Fast Feature-Based Metamorphosis and Operator Design

Tong-Yee LEE[1], Young-Ching LIN, Leeween LIN[2], Y.N. SUN

Department of Computer Science and Information Engineering

National Cheng-Kung University Tainan, Taiwan, R.O.C

Abstract

Metamorphosis is a powerful visual technique, for producing interesting transition between two images or volume data. Image or volume metamorphosis using simple features provides flexible and easy control of visual effect. The feature-based image warping proposed by Beier and Neely is a brute-force approach. In this paper, first, we propose optimization methods to reduce their warping time without noticeable loss of image quality. Second, we extend our methods to 3D volume data and propose several interesting warping operators allowing global and local metamorphosis of volume data.

1. Introduction

Metamorphosis or warping technique is a powerful tool to transform one image into another or generate new 3D model from a given one. Using this technique, many exciting visual effects in film and television is realized by animating complex models and changing their physical attributes such as shapes and positions. In general, 2D warping can be achieved by generating 2D images from 3D metamorphosis. Levoy et al [1] discuss the shortcomings of 2D warping and thus suggest 3D morphing. In the past, there have been many efforts done in 2D and 3D metamorphosis. Lee et al present a survey of 2D warping algorithms including those based on mesh morphing, field morphing, radial basis functions, thin plate splines, and energy minimization [2]. Prior work on field morphing [3] will be discussed in section 2. For mesh morphing [4], two meshes from the source and target images are used to define the spatial transformation that maps all points in the source image onto the target image. There is no folding or discontinuity allowed on mesh morphing. The radial basis functions and thin plate splines use more general form of features such as lines and curves [5]. On energy minimization method, the morphing transition is specified by physically meaningful energy terms and satisfied by minimizing their sum [6]. It results in a natural warping but with very high computation cost.

Several prior researches on volume morphing were presented in the past. Levoy et al extend [3] to 3D volume data and present optimization for computational efficiency. [7] attempts to automatically generate morphing without the aid of user input. Some work [7,8] transforms the volume data into frequency domain to perform morphing. Recently, Yagel et al [9] use ray deflector technique for volume morphing and allow local control of shape deformation. This approach computes morphing in those regions of volume data that contribute to the final image.

Beier and Neely propose a feature-based (field) metamorphosis method [3]. Using this method, an animator begins with establishing correspondence with pairs of feature primitives such as points and line segments between two images or models. In this paper, we attempt to optimize this method. The main contributions of this paper are as follows. First, we present two methods to improve [3] by taking into account the positions of features. The fastest of two proposed schemes achieves 20 times faster for our test images. Second, we extend the fastest method to 3D volume data as did in [1]. However, our extension allows both global and local control of volume deformation. Additionally, we describe several interesting morphing operators similar to

[1] Corresponding author: tonylee@mail.ncku.edu.tw

[2] Institute of Computer and Information Engineering, National Sun Yat-Sen Univ., Kaohsiung, Taiwan, ROC

[9], but allow more intuitive control of volume deformation. This paper is organized as follows. Section 2 will first review [3] and then describe proposed methods. Our experimental results on image metamorphosis will be shown and discussed in Section 3. In Section 4, we will describe several morphing operators for 3D volume data. Some concluding remarks and future work are given in Section 5.

2. Optimization on 2D Feature-based Morphing

In this section, we will first revisit a 2D feature-based morphing proposed by [3]. Then, we propose two optimized methods to reduce its morphing time. The proposed methods are termed: scan-line and optimized subdivision, respectively.

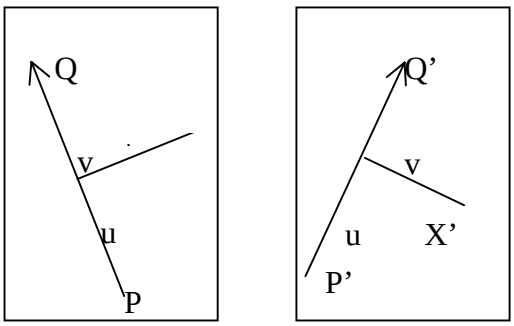
2.1 Feature-based Image Morphing

The metamorphosis technique [3] is accomplished based upon fields of influence surround two dimensional control primitives. First, the influence of all feature pairs on all pixels is computed. Second, a blending process is used to determine the colors of the morphed image. The features can be a single or multiple line pairs. In the following, we use identical equations used in [3] to describe this method. In [3], equations (1)(2)(3) are used to compute the influence of a single line pair on pixels (as shown in Figure 1).

$$u = \frac{(X\_P)\bullet(Q-P)}{\|Q-P\|^2} \quad (1)$$

$$v = \frac{(X-P)\bullet Perpendicular(Q-P)}{\|Q-P\|} \quad (2)$$

$$X' = P' + u\bullet(Q'-P') + \frac{v\bullet Pendicular(Q'-P')}{\|Q'-P'\|} \quad (3)$$



Destination Image Source Image

Figure 1: Single line pair

A single line pair for the source and destination images are $\overline{P'Q'}$ and $\overline{PQ}$, respectively. Using inverse mapping, this feature pair finds the pixel X in the destination image that corresponds to X' in the source image. *Perpendicular*() returns the vector perpendicular to the input vector. For the multiple line pairs, the *weight* of each pair is computed by the following equation:

$$weight = \left[\frac{length^p}{(a+dist)}\right]^b \quad (4)$$

where *length* is the length of a line, *dist* is the distance from the pixel to the line, and *a*,*b*, *p* are constants to control the effect of the lines. Finally, the morphing by the multiple feature pairs is computed by the multiple line algorithm presented in [3].

2.2 Scan-line Algorithm

The method proposed by Beier and Neely is a brute-force approach which computes every pixel to new location according to all feature line pairs. We propose a simple method termed as scan-line algorithm to approximate this method but with cheaper computational cost. Using equation (1)(2)(3) to transform a scan-line, we can get a line but rotated, scaled or translated (see Figure 2). Latter, we will show this transformation is a linear function. Therefore, on this basis, we morph two ending points only for every scan-line using (1)(2)(3) and interpolate warping for the remaining points using these two ending points. In this manner, we approximate the influence of a line pair, say F_i on a given scan-line, and term each transformed line T_i. Then, we use equation (4) and multiple line algorithm [3] to combine all T_is and accomplish warping of this scan-line. This combination is not a linear function, and thus a straight line could be distorted into a curve (see Figure 3). Next, we will show the transformation using by (1)(2)(3) is a linear function.

Assume that *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 mapped onto new coordinates A', B' and C'.

$$C = (1\text{-}s)A + sB \text{ and } s \in (1,0) \quad (5)$$

Using equation (1)(2)(3), we have

$$\begin{aligned} C' &= P' + u\bullet(Q'-P') + \frac{v\bullet Perpendicular(Q'-P')}{\|Q'-P'\|} \\ &= P' + P' - P' + \frac{(A - sA + sB - P + P - P)\bullet Perpndicular(Q-P)}{\|(Q-P\|^2}\bullet(Q'-P') \\ &\quad + \frac{\frac{(A - sA + sB + P - P + P)\bullet Perpendicular(Q-P)}{\|Q-P\|}\bullet Perpendicular(Q'-P')}{\|Q'-P'\|} \\ &= A' - sA' + sB' = (1-s)A' + B' \end{aligned} \quad (6)$$

From equation (5)(6), we know the transformation specified by a single line pair is a linear function.

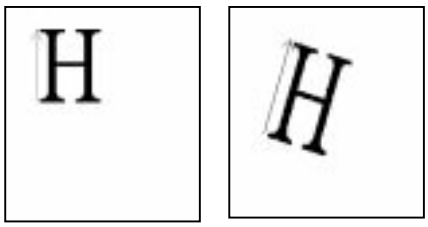


Figure 2. A single line pair

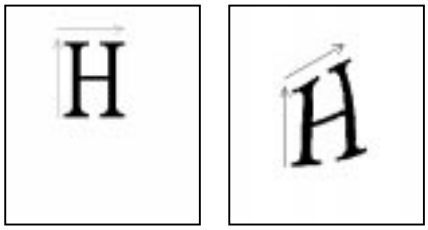


Figure 3. The multiple line pair

2.3 Optimized Subdivision Algorithm

This proposed method is similar to the piecewise linear approximation proposed by Levoy [1] but with further optimization. For the morphing in highly linear regions, the influence of all feature lines on every pixel within these regions is quite similar. Therefore, we decide to compute the morphing using simple linear approximation to reduce the computation time. We perform this approximation as follows. First, the target image is divided into a coarse regular grid and morph the grid vertices into the source image, equation (1)(2)(3) (see Figure 4). Then, we approximate the morphing for the pixels within the grid using bilinear interpolation. However, this approximation could not work for some grids where are highly non-linear influenced by the feature lines. To avoid this, we will pick some pixels within a given grid and compute these selected pixels by (1)(2)(3) as well as by using approximation. In case that the difference between two methods is large than a threshold, we will subdivide this grid more finely and then perform approximation. In this manner, we can potentially save a lot of computation since we just compute (1)(2)(3) for a small fraction of the pixels. The above process is identical to [1]. To further improvement, we would like to select test pixels in a smart manner as follows. For the highly non-linear regions, they are likely to be subdivided into the finer grids and thus, the test pixels are possibly computed several times using (1)(2)(3). To eliminate these redundant computations, we select candidate pixels in a uniform fashion as shown in Figure 5. The number of candidates is specified by the user input. In this figure, a large grid (*ABCD*) consists of several small regions (*w* by *h*). The black points shown in this figure are our selected pixels. In case a large grid (*ABCD*) requires further subdivision, we need not to compute ``black'' points in each small grid region (i.e, since their values are stored, and have been calculated early to test if *ABCD* grid is required for subdivision). Next, we take equation (4) into account to further improvement. In (4), the *weight* is in proportion to the inverse of the b^{th} power of distance. So, when a pixel is far from a feature line, it will be less influenced. Some feature lines will be likely to be ignored by taking (4) into account.

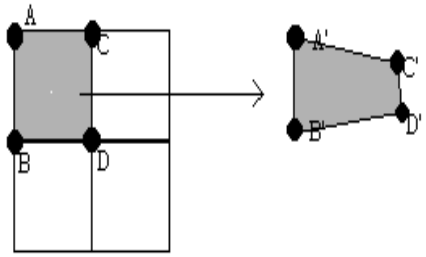


Figure 4. Bilinear interpolation

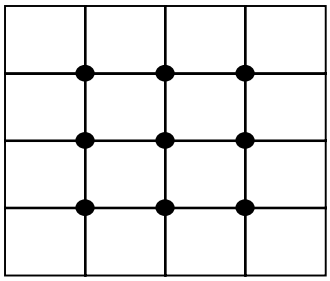


Figure 5. Uniform pixel selection

Assume the i^{th} feature line pair is denoted as F_i and transforms any pixel X onto X'. We have $D_i = X'_i - X$ and its *weight* is represented by *weight* (*i*). We formulate the morphing as:

$$X' = X + \sum_{i=1}^{N} \left(\frac{weight(i) \times D_i}{\sum_{j=1}^{N} weight(j)} \right) \qquad (7)$$

where N is the number of line pairs. This equation implies that *weight* (i) can be ignored if it is a small portion of $\sum_{j=1}^{j=N} weight(j)$. A threshold, say τ, is specified by the user input and S is defined as a set of F_i where *weight* (i)/ $\sum_{j=1}^{j=N} weight(j)$ is larger than τ. We approximate the formula (7) by:

$$X' = X + \sum_{i \in S} \left(\frac{weight(i) \times D_i}{\sum_{j \in S} weight(j)} \right) \qquad (8)$$

3. Experimental Results and Discussion

We implement proposed methods on the Intel Petium 133 PC platform. The experimental results are shown in Table 1. We use two 320 by 200 resolution images and impose 22 line pairs to conduct experiments. The scan-line method is slightly faster than [3], even though the number of pixels computed by (1)(2)(3) is reduced from 320 by 200 by 22 to 320 by 2 by 22. The main reason is that the cost spent in (1)(2)(3) is a small portion of the total warping time. In our experiments, it is about 20%. The other 80% time is spent on (4) and the multiple line algorithm. The method proposed by Levoy et al performs the next best; it achieves about 15 seconds to finish a image warping. This method uses bilinear interpolation for the highly linear regions to reduce computing time. The scan-line employs 1D interpolation but Levoy et al use 2D approach. The latter is expected faster and verified from our results. The optimized subdivision method is the best performer; it is faster than [3] and [1] by approximately 20 and 7.5 times, respectively. The improvement is due to that the number of feature pairs is decreased in each grid. For example in our experiments, the grids near to the corners are hardly influenced, and most feature pairs can be ignored. Finally, a morph sequence is shown in Figure 6 (shown in the end). Figures 7 (a) and (b) show feature lines on source and target images. To compare morphing quality, we measure average ($E(d)$) and variance ($\delta^2(d)$) of pixel distance (d) over two images between Beier et al's and proposed methods. The results are shown in Table 2.

	320 by 200 image and 22 line pairs
T. Beier et al	40 sec.
Scan-line algorithm	32 sec.
Levoy et al	15 sec.
Optimized Subdivision	2 sec.

Table 1. Experimental Results

	$E(d)$	$\delta^2(d)$
Scan_line	0.51	0.29
Levoy et al	2.10	10.61
Optimized subdivision	3.16	16.09

Two 320*200 images, 22 feature line pairs

Table 2. Morphing quality comparisons with Beier et al's method

Figure 7 (a) Feature lines on source image

Figure 7 (b) Feature lines on target image

4. Volume Deformation Using Features

Levoy et al [1] extend [3] to warp 3D volume data. In this section, we do identical extension but with the following difference. First, we use optimized subdivision to reduce warping time. Second, we design several interesting operators allowing both global and local deformation of volume data. Only global warping is reported in [1]. Yagel et al [9] propose similar operators using ray deflector technique. Using ray deflector, the data itself is not changed and only the directions of traced rays are altered. Our proposed scheme does change volume data. We can further deform this resulting data using additional features. Using ray deflector [9], an animator must impose correct order on the activation of various deflectors on each ray. If further warping is required, the rays are deflected using both newly added operators plus the older operators. From this view of point, our proposed operators seem more intuitive and efficient. In our extension, by defining a pair of elements (e, e'). Using inverse mapping, we can find a point *p* in the source volume that corresponds to p' in the destination volume.

4. 1Translate Operator

When we want to achieve the visual effect of pulling out the object, we can impose a translator operator on the object. See an example in Figure 8. In this figure, we want to pull out the nose starting with a MRI head scan. In our design, a translate operator is defined by a pair of $F = < e_0, e_1 \sim e_8 >$, where e_0 is a point and $e_1 \sim e_8$ are line segments to define a closed area. A simple translate operator is shown in Figure 8 (a)(b)(c). A e_0 pair is used to control the effect of translation. A pair of $<e_1 \sim e_8>$ using the same line segments is used to freeze the area of volume bounded by $<e_1 \sim e_8>$. The warping by a pair of *F* is computed by equation (7). Using (7), since the area bounded by $<e_1 \sim e_8>$ is far from e_0, it is less influenced by this feature point. In this manner, this area can be seen as unchanged. Figure 8 shows an example of using a translate operator. In this example, the nose can be further pulled longer (Figure 8 (c)) on the basis of intermediate volume (Figure 8(b)). The exact deformation for the pulled nose is determined by (e_0, e'_0).

4.2 Discontinuous Operator

As suggested in [9], a discontinuous operator can be used to generate cuts on volume data. Our discontinuous operator is designed as follows. First, we specify two line segments <*L*, *L*> (i.e., same line segments) for the source features and $<L_1, L_2>$ for the destination features (define the crack on volume data) as shown in Figure 9. This operator will generate two cutting planes with the normal vectors n_1 and n_2, respectively. This is a localized operator and is limited by a constant *D* measured by the distance from each cutting plane to *L*. The warping by the discontinuous operator is computed by following formulas,

$$p_i = \begin{cases} p'_i & \text{if } d > D \quad or \quad u > 1 \\ p'_i + (1 - \frac{d^2}{D^2})^k (p_i - p'_i) & \text{if } d \le D \end{cases}$$

$$u = \frac{(X - P) \cdot (Q - P)}{\|Q - P\|^2}$$

$$w_i = \begin{cases} 0 \ \ if & \overrightarrow{p'c} \cdot \overrightarrow{n_i} > 0 \\ w_i \ if & \overrightarrow{p'c} \cdot \overrightarrow{n_i} \le 0 \end{cases}$$

where *d* is the shortest distance from the point p_i to the closer one of two cutting planes. Similarly, we use equation (4) to compute *weight* but with an additional condition: only the left side of *L* is effected by L_1 and only the right side of *L* is effected by L_2. This test can be determined by the sign of $\overline{p'_i c} \cdot \overline{n_i}$, where *c* is the central point of L_i. $u > 1$ means P_i is out of line segment. In our experiment, *k* is varying from 1 to 4. Figure 10 shows an example of cutting into a MRI scan head using this operator.

4.3 Scale Operator

Another localized operator introduced is scale operator. We defined this operator by the following equation,

$$p_i = \begin{cases} p'_i & \text{if } d > D \\ p'_i + (1 - \frac{d^2}{D^2})^2 (p_i - p'_i) & \text{if } d \le D \end{cases}$$

For a scale operator, we will specify a line segment *e* and *D*. The shortest distance between *p* and *e* is *d*. If *d* is less than *D* ,then *p* will be influenced by this local operator. In Figure 11, we first use a discontinuous operator to create a crack on the sphere and then use a scale operator to enlarge the crack.

4.4 Optimization

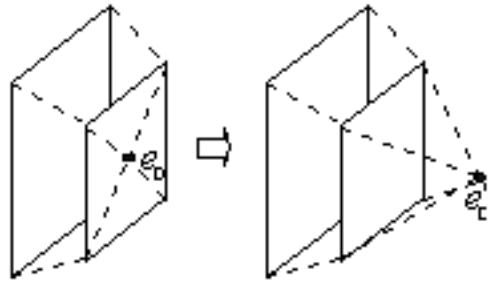
Finally, we show the experimental results for our designed operators. The experiments were conducted on SUN sparc-20 workstation. The size of the MRI head scan is 128 by 128 by 84 and the size of concentric spheres is 128 by 128 by 128. We use the proposed optimization to speed up computation. Table 3 shows the timings for both non-optimized and optimized methods. From table 3, we see morphing with optimization is consistently faster than non-optimized method. For the global operator like translate operator, it achieves better speed up. For the local operator like discontinuous operator, its speed up is not so significant.

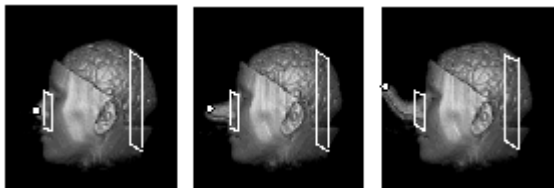
Examples	Dataset	Optimize	Execution time(sec)
1st nose	Head 128*128*84	Non	202.5052
		Yes	9.17017
2nd nose		Non	201.7112
		Yes	9.210818
cut brain		Non	68.2796
		Yes	24.27296
scale	Sphere 128*128*128	Non	49.22352

Table 3. Experimental results for volume morphing

5. Conclusion and Future Work

In this paper, we propose efficient approximation methods for the feature-based image warping technique. We extend proposed methods to 3D volume data and achieve faster warping computation. Several warping operators are proposed to achieve special visual effects on volume data. These operators can be either global (translate) or local (discontinuous and scaling). An animator can impose operators on data in a very intuitive manner. Modeling with multiple operators is easy. An example is shown in Figure 11. In this example, a discontinuous operator is first imposed and then a scaling operator is imposed next. Finally, we plan to design more types of operators such as twisting. We will use these designed operators in our surgical simulation project collaborating with Hospital of National Cheng-Kung University in Taiwan.





(a) (b) (c)

Figure 8. A translator operator and its application (a)(b)(c)

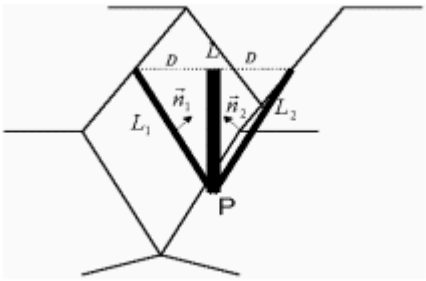


Figure 9 A discontinuous operator

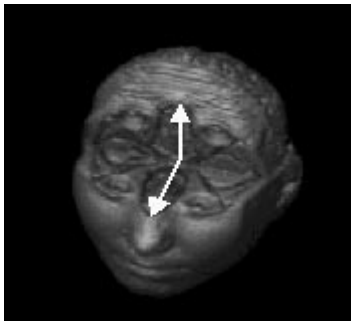
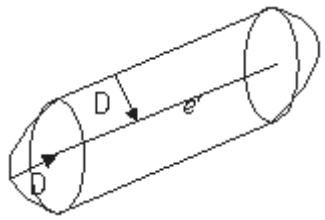
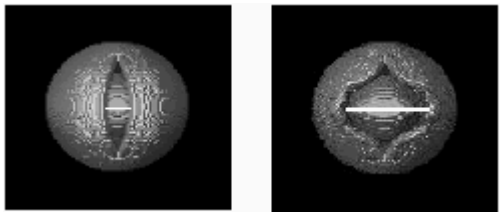


Figure 10. An example of a cutting using a discontinuous operator



Scale: $e' = e$ but with a D



Cut: Scale: enlarge the cut

Figure 11. Scale operator and its application

Reference

1. A. Lerios, C. Garfinkle, and M. Levoy, ``Feature-based Volume Metamorphosis,'' Proceedings of SIGGRAPH'95, pp. 449-456, 1995.
2. Seungyong Lee, Geroge Wolberg, Kyung-Yong Chwa, and Sung Yong Shin, ``Image Metamorphosis with Scattered Feature Constraints,'' IEEE Transactions on Visualiaztion and Computer Graphics, Vol. 2, No. 4, Dec., 1996, pp. 337-354.
3. T. Beier and S. Neely, ``Feature-based image metamorphosis,'' in Computer Graphics, vol 26(2),pp 35-42,New York, NY, July 1992. Proceeding of SIGGRAPH '92.
4. Wolberg,G.,"Digital Image Warping". IEEE Computer Society Press, 1990.
5. D. Ruprecht and H. Muller, ``Image Warping with Scattered Data Interpolation,'' IEEE Computer Graphics and Applications, vol. 15, pp. 37-43, Mar. 1995.
6. S. Lee, K-Y, Chwa, J. Hahn, and S.Y. Shin, ``Image Morphing Using Deformation Techniques,'' J. Visualization and Computer Animation, Vol. 7, no. 1, pp.3-23, 1996.
7. T. He, S. Wang, and A. Kaufman, ``Wavelet-based volume morphing,'' Proceedings of Visualization'94, pp. 85-91.
8. J. F. Hughes, ``Scheduled Fourier Volume Morphing,'' Proceedings of SIGGRAPH'92, pp. 43-46.
9. Y. Kurzion and R. Yagel, ``Space Deformation Using Ray Deflectors,'' 6th Eurographics Workshop on Rendering'95, pp. 21-32.

Acknowledgements

This research is supported by the National Science Council of Taiwan, ROC, project No. NSC-87-2213-E-006-012

Figure 6. A morph sequence