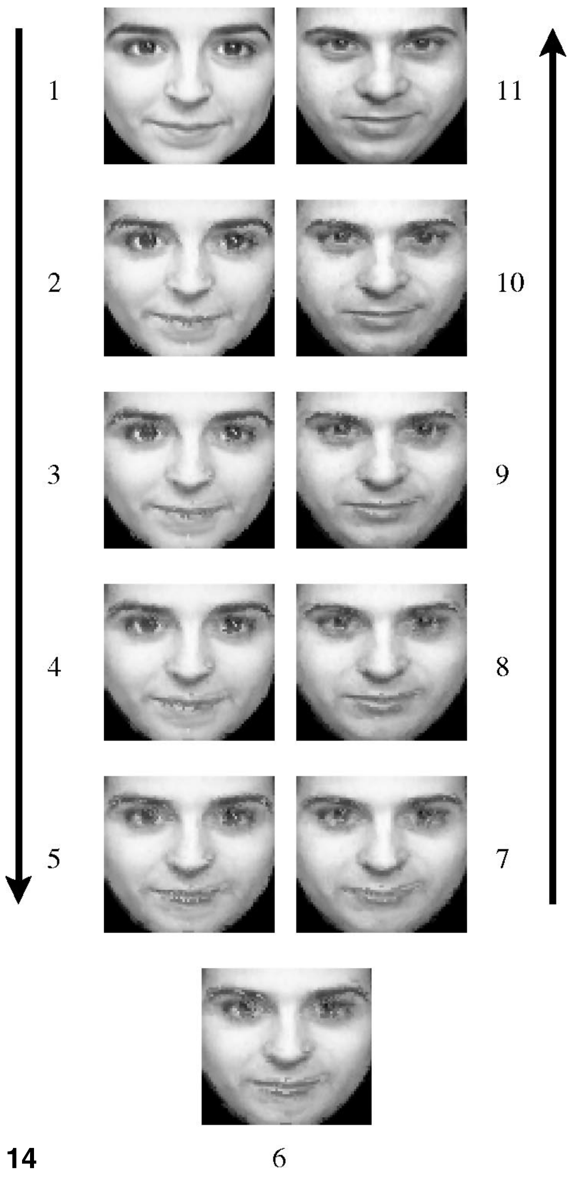
However, a number of artifacts are visible. The most noticeable problem is that the upper lip area of face 1 gets matched to the lower lip of face 2, causing a strange "flipping over" of the upper lip to form the lower lip. The main reason for this is that the lower lip of face 1 is much smaller than the rather prominent lower lip of face 2. In addition, the bottom edge of the lower lip of face 2 is similar in shape to the upper lip of face 1. Most automatic methods would have problems with this type of situation because most algorithms do not know the difference between upper and lower lips. In our case, the problem can be corrected by a small amount of manual feature specification. The mouth can be forced to match properly simply by increasing the pixel intensities in the area *between* the lips to highlight this region in both faces. In our case, we use a drawing program to "paint" a white line between the lips in the two original faces. Figure 14 shows the interpolation done with this minor modification. The lips are now interpolated nicely. If the user wants to further enhance other parts of the interpolated images, he is free to manually specify other features as desired.

For the CT data, the most important matches are in the graft wire supports that appear as small, bright white patches around the circumference of the aorta in the original images. As can be seen in Fig. 3b, some of the wires move toward one another, while others move apart. In addition, the aorta and some of the wires change shape between the two images. The goals of this interpolation are very similar to that of the contour correspondence problem (Meyers et al. 1991), where the matching and interpolation of the shapes of contours are primary objectives. Figure 15 shows the results from applying our algorithm. Frame 1 is the first original slice, frames 2 to 10 are interpolated images, and frame 11 is the second original slice. Even with no manual feature specification, our algorithm effectively interpolates between the two original slices. In order to further assess the quality of the interpolation, we perform a 3D volumetric reconstruction using the new slices. Figure 9d shows a volume rendering of all 11 slices. It is very clear in this figure that the wire supports are not parallel, which reflects what is seen in a real graft. Using our interpolation method, we are able to obtain a reconstruction with great detail using only two initial slices.

The data for our third test case is the image of the two legs shown in Fig. 3c. The mirror image of the left leg and the image of the right leg are the input to our algorithm. Our software forms UoS representations of the images, performs the sphere matching, then calculates the transformation ma-



13

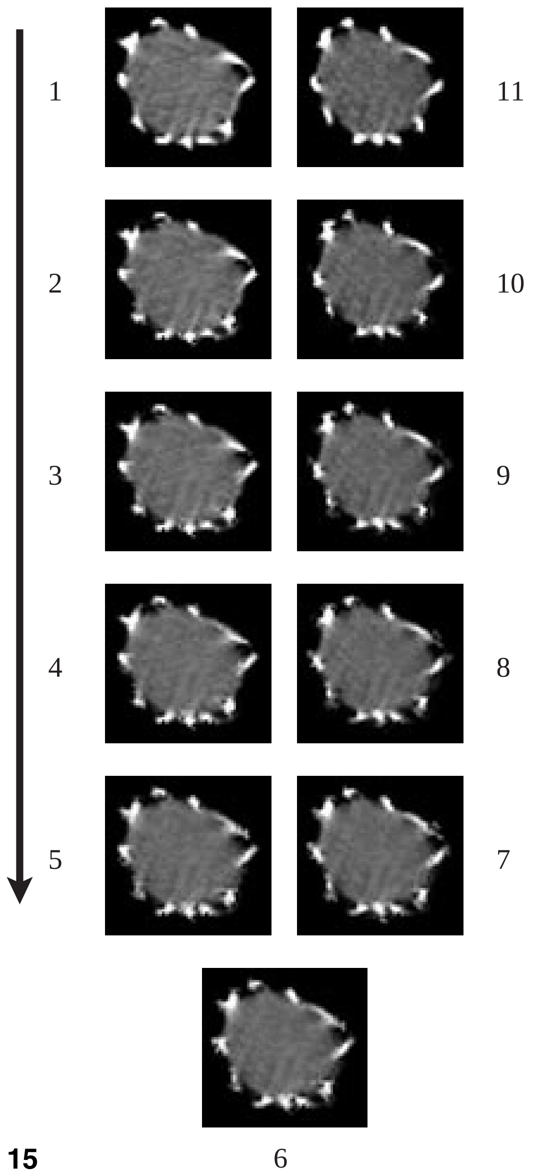


14

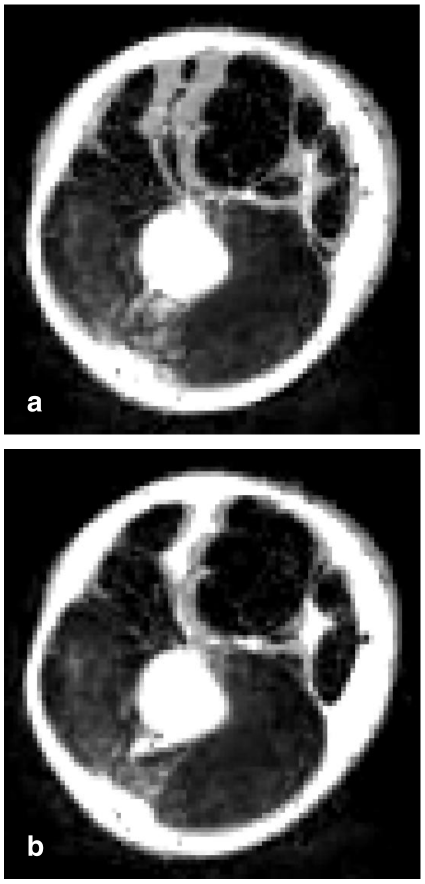
Fig. 13. Interpolation between face 1 and face 2 with no manual feature specification

Fig. 14. Interpolation between face 1 and face 2 with manual feature specification for the lips

trix that should be applied to the left side to register the two images. Figure 16a shows the mirror image of the left leg manually superimposed on the image of the right leg. This shows that one of the transformations should be a clockwise rotation of left leg about the z-axis. Our results are very positive because the transformation matrix calculated by our algorithm performs a translation and a rotation (6.6 degrees) about the z-axis and results in the image shown in Fig. 16b. Qualitatively speaking, the two legs are well registered, with similar features very close together. As with the CT data, no manual feature specification is required for the matching process.



15



16

Fig. 15. Interpolation between CT slices

Fig. 16a, b. Visible Man legs: **a** unregistered **b** registered

3.1 Processing time

This subsection gives an example of the amount of processing time required by the current version of our algorithm to create a simplified UoS model from an image, match the features between two UoSs, and render a number of interpolated frames. The facial images shown in Fig. 3a are used for the timing tests. Each image is of size 72×66×16 (width×height×depth) bits. The number of spheres in each unsimplified model is about 7000, and the number of spheres in each simplified model is about 700. The final output of the algorithm consists of 11 images, 9 of which are interpolated (the start and end images are re-rendered from their respective UoSs to make sure they match the original images). Our tests are performed on a Silicon Graphics Indigo II Impact workstation with an R1000 CPU. Table 1 summarizes the timing results.

While the UoS generation and rendering processes take the most time, optimal efficiency is not criti-

Table 1. Timing results for UoS interpolation algorithm

Process	Time (seconds)
Height field generation	7 (per image)
UoS generation	104 (per image)
Clustering	39 (per model)
Distance calculation	4
Matching and interpolation	20
Rendering from interpolated UoS's	724 (11 frames)

cal for these processes because they are only used once in the entire procedure. In addition, they do not require supervision while running. Therefore, not much effort has been put into optimizing them. In contrast, distance calculation, matching and interpolation are usually performed several times while the user determines the appropriate distance weights. For these processes, the user has to wait until completion, visualize the results, then make adjustments as necessary. Work is currently being done to reduce waiting time during these interactive sessions.

4 Summary and conclusions

We have presented an algorithm for image interpolation using UoSs. The algorithm is designed for good shape interpolation and increased stability with respect to changes in the input data. It also provides the user with a reasonable degree of control. We have shown that this method can interpolate between images with minimal preprocessing. For many applications, our algorithm requires no user specification of image features, but the user still has flexible control of the matching process. We have shown that the method easily facilitates manual feature specification where desired. Visualization of the UoSs allows the user to select suitable values for a number of parameters that control the simplification and matching of the models.

Three test cases, one of facial images and two of medical data, were used to demonstrate the capabilities of the method. The face data revealed some of the limitations inherent in automatic interpolation methods, and we showed how manual specification can be used to augment our technique to overcome these limitations. The CT and Visible Man leg images were used as examples of the types of data for which no manual feature specification is required for effective matching. The CT data was used in a volumetric reconstruction example to show the quality of the interpolation. The Visible Man legs were used in a registration example to demonstrate the feature matching capabilities of our algorithm.

Even though our algorithm has a number of advantages, it also has a number of weaknesses. Of primary concern is the fact that certain processes can be computationally intensive and quite time consuming, especially for larger images (i.e. $\geq 256\times 256$ pixels). Work is currently being done to improve the algorithm's efficiency.

5 Future work

There are many areas that we are currently exploring or intend to work on in the future. While we are experimenting to further test the capabilities of the algorithm with a minimal amount of preprocessing of the images, we are also investigating a number of preprocessing methods that could enhance the results of using our method for certain applications. For example, various nonlinear mappings from pixel intensities to height values are being tested.

We also plan to further extend the range of input data to test the robustness and stability of the representation. For example, in theory, our algorithm should be able to handle pairs of images that differ significantly in size or the amount of detail present. Also, at present, we have only used greyscale images; the extension to colour would be straightforward, but still very useful and interesting.

In addition, other applications of the use of the UoS representation of images will be explored. For example, because the method provides a way to measure distances between features, it can potentially be used for searching images in a database. Another potential application is image "warping" via manual manipulation of the UoS model.

We are also interested in extending the basic concept of this paper to higher dimensions. The next step would be to use four-dimensional models to represent three-dimensional shapes, and to apply simplification and matching techniques, analogous to the ones used here, to the new models.

Acknowledgements. We gratefully acknowledge the Natural Sciences and Engineering Research Council of Canada for providing support in the form of research grants and graduate scholarships. We thank Dr. B. Flak from the Vancouver Hospital and Health Sciences Centre for providing the CT data. The facial images were obtained from the Olivetti and Oracle Research Laboratory database of faces (http://www.cam-orl.co.uk/facesataglance.html). The Visible Man image was obtained from the U.S. National Library of Medicine Visible Human website (http://www.nlm.nih.gov/research/visible/visible_human.html).

References

Attali D, Montanvert A (1997) Computing and simplifying 2D and 3D continuous skeletons. Computer Vision and Image Understanding 67(3):261–273

Beier T, Neely S (1992) Feature-based image metamorphosis. ACM SIGGRAPH, Comput Graph 26:35–42

Brandt JW, Algazi VR (1992) Continuous skeleton computation by Voronoi diagram. CVGIP: Image Understanding 55:329–337

Brown LG (1992) A survey of image registration techniques. ACM Comput Surveys 24:325–376

Carmel E, Cohen-Or D (1997) Warp-guided object-space morphing. Visual Comput 13:465–478

Lee SY, Chwa KY, Shin SY, Wolberg G (1995) Image metamorphosis using snakes and free-form deformations. ACM SIGGRAPH. Comput Graph 29:439–448

Meyers D, Skinner S, Sloan K (1991) Surfaces from contours: the correspondence and branching problems. Proceedings of Graphics Interface ’91, Calgary, Canadian Information Processing Society, Toronto, pp 246–254

Ranjan V (1996) A union of spheres representation for 3D objects. PhD Thesis, Department of Computer Science, University of British Columbia, Vancouver, Canada

Ranjan V, Fournier A (1994) Volume models for volumetric data. IEEE Computer, Special Issue on Volume Visualization 27:28–36

Ranjan V, Fournier A (1996) Matching and interpolation of shapes using unions of circles. Proceedings of Eurographics ’96, Volume 15:35–42

ROGER TAM is a PhD student in the Department of Computer Science at the University of British Columbia (UBC), Vancouver, Canada. He is a member of the Imager Computer Graphics Laboratory at UBC. He received his MSc from the University of British Columbia in 1997. His research interests include scientific visualization, computer graphics and image processing.

ALAIN FOURNIER is a Professor in the Department of Computer Science at the University of British Columbia (UBC), Vancouver, Canada. He is Co-director of the Imager Computer Graphics Laboratory at UBC. He received his PhD from The University of Texas at Dallas in 1980. He has been a Professor of Computer Science since 1980, and has been at UBC since 1989. His research interests include computer graphics, object modelling, graphic algorithms, display systems architecture and computational geometry.