# Feature-Guided Shape-Based Image Interpolation

Tong-Yee Lee* and Chao-Hung Lin

***Abstract*—A feature-guided image interpolation scheme is presented. It is an effective and improved, shape-based interpolation method used for interpolating image slices in medical applications. The proposed method integrates feature line-segments to guide the shape-based method for better shape interpolation. An automatic method for finding these line segments is given. The proposed feature-guided shape-based method can manage translation, rotation and scaling situations when the slices have similar shapes. It can also interpolate intermediate shapes when the successive slices do not have similar shapes. This method is experimentally evaluated using artificial and real two-dimensional and three-dimensional data. The proposed method generated satisfactory interpolated results in these experiments. We demonstrate the practicality, effectiveness and reproducibility of the proposed method for interpolating medical images.**

***Index Terms*—Blending, distance map, interpolation, shape-based, warping.**

## I. Introduction

MEDICAL imaging devices produce medical data in the form of image slices. In such images, the distance between consecutive slices is larger than the distance between two neighboring pixels within a slice. This problem has an adverse effect on the subsequent visualization and analysis processes [1]–[4]. Many interpolation techniques have been proposed for processing such data. Classical image interpolation procedures fall into two main categories: gray level and shape-based. In the following, we will overview the work related to the proposed method.

The simplest method involves linearly interpolating the gray values in the slices to fill in the gray values in the missing slices [1]–[4]. However, artifacts are produced when the contour locations on two given slices shift considerably. To reduce these artifacts, Keys *et al.* [6] attempted to use higher order functions to perform interpolation. Later, a dynamically elastic surface interpolation scheme was proposed to manage the branching problem [7]. The key concept in this method involved determining the force field acting on the contour of one base image and to deform it gradually, making it closer to the contour on the other base image. Similarly, a hybrid approach combining elastic interpolation, the spline theory and the surface consistency theorem was proposed to produce further improvement [8].

Grevera and Udupa [5] and Herman *et al.* [10] proposed shape-based methods by encoding the segmented image with distance codes. This approach interpolates the distance instead of the gray values and, therefore, maintains better geometric changes. Because the shape-based method can be implemented efficiently and achieves reasonable interpolation results, it has become a widely used method. However, it cannot deal effectively with objects with holes, large offsets, or heavy invagination. To overcome these drawbacks, Guo *et al.* [11] developed a morphology-based interpolation method. This method first overlaps the two slices to obtain a morphologically difference image and then applies a sequence of dilation and erosion operations on nonoverlapping regions to achieve interpolation. This method successfully resolves the problems in objects with holes and large offsets. However, it still cannot handle objects with heavy invagination [12]. Recently, Lee *et al.* [13] proposed another morphology-based scheme. In contrast to [11], this approach is simpler in computational complexity and can handle more cases such as branching and invagination.

Several modified shape-based methods that utilize knowledge extracted from images have been developed. For example, Goshtasby *et al.* [14] selected feature points from successive frames to control the gray-level interpolation. Feature points are selected using the gradient value as a criterion. Feature points were used in [15], [16] defined using a fuzzy measure of boundaries and medial axis transforms. In [17] morphological skeleton interpolation was proposed based on object representation using mathematical morphological skeletonization.

In this paper, we propose a feature-guided shape-based image interpolation scheme. We borrowed the image-warping concept from T. Beier *et al.* [18] to compute interpolation with feature line-segment control. We extended this basic idea in two directions. First, the proposed scheme can automatically compute feature line-segments to control shape-based methods. Second, the original warping method in [18] is very computationally expensive. We propose methods to speed up the warping computation. In computer graphics, Cohen *et al.* [19] proposed three-dimensional (3-D) warping to interpolate 3-D volumetric objects. There are two main drawbacks that prevent this method from being directly used to interpolate medical images. First, this algorithm is very computationally expensive. Second, the success of this method is highly dependent on manual feature specification. It is impractical to accurately specify corresponding features in a real 3-D medical image that often contains hundreds of slices. This is very tedious and labor intensive work. In addition, questions about the reproducibility and accuracy of the method arise if different people specified the features.

Manuscript received March 25, 2002; revised July 23, 2002. This work was supported by National Science Council, Republic of China under Contract NSC-91-2213-E-006-078. The Associate Editor responsible for coordinating the review of this paper and recommending its publication was W. Higgins. *Asterisk indicates corresponding author.*

T.-Y. Lee is with the Computer Graphics Group/Visual System Laboratory, Department of Computer Science and Information Engineering, National Cheng Kung University, Tainan, Taiwan, R.O.C. (e-mail: tonylee@mail.ncku.edu.tw).

C.-H. Lin is with the Visual System Laboratory, Department of Computer Science and Information Engineering, National Cheng Kung University, Tainan, Taiwan, R.O.C.

Digital Object Identifier 10.1109/TMI.2002.806574

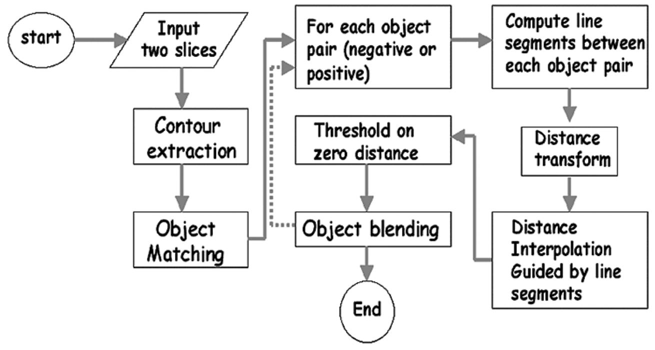
start
Input two slices
Contour extraction
Object Matching
For each object pair (negative or positive)
Threshold on zero distance
Object blending
End
Compute line segments between each object pair
Distance transform
Distance Interpolation Guided by line segments

Fig. 1. Flowchart of the feature-guided shape-based interpolation algorithm.

The paper is organized as follows. The feature-guided shape-based interpolation method is presented in Section II. In Section III, the method for automatically finding feature line-segments is presented. Experimental results when the proposed method is applied to artificial and real-world data are given in Section IV. Conclusions are drawn in Section V.

## II. Feature-Guided Shape-Based Image Interpolation

The proposed method is shown in Fig. 1. This method can be divided into the following steps. For any given two consecutive slices (binary or gray images), segment them (i.e., if gray images) and extract the contours for the objects of interest as in most of the shape-based interpolation techniques. Our previous techniques in [13] are exploited to compute the object matching and create positive and negative object pairs. These tasks will be briefly explained as we discuss branching and hole situations in Section IV. For each object pair (positive or negative), the standard shape-based method [9], [10] is used to generate the corresponding distance maps. The feature line-segments are found automatically and warping is used to interpolate the intermediate distance maps. The work presented in this section is the core of the proposed method. Next, the threshold is set to zero for values stored on the interpolated distance maps to obtain the interpolated contours. In case we need to create multiple contours, such as branching or holes, some of the above procedures must be processed several times (see the dashed line in Fig. 1). Finally a contour-blending task is required to combine all interpolated contours together to obtain the correct results. In these above procedures, some well-known shape-based methods [9], [10] and techniques were used from our previous work [13] will not be explained here in detail. We will concentrate on the fourth part of the above procedures. In the following subsections, we will present the details on 1) how warping [15] using corresponding line segments to control the interpolation and 2) speeding up warping computation in the proposed method and 3) how to automatically compute the control line segments.

### A. Shape-Based Interpolation Using Warping

In Fig. 1, for given an object contour pair $(C_s, C_{s+1})$ (i.e., after contour extraction) two distance maps $(D_s, D_{s+1})$ are computed using the standard shape-based method. After the corresponding feature line-segments are computed (will be described later) for $(C_s, C_{s+1})$, a warping technique [18] is used to fill in the distance information for the intermediate distance map $D_t$. To create $D_t$, the warping is divided into two main steps. First, we compute two deformed distance maps $(D'_s, D'_{s+1})$ from $(D_s, D_{s+1})$ according to the control line-segments. The two deformed $(D'_s, D'_{s+1})$ maps are then linearly blended to generate $D_t$. Each control line segment is directed. For each corresponding pair of line segments $\overline{P_sQ_s}$ and $\overline{P_{s+1}Q_{s+1}}$ on $(D_s, D_{s+1})$, the warping computes an intermediate line segment $\overline{P_tQ_t}$, where $0 \leq t \leq 1$, $P_t = P_s*(1-t)+t*P_{s+1}$ and $Q_t = Q_s*(1-t)+t*Q_{s+1}$. The deformed $D'_s$ can be then computed as follows [18]. For each pixel coordinate $X$ of $D'_s$ and its corresponding image pixel $X'$ on $D_s$, the warping computes $X'$ using

$$u = \frac{(X-P_t)\bullet(Q_t-P_t)}{||Q_t-P_t||^2} \tag{1}$$

$$v = \frac{(X-P_t)\bullet \mathrm{Perpendicular}(Q_t-P_t)}{||Q_t-P_t||} \tag{2}$$

$$X' = P_s + u\bullet(Q_s-P_s) + \frac{v\bullet \mathrm{Perpendicular}(Q_s-P_s)}{||Q_s-P_s||} \tag{3}$$

where the value $u$ is the position along the line normalized by the distance $\overline{P_tQ_t}$ and $v$ is the distance from the line and procedure $\mathrm{Perpendicular}(\ )$ returning a vector that is perpendicular to the input vector. The idea is very simple in the above equations. Both directed lines $\overline{P_tQ_t}$ and $\overline{P_sQ_s}$ define their local coordinate systems. For the corresponding $X$ and $X'$, their local coordinates are $(u, v)$ in these two local systems defined by $\overline{P_tQ_t}$ and $\overline{P_sQ_s}$. In [18], the warping method uses multiple line segments. Assume that we have computed $X'_1, X'_2 \cdots X'_n$ due to $n$ corresponding line segments. We calculate the combined point $X'$ for $X$ as follows:

$$X' = \frac{\sum_{i=1}^{n} w_i * X'_i}{\sum_{i=1}^{n} w_i} \quad \text{and} \quad w_i = (a + \mathrm{distance})^{-b} \tag{4}$$

where the distance is the distance from point $X$ to each line segment on $D'_s$. The parameter $a$ ($a = 0.01$ in all experiments) is a small number to avoid $w_i$ being a zero value and the parameter $b$ (i.e., $b = 2$ in all experiments) is used to control the rate of degradation influence per each line segment. In this manner, the warping function calculates $X'$ for $X$. We then let the distance value for $X$ on $D'_s$ be the distance for $X'$ on $D_s$. Using procedures similar to those above we can also compute the deformed $D'_{s+1}$. Now, we have computed two deformed $(D'_s, D'_{s+1})$ maps for the intermediate $D_t$. Next, both $D'_s$ and $D'_{s+1}$ are blended in a linear manner to calculate $D_t$ using

$$D_t(X) = D'_s(X)*(1-t) + D'_{s+1}*t \tag{5}$$

where $0 \leq t \leq 1$ and $X$ is a pixel coordinate. We call the above procedures a feature-guided shape-based interpolation method in this paper.

### B. Optimization

The warping technique [18] is a brute-force approach. It computes every pixel to a new location according to all feature line pairs. In this section, we will propose methods to speed up the warping computation. We suggest warping computation speed improvements using the linearity property of each warping scan-line and the bounding boxes.

*1) Scan-Line Linearity of Warping Function:* Using (1)–(3) to transform a scan-line, we can obtain a transformed line that is rotated, scaled or translated. In the following, we will show that this transformation is a linear function. Assume that pixels $A$ and $B$ are two ending points of a scan-line and $C$ is a point on $\overline{AB}$. For a feature line pair, the locations of $A$, $B$ and $C$ will be mapped onto new coordinates $A'$, $B'$ and $C'$. Since $A$, $B$ and $C$ are located on the same scan-line, we have

$$C = (1-s) * A + s * B \quad \text{and} \quad s \in (1, 0). \tag{6}$$

For the pixel $C$ on either $D'_s$ or $D'_{s+1}$, we use (1)–(3) to find $C'$'s corresponding location $C'$ on either $D_s$ or $D_{s+1}$ by (7), as shown at the bottom of the page, where $P'$ can be either $P_s$ or $P_{s+1}$ and $Q'$ can be either $Q_s$ or $Q_{s+1}$ and $P$ and $Q$ are $P_t$ and $Q_t$ in (1). Therefore, (7) indicates the transformation of a given scan-line is a linear function. On this basis, we morph two ending points for each scan-line using only (1)–(3) and linearly interpolate the remaining points rather than compute the remaining points using (1)–(3). The warping computation can, thus, be reduced. With this optimization, we approximate the influence of a line pair, say $F_i$ on a given scan-line, and term each transformed line $T_i$. We then use (4) to combine all $T_i$s and accomplish warping this scan-line. This combination using (4) is not a linear function and, thus, a straight line could be distorted into a curve.

*2) Bounding Box:* The computation cost for the original warping technique is in proportion to the number of feature lines and image resolution. We can reduce the computation cost by skipping some empty pixels in our application. For each contour pair ($C_s$, $C_{s+1}$), we find the union of their bounding boxes and need only to perform the same interpolation procedures described in Section II-A to this union area, as shown in Fig. 2. Only a distance transform is performed on this area. Later in Section IV, we will show how this bounding box technique combined with scan-line optimization (in Section II-B1) will significantly improve the warping computation.

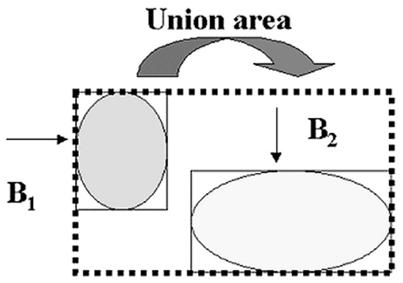

Fig. 2. B1 and B2 are bounding boxes for a given contour pair ($C_s$, $C_{s+1}$). Both the distance transform and interpolation are performed only on the union area instead of the whole image to save computation cost.

## III. AUTOMATICALLY COMPUTING CONTROL LINE SEGMENTS

In the following, we will propose an approach to compute corresponding line segments for a given contour pair ($C_s$, $C_{s+1}$). This approach consists of three main tasks: 1) finding the principle axis of each contour, 2) simplifying the input contours and 3) contour matching. These three tasks will be presented in the following sections.

### *A. Principle Axis Alignment*

Given two input contours, we need to align their principle axes before we find their corresponding line segments. A two-dimensional (2-D) contour $C$ consists of $n$ points and any two consecutive points $p_i$ and $p_{i+1}$ of these $n$ points can form a line segment. The coordinate of $p_i$ is denoted as $(p_i.x, p_i.y)$. The principle axis of a contour $C$ can be computed from its $\Sigma$, covariance matrix [20], where $\Sigma$ is defined by (8), as shown at the bottom of the page. In (8), $(m.x, m.y)$ represents the average of $n$ points. We then compute the maximum eigenvalue $\lambda$ of $\Sigma$ [20]. The corresponding eigenvector of $\lambda$ defines the principle axis of the contour $C$. We compute the principle axes for the two

$$\begin{aligned} C' &= P' + u \bullet (Q' - P') + \frac{v \bullet \text{Perpendicular}(Q' - P')}{\|Q' - P'\|} \\ &= P' + P' - P' + \frac{(A - s*A + s*B - P + P - P)\text{Perpendicular}(Q-P)}{\|Q-P\|^2} \bullet (Q' - P') \\ &\quad + \frac{\frac{(A - s*A + s*B + P - P + P) \bullet Perpendicular(Q-P)}{\|Q-P\|} \bullet \text{Perpendicular}(Q' - P')}{\|Q' - P'\|} \\ &= A' - s*A' + s*B' = (1-s)*A' + B' \end{aligned} \tag{7}$$

$$\Sigma = \begin{bmatrix} \frac{1}{n}\sum_{i=1}^{n}\{(p_i.x - m.x)(p_i.x - m.x)\} & \frac{1}{n}\sum_{i=1}^{n}\{(p_i.x - m.x)(p_i.y - m.y)\} \\ \frac{1}{n}\sum_{i=1}^{n}\{(p_i.y - m.y)(p_i.x - m.x)\} & \frac{1}{n}\sum_{i=1}^{n}\{(p_i.y - m.y)(p_i.y - m.y)\} \end{bmatrix} \tag{8}$$

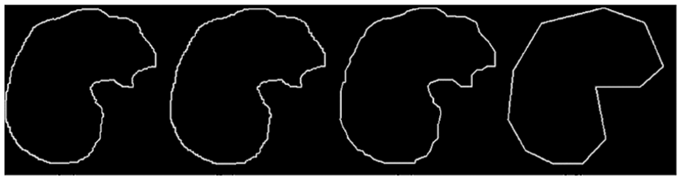

Fig. 3. Contour simplification. (a) Input contour consists of 999 points. (b)–(d) Simplified contours of (a) consisting of 250, 60, and 12 points, respectively.

input contours ($C_s$ and $C_{s+1}$) and their axes are denoted $V_s$ and $V_{s+1}$, respectively. The included angle $\theta$ between $V_s$ and $V_{s+1}$ is then determined. Using this included angle $\theta$, we then rotate each $p_i$ of $C_s$ to achieve alignment using

$$p'_i = R(\theta) * p_i \tag{9}$$

where $R(\theta)$ is the rotation matrix.

## B. Contour Simplification

In our approach, line segments are used to control the shape-based interpolation. Ideally, we will not use too many line segments and these line segments should be feature line segments of the given contours. For this purpose, we implemented a method for the optimal polygonal approximation of digitized curves [21]. This method uses A* heuristic search algorithm to speed up finding the optimal solution. For a given $n$-point contour, its complexity is close to $O(n^2)$. In [21], the error function *error* $(i\,j)$ is used to estimate the local approximation and is equal to the square Euclidean distance from contour each point from $(x_i, y_i)$ to $(x_j, y_j)$ to its orthogonal projection onto the line $y = ax + b$ defined by $(x_i, y_i)$ and $(x_j, y_j)$ using

$$error(i, j) = \sum_{t=i}^{t=j} \frac{(y_{P_t} - ax_{P_t} - b)^2}{a^2+1} \tag{10}$$

where $P_t$ is a point from $(x_i, y_i)$ to $(x_j, y_j)$. Fig. 3 shows an example using this method (for more details, see [21]. After simplification, only the feature points of a given contour are left. In the next subsection, we will describe a matching method to determine the correspondence among these feature points.

## C. Contour Matching

Let $C_s(u)$, $u \in [0, 1]$ and $C_{s+1}(v)$, and $v \in [0, 1]$ be two parametric curves for the source and input contours. In this section, we need to establish the correspondence between the two curves. To establish correspondence, two matching criteria are considered: intensity and geometry similarities. We use the image correlation to evaluate the intensity similarity. Assume that two contours are originally extracted from two given gray-level (or binary level for artificial data) image slices $I_s$ and $I_{s+1}$. For two contour points $p_s^i$ and $p_{s+1}^j$ on $I_s$ and $I_{s+1}$, their image correlation can be computed using

$$C(p_s^i, p_{s+1}^j) = \frac{\sigma_{s,\,s+1}^2}{(\sigma_s^2 \sigma_{s+1}^2)^{1/2}} \tag{11}$$

where $\sigma_s^2$ and $\sigma_{s+1}^2$ are the variances in intensity value for two $m*n$ blocks centered on $p_s^i$ and $p_{s+1}^j$, respectively. $\sigma_s^2$ and $\sigma_{s+1}^2$ are computed using

$$\sigma_k^2 = \sum_{j=1}^{n} \sum_{i=1}^{m} \{I_k(i, j) - \mu_k\}^2 \Big/ (mn) \quad \text{for } k = s \text{ or } s+1. \tag{12}$$

And $\sigma_{s,\,s+1}^2$ is covariance of $I_s$ and $I_{s+1}$, and can be computed using

$$\sigma_{s,\,s+1}^2 = \sum_{j=1}^{n} \sum_{i=1}^{m} [\{I_s(i, j) - \mu_s\} \cdot \{I_{s+1}(i, j) - \mu_{s+1}\}]/(mn). \tag{13}$$

For a continuous parametric curve $C(x)$, we can compute its unit tangent vector $T(x)$ using the following:

$$T(x) = \frac{\frac{dC(x)}{dx}}{\left\| \frac{dC(x)}{dx} \right\|} = \frac{C'(x)}{\|C'(x)\|}. \tag{14}$$

In this paper, the geometric similarity of $C_s(p_s^i)$ and $C_{s+1}(p_{s+1}^j)$ is evaluated using the inner product of $T_s(p_s^i)$ and $T_{s+1}(p_{s+1}^j)$. This inner product is denoted using

$$\langle T_s(p_s^i),\, T_{s+1}(p_{s+1}^j) \rangle. \tag{15}$$

The basic idea is that when two curves are matched, each correspondence pair $(p_s^i, p_{s+1}^j)$ has an equal tangent vector (i.e., inner product is equal to 1). Therefore, when we find all of the equal tangent vectors between two curves, the best match occurs when the sum of (15) for all correspondences is the maximum. After the contour simplification discussed in Section III-B, let us assume that both $C_s(u)$ and $C_{s+1}(v)$ consists of $m$ points and each point is denoted as $C_s(p_s^i)$ and $C_{s+1}(p_{s+1}^j)$, where $1 \le i, j \le m$. To find the best matched points between $C_s(u)$ and $C_{s+1}(v)$, we optimally compute the solution using the following:

$$\max_{j(i)} \sum_{i=1}^{m} \left\{ w_1 * \left\langle T_s(p_s^i),\, T_{s+1}\left(p_{s+1}^{j(i)}\right) \right\rangle + w_2 * C\left(p_s^i,\, p_{s+1}^{j(i)}\right) \right\}. \tag{16}$$

To evaluate (16), we need to reparameterize [22] $C_{s+1}(p_{s+1}^j)$ using $C_{s+1}(p_{s+1}^{j(i)})$ to find solutions. $w_1$ and $w_2$ are the weights for the intensity and geometry similarities. In addition, the reparameterization must be subject to $j(1) = 1$, $j(m) = m$ and $j(i) \le j(i+1)$. Directly solving (16) is a hard problem. We adopted Cohen *et al.* [23] approach by exploiting dynamic programming over the $m$ points of $C_s(u)$ and $C_{s+1}(v)$. This dynamic programming approach recursively defines a cost function $Cost(i, j)$ using

$$\begin{aligned} Cost(i, j) = \min(&Cost(i-1, j-1), \\ &Cost(i-1, j), Cost(i, j-1)) \\ &+ w_1 * \langle T_s(p_s^i),\, T_{s+1}(p_{s+1}^j) \rangle \\ &+ w_2 * C(p_s^i,\, p_{s+1}^j). \end{aligned} \tag{17}$$

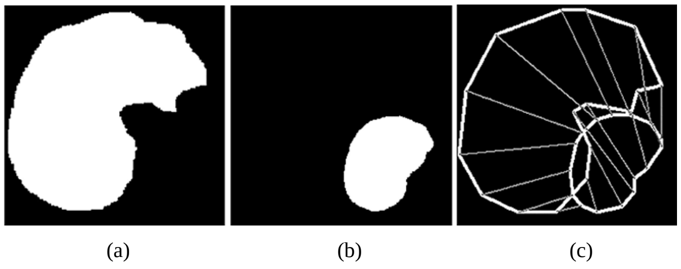

Fig. 4. Contour matching. (a) and (b) Input contours on $I_s$ and $I_{s+1}$, respectively. Each corresponding point pair is shown using a line connecting two corresponding points on two contours in (c).

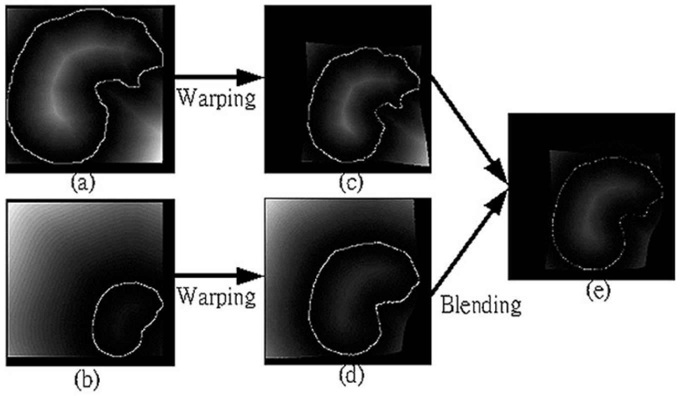

Fig. 5. Warping procedures to interpolate contours.

The meaning of $Cost(i, j)$ is interpreted as the optimal cost for matching the first $i$ samples of $C_s(u)$ with the first $j$ samples of $C_{s+1}(v)$. The time complexity of $Cost(i, j)$ is $O(ij)$ using the dynamic programming approach [23]. Fig. 4 shows an example of matching two curves using the proposed method. In this example, each corresponding point pair is shown using a line connecting two corresponding points. After the matching task, these matched points will define corresponding points between the two contours and two consecutive feature-points from a feature line-segment on each contour. In all examples in this paper, when computing (17), we let $w_1 = 0.7$ and $w_2 = 0.3$ for the real data and $w_1 = 0.9$ and $w_2 = 0.1$ for the artificial data.

## D. Solving Interpolation Problems

We show an example in Fig. 5 using the proposed algorithm to control shape-based interpolation. In this example, we need to create distance maps $D_s$ [Fig. 5(a)] and $D_{s+1}$ [Fig. 5(b)] for two input contours $C_s$ and $C_{s+1}$ on image slices $I_s$ and $I_{s+1}$. We used feature line-segments computed from the algorithms presented in Section III-C to generate two warping distance maps $D'_s$ [Fig. 5(c)] and $D'_{s+1}$ [Fig. 5(d)]. Next, we used (5) to linearly interpolate any number of intermediate distance maps. This simple approach performs well as the orientation for the principle axes of $C_s$ and $C_{s+1}$ are not too different. This approach would not otherwise be able to generate a smooth interpolation (see an example in Fig. 6). To avoid this situation, we replaced (5) with

$$D_t(X) = [D'_s(X) * R(\theta) * (1-t) + D'_{s+1}(X) * t] * R(-\theta(1-t)) \quad (18)$$

where $0 \le t \le 1$, $D'_s(X)$, $D'_{s+1}(X)$ are defined in (5), $\theta$ and $R(\,)$ are defined in (9). Fig. 5(e) shows an interpolated map using (18). We used an example in Fig. 6 to demonstrate the difference using (5) and (18). In (5), $C_s$ and $C_{s+1}$ are shown in (a) and (b). The matched points between the two contours are illustrated in Fig. 5(c). Fig. 5(d) and (e) represents the rendered results of the interpolated volume using (5) and (18). From this example, we can observe that Fig. 5(e) has a smoother interpolated volume than Fig. 5(d).

## IV. Experimental Results and Discussions

Experiments were performed using artificial and real-world data. The artificially generated data were selected from previous work or selected to present the features of the proposed algorithm.[1] In the following examples, all corresponding point pairs are shown using lines connecting the corresponding points among the contours. The first example was presented in Section III-D as shown in Fig. 6. This example demonstrates the capability of the proposed method to interpolate slices of a rotating stick-like object. Fig. 7 shows that the shape of the stick-like object is preserved and the object in the first original frame is rotated to the orientation of the second original frame. The algorithm was next applied on slices that contain two circles [11], [12], [17] in Fig. 8. In this example, the shape of the large circle on the first original frame is preserved and the circle is translated and shrunk in size toward the small circle in the second original frame. This experiment shows the ability of this algorithm to interpolate slices contained in objects with large offsets and that differ in size. From these two examples, the proposed feature-guided shape-based method can manage translation, rotation and scaling situations, when the slices have similar shapes.

The next two examples were hole and branching problems shown in Figs. 9 and 10. In Fig. 9, the original frames are (a) and (b). Both (a) and (b) have a hole. This example demonstrates the capability of the proposed method to interpolate object slices with holes. In this example two contour pairs are created: a positive pair for the outer contours and a negative pair for the inner hole-contours. We then used the proposed method to generate two interpolated contours: one for a positive pair and one for a negative pair. The two contours were then blended to generate a desired contour. To deal with a hole, the blending is simply a set subtraction operation [13]. For multiple negative and positive pairs, we separately interpolate the positive and negative object pairs. Afterwards, we blend all of the same type interpolated contours (positive or negative). In this case, the blending is simply a set union operation. We then subtract the union of negative contours from the union of positive contours to produce the final interpolated contour. More details about this blending can be found in our previous work [13]. Fig. 10 demonstrates the capability of the proposed method to interpolate object slices with branches [7], [8], [11]–[13]. In [13], we scored each potential matched object pair based on the overlapping information and distance between them. If the score was over a selected threshold, the potential pair was considered a matched pair. We generated interpolated contours for all matched pairs.

[1]More experiments can be found at our research web site: http://couger.csie.ncku.edu.tw/~vr/TMI_interpolation.html.

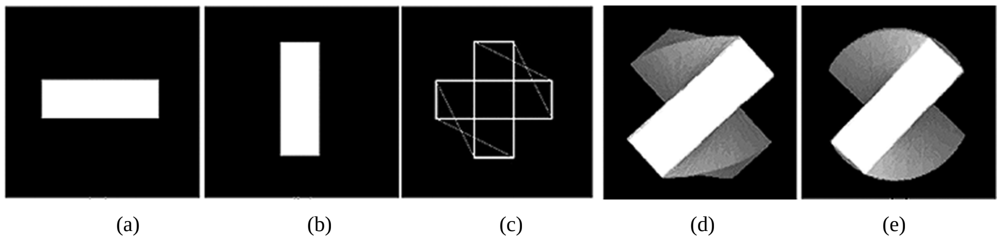

Fig. 6. Difference using (5) and (18) is shown. (a) and (b) Original contours $C_s$ and $C_{s+1}$, respectively. (c) Matched points between the two contours. (d) and (e) Interpolated volume rendering results using (5) and (18), respectively.

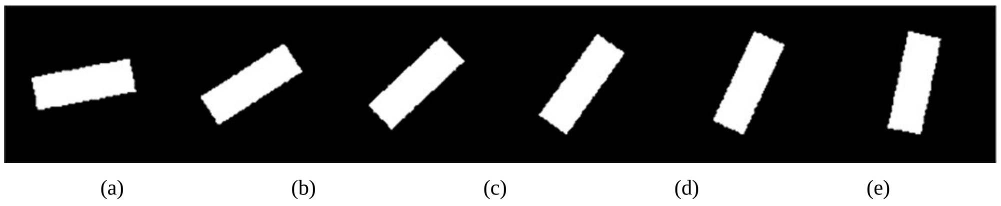

Fig. 7. Interpolated slices for the example in Fig. 6.

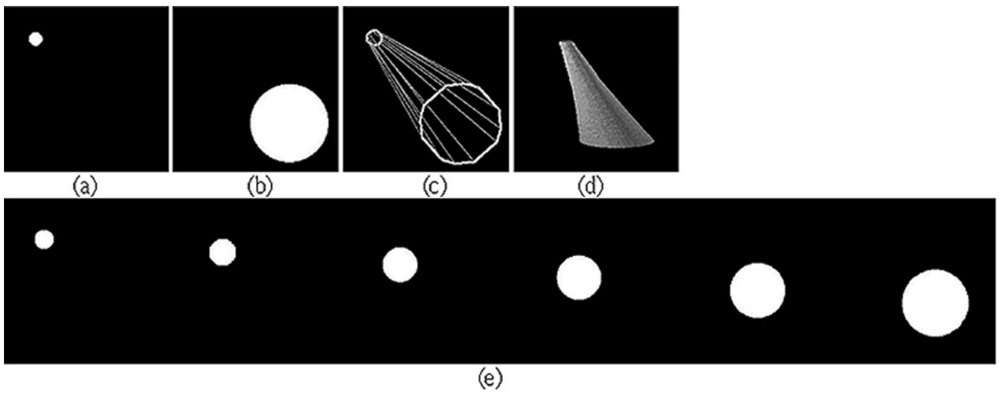

Fig. 8. Objects with large offsets [11], [12], [17]. (a) and (b) Two input contours. (c) Matched points. (d) Interpolated volume rendering results. (e) Sequence of interpolated slices.

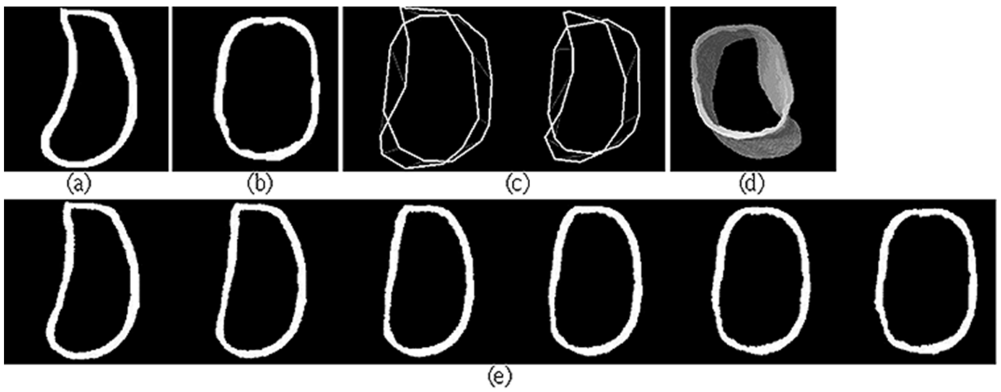

Fig. 9. Objects with a hole [11]. (a) and (b) Original frames. (c).(left) positive pair and (right) negative pair. (d) Rendered result of interpolated volume and (e) shows a sequence of interpolated object.

Fig. 10 shows the independent interpolation of three positive object pairs. The final contour can then be reconstructed from the union of these three interpolated contours. This example was widely tested in [7], [8], [11]–[13]. The proposed scheme yields very satisfactory results.

Figs. 11 and 12 are termed "moderate" and "extreme" concave cases in [7]. The proposed method has the capability to deal well with these two cases. Both examples were also tested in [7]. To deal with these examples, this approach [7] employs a very computationally intensive method to distort one contour to be like another one. Using the simpler proposed scheme, we see the shapes of the intermediate contour change smoothly regardless of being a "moderate" or "extreme" concave case. Our results are similar to those presented in [7].

The last two artificial examples were evaluated in [12] as shown in Figs. 13 and 14. Sun *et al.* [12] mentioned that both cases were difficult and were not solved well using the morphological-based interpolation method [11]. Many previous studies

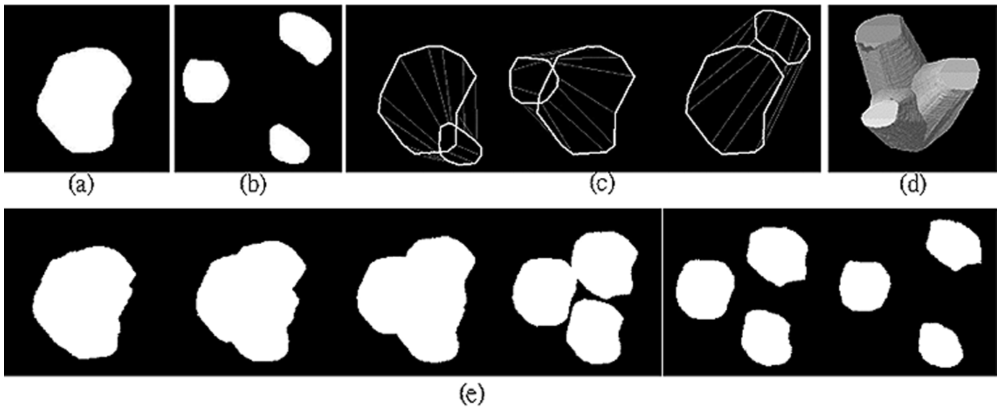

Fig. 10. Branching case [7], [8], [11]–[13]. (a) and (b) Original slices. (c) Matched object pairs. (d) Rendered result of interpolated volume. (e) Sequence of interpolation.

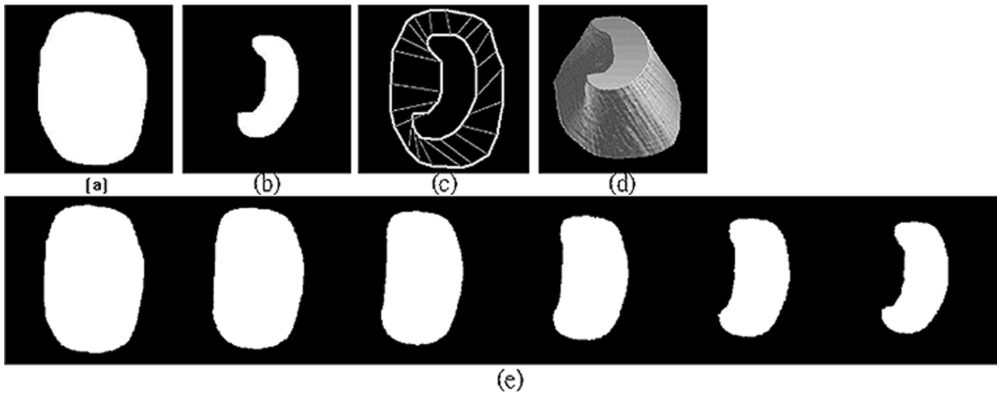

Fig. 11. A "moderate" concave case [7]. (a) and (b) Original slices. (c) Correspondence. (d) Rendered result of interpolated volume. (e) Sequence of interpolation.

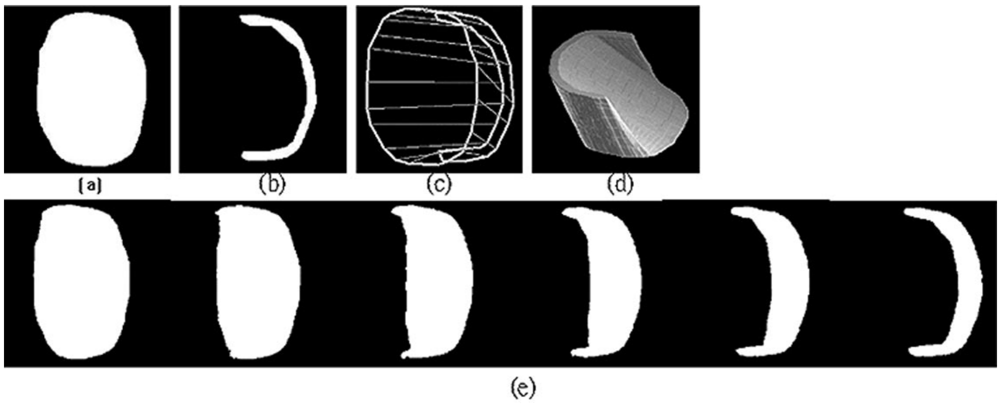

Fig. 12. An "extreme" concave case [7]. (a) and (b) Original slices. (c) Correspondence. (d) Interpolated volume rendering result. (e) Interpolation sequence.

suggested applying object centralization to have one object enclosed by another before interpolation [17]–[19]. Sun *et al.* pointed out that this conventional centralization (i.e., aligning the centroids of the two objects) sometimes failed when the objects were concave. To solve this issue, Sun *et al.* [12] iteratively employed object centralization and object enlargement to ensure that object enclosure could occur. After interpolation, this approach requires contour shrinking using erosion to compensate for the object enlargement effect. Furthermore, this process cannot always guarantee object enclosure even when the enlarging factor becomes extremely large [12]. This entire process is not very efficient with respect to the computational complexity. Our proposed scheme is more practical than this approach. Four examples using the proposed method applied to real-world medical data follows in Figs. 15–18.

All of the above experiments were run on the 1-GHz Intel Pentium III with 128-MB memory. All original slices were $256 * 256$ images. We interpolated 100 slices for Figs. 7–14. For Figs. 15–17, we interpolated eight slices. Table I shows the timing information used in the proposed method and the number of feature line segments for interpolating eight slices in Figs. 15–17. The information includes the time used in image preprocessing (i.e., contour extraction, object matching, bounding boxes, computing the principle axes and distance

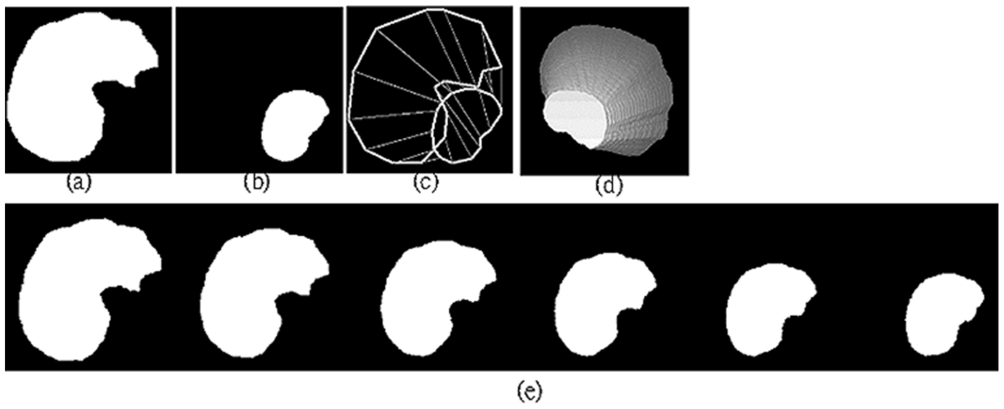

Fig. 13. A difficult case tested in [12]. (a) and (b) Original slices. (c) Correspondence. (d) Interpolated volume rendering results. (e) Sequence of interpolation.

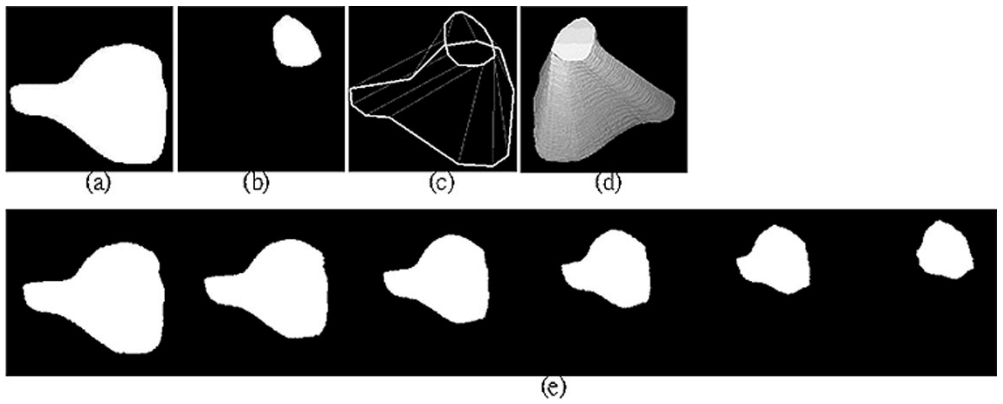

Fig. 14. The invagination case (abrupt change in shape) [12]. (a) and (b) Original slices. (c) Correspondence. (d) Interpolated volume rendering results. (e) Sequence of interpolation.

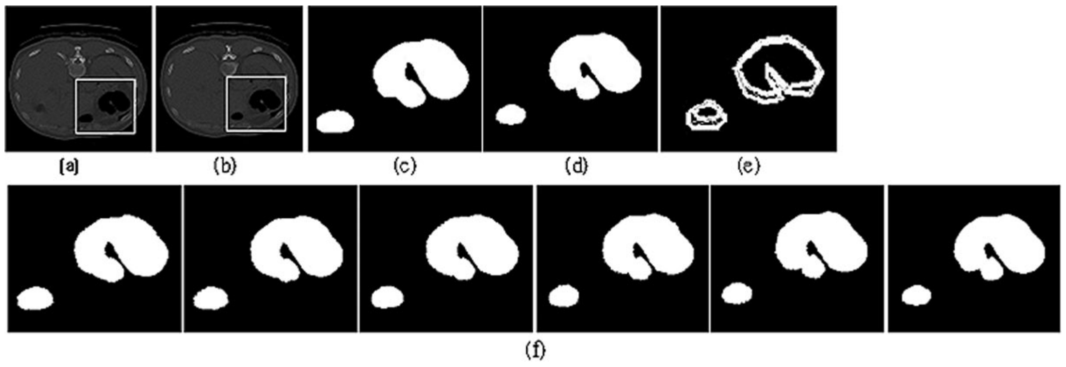

Fig. 15. Colon CT image slices. (a) and (b) Original slices and colon contours are extracted in (c) and (d). (e) Correspondence. (f) Sequence of interpolation.

transform etc.), contour simplification, contour matching and interpolation with warping. From this table, we can see that most of the components of the proposed algorithm compute very fast. Only the interpolation component is not fast. The average time per interpolated slice was 4.16 s, achieved using the nonoptimized proposed method for all experiments in this paper. Tables II and III show the improvement in warping computation using the scan-line linearity and bounding boxes. From these two tables, we clearly see the improvement using the proposed optimized schemes. The average time per interpolated slice is reduced from about 4.16 s to 2.99 (using scan-line linearity) and 1.097 s (using scan-line linearity with bounding boxes). From these tables, we see that the interpolation cost is in proportion to the number of line segments. The empty area is also an important factor. As we discussed in Section II-B2, we can save computation time, if we skip pixels outside the union of the bounding boxes. The more pixels we skip, the greater the potential improvement achieved.

Finally, we applied the proposed method to real 3-D molar data. The original slices of the molar volume were scanned from [17]. Fig. 18(a) shows 16 slices from the original 30 slices. Fig. 18(b) shows two views of the reconstructed results.

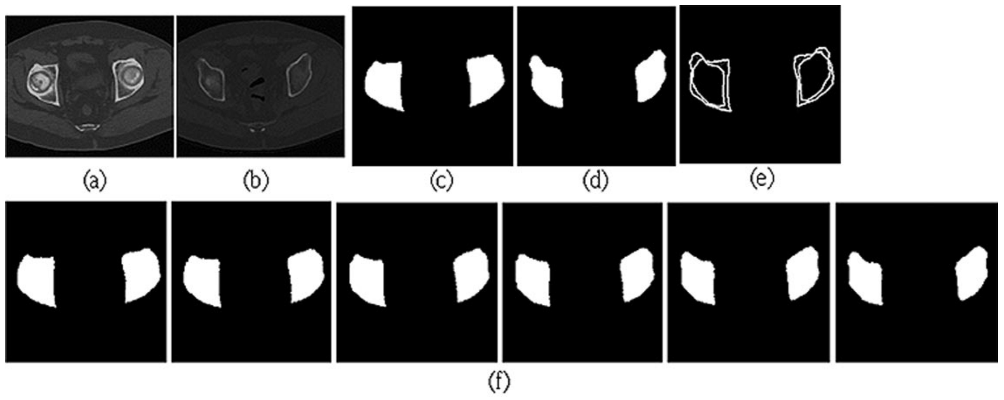

Fig. 16. CT pelvis image slices: (a) and (b) are original slices; (c) and (d) are extracted contours with interest; (e) shows correspondence between (c) and (d). (f) Interpolation sequence.

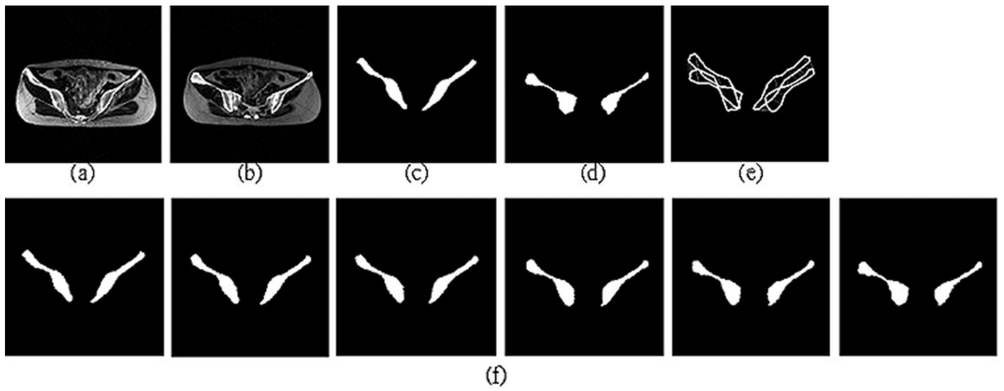

Fig. 17. MRI image slices. (a) and (b) Original slices. (c) and (d) Extracted contours with interest. (e) Correspondence between (c) and (d). (f) Interpolation sequence.

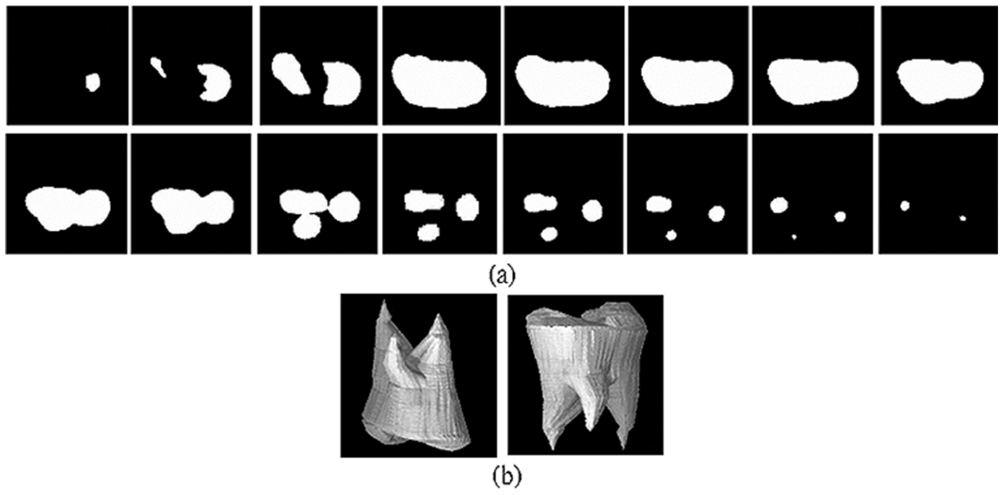

Fig. 18. Molar volume reconstruction ($256 * 256 * 240$). Original input images were scanned from [17].

Table IV shows the experimental timing information for reconstructing Fig. 18. Since we have 30 original slices, we need to perform image preprocessing, contour simplification and matching 30 times. A total of 240 slices were interpolated. We also reconstructed other 3-D data.[2]

[2]Available: http://couger.csie.ncku.edu.tw/~vr/TMI_interpolation.html

## V. Conclusion

We proposed using features to control shape-based interpolation. The proposed scheme was experimentally shown to successfully resolve complex interpolation problems that could not be managed using the original shape-based method or other previous approaches. We applied the proposed scheme to real-world data. All of the experimental results showed that the proposed method could generate satisfactory interpolation. If feature-lines are specified manually, there is no guarantee of

TABLE IV
EXPERIMENTAL TIMINGS (UNIT: SECONDS) FOR FIG. 18

| Molar 256*256*240 | Image Preprocessing (30 times) | Contour Simplification (30 times) | Contour Matching (30 times) | Interpolation (240 slices) | | |
|---|---|---|---|---|---|---|
| | | | | (1) Non-optimized | (2) Scan-line linearity | (1) & (2) |
| Fig. 18 | 4.451 s | 3.622 s | 4.413 s | 788.2 s | 568.9 s | 387.7 s |

TABLE I
TIME (UNIT: SECONDS) ANALYSIS FOR EXPERIMENTS IN FIGS. 15–17. FOR FIG. 17, 12–16 MEANS THAT THERE ARE TWO CONTOUR PAIRS AND ONE PAIR USED 12 LINE SEGMENTS AND THE OTHER USED 16 LINE SEGMENTS TO CONTROL INTERPOLATION

| | Number of line segments | Image Preprocessing | Contour Simplification | Contour Matching | Interpolation with warping |
|---|---|---|---|---|---|
| Fig. 15 | 12-12 | 0.163 | 0.070 | 0.045 | 34.259 |
| Fig. 16 | 18-18 | 0.134 | 0.170 | 0.102 | 50.433 |
| Fig. 17 | 12-16 | 0.144 | 0.221 | 0.072 | 38.896 |

TABLE II
PERFORMANCE COMPARISON (UNIT: SECONDS) AMONG THE NON-OPTIMIZED, OPTIMIZED BY SCAN-LINE LINEARITY AND OPTIMIZED USING A COMBINATION OF SCAN-LINE LINEARITY AND BOUNDING BOXES. IN THESE EXAMPLES: FIGS. 15–18, WE INTERPOLATED EIGHT SLICES

| | Non-optimized | Optimized | Optimized & Bounding Box | No. of features |
|---|---|---|---|---|
| Fig. 15 | 34.537 | 24.321 | 1.502 | 12-12 |
| Fig. 16 | 50.839 | 34.471 | 3.391 | 18-18 |
| Fig. 17 | 39.333 | 28.989 | 3.512 | 12-16 |

TABLE III
PERFORMANCE COMPARISON (UNIT: SECONDS) AMONG THE NON-OPTIMIZED, OPTIMIZED USING SCAN-LINE LINEARITY AND OPTIMIZED USING A COMBINATION OF SCAN-LINE LINEARITY AND BOUNDING BOXES. IN THESE EXAMPLES (FIGS. 7–14), WE INTERPOLATED 100 SLICES

| | Non-optimized | Scan-line linearity | Scan-line linearity & bounding box | Number of feature lines |
|---|---|---|---|---|
| Fig. 7 | 74.612 | 54.422 | 22.046 | 4 |
| Fig. 8 | 208.411 | 150.420 | 95.730 | 12 |
| Fig. 9 | 416.023 | 299.324 | 192.743 | 12-12 |
| Fig. 10 | 625.442 | 473.916 | 207.528 | 12-12-12 |
| Fig. 11 | 378.198 | 279.736 | 165.217 | 22 |
| Fig. 12 | 378.641 | 271.183 | 164.548 | 22 |
| Fig. 13 | 276.498 | 202.954 | 158.561 | 16 |
| Fig. 14 | 208.179 | 156.121 | 124.975 | 12 |

reproducibility in daily practice because different persons can specify different features. The proposed method automatically finds the feature line-segments. This method can be used easily in practice experiments and there is no reproducibility problem. We also proposed techniques to optimize the warping speed. On average, the achieved speedup varies from 1.39 to 3.79 times. The proposed method has demonstrated practicality, effectiveness, reproducibility and accuracy in medical image interpolation. Finally, we should also mention that the prior techniques for shape-based interpolation are effective in the large majority of circumstances in our experience. Furthermore, it is very difficult to state rigorously that our method produces theoretically correct interpolated shape information, since the input data are subsampled. However, in contrast to other work, the proposed method generally and successfully interpolates data that were reported in other previous work. On the other hand, with improving imaging scanners, the interpolation problem is becoming less necessary for 2-D/3-D data. However, it will be another research direction to apply the proposed method to four-dimensional (4-D) (plus time) data. It is not common to have a very high resolution of 4-D data at most hospitals.

## ACKNOWLEDGMENT

The authors would like to give their special thanks to the authors of [17], since they scanned their test input images to conduct experiments in Fig. 18. Also, their sincere thanks go to anonymous readers to give them many helpful comments in their earlier drafts.

## REFERENCES

[1] H. Fuchs, Z. M. Kedem, and S. P. Uselton, “Optimal surface reconstruction from planar contours,” *Commun. ACM*, vol. 20, pp. 693–702, 1977.

[2] G. Herman and C. Coin, “The use of 3D computer display in the study of disk disease,” *J. Comput. Assist. Tomogr.*, vol. 4, no. 4, pp. 564–567, Aug. 1980.

[3] C. C. Liang, C. T. Chen, and W. C. Lin, “Intensity interpolation for reconstructing 3-D medical images from serial cross-sections,” in *Proc. IEEE Eng. Med. Bio. Soc. 10th Int. Conf.*, New Orleans, LA, Nov. 1988, Paper CH2566-8, pp. 1389–1390.

[4] ——, “Intensity interpolation for branching in reconstructing three-dimensional objects from serial cross-sections,” in *Proc. SPIE Conf. Medical Imaging V: Image Processing*, vol. 1445, San Jose, CA, Feb.–Mar. 1991.

[5] G. J. Grevera and J. K. Udupa, “Shape-based interpolation of multidimensional grey-level images,” *IEEE Trans. Med. Imag.*, vol. 15, pp. 881–892, Dec. 1996.

[6] R. G. Keys, “Cubic convolution interpolation for digital image processing,” *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-29, pp. 1153–1160, Dec. 1981.

[7] W.-C. Lin, C.-C. Liang, and C.-T. Chen, “Dynamic elastic interpolation for 3-D medical image reconstruction from cross sections,” *IEEE Trans. Med. Imag.*, vol. 7, pp. 225–232, Sept. 1988.

[8] S.-Y. Chen and W.-C. Lin, “Automated surface interpolation technique for 3-D object reconstruction from serial cross sections,” *Computerized Med. Imag. Graphics*, vol. 15, no. 4, pp. 265–276, 1991.

[9] S. P. Raya and J. K. Udupa, “Shape-based interpolation of multi-dimensional objects,” *IEEE Trans. Med. Imag.*, vol. 9, pp. 32–42, Mar. 1990.

[10] G. T. Herman, J. Zheng, and C. A. Bucholtz, “Shaped-based interpolation,” *IEEE Comput. Graphics Applicat.*, pp. 69–79, May 1992.

[11] J.-F. Guo, Y.-L. Cai, and Y.-P. Wang, "Morphology-based interpolation for 3D medical image reconstruction," *Computerized Med. Imag. Graphics*, vol. 19, no. 3, pp. 267–279, 1995.

[12] Y.-H. Liu, Y.-N. Sun, C.-W. Mao, and C.-J. Lin, "Edge-shrinking interpolation for medical images," *Comput. Vis., Graphics Image Processing*, vol. 21, no. 2, pp. 91–101, 1997.

[13] T.-Y. Lee and W.-H. Wang, "Morphology-based three-dimensional interpolation," *IEEE Trans. Med. Imag.*, vol. 19, pp. 711–721, July 2000.

[14] A. Goshtasby, D. A. Tuner, and L. V. Ackerman, "Matching tomographic slices for interpolation," *IEEE Trans. Med. Imag.*, vol. 11, pp. 507–516, Apr. 1992.

[15] D. T. Puff, D. Eberly, and S. M. Pizer, "Object-based interpolation via cores," in *Proc. SPIE*, vol. 2167, Medical Imaging'94, Image Processing, pp. 143–150.

[16] B. S. Morse, S. M. Pizer, and D. S. Fritsch, "Robust object representation through object-relevant use of scale," in *Proc. SPIE*, vol. 2167, Medical Imaging'94, Image Processing, 1994, pp. 104–115.

[17] V. Chatzis and I. Pitas, "Interpolation of 3-D binary images based on morphological skeletonization," *IEEE Trans. Med. Imag.*, vol. 19, pp. 699–710, July 2000.

[18] T. Berier and S. Neely, "Feature-based image metamorphosis," in *Comput. Graphics (SIGGRAPH'92 Proceedings), 26*, vol. 2, 1992, pp. 35–42.

[19] D. Cohen-Or, A. Solomovici, and D. Levin, "Three-dimensional distance field morphing," *ACM Trans. Graphics*, vol. 17, no. 2, Apr. 1998.

[20] I. Kompatsiaris, D. Tzovaras, V. Koutkias, and M. G. Strintzis, "Deformable boundary detection of stents in angiographic images," *IEEE Trans. Med. Imag.*, vol. 19, pp. 652–661, June 2000.

[21] M. Salotti, "An efficient algorithm for the optimal polygonal approximation of digitized curves," *Pattern Recogn. Lett.*, vol. 22, pp. 215–221, 2001.

[22] M. D. Carmo, *Differential Geometry of Curves and Surfaces*. Englewood Cliffs, NJ: Prentice-Hall, 1976.

[23] S. Cohen, G. Elber, and R. Bar Yehuda, "Matching of freeform curves," *Comput.-Aided Design*, vol. 29, no. 5, pp. 369–378, 1997.