Image Metamorphosis with Scattered Feature Constraints

Seungyong Lee, George Wolberg, Kyung-Yong Chwa, and Sung Yong Shin

Abstract—This paper describes an image metamorphosis technique to handle scattered feature constraints specified with points, polylines, and splines. Solutions to the following three problems are presented: feature specification, warp generation, and transition control. We demonstrate the use of snakes to reduce the burden of feature specification. Next, we propose the use of multilevel free-form deformations (MFFD) to compute C^2-continuous and one-to-one mapping functions among the specified features. The resulting technique, based on B-spline approximation, is simpler and faster than previous warp generation methods. Furthermore, it produces smooth image transformations without undesirable ripples and foldovers. Finally, we simplify the MFFD algorithm to derive transition functions to control geometry and color blending. Implementation details are furnished and comparisons among various metamorphosis techniques are presented.

Index Terms—Image metamorphosis, morphing, snakes, multilevel free-form deformation, multilevel B-spline interpolation.

1 Introduction

Image metamorphosis has proven to be a powerful tool for visual effects. There are now many breathtaking examples in film and television depicting the fluid transformation of one digital image into another. This process, commonly known as *morphing*, is realized by coupling image warping with color interpolation. Image warping applies 2D geometric transformations on the images to retain geometric alignment between their features, while color interpolation blends their color.

Image metamorphosis between two images begins with an animator establishing their correspondence with pairs of feature primitives, e.g., mesh nodes, line segments, curves, or points. Each primitive specifies an image feature, or landmark. The feature correspondence is then used to compute mapping functions that define the spatial relationship between all points in both images. Since mapping functions are central to warping, we shall refer to them as warp functions in this paper. They will be used to interpolate the positions of the features across the morph sequence. Once both images have been warped into alignment for intermediate feature positions, ordinary color interpolation (i.e., cross-dissolve) is performed to generate an inbetween image.

The explosive growth of image morphing is due to the compelling and aesthetically pleasing effects possible through warping and color blending. The extent to which artists and animators can effectively use morphing tools is directly tied to solutions to the following three problems: feature specification, warp generation, and transition control. Together, they influence the ease and effectiveness in generating high-quality metamorphosis sequences. We have already reported effective solutions towards these problems in [1]. In this paper, we describe further insights and implementation details.

Feature specification is the most tedious aspect of morphing. Although the choice of allowable primitives may vary, all morphing approaches require careful attention to the precise placement of primitives. In this paper, we demonstrate the use of snakes to reduce this burden. Snakes are energy minimizing splines that move under the influence of image and constraint forces. They were first adopted in computer vision as an active contour model [2]. Derived from primitives, snakes assist us in feature specification because we need only position them near the features. Image forces push snakes toward salient edges, thereby refining their final positions and making it possible to capture the exact position of a feature easily and precisely.

Given feature correspondence constraints between both images, a warp function over the whole image plane must be derived. This process, which we refer to as warp generation, is essentially an interpolation problem. To derive warp functions from scattered feature (positional) constraints, we propose the multilevel free-form deformation (MFFD) as an extension to free-form deformation (FFD) [3]. We take the bivariate cubic B-spline tensor product as the deformation function of FFD. A new direct manipulation technique for FFD, based on 2D B-spline approximation, is developed in this paper. We apply it to a hierarchy of control lattices to exactly satisfy the feature constraints. Furthermore, we present a sufficient condition for a 2D cubic B-spline surface to be one-to-one. This guarantees that the resulting warp is one-to-one, i.e., the distorted image does

- *S. Lee is with the Department of Computer Science and Engineering, Pohang University of Science and Technology (POSTECH), Pohang, 790-784, Korea. E-mail: leesy@postech.ac.kr.*
- *G. Wolberg is with the Department of Computer Science, City College of New York, New York, NY 10031. E-mail: wolberg@cs-mail.engr.ccny.cuny.edu.*
- *K.-Y. Chwa and S.Y. Shin are with the Department of Computer Science, Korea Advanced Institute of Science and Technology (KAIST), Taejon, 305-701, Korea. E-mail: {kychwa, syshin}@jupiter.kaist.ac.kr.*

For information on obtaining reprints of this article, please send e-mail to: transvcg@computer.org, and reference IEEECS Log Number V96030.

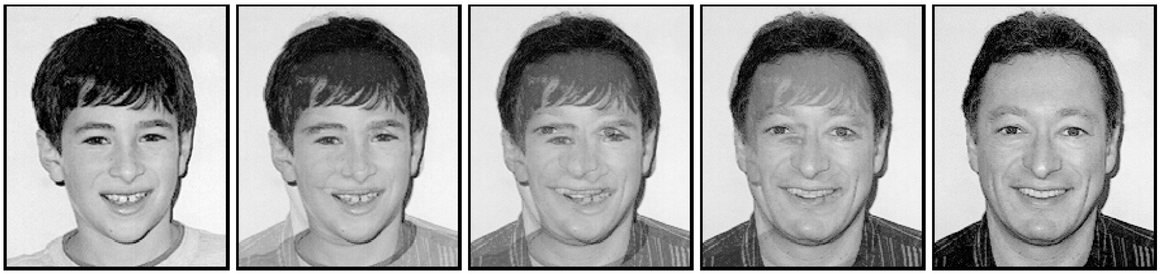


Fig. 1. Cross-dissolve. Image transitions are achieved by linearly interpolating pixel colors to fade from one image to another.

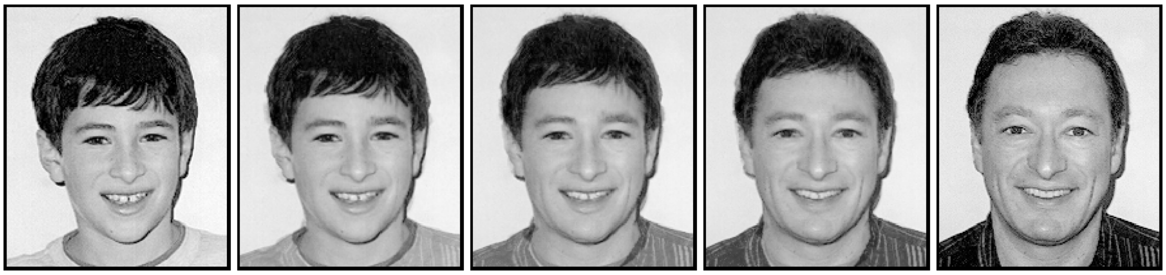


Fig. 2. Morph sequence. Warping maintains geometric alignment throughout the cross-dissolve to achieve fluid transformations.

not fold back upon itself. The MFFD generates C^2-continuous and one-to-one warps which yield fluid image distortions. It is much simpler and faster than a related energy minimization method [4]. We also present a hybrid approach that combines the two methods.

Another interesting problem in image morphing is transition control. If transition rates are allowed to vary locally across inbetween images, more interesting animations are possible. An effective method for transition control has been proposed in [4]. Transition rates on an inbetween image are derived from transition curves by constructing a smooth surface. The surface represents the propagation of transition rates defined by the user at sparse positions across the image. In this work, the MFFD technique for warp generation is simplified and applied to efficiently generate a C^2-continuous surface. The simplification is due to relaxation of the one-to-one property.

A tradeoff exists between the complexity of feature specification and warp generation. As feature specification becomes more convenient, warp generation becomes more formidable. The recent introduction of spline curves to feature specification raises a challenge to the warp generation process, making it the most critical component of morphing. It influences the smoothness of the transformation and dominates the computational cost of the morphing process. We shall comment on these tradeoffs and describe their role in influencing recent progress in this field.

This paper is organized as follows. Section 2 reviews previous work. A general metamorphosis framework is given in Section 3. The use of snakes for feature specification is described in Section 4. Section 5 introduces the MFFD technique for warp generation. Section 6 describes a simple variation to the MFFD technique for obtaining multilevel B-spline interpolation for transition control. Examples and conclusions are given in Sections 7 and 8, respectively.

2 Previous Work

Before the development of morphing, image transitions were generally achieved through the use of cross-dissolves, e.g., linear interpolation to fade from one image to another. Fig. 1 depicts this process applied over five frames. The result is poor, owing to the double-exposure effect apparent in misaligned regions. This problem is particularly apparent in the middle frame, where both input images contribute equally to the output. Morphing achieves a fluid transformation by incorporating warping to maintain geometric alignment throughout the cross-dissolve process. The important role that warping plays is readily apparent by comparing the morph sequence in Fig. 2 with the cross-dissolve result in Fig. 1.

In this section, we review several morphing algorithms, including those based on mesh warping, field morphing, radial basis functions, thin plate splines, and energy minimization. This review is intended to motivate the discussion of progress in feature specification and warp generation.

2.1 Mesh Warping

Mesh warping was pioneered at Industrial Light & Magic (ILM) by Douglas Smythe for use in the movie *Willow* in 1988. It has been successfully used in many subsequent motion pictures. A full description of this method appears in [5]. To illustrate the two-pass mesh warping algorithm, consider the image sequence shown in Fig. 2. The five frames in the figure represent a metamorphosis (or morph) between the two faces at both ends of the row. We will refer to these two images as I_0 and I_1, the source and the target images, respectively. The source image has mesh M_0 associated with it that specifies the coordinates of control points, or landmarks. A second mesh, M_1, specifies their corresponding positions in the target image. Meshes M_0 and M_1 are, respectively, shown overlaid on I_0 and I_1 in Fig. 3. Notice that landmarks such as the eyes, nose, and lips lie below corresponding grid lines in both meshes.

Together, M_0 and M_1 are used to define the spatial transformation that maps all points in I_0 onto I_1. The meshes are constrained to be topologically equivalent, i.e., no folding or discontinuities are permitted. Therefore, the nodes in M_1 may wander as far from M_0 as necessary, as long as they do not cause self-intersection. Furthermore, for simplicity, the meshes are constrained to have frozen borders.

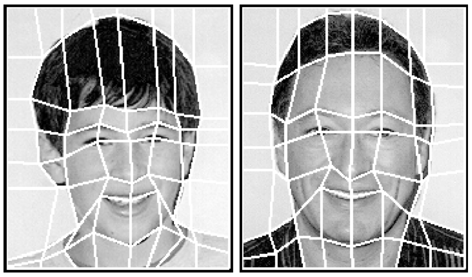


Fig. 3. The source and target images with overlaid meshes.

The use of meshes for feature specification facilitates a straightforward solution for warp generation: bicubic spline interpolation. The morph sequence in Fig. 2 was produced using mesh warping with Catmull-Rom spline interpolation to determine the correspondence of all pixels. Fant's algorithm was used to resample the image in a separable implementation [6], [5].

2.2 Field Morphing

While meshes appear to be a convenient manner of specifying pairs of feature points, they are, however, sometimes cumbersome to use. In particular, a control mesh is always required although the structure of the image features may be arbitrary. The field morphing algorithm developed by Beier and Neely [7] at Pacific Data Images grew out of the desire to simplify the user interface to handle correspondence by means of line pairs with arbitrary configurations. A pair of corresponding lines in the source and target images defines a coordinate mapping between the two images. In addition to the straightforward correspondence provided for all points along the lines, the mapping of points in their vicinity can be determined by their distance from the line. Since multiple line pairs are usually given, the displacement of a point in the source image is actually a weighted sum of the mappings due to each line pair, with the weights attributed to distance and line length.

This approach has the benefit of being more expressive than mesh warping. For example, rather than requiring the correspondence points of Fig. 3 to all lie on a mesh, line pairs can be drawn along the mouth, nose, eyes, and cheeks of the source and target images. Therefore, only key feature points need be given.

Although this approach simplifies the specification of feature correspondence, it complicates warp generation. This is due to the fact that all line pairs must be considered before the mapping of each source point is known. This global algorithm is slower than mesh warping, which uses bicubic interpolation to determine the mapping of all points not lying on the mesh. A more serious difficulty, though, is that unexpected displacements may be generated after the influence of all line pairs are considered at a single point. The resulting distortions, referred to as *ghosts*, may prevent an animator from realizing a precise warp in a complex metamorphosis. Additional line pairs must sometimes be supplied to counter the ill-effects of a previous set. In the hands of talented animators, though, the mesh warping and field morphing algorithms have both been used to produce startling visual effects.

2.3 Radial Basis Functions / Thin Plate Splines

The most general form of feature specification permits the feature primitives to consist of points, lines, and curves. Since lines and curves can be point sampled, it is sufficient to consider the features on an image to be specified by a set of points. In that case, the x- and y-components of a warp can be derived by constructing the surfaces that interpolate scattered points. Consider, for example, n feature points labeled (u_k, v_k) in the source image and (x_k, y_k) in the target image, where $1 \le k \le n$. Deriving warp functions that map points from the target image to the source image is equivalent to determining two smooth surfaces: one that passes through points (x_k, y_k, u_k) and the other that passes through (x_k, y_k, v_k) for $1 \le k \le n$.

This formulation permits us to draw upon a large body of work on scattered data interpolation to address the warp generation problem. All subsequent morphing algorithms have facilitated general feature specification by appealing to scattered data interpolation.

Warp generation by this approach was extensively surveyed in [8], [5]. Recently, two similar methods were independently proposed using the thin plate surface model [9], [10]. Another method using radial basis functions was described in [11]. These techniques generate smooth warps that exactly reflect the feature correspondence. Furthermore, they offer the most general form of feature specification since any primitive (e.g., spline curves) may be sampled into a set of points. Elastic Reality, a popular morphing package from Avid Technology, uses curves to enhance feature specification. Their warp generation method, however, is unpublished.

2.4 Energy Minimization

All of the methods described above do not guarantee the one-to-one property of the generated warp functions. When a warp is applied to an image, the one-to-one property prevents the warped image from folding back upon itself. An energy minimization method has been proposed for deriving one-to-one warp functions in [4]. That method allows extensive feature specification primitives such as points, polylines, and curves. Internally, all primitives are sampled and reduced to a collection of points. These points are then used to generate a warp, interpreted as a 2D deformation of a rectangular plate. A deformation technique is provided to derive C^1-continuous and one-to-one warps from the positional constraints. The requirements for a warp are represented by energy terms and satisfied by minimizing their sum. The technique generates natural warps since it is based on physically meaningful energy terms. The performance of that method, however, is hampered by its high computational cost.

2.5 Discussion

The progression of morphing algorithms has been marked by more expressive and less cumbersome tools for feature specification. A significant step beyond meshes was made possible by the specification of line pairs in field morphing. The complications that this brought to warp generation, however, sometimes undermined the usefulness of the approach. For instance, the method sometimes demonstrated undesirable artifacts (e.g., ghosts) due to the computed warp function [7]. To counter these problems, the user is required to specify additional line pairs, beyond the minimal set that would otherwise be warranted. All subsequent algorithms, including those based on radial basis functions, thin plate splines, and energy minimization, formulated warp generation as a scattered data interpolation problem and sought to improve the quality (smoothness) of the computed warp function. They do so at relatively high computational cost.

This paper advances a new approach based on multilevel free-form deformation (MFFD). It is much simpler and faster than the related energy minimization method in [4]. The MFFD algorithm facilitates C^2-continuous warp functions and significantly improves performance by accelerating warp generation. The method was first described in [1]. In this paper, we have broadened the presentation to include further insights and implementation details.

One important benefit of MFFD-based morphing is that feature specification is more expressive and less cumbersome. Rather than editing a mesh, for instance, only a small set of features must be specified. Consider the features depicted in Fig. 4. They were used to generate a morph sequence that is visually indistinguishable from that of Fig. 2. To further assist the user, snakes can be introduced to reduce the burden of feature specification. Snakes streamline feature specification because primitives must only be positioned near the features. Image forces push snakes toward salient edges, thereby refining their final positions and making it possible to capture the exact position of a feature easily and precisely.

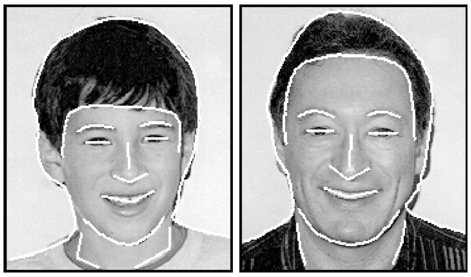


Fig. 4. Feature specification for MFFD-based morphing.

3 Metamorphosis Framework

In this section, we summarize the general framework presented in [4], which encompasses the three components shared by all morphing algorithms: feature specification, warp generation, and transition control. The framework serves to highlight the interaction of these components, independent of implementation methodologies. The role of the transition control function for enhancing morph sequences is demonstrated for the uniform and nonuniform cases.

Let F_0 and F_1 be two sets of features specified by an animator on the source and target images I_0 and I_1, respectively. For each feature f_0 in F_0, there exists a corresponding feature f_1 in F_1. Let W_0 and W_1 be the warp functions that specify the corresponding point in I_1 and I_0 for each point in I_0 and I_1, respectively. When it is applied to I_0, W_0 generates a distorted image so that the features in F_0 coincide with their corresponding features in F_1. W_1 is required to map features f_1 onto f_0 when it distorts I_1. Although W_1 is the inverse of W_0 at the features, this is not necessarily true at other positions across the image. Together, W_0 and W_1 serve to retain geometric alignment of the features during the morph.

Transition functions specify a transition rate for each point on the given images over time. Let T_0 be a transition function defined for source image I_0. For a given time t, $T_0(p;\, t)$ is a real-valued function that determines how fast each point p in I_0 moves towards the corresponding point q in target image I_1. $T_0(p;\, t)$ also determines the color contribution of each point p in I_0 to the corresponding point in an inbetween image $I(t)$.

Let T_1 be the transition function for target image I_1. For each point q in I_1, $T_1(q;\, t)$ is defined to have the same transition rate as $T_0(p;\, t)$ if q corresponds to p in I_0. Hence, $T_1(q;\, t)$ can be derived from $T_0(p;\, t)$ using warp function W_1. That is, $T_1(q;\, t) = T_0(W_1(q);\, t)$. For simplicity, we treat the transition functions for both geometry and color to be identical, although they may be different in practice.

Let $W \bullet I$ denote the application of warp function W to image I. The procedure for generating an inbetween image $I(t)$ can be described as follows.

$$\overline{W_0}(p;t) = \left(1 - T_0(p;t)\right)\cdot p + T_0(p;t)\cdot W_0(p)$$
$$\overline{W_1}(q;t) = T_1(q;t)\cdot q + \left(1 - T_1(q;t)\right)\cdot W_1(q)$$
$$\bar{I}_0(p;t) = \overline{W_0}(p;t)\bullet\left(\left(1 - T_0(p;t)\right)\cdot I_0(p)\right)$$
$$\bar{I}_1(q;t) = \overline{W_1}(q;t)\bullet\left(T_1(q;t)\cdot I_1(q)\right)$$
$$I(r;t) = \bar{I}_0(r;t) + \bar{I}_1(r;t).$$

Note that $0 \le T_0, T_1 \le 1$, and $0 \le t \le 1$. Transition rates zero and one imply the source and target images, respectively.

Figs. 5 and 6 show examples of the use of uniform and nonuniform transition functions, respectively. The upper left and lower right images of Fig. 5 are the source and target images I_0 and I_1, respectively. The feature sets F_0 and F_1 used to define warp functions W_0 and W_1 are shown overlaid on the two images. Functions $\overline{W_0}$ and $\overline{W_1}$ are generated using uniform transition functions $T_0(p;\, t) = t$ and $T_1(p;\, t) = t$ for all points p. Applying $\overline{W_0}$ to I_0 yields the set of inbetween images given in the top row of Fig. 5. Applying $\overline{W_1}$ to I_1 yields the set of inbetween images given in the bottom row of that figure. Those two rows are attenuated by T_0 and T_1, respectively, and added together to yield the middle row of inbetween images. Notice that geometric alignment is maintained among the two sets of warped inbetween

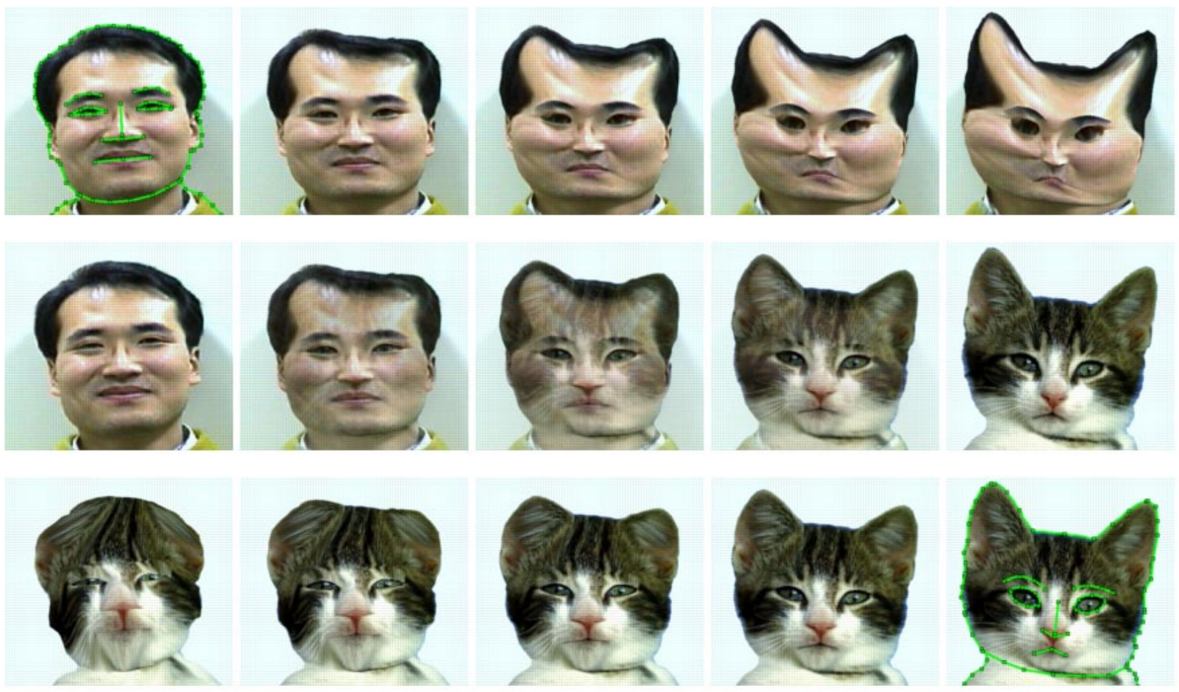


Fig. 5. Uniform metamorphosis. All pixels change uniformly between their positions and colors in the source and target images.

images before color blending merges them into the final morph sequence.

The example in Fig. 6 demonstrates the effects of a non-uniform transition function applied to the same source and target images. In this example, a transition function was defined that accelerated the deformation of the nose of the source image, while leaving the shape of the head intact for $t \leq 0.5$. The deformation of the head begins at $t = 0.5$ and continues linearly to $t = 1$. The same transition function was used for T_1. Notice that this use of nonuniform transition functions is responsible for the dramatic improvement in the morph sequence.

In the remainder of this paper, we address the following three problems common to all metamorphosis techniques:

- how to specify feature sets F_0 and F_1,
- how to derive warp functions W_0 and W_1, and
- how to derive transition functions T_0 and T_1.

4 Feature Specification

Feature specification plays an important role in morphing. It permits us to identify those image landmarks that must remain in geometric alignment as the morph proceeds from the source image to the target image. This task is critical to produce the warps that must precede color blending (see Section 3). Failure to accurately match corresponding features gives rise to the annoying double-exposure artifacts typical of ordinary cross-dissolves.

The position of a feature is identified by the user with feature primitives overlaid upon edges, where color values change abruptly. Features on a facial image, for example, may consist of the profile, eyes, nose, and mouth. Several forms of primitives are popular, including meshes [5], [12], line segments [7], points, polylines, and curves [9], [10], [4].

In this work, we also use points, polylines, and curves to specify features. To further reduce the burden of an animator, we adopt snakes [2], an active contour model made popular in computer vision. Snakes simplify feature specification because the user need only position a primitive near the feature. Large nearby image gradients refine the initial placement of primitives, as they slither towards the precise feature position. Furthermore, snakes permit us to define primitives with fewer control points because they can be made to lie on the underlying features with the guidance of only a small set of well-placed control points.

4.1 Snakes

Snakes [2] are energy-minimizing splines under the influence of image and constraint forces. The spline energy serves to impose a piecewise smoothness constraint on a snake. The image forces push the snake toward salient image features such as lines, edges, and subjective contours. The constraint forces are used for pulling the snake to a desired image feature among the nearby ones. Snakes have proven to be useful for the interactive specification of image features.

Representing the position of a snake in parametric form, $v(s) = (x(s), y(s))$, where $s \in [0, 1]$, its energy functional can be written as

$$E_{snake}(v) = \int_0^1 \left[E_{spline}(v) + E_{image}(v) \right] ds.$$

E_{spline} represents the spline energy due to bending, and E_{image} is the energy defined from the intensity distribution of an image. We have removed the term related to the constraint forces because it is not used in this work. We also simplify the spline energy to $E_{spline} = \beta \left| \frac{d^2 v}{ds^2} \right|^2$, which makes a snake act like a thin plate.

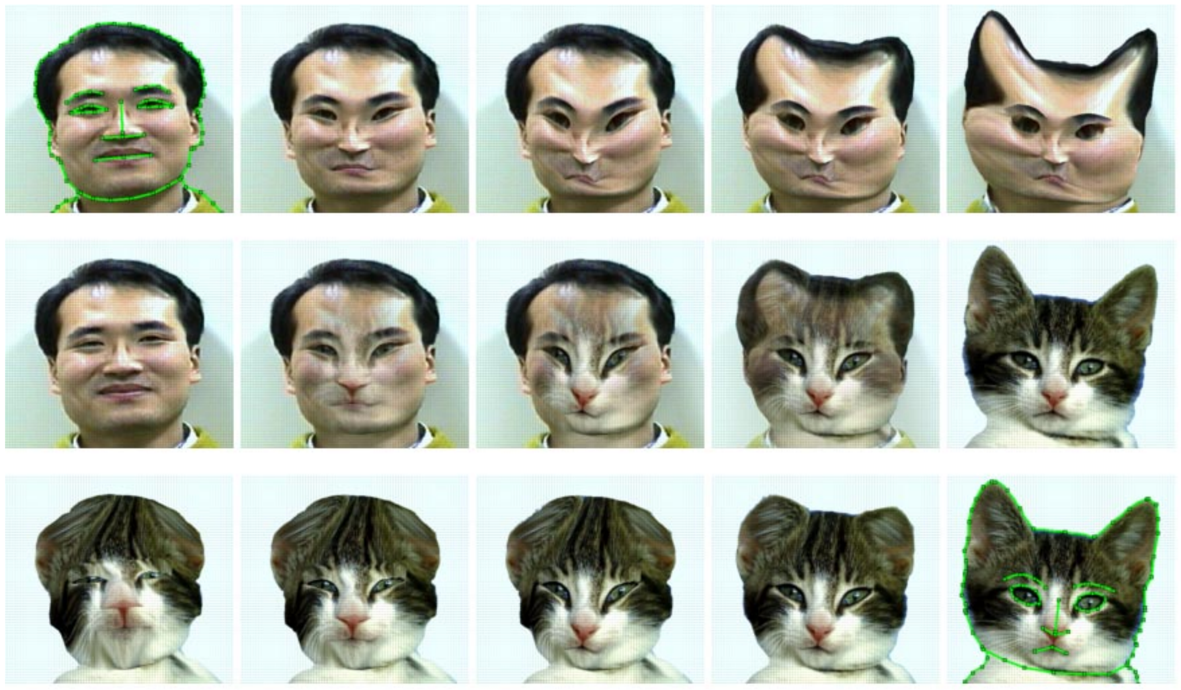


Fig. 6. Nonuniform metamorphosis. Each pixel may migrate at different rates between the source and target images.

For a grayscale image I, the gradient ∇I measures the local changes of image values and can be computed by a difference operator or the Sobel operator [13]. The image energy functional can be defined by $E_{image} = -\nabla^2 I = -\left|\nabla I(x, y)\right|^2$. It makes a snake precisely localize a feature at a boundary having large image gradients. While minimizing the energy functional E_{snake}, the snake slithers from its initial position to a nearby feature.

A feature is allowed to attract a distant snake if image gradients are convolved with a smoothing filter. For example, the convolution results in an image energy functional, $E_{image} = -(G_\sigma * \nabla^2 I)$, where G_σ is a Gaussian of standard deviation σ. Other image energy functionals can be found in [2].

To derive the desired snake, the energy minimization problem is converted to a differential equation. The equation is then discretized to a system of nonlinear equations by the finite difference method. The solution of the system is computed by iteratively solving the linear system until its equilibrium is reached. Hence, a snake consists of a sequence of points that gets updated at each iteration in the loop. The illusion of a moving snake can be rendered by repeatedly drawing a polyline through the updated points.

4.2 Specification of Features and Their Correspondence

In this paper, the feature specification primitives include points, polylines, and curves. Snakes are provided as an operation applicable to polyline or curve primitives. They permit us to place a primitive near a feature and then subject it to image forces that refine its position. Snakes work best when the feature has large image gradients.

When the snake operation is applied to a polyline or curve, a sequence of points is uniformly sampled on it, e.g., 20 points per segment. Then, a constant number of iterations in the energy minimization loop is performed to move the sampled points toward the features. While the snake operation is applied several times, the points slither and finally lock onto the feature by the image force.

To tailor the response of the snake operation, the user may clamp any of the sampled points in place. Internally, this is achieved by assigning a large value to the parameter γ in [2] for the selected points. Since it is often tedious to select among the many sampled points, we provide an option for fixing those that lie on the control points of the initial primitive.

If the results of the snake operation are not deemed satisfactory, a polyline or curve can be recovered by the unsnake operation. In that case, the polyline or curve is generated from those snake points that correspond to the control points of the initial primitive. After editing the position of these control points, the user can again apply the snake operation to refine the position of the specified feature.

When there are several features near a snake, the snake operation can sometimes fail to capture the desired feature. In that case, the user can select any snake point and move it toward the feature. Then, the following snake operation may give a successful response because some parts of the snake are now closer to the desired feature than others.

Once a feature specification primitive f_0 is placed on image I_0, a primitive f_1 is also deposited on the other image I_1. To guarantee that f_0 and f_1 have the same number of control points, we let f_1 be a copy of f_0. We either move f_0 repeatedly or use the snake operation to identify a feature on I_0. Primitive f_1 must then be moved to designate the corresponding feature on I_1. Again, the snake operation may be applied to f_1, if necessary.

To derive the set of feature point pairs from the specified features, a set of points are sampled on polylines or curves. For the sampling, we apply the same method used for deriving snake points from polylines or curves. This makes all nonpoint features have the same number of sampled points on each segment. Feature point pairs between two images are then derived from the sampled points on corresponding feature primitives.

Fig. 7 shows an example. Fig. 7a is the input image. We convert it to a grayscale image and apply the Sobel operator [13] to compute image gradients. Fig. 7b shows the image gradients convolved with a Gaussian filter, where bright intensities denote large gradients. In Fig. 7c, we place a polyline near the profile of the image. The snake starting from the polyline exactly captures the profile, as in Fig. 7d. Fig. 7e illustrates the specified feature primitives overlaid on the image. The cyan points in Fig. 7f represent internally sampled feature points on the primitives. We typically use the sampling rate of 20 points per primitive segment, although only four points per segment are shown in the figure.

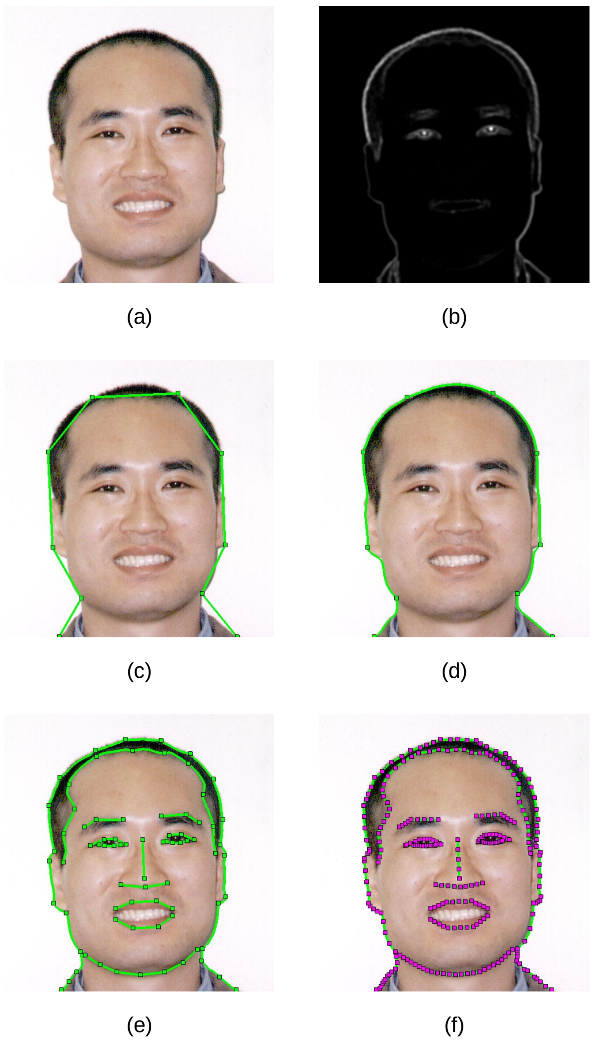


Fig. 7. Feature specification. (a) Input image; (b) image gradients; (c) polyline placed near profile; (d) snake locks onto profile; (e) feature primitives; (f) internally sampled feature points.

4.3 Discussion

The snake operation presented in this section facilitates the precise specification of image features with moderate user interaction. The effectiveness of snakes drops when several features are close to each other or when the image gradient near a feature is low. Although editing snake points can be useful in such cases, it is often necessary to repeatedly apply the edit and snake operations.

When two polylines or curves are used for feature correspondence, their control points are placed on the matching parts of the features. The feature point pairs derived by uniformly sampling the primitives properly reflect the correspondence specified by the control points. When the snake operation is applied to one or both of them, the sampled points move freely and independently of each other. This compromises the accuracy of feature correspondence because the matching points may move to other parts of the features, thereby, misregistering the user-specified point pairs.

The point clamp option makes the control points of a polyline or curve stay in place when the snake operation is applied. With this option, a snake operation does not hamper the feature correspondence information specified by the control points. Furthermore, in that case, the points not sampled from control points do not move arbitrarily because they are related to the fixed control points by the spline force of a snake. Hence, the careful handling of the control points alone is sufficient to derive precise feature positions and their correspondence.

5 Warp Generation

Feature correspondence among images I_0 and I_1 is established by two sets of points, P and Q. The points may either be explicitly given constraints or, more commonly, they are samples from user-specified feature primitives. In either case, they provide correspondence information at a set of sparse and irregular positions. The warp generation process is responsible for smoothly propagating this information to all points in the image plane to determine warp functions W_0 and W_1 (see Section 3). In this work, we shall be interested in deriving smooth and one-to-one functions. The smoothness of a warp makes it possible to obtain a distorted image with no discontinuities or abrupt deformations. The one-to-one property guarantees that the distorted image does not fold back upon itself. Hence, the warp generation problem requires us to find a smooth one-to-one 2D mapping from positional constraints. It may be formulated as follows:

Given a set of point pairs $(P, Q) = \{(p_i, q_i)\}$, *find a smooth one-to-one function* $\mathbf{w} : R^2 \to R^2$ *such that* $\mathbf{w}(p_i) = q_i$ *for each point* p_i *in* P *and the corresponding point* q_i *in* Q.

In this paper, we propose the multilevel free-form deformation (MFFD) to provide a solution to the warp generation problem. The MFFD method contains a new direct manipulation technique for free-form deformation (FFD) [3] and applies it to a hierarchy of control lattices. To guarantee the one-to-one property, we present a sufficient condition for a 2D cubic B-spline surface to be one-to-one.

The MFFD generates C^2-continuous and one-to-one warps which yield fluid image distortions. It is much simpler and faster than the energy minimization method [4]. We also present a hybrid approach that combines the two methods.

5.1 Free-Form Deformation

Free-form deformation (FFD) was proposed by Sederberg and Parry as a powerful modeling tool for 3D deformable objects [3]. The basic idea of FFD is to deform an object by manipulating a 3D parallelepiped lattice containing the object. The manipulated lattice determines a deformation function that specifies a new position for each point on the object. Coquillart extended the FFD method to handle non-parallelepiped lattices [14] and proposed a technique for animating objects modeled by FFD [15]. Hsu et al. employed the FFD method to directly control the shape of an object under complex deformations [16]. They took the trivariate cubic B-spline tensor product as the deformation function instead of the Bernstein polynomials used by Sederberg and Parry.

In this paper, we consider a 2D FFD to deform a rectangular plate in the xy-plane by manipulating a regular lattice overlaid on it. The bivariate cubic B-spline tensor product is taken as the deformation function of FFD because a B-spline has local control [17]. This property makes it possible to locally manipulate the lattice when a point on the plate is moved to the specified position. Therefore, the new lattice producing this movement can be efficiently computed even for a large number of control points.

Let Ω be a rectangular plate placed on the xy-plane. We assume that Ω contains points $p = (u, v)$ where $1 \le u \le m$ and $1 \le v \le n$. When plate Ω is deformed in the xy-plane, its shape can be represented by a vector-valued function, $\mathbf{w}(p) = (x(p), y(p))$. Let Φ be an $(m + 2) \times (n + 2)$ lattice of control points overlaid on plate Ω. In the initial configuration of Φ, the ijth control point lies at its initial position, $\phi_{ij}^0 = (i, j)$. With the FFD method, a desired deformation $\mathbf{w}$ of plate Ω is derived by displacing the control points on lattice Φ from their initial positions (Fig. 8).

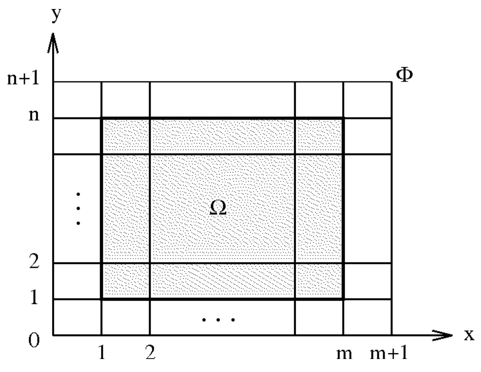


Fig. 8. The initial arrangement of the plate and control lattice.

Let ϕ_{ij} be the position of the ijth control point on lattice Φ. The function $\mathbf{w}$ is defined in terms of ϕ_{ij} by

$$\mathbf{w}(u,v) = \sum_{k=0}^{3}\sum_{l=0}^{3} B_k(s)B_l(t)\phi_{(i+k)(j+l)}, \qquad (1)$$

where $i = \lfloor u \rfloor - 1$, $j = \lfloor v \rfloor - 1$, $s = u - \lfloor u \rfloor$, and $t = v - \lfloor v \rfloor$. $B_k(s)$ and $B_l(t)$ are the uniform cubic B-spline basis functions evaluated at s and t, respectively. They are defined as

$$\begin{aligned} B_0(t) &= \left(-t^3 + 3t^2 - 3t + 1\right)/6 \\ B_1(t) &= \left(3t^3 + 6t^2 + 4\right)/6 \\ B_2(t) &= \left(-3t^3 + 3t^2 + 3t + 1\right)/6 \\ B_3(t) &= t^3/6, \end{aligned}$$

where $0 \le t < 1$.

Since a B-spline curve through collinear control points is itself linear, the initial configuration of lattice Φ generates the undeformed shape of the plate. That is,

$$\mathbf{w}^0(u,v) = (u,v) = \sum_{k=0}^{3}\sum_{l=0}^{3} B_k(s)B_l(t)\phi^0_{(i+k)(j+l)}. \qquad (2)$$

From (1), we know that the deformed position $\mathbf{w}(p)$ of point p on plate Ω relates to the sixteen control points in its neighborhood.

5.2 One-to-One Property of Free-Form Deformation

Function $\mathbf{w}$ defined in (1) can be regarded as a 2D uniform cubic B-spline surface where plate Ω is the parameter space. The one-to-one property of a 2D B-spline surface has not been studied because B-spline surfaces are usually considered in three dimensions to model free-form surfaces.

Recently, Goodman and Unsworth presented a sufficient condition for a 2D Bézier surface to be one-to-one [18]. They commented that the condition can also be applied to a 2D B-spline surface. For an $m \times n$ lattice of control points, the condition contains $2m(m + 1) + 2n(n + 1)$ linear inequalities. If the number of control points is large, the time to check the condition becomes prohibitive. Moreover, if the condition does not hold, there is no simple way for manipulating the control lattice to satisfy the condition.

In this paper, we present a sufficient condition for the function $\mathbf{w}$ to be one-to-one in terms of the displacements of control points. With the following theorem, a 2D uniform cubic B-spline surface can be made one-to-one by limiting the displacements of control points. Let $\Delta\phi_{ij} = \phi_{ij} - \phi_{ij}^0$ be the displacement of the ijth control point from its initial position. Let $|\delta|_\infty = \max(|\delta_1|, |\delta_2|)$, where $\delta = (\delta_1, \delta_2)$.

THEOREM 1. *The function* $\mathbf{w}$ *given in* (1) *is one-to-one if* $\left|\Delta\phi_{ij}\right|_\infty \le 0.48$ *for all i, j.*

PROOF. Let $\Delta\phi_{ij} = \phi_{ij} - \phi_{ij}^0$ and $\mathbf{w} = (x, y)$. Suppose that $\frac{\partial x}{\partial u} > \left|\frac{\partial x}{\partial v}\right|$ and $\frac{\partial y}{\partial v} > \left|\frac{\partial y}{\partial u}\right|$ at each point on the domain Ω when $(-0.48, -0.48) \le \Delta\phi_{ij} \le (0.48, 0.48)$ for all i, j. Then, the Jacobian, $J = \frac{\partial x}{\partial u}\frac{\partial y}{\partial v} - \frac{\partial x}{\partial v}\frac{\partial y}{\partial u}$, is greater than zero at all points in Ω including the boundary, which implies that function $\mathbf{w}$ is one-to-one [19]. Let $\phi_{ij} = (x_{ij}, y_{ij})$, $\phi_{ij}^0 = (x_{ij}^0, y_{ij}^0)$, and $\Delta y_{ij} = y_{ij} - y_{ij}^0$. $\frac{\partial y}{\partial v} > \left|\frac{\partial y}{\partial u}\right|$ if and only if $\frac{\partial y}{\partial v} > \frac{\partial y}{\partial u}$ and $\frac{\partial y}{\partial v} > -\frac{\partial y}{\partial u}$. In what follows, we only show that $\frac{\partial y}{\partial v} > \frac{\partial y}{\partial u}$ if $-0.48 \le \Delta y_{ij} \le 0.48$. A symmetrical argument can be applied to the case when $\frac{\partial y}{\partial v} > -\frac{\partial y}{\partial u}$. Similarly, we can prove the remaining case, $\frac{\partial x}{\partial u} > \left|\frac{\partial x}{\partial v}\right|$.

From (1) and (2), we have

$$\begin{aligned} y &= \sum_{k=0}^{3}\sum_{l=0}^{3} B_k(s)B_l(t)y_{(i+k)(j+l)} \\ &= v + \sum_{k=0}^{3}\sum_{l=0}^{3} B_k(s)B_l(t)\Delta y_{(i+k)(j+l)}, \end{aligned}$$

and hence,

$$\begin{aligned} &\frac{\partial y}{\partial v} - \frac{\partial y}{\partial u} \\ &\quad = 1 + \sum_{k=0}^{3}\sum_{l=0}^{3}\left(B_k(s)B_l'(t) - B_k'(s)B_l(t)\right)\Delta y_{(i+k)(j+l)}. \end{aligned}$$

Let $c_{kl} = B_k(s)B_l'(t) - B_k'(s)B_l(t)$. From the formulae of B-spline basis functions, it holds that $B_k(t) \ge 0$, for $i = 0, 1, 2, 3$, $B_0'(t) \le 0$, $B_1'(t) \le 0$, $B_2'(t) \ge 0$, and $B_3'(t) \ge 0$ when $0 \le t \le 1$. Therefore, it immediately follows that $c_{20}, c_{30}, c_{21}, c_{31} \le 0$ and $c_{02}, c_{12}, c_{03}, c_{13} \ge 0$. If we let $t = s + \Delta t$, then $c_{10} = -(s + \Delta t - 1)^2((s-2)^2 + (4s - 3s^2)\Delta t)/12$. Since $\Delta t \ge -s$ and $(4s - 3s^2) \ge 0$ when $0 \le s \le 1$, we get $(4s - 3s^2)\Delta t \ge -s\,(4s - 3s^2) \ge -1$, which implies $c_{10} \le 0$. Similarly, it can be proved that $c_{32} \le 0$, $c_{01} \ge 0$, and $c_{23} \ge 0$ when $0 \le s, t \le 1$.

From the fact that $c_{00} = (s-t)(s-1)^2(t-1)^2/12$ and $c_{33} = (s-t)s^2t^2/12$, it follows that if $s \ge t$, then $c_{00}, c_{33} \ge 0$ and if $s < t$, then $c_{00}, c_{33} \le 0$. By manipulating the formula of c_{11}, we get $c_{11} = (s-t)(3(st+1)(3st - 4s - 4t + 5) + 1)/12$. Let $f(s, t) = 3st - 4s - 4t + 5$. Then, $\frac{\partial f}{\partial s} = 3t - 4 < 0$ and $\frac{\partial f}{\partial t} = 3s - 4 < 0$ when $0 \le s, t \le 1$, which implies that f has a global minimum at $(1, 1)$. Because $f(1, 1) = 0$, $c_{11} \ge 0$ if $s \ge t$ and $c_{11} < 0$ if $s < t$. Similarly, it can be shown that $c_{22} \ge 0$ if $s \ge t$ and $c_{22} < 0$ if $s < t$.

In summary, $c_{kl} \ge 0$ if $k < l$ and $c_{kl} \le 0$ if $k > l$ when $0 \le s, t \le 1$. Also, if $s \ge t$, then $c_{kk} \ge 0$, and if $s < t$, then $c_{kk} \le 0$. We consider the case when $s \ge t$. Let

$$C = \sum_{k=0}^{3}\sum_{l=0}^{k-1} c_{kl} - \sum_{k=0}^{3}\sum_{l=k}^{3} c_{kl}.$$

C is a function of s and t defined on $0 \le s, t \le 1$. From the condition that $-0.48 \le \Delta y_{ij} \le 0.48$ and the properties of the values of c_{kl}, it holds that

$$\begin{aligned} &\frac{\partial y}{\partial v} - \frac{\partial y}{\partial u} \\ &\quad = 1 + \left(\sum_{k=0}^{3}\sum_{l=0}^{k-1} c_{kl}\Delta y_{(i+k)(j+l)} + \sum_{k=0}^{3}\sum_{l=k}^{3} c_{kl}\Delta y_{(i+k)(j+l)}\right) \\ &\quad \ge 1 + 0.48\left(\sum_{k=0}^{3}\sum_{l=0}^{k-1} c_{kl} - \sum_{k=0}^{3}\sum_{l=k}^{3} c_{kl}\right) \\ &\quad = 1 + 0.48C. \end{aligned}$$

To derive a lower bound of C, we partition the domain $0 \le s, t \le 1$ to the grid in which the internode distance Δd is 0.0001. When C is evaluated at each grid point by 64 bits double-precision arithmetic, the minimum value is -2.0463927 at $(s_0, t_0) = (0.7552, 0.2448)$. Let (s_g, t_g) be a grid point and $D = C(s_g + \Delta s, t_g + \Delta t) - C(s_g, t_g)$, where $0 \le \Delta s, \Delta t < \Delta d$. D consists of terms $s_g^{\alpha} t_g^{\beta} \Delta s^{\gamma} \Delta t^{\delta}$, where $\alpha, \beta, \gamma, \delta = 0, 1, 2, 3$. To simplify the formula of D, we assign $s_g = t_g = 0$ and $s_g = t_g = 1$ to the terms in D having positive and negative coefficients, respectively. Then, from $\Delta d > (\Delta s)^{\gamma}$ and $\Delta d > (\Delta t)^{\delta}$, it holds that $D > -\varepsilon\Delta d$ for $\varepsilon = \frac{4{,}230}{36}$.

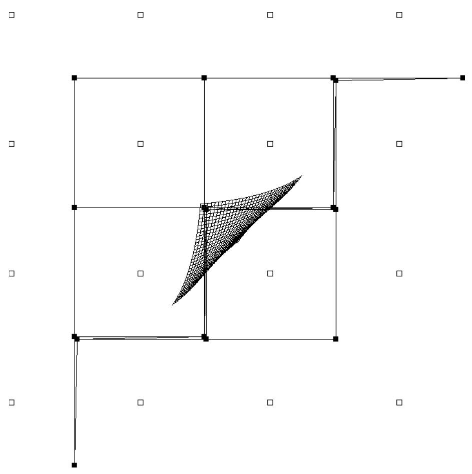
Let (s, t) be a point on the domain $0 \le s, t \le 1$. Let $s_g = \Delta d\lfloor s/\Delta d\rfloor$ and $t_g = \Delta d\lfloor t/\Delta d\rfloor$. Let $\Delta s = s - s_g$ and $\Delta t = t - t_g$. Then, $C(s, t) = C(s_g + \Delta s, t_g + \Delta t) > C(s_g, t_g) - \varepsilon\Delta d \ge C(s_0, t_0) - \varepsilon\Delta d \ge -2.0581427$. Hence, $1 + 0.48\,C > 0$ on the domain $0 \le s, t \le 1$, which implies that $\frac{\partial y}{\partial v} > \frac{\partial y}{\partial u}$. The case when $s < t$ can be treated similarly. □

Theorem 1 provides a tight, sufficient, although not necessary, condition. From the proof of the theorem, an example of a B-spline surface that violates the one-to-one property can be constructed when all control points are displaced by amounts less than 0.5.

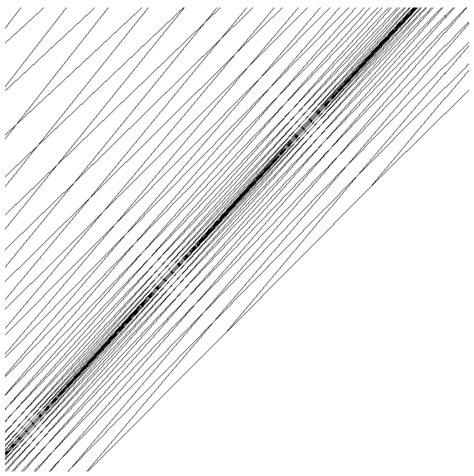
Consider a single B-spline patch given by a 4×4 control lattice. Let $\Delta\phi_{kl} = (0.49, -0.49)$ if $k \le l$ and $\Delta\phi_{kl} = (-0.49, 0.49)$ if $k > l$ for $k, l = 0, 1, 2, 3$. Then, the Jacobian of the function $\mathbf{w}$ is less than zero at the point where $u = 0.75519$ and $v = 0.24483$. Since the Jacobian values are positive at other points, this implies that $\mathbf{w}$ contains a foldover and so is not one-to-one. Fig. 9 shows the single B-spline patch and a magnification of the region near the foldover. In the figure, black and white rectangles denote the displaced and original positions, respectively, of control points. This example shows that a B-spline surface may violate the one-to-one property even when control lattice gridlines do not self-intersect.

5.3 Manipulation of Free-Form Deformation

Suppose that plate Ω should be deformed to place a point p at the specified position q, that is, $\mathbf{w}(p) = q$. Without loss of generality, we may assume that $p = (u, v)$, $1 \le u, v < 2$. Then, the displacements of the klth control points, $k, l = 0, 1, 2, 3$, on lattice Φ determine the deformed position $\mathbf{w}(p)$ of point p. See Fig. 10a. Let $\Delta q = \mathbf{w}(p) - \mathbf{w}^0(p) = q - p$ be the movement of the point p from its original position.



(a) Surface patch



(b) Magnified foldover region

Fig. 9. A violation of the one-to-one property on a single B-spine surface patch.

From (1) and (2), the displacements $\Delta\phi_{kl}$ must satisfy (3):

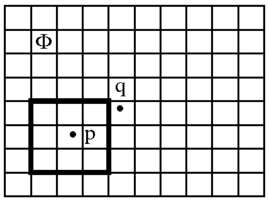
$$\Delta q = \sum_{k=0}^{3}\sum_{l=0}^{3} w_{kl}\Delta\phi_{kl}, \tag{3}$$

where $w_{kl} = B_k(s)B_l(t)$ and $s = u - 1$, $t = v - 1$.

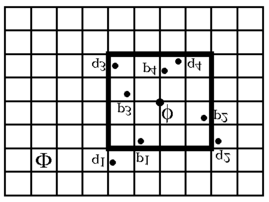
There are many values of $\Delta\phi_{kl}$ that are solutions to (3). Since B-spline basis functions have the property that $\sum_{k=0}^{3}\sum_{l=0}^{3} w_{kl} = 1$, one obvious solution is $\Delta\phi_{kl} = \Delta q$, for k, l = 0, 1, 2, 3. However, this does not properly reflect the movement of its neighboring point p. The resulting deformation **w** is the same regardless of the undeformed position of p for $1 \le u, v < 2$. Another solution is to choose the control point nearest to p and displace it until p reaches the desired position q. This solution also generates an improper deformation **w** because only the parts near the displaced control point are deformed.

Hence, we choose the solution in the least-squared sense such that

$$\Delta\phi_{kl} = \frac{w_{kl}\Delta q}{\sum_{a=0}^{3}\sum_{b=0}^{3} w_{ab}^2}. \tag{4}$$



(a) single constraint



(b) multiple constraints

Fig. 10. Examples of the positional constraints.

Among all the solutions to (3), Hsu et al. showed that this minimizes the squared sum of control point displacements [16]. In this solution, the control points near point p gets larger displacements than the others because w_{kl} depends on the distance between the klth control point and point p. It generates the deformation **w** whereby the effect of the movement of p tapers off smoothly.

Now, suppose that plate Ω should be deformed to place a set of points P at a set of positions Q. That is, $\mathbf{w}(p) = q$ for each point p in P and its position q in Q. A point p in P can be moved to the specified position q if its surrounding control points are displaced by the amount $\Delta\phi_{kl}$ given in (4). However, these displacements may mislead another point in P to another position than the one specified in Q.

Let $P' = \{(u_c, v_c)\}$ be the set of points in P such that $i - 2 \le u_c < i + 2$ and $j - 2 \le v_c < j + 2$. Let ϕ be the ijth control point of lattice Φ whose initial position is (i, j), as in Fig. 10b. The displacements of control point ϕ influences the movement of points in P' when we evaluate the deformation function **w**. For each point p_c in P', (4) gives the displacement $\Delta\phi_c$ of control point ϕ required for moving p_c to the specified position. Since the displacement $\Delta\phi_c$ may be different from point to point in P', the displacement $\Delta\phi$ of control point ϕ is chosen to minimize an error.

The error is defined as the squared sum of differences between $w_c\Delta\phi_c$ and $w_c\Delta\phi_c$, where $w_c = B_k(s)B_l(t)$, $k = (i + 1) - \lfloor u_c \rfloor$, $l = (j+1) = \lfloor v_c \rfloor$, $s = u_c - \lfloor u_c \rfloor$, and $t = v_c - \lfloor v_c \rfloor$, for each point $p_c = (u_c, v_c)$ in P'. That is, the error is

$$\sum_c (w_c\Delta\phi - w_c\Delta\phi_c)^2.$$

$w_c\Delta\phi$ is the movement of point p_c due to the displacement $\Delta\phi$ of control point ϕ. $w_c\Delta\phi_c$ represents the contribution of control point ϕ, determined by (4), for moving p_c to its specified position. By differentiating the error with respect to $\Delta\phi$ and equating the derived formula to zero, we get

$$\Delta\phi = \frac{\sum_c w_c^2\Delta\phi_c}{\sum_c w_c^2}. \tag{5}$$

If the set P' is empty, the control point ϕ does not play a role in making the positional constraints hold. The displacement $\Delta\phi$ can have an arbitrary value without interrupting movements of points in P to the positions in Q. We define $\Delta\phi$ to be zero in that case to keep the deformation of plate Ω as small as possible.

When the necessary control point displacements have been computed by (5), the resulting deformation function $\mathbf{w}$ is not guaranteed to be one-to-one. To make $\mathbf{w}$ one-to-one, we truncate the displacement $\Delta\phi$ of a control point ϕ so that $|\Delta\phi|_\infty \le 0.48$. Then, the condition of Theorem 1 holds and the derived function $\mathbf{w}$ is one-to-one. The following algorithm summaries the scheme to manipulate control lattice Φ from point sets P and Q.

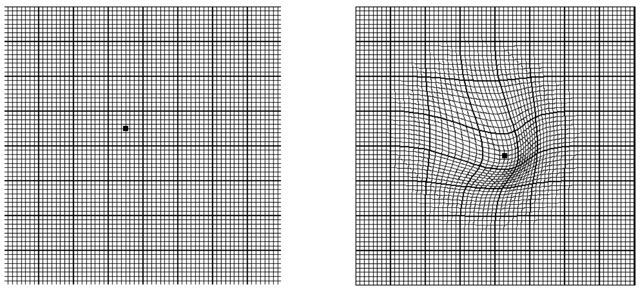
```
Algorithm FFD Manipulation
Input: point sets $P$ and $Q$
Output: control point displacements $\Delta\Phi = \{\Delta\phi_{ij}\}$
for all $i, j$ do let $\delta_{ij} = 0$ and $\omega_{ij} = 0$
for each point $p = (u, v)$ in $P$ do
    let $i = \lfloor u \rfloor - 1$ and $j = \lfloor v \rfloor - 1$
    let $s = u - \lfloor u \rfloor$ and $t = v - \lfloor v \rfloor$
    for $k, l = 0, 1, 2, 3$ do
        compute $\Delta\phi_{kl}$ with (4)
        add $w_{kl}^2 \, \Delta\phi_{kl}$ to $\delta_{(i+k)(j+l)}$
        add $w_{kl}^2$ to $\omega_{(i+k)(j+l)}$
    end
end
for all $i, j$ do
    if $\omega_{ij} \neq 0$ then do
        compute $\Delta\phi_{ij} = \delta_{ij} / \omega_{ij}$
        truncate $\Delta\phi_{ij}$ so that $|\Delta\phi_{ij}|_\infty \le 0.48$
    else let $\Delta\phi_{ij} = 0$
end
```

Hsu et al. presented a technique for manipulating control points so that the points on an object modeled by FFD may be moved to the specified positions [16]. That technique calculates the pseudoinverse of a matrix to derive the displacements of control points that minimize the squared sum of distances between the specified and actually moved positions. The matrix contains the values of B-spline basis functions and its size depends on the number of positional constraints. When a large number of points must be moved, the computation for calculating the pseudoinverse is prohibitive.

On the other hand, the technique proposed in this section runs very fast even when the number of moved points is large. The deformation of plate Ω nicely reflects the movements of points because the displacement of each control point minimizes a reasonable error. Fig. 11 shows examples. In the figures, black spots represent the positions of the selected points in the undeformed and deformed shapes. Thick curves show the control lattice Φ overlaid on plate Ω. The control lattice is a rectangular grid in its initial configuration. It is transformed when plate Ω is deformed.

5.4 Multilevel Free-Form Deformation

Let P be a set of points on plate Ω and Q be a set of corresponding positions. An application of the FFD manipulation presented in Section 5.3 cannot always deform plate Ω to place each point in P at its specified position in Q. One reason is that the displacement of a control point on lattice Φ is the weighted average of the displacements required for moving its neighboring points in P. The other reason is that we limit the maximum displacement of a control point to approximately a half of the spacing between control points in order to make the deformation function one-to-one.


(a) single positional constraint

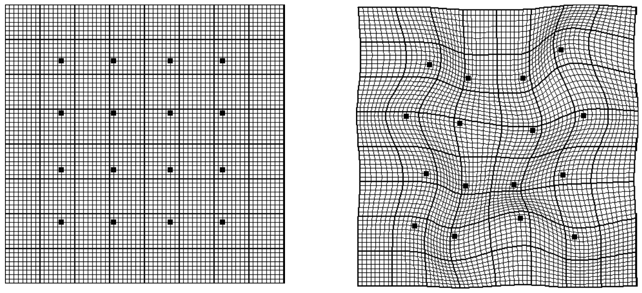

(b) multiple positional constraints

Fig. 11. Examples of the FFD manipulation.

We may circumvent the first problem if we make the control lattice finer until every point in P can be moved by its surrounding control points without interfering with other points in P. The second can be overcome if we repeatedly apply the FFD manipulation to plate Ω so that the accumulated movement of a point in P can be sufficiently large. Hence, an obvious method for deriving a one-to-one deformation function from the positional constraints is to overlay a sufficiently fine control lattice over plate Ω and iterate the FFD manipulation. However, in this case, the resulting shape of plate Ω will show only sharp local deformations near the points in P. Moreover, a large number of FFD manipulations may be required to satisfy the positional constraints because a point in P can only move a short distance by the FFD manipulation when a fine control lattice is used. In this section, we present the multilevel free-form deformation (MFFD) technique that overcomes the drawbacks of the straightforward method.

In MFFD, a hierarchy of control lattices, $\Phi_0, \Phi_1, ..., \Phi_m$, is used to derive a sequence of deformation functions with the FFD manipulation. Let h_k be the spacing between control points on the initial configuration of lattice Φ_k. We assume that h_0 and h_m are given and that $h_k = 2h_{k+1}$. When plate Ω is deformed with a coarse control lattice, the positional constraints merge with each other and result in a smooth deformation, although they are not exactly satisfied. The remaining deviations between the deformed and specified positions will be handled by subsequent deformations with finer control lattices.

Let $\mathbf{w}_0, \mathbf{w}_1, ..., \mathbf{w}_n$ be the sequence of deformation functions derived in the MFFD. Then, the deformation of plate Ω is defined by the composite function $\mathbf{w} = \mathbf{w}_n \circ \mathbf{w}_{n-1} \circ ... \circ \mathbf{w}_0$.

That is, $\mathbf{w}(\Omega) = \mathbf{w}_n(\Omega_n)$, where $\Omega_0 = \Omega$ and $\Omega_{i+1} = \mathbf{w}_i(\Omega_i)$. $\mathbf{w}(\Omega)$ and $\mathbf{w}_i(\Omega_i)$ denote the resulting shapes when the deformation functions $\mathbf{w}$ and $\mathbf{w}_i$ are applied to the plate Ω and deformed plate Ω_i, respectively. Let $P_{i+1} = \mathbf{w}_i(P_i)$, where $P_0 = P$. P_i is the set of points on the deformed plate Ω_i that lie at the deformed positions of the points in P. The deformation function $\mathbf{w}_i$ is computed to move the points in P_i to their specified positions in Q. When the deformation function $\mathbf{w}$ is applied to plate Ω, we define the error as

$$err(\mathbf{w}) = \max_c \left\| \mathbf{w}(p_c) - q_c \right\|^2,$$

where q_c is the position in Q specified for point p_c in P.

When we deform a plate Ω_i with a control lattice Φ_k, a point in P_i can move at most $(0.48h_k, 0.48h_k)$ if and only if all 16 surrounding control points are displaced by $(0.48h_k, 0.48h_k)$. Note that this maximum movement follows from Theorem 1 whereby the displacements of control points must be truncated to keep the one-to-one property of the deformation function. If each point in P_i moves by $(0.48h_k, 0.48h_k)$, the error decreases by at least $(0.48h_k)^2$. In this case, more FFD manipulations with the control lattice Φ_k may be helpful for moving the points in P_i to their specified positions.

In MFFD, the FFD manipulation starts with the coarsest control lattice Φ_0. With a control lattice Φ_k, the FFD manipulation iterates until the change in error falls below $\alpha(0.48h_k)^2$. Then, the next finer control lattice Φ_{k+1} is used for the successive FFD manipulation, as long as Φ_k is not the finest control lattice. This process continues while the error exceeds a user-specified threshold. The parameter α is a real value between zero and one. A small α generates a smooth deformation of plate Ω because FFD manipulations tends to be performed on coarser control lattices. We usually use 0.5 as the value of α.

The following pseudocode outlines the MFFD algorithm. In this algorithm, FFD manipulation (as outlined in the previous algorithm) is successively applied with a hierarchy of control lattices to make the points in P gradually approach their positions in Q. The deformation function $\mathbf{w}$ from the MFFD algorithm is the composition of several functions derived by FFD manipulations. Function $\mathbf{w}$ is C^2-continuous and one-to-one because FFD manipulation generates a C^2-continuous and one-to-one function.

```
**Algorithm MFFD**
Input: point sets $P = \{p_i\}$ and $Q = \{q_i\}$
Output: 2D function $\mathbf{w}$ such that $\mathbf{w}(p_i) = q_i$
let $\mathbf{w}$ be the identity function
let $\Phi$ be the coarsest control lattice (e.g., $5 \times 5$)
let $h$ be the initial control point spacing in $\Phi$
compute $error = err(\mathbf{w})$
let $oldError = error$
while $error > theshold$ do
    compute $\Delta\Phi$ from $P$ and $Q$ by FFD Manipulation
    evaluate deformation function $ffd$ from $\Phi' = \Phi + \Delta\Phi$
    update function $\mathbf{w} = ffd \circ \mathbf{w} = ffd(\mathbf{w})$
    update point set $P = ffd(P)$
    compute $error = err(\mathbf{w})$
    let $\Delta error = error - oldError$
    if $\Delta error < \alpha(0.48h)^2$ and $\Phi$ is not finest lattice
    then do
        let $\Phi$ be the next finer control lattice
        let $h$ be the initial control point spacing in $\Phi$
    end
    let $oldError = error$
end
```

Fig. 12 gives an example in which the MFFD is applied to generate a deformation from positional constraints. Fig. 12a shows the selected points in the undeformed shape. Figs. 12b through 12e show a sequence of deformations from successive FFD manipulations. In this example, the FFD manipulations are performed no more than twice at each level of the control lattice. The final deformation is given in Fig. 12f with the specified positions of selected points. When the function $\mathbf{w}$ is evaluated on a 64×64 grid, it takes 0.2 seconds for an SGI Crimson to generate that final deformation. When the size of the grid is 512×512, the computation time is 9.2 seconds.

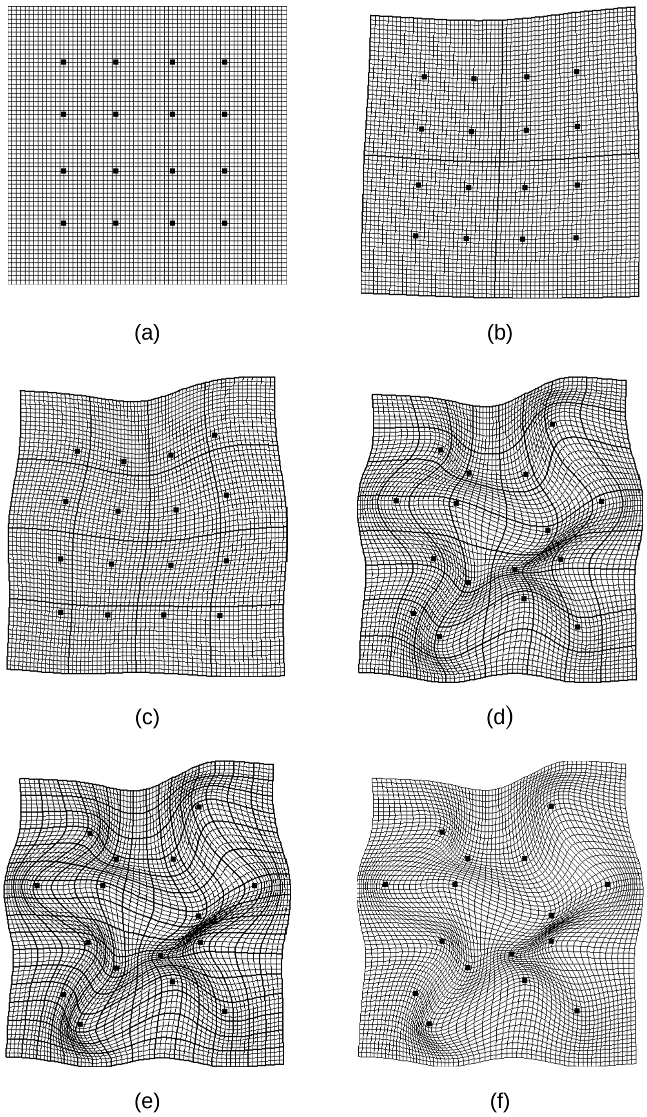


Fig. 12. A sequence of deformations produced by the MFFD.

5.5 A Hybrid Approach

An energy minimization method has been proposed to derive C^1-continuous and one-to-one warps from positional constraints specified by point pairs [4]. It generates a natural warp by numerically solving a differential equation to minimize the sum of physically meaningful energy terms. The computational cost increases drastically with the size of the grid on which warps are evaluated. The MFFD can be combined with that method to obtain nice warps more efficiently when the grid size is large.

Suppose that warps are to be evaluated on an $m \times n$ grid Ω. Let Ω' be a coarse grid of size $m' \times n'$. The positional constraints on grid Ω can be approximated onto grid Ω' by weighted averaging. We then use the energy minimization method to obtain a warp $\mathbf{w}_e$ on grid Ω' from the resulting constraints. $\mathbf{w}_e$ gives a C^1-continuous and one-to-one deformation of grid Ω'.

To propagate $\mathbf{w}_e$ onto the original grid Ω, we construct a 2D uniform cubic B-spline surface with a $(m' + 2) \times (n' + 2)$ control lattice. The control points are handled to make the surface interpolate the value of function $\mathbf{w}_e$ at each point of grid Ω'. A warp $\mathbf{w}_0$ is then derived by evaluating the surface on the original grid Ω. $\mathbf{w}_0$ provides a C^2-continuous and one-to-one deformation of grid Ω. However, it does not exactly satisfy the given positional constraints specified on Ω.

To resolve the error, we use the MFFD method with $\mathbf{w}_0$ as its first deformation function on Ω. The size of the coarsest control lattice is then chosen as $(m' + 2) \times (n' + 2)$. Any control lattice coarser than that need not be considered because a smooth deformation of Ω has been a1ready made by $\mathbf{w}_0$. The MFFD finally generates a C^2-continuous and one-to-one deformation of Ω which exactly satisfy the given positional constraints.

In the hybrid approach, the global shape of a warp is determined by the energy minimization method. It can be computed quickly by running the method on a coarse grid. The MFFD on the original grid derives local deformations near the selected points to move them exactly to the specified positions. Then, it is possible to avoid the excessive computation required for energy minimization on a fine grid. Hence, the hybrid approach generates a nice warp similar to the energy minimization method in a computation time comparable to the MFFD.

Fig. 13 gives an example. Fig. 13a shows the selected points in the undeformed shape of the plate. Figs. 13b, 13c, and 13d show the deformations of the plate derived by energy minimization, MFFD, and the hybrid method, respectively. In the figures, warps are evaluated on a 512 × 512 grid. For the hybrid approach, the energy minimization method is applied to a 128 × 128 grid. The computation times for Figs. 13b, 13c, and 13d on an SGI Crimson are 26.7, 6.4, and 7.5 seconds, respectively.

5.6 Discussion

Positional constraints from sets P and Q specify the linear movement of points to their final positions. In the MFFD method, the deformation reflects the virtual forces due to these movements. In general, if positional constraints permit the paths to self-intersect, the resulting warp may violate the one-to-one property and give rise to foldovers.

The MFFD guarantees the one-to-one property of a warp even when positional constraints are prone to foldovers. This is achieved by simply relaxing the requirement to exactly satisfy the positional constraints. Fig. 14 shows an example of the resulting deformation when two points swap their positions.

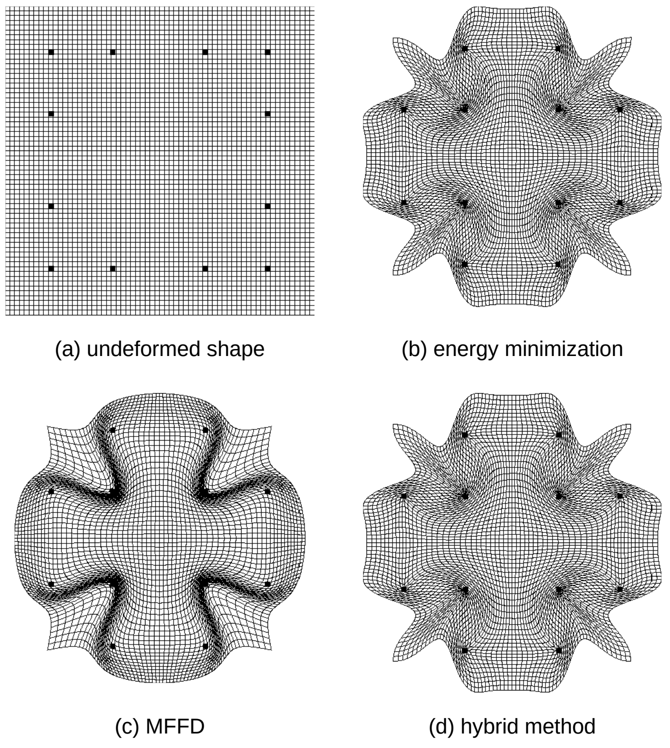


Fig. 13. Comparison of the deformed shapes of a plate.

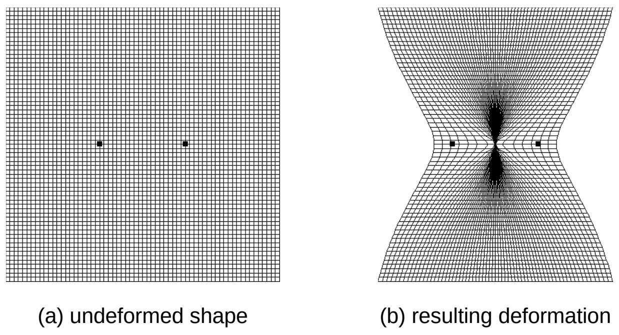


Fig. 14. A deformation from foldover-prone positional constraints.

Warp $\mathbf{w}$ from the MFFD method is the composition of a sequence of deformation functions derived by FFD. Algorithm MFFD iteratively updates warp $\mathbf{w}$ by composing it with a new deformation function *ffd*. When warp $\mathbf{w}$ is evaluated on a large grid, it takes time to obtain a new value of $\mathbf{w}$ at each grid point, which is transformed by function *ffd*. This repeated step dominates the computation time of Algorithm MFFD.

To accelerate this step, we evaluate function *ffd* on a regular grid and use bilinear interpolation to compute the function composition $ffd \circ \mathbf{w}$. Values of function *ffd* on the grid can be derived efficiently by the forward difference method [5], [17] and a lookup table. The forward difference method is used for a coarse control lattice where the function values can be obtained by simple addition at a large number of grid points. When the control lattice gets finer, the forward difference method slows down because a setup is required for each control patch. Then, a table is used to compute B-spline basis functions in evaluating function *ffd*. In that case, function *ffd* is initialized to an identity function and the function values are updated only for grid points near control points with nonzero displacements.

6 Transition Control

Transition behavior in a metamorphosis sequence can be controlled by a set of transition curves defined along primitives [4]. These primitives may differ from those used to specify features. Again, there are no restrictions on their placement. The transition rate for each pixel is computed by propagating the transition control information specified at sparse positions. This process is analogous to that of warp generation.

Let P be a set of points on the source image I_0 for which transition curves are assigned. The curves specify a transition rate to each point in P over time t. For a given time t, transition function $T_0(t)$ is a real-valued function defined on I_0. At each point p in P, $T_0(t)$ must have the transition rate specified by the assigned curve. $T_0(t)$ is also required to be smooth so that an inbetween image does not contain an abrupt change of transition rates. Transition function $T_1(t)$ can be derived from $T_0(t)$ using warp W_1, as described in Section 3. Hence, the transition control problem can be formulated as follows

Given points $\{(u_c, v_c)\}$ *on a plane and real values* $\{t_c\}$, *find a smooth function* $f : R^2 \rightarrow R$ *such that* $f(u_c, v_c) = t_c$.

This problem reduces to surface interpolation, where function f is a surface that interpolates scattered constraints, each denoted by a surface height value. In a previous method [4], the thin plate surface model was employed to construct C^1-continuous surfaces through given points. To solve the same problem, we simplify the MFFD to obtain multilevel B-spline interpolation. It generates a C^2-continuous surface more efficiently than the previous method.

6.1 Manipulation of a B-Spline Surface

B-spline surfaces are widely used to model free-form surfaces because they offer nice properties such as continuity and local control. In this section, we consider uniform cubic B-splines to generate a surface that interpolates a scattered set of 3D height field points.

Let Ω be a rectangular region in the *uv*-plane which contains points $p = (u, v)$ such that $1 \le u \le m$ and $1 \le v \le n$. Let Φ be an $(m + 2) \times (n + 2)$ lattice of control points overlaid on the region Ω. In the initial configuration of Φ, the *ij*th control point lies at its initial position (i, j) in the *uv*-plane. When the control points on lattice Φ are displaced only in the direction perpendicular to the *uv*-plane, the resulting B-spline surface can be represented by a real-valued function f. The function value $f(p)$ for a point $p = (u, v)$ on Ω implies that the point p is placed at the position $(u, v, f(p))$ when the surface is generated.

Let ϕ_{ij} be the height of the *ij*th control point from the *uv*-plane. Then, the function f is given by

$$f(u,v) = \sum_{k=0}^{3}\sum_{l=0}^{3} B_k(s)B_l(t)\phi_{(i+k)(j+l)},$$

where $i = \lfloor u \rfloor - 1, j = \lfloor v \rfloor - 1, s = u - \lfloor u \rfloor$, and $t = v - \lfloor v \rfloor$. $B_k(s)$ and $B_l(t)$ are the uniform cubic B-spline basis functions evaluated at s and t, respectively. The above formula for f is in the same form as (1) for the deformation function $\mathbf{w}$ in Section 5.1.

Suppose that a B-spline surface is required to interpolate a set of scattered points (u_c, v_c, t_c), where (u_c, v_c) is a point in the region Ω. That is, $f(u_c, v_c) = t_c$ for each point in the set. A surface that approximately satisfies the positional constraints can be obtained by following the same approach for FFD manipulation described in Section 5.3. The required heights of control points from the *uv*-plane are derived by (4) and (5), replacing Δq with t_c. The computed heights of control points are not truncated in this case because it is not necessary to consider the one-to-one property.

6.2 Multilevel B-Spline Interpolation

Let P be a set of points (u_c, v_c, t_c) in 3D space, where (u_c, v_c) is a point in the region Ω. As in the case for a warp, the B-spline surface derived by (4) and (5) does not necessarily interpolate the points in P. A straightforward solution is to use a sufficiently fine control lattice so that every point in P can be interpolated without interfering with other points. However, the resulting surface will show only sharp local deformations near the points in P. Thus, we introduce multilevel B-spline interpolation to overcome this drawback.

In multilevel B-spline interpolation, a hierarchy of control lattices, $\Phi_0, \Phi_1, \ldots, \Phi_m$, is overlaid over the region Ω to derive a sequence of functions, $f_0, f_1, \ldots, f_m$. Let h_i be the spacing between control points in the initial configuration of lattice Φ_i. We assume that h_0 and h_m are given and that $h_i = 2h_{i+1}$. The final function f is defined by the sum of the functions f_i, that is, $f(p) = \sum_i f_i(p)$, for each point p on Ω.

The manipulation of a B-spline surface starts with the coarsest control lattice Φ_0. The heights of control points on lattice Φ_0 are derived to generate the surface f_0 that interpolates the points in P. Sometimes, however, surface f_0 only passes near the points in P, leaving the deviation $\Delta^0 t_c = t_c - f_0(u_c, v_c)$ for each point (u_c, v_c, t_c) in P. Then, the next finer control lattice Φ_1 is used to obtain the B-spline surface f_1 that interpolates the set of points $(u_c, v_c, \Delta^0 t_c)$.

In general, we manipulate the control points on lattice Φ_{k+1} to derive the B-spline surface f_{k+1} that interpolates the set of points $(u_c, v_c, \Delta^k t_c)$, where $\Delta^k t_c = t_c - \sum_{i=0}^{k} f_i(u_c, v_c)$. This process continues to the finest control lattice Φ_m until $err(f)$ falls below a given threshold, where $err(f)$ is the maximum difference between the points in P and the final surface f. Unlike the MFFD for warp generation, we do not enforce the one-to-one property. Consequently, the B-spline surface manipulation is applied only once on each control lattice because the heights of the control points are not truncated.

The following pseudocode outlines the multilevel B-spline interpolation algorithm. It resembles the MFFD algorithm except for the following key differences. First, the final surface *f* is not due to function composition but rather it is derived from the sum of a sequence of surfaces. Second, the computed heights of the control lattice are not truncated in the B-spline manipulation. Finally, the B-spline manipulation is applied only once on each control lattice. A surface generated by the algorithm is C^2-continuous because it is the sum of C^2-continuous B-spline surfaces. The input to the algorithm consists of point set $P = \{(u_c, v_c)\}$ and value set $Q = \{t_c\}$. The output is a real-valued, C^2-continuous, interpolation function *f* such that $f(u_c, v_c) = t_c$.

```
Algorithm Multilevel B-Spline Interpolation
Input: point set $P$ and value set $Q$
Output: real-valued interpolation function $f$
let $f = 0$ at all points
let $\Phi$ be the coarsest control lattice (e.g., $5 \times 5$)
compute $error = err(f)$
while $error > theshold$ do
    compute the values for $\Phi$ from sets $P$ and $Q$
    evaluate the B-spline surface $bss$ from $\Phi$
    update function $f = f + bss$
    update point set $P = P - bss(P)$
    compute $error = err(f)$
    if $\Phi$ is not the finest control lattice then
        let $\Phi$ be the next finer control lattice
    else exit
end
```

Fig. 15 shows an example. Black spots in the figure represent the interpolated points. Most of the computation time is spent evaluating function *f* in the region Ω. For a 64 $\times$ 64 grid, it takes 0.1 seconds for an SGI Crimson to generate the surface given in Fig. 15. For a 512 $\times$ 512 grid, the surface is generated in 1.5 seconds. As in the MFFD method, we used the forward difference method and a lookup table to efficiently evaluate a B-spline surface.

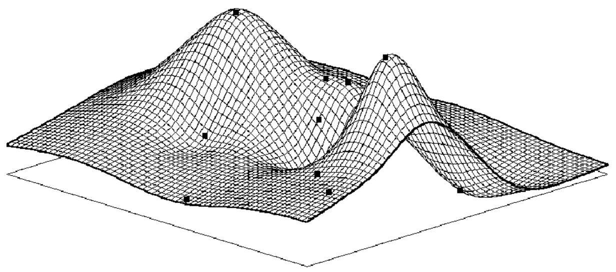


Fig. 15. An example of multilevel B-spline interpolation.

6.3 Discussion

In multilevel B-spline interpolation, the coarsest control lattice Φ_0 determines the area of the resulting surface on which an interpolated point has influence. When Φ_0 is coarse, large spacing between control points merges the effects of interpolated points and generates smooth hills. On the other hand, with a fine Φ_0, the surface tends to contain local bumps near interpolated points. Fig. 16 shows the surface from the same constraints as Fig. 15 when Φ_0 is 19 $\times$ 19. In Fig. 15, Φ_0 is 5 $\times$ 5. The density of the finest control lattice Φ_m controls the precision to which the constructed surface interpolates the given points.

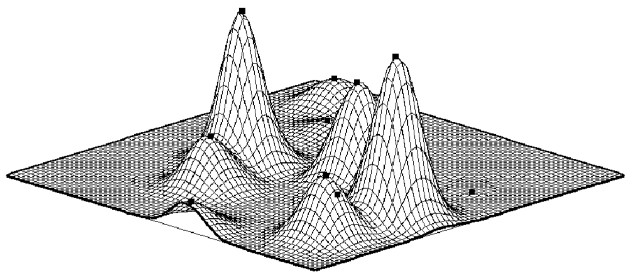
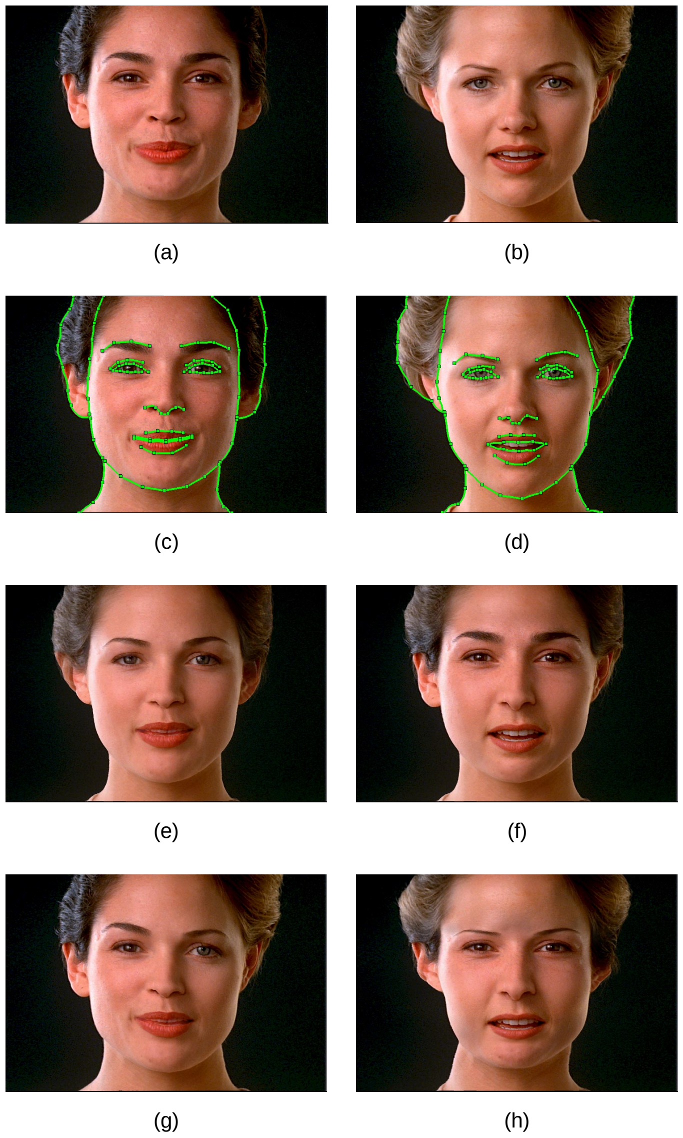


Fig. 16. A surface when the coarsest lattice is fine.



(a) (b) (c) (d) (e) (f) (g) (h)

Fig. 17. Metamorphosis examples.

7 Metamorphosis Examples

Fig. 17 gives metamorphosis examples. Figs. 17a and 17b show source and target images, Melissa and Tiffany, respectively. Figs. 17c and 17d show the specified features overlaid on the input images. In feature specification, snakes are used to capture the precise positions of features around the eyes, mouth, and profiles. Fig. 17e is the inbetween image at $t = 0.5$ in which the same transition rate is applied over all pixels. The image demonstrates that the features are accurately interpolated. For example, the mouth and eyes of the inbetween image are blended smoothly. A drawback with applying the same transition function across the image is evident in the dark region on the ear. The problem is due to the uniform blending between corresponding ear and hair pixels. Nonuniform transition functions are necessary to correctly blend regions to get the desired effect.

To relate a transition behavior in a morph sequence, we assign transition curves at primitives specified on the source image. Fig. 19a shows those primitives used to obtain the transition effects shown in Fig. 17f. Each primitive

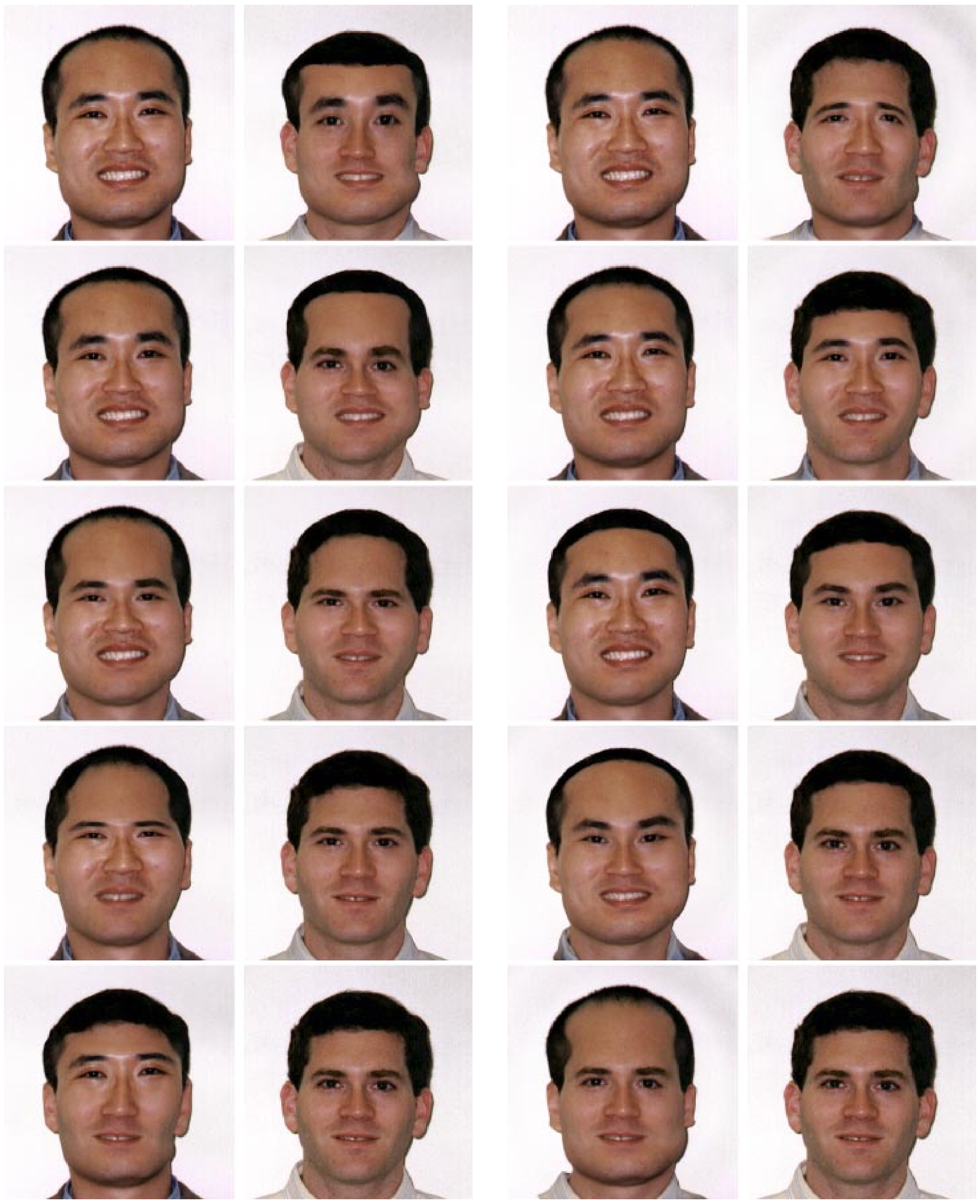

(a) (b)

Fig. 18. Procedural transition control. A sinusoidal function varies the transformation along the (a) vertical and (b) radial directions.

may have a different transition curve, with all points along a primitive sharing the same transition rate. Figs. 19b and 19c depict the transition curves for the left and right primitives of Fig. 19a, respectively. Fig. 17f is the resulting inbetween image at $t = 0.5$. The eyes and left ear are taken from Melissa, while the mouth and the right ear are from Tiffany.

In Figs. 17g and 17h, the transition rate of a pixel is determined procedurally, i.e., based on its position. Fig. 17g shows an inbetween image in which the transition function varied linearly across the scanlines to change Melissa to Tiffany from left to right. A sinusoid transition function along the columns was used to generate the image in Fig. 17h. Note that the differences between Figs. 17f and 17h are prominent near the eyes and eyebrows.

Fig. 20a illustrates the warp function generated for transforming Melissa to Tiffany in Fig. 17. Dark lines represent the new positions of feature points that have been internally sampled on the source image. Fig. 20b shows the surface interpolating through the transition rates that are evaluated from the transition curves along the primitives in Fig. 19.

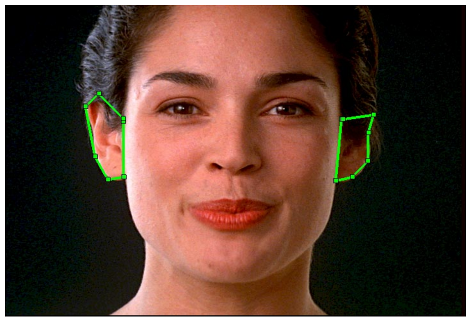
Fig. 18 demonstrates the use of a sinusoidal transition control function to vary the transformation between the source and target images in the upper left and lower right corners of the figure. In Fig. 18a, the sinusoid moves vertically along the image while its amplitude and base offset vary over time. The source (target) image dominates the output at pixels that correspond to sinusoidal valleys (peaks) in the transition function. In Fig. 18b, the sinusoid moves along concentric circles emanating from the image center.

All images shown in Fig. 17 are 720×486 and were generated on a SUN SPARCsystem 10. We used the hybrid method to derive the warp functions and multilevel B-spline interpolation to compute the surfaces for transition control. It took 22.0 and 1.0 seconds, respectively, to generate the warp function and surface in Fig. 20.

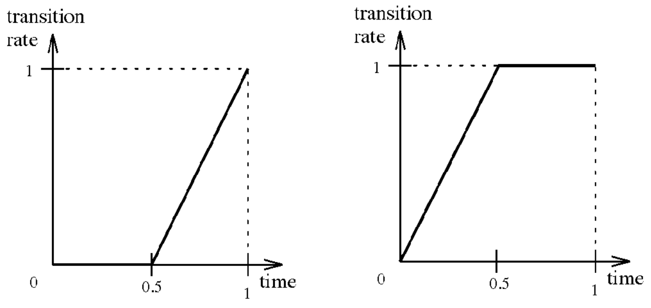
8 Conclusions

This paper has presented solutions to the following three problems in image morphing: feature specification, warp generation, and surface generation for transition control. The features in an image can be specified with snakes [2], a popular computer vision technique. Snakes help an animator to easily and precisely capture the exact position of a feature. They also may reduce the work of an animator in establishing the feature correspondence between two image sequences. We presented a detailed description of MFFD, a new deformation technique, which derives C^2-continuous and one-to-one warps from feature point pairs. The technique is fast, even when the number of features is large. The resulting warps provide visually pleasing image distortions. The issue of foldover prevention was discussed and implementation details were furnished. We also presented multilevel B-spline interpolation to construct smooth surfaces that are used to control geometry and color blending. The method efficiently generates a C^2-continuous transition control surface that interpolates a set of scattered points.

The warp and surface generation techniques in this paper may be applied to other areas of computer graphics. The MFFD can be readily extended to 3D and used to directly manipulate the shape of deformable objects. Multilevel B-spline interpolation can be used to rapidly generate free-form surfaces from positional constraints.

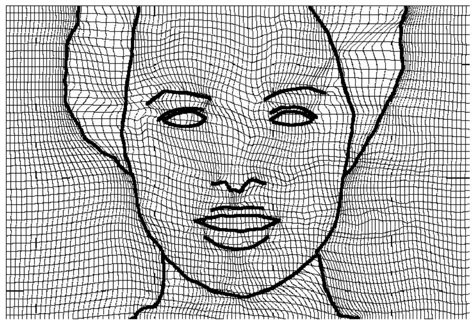


(a)

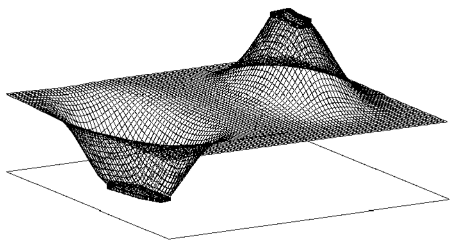


(b) (c)

Fig. 19. Primitives with transition curves.



(a)



(b)

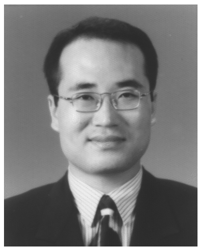
Fig. 20. Warp function and surface.

ACKNOWLEDGMENTS

Dr. Seungyong Lee performed work on this paper while he was a visiting scholar in the Department of Computer Science at the City College of New York. This work was supported in part by the Korean Ministry of Science and Technology (contract 94-S-05-A-03 of STEP 2000), U.S. National Science Foundation Presidential Young Investigator award IRI-9157260, and PSC-CUNY grant RF-666338.

REFERENCES

[1] S. Lee, K.-Y. Chwa, S.Y. Shin, and G. Wolberg, "Image Metamorphosis Using Snakes and Free-Form Deformations," *Computer Graphics (Proc. SIGGRAPH '95)*, pp. 439-448, 1995.
[2] M. Kass, A. Witkin, and D. Terzopoulos, "Snakes: Active Contour Models," *Int'l J. Computer Vision*, pp. 321-331, 1988.
[3] T.W. Sederberg and S.R. Parry, "Free-Form Deformation of Solid Geometric Models," *Computer Graphics (Proc. SIGGRAPH '86)*, vol. 20, no. 4, pp. 151-160, 1986.
[4] 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.
[5] G. Wolberg, *Digital Image Warping*. Los Alamitos, Calif.: IEEE CS Press, 1990.
[6] K.M. Fant, "A Nonaliasing, Real-Time Spatial Transform Technique," *IEEE Computer Graphics and Applications*, vol. 6, pp. 71-80, Jan. 1986.
[7] T. Beier and S. Neely, "Feature-Based Image Metamorphosis," *Computer Graphics (Proc. SIGGRAPH '92)*, vol. 26, no. 2, pp. 35-42, 1992.
[8] D. Ruprecht and H. Müller, "Image Warping with Scattered Data Interpolation," *IEEE Computer Graphics and Applications*, vol. 15, pp. 37-43, Mar. 1995.
[9] S. Lee, K.-Y. Chwa, J. Hahn, and S.Y. Shin, "Image Morphing Using Deformable Surfaces," *Proc. Computer Animation '94*, pp. 31-39, 1994.
[10] P. Litwinowicz and L. Williams, "Animating Images with Drawings," *Computer Graphics (Proc. SIGGRAPH '94)*, pp. 409-412, 1994.
[11] N. Arad, N. Dyn, D. Reisfeld, and Y. Yeshurun, "Image Warping by Radial Basis Functions: Applications to Facial Expressions," *CVGIP: Graphical Models and Image Processing*, vol. 56, pp. 161-172, Mar. 1994.
[12] T. Nishita, T. Fujii, and E. Nakamae, "Metamorphosis Using Bézier Clipping," *Proc. First Pacific Conf. Computer Graphics and Applications*, pp. 162-173, Seoul, 1993.
[13] R.C. Gonzalez and R.E. Woods, *Digital Image Processing*. Reading, Mass.: Addison-Wesley, 1992.
[14] S. Coquillart, "Extended Free-Form Deformation: A Sculpturing Tool for 3D Geometric Modeling," *Computer Graphics (Proc. SIGGRAPH '90)*, vol. 24, no. 4, pp. 187-196, 1990.
[15] S. Coquillart and P. Jancene, "Animated Free-Form Deformation: An Interactive Animation Technique," *Computer Graphics (Proc. SIGGRAPH '91)*, vol. 25, no. 4, pp. 23-26, 1991.
[16] W.M. Hsu, J.F. Hughes, and H. Kaufman, "Direct Manipulation of Free-Form Deformations," *Computer Graphics (Proc. SIGGRAPH '92)*, vol. 26, no. 2, pp. 177-184, 1992.
[17] J.D. Foley, A. van Dam, S.K. Feiner, and J.F. Hughes, *Computer Graphics: Principles and Practice,* second edition. Reading, Mass.: Addison-Wesley, 1990.
[18] T. Goodman and K. Unsworth, "Injective Bivariate Maps," Technical Report CS94/02, Dundee Univ., UK, 1994.
[19] G. Meisters and C. Olech, "Locally One-to-One Mappings and a Classical Theorem on Schlicht Functions," *Duke Mathematical J.*, vol. 30, pp. 63-80, 1963.



Seungyong Lee received his BS degree in computer science and statistics from Seoul National University, Korea, in 1988, and his MS and PhD degrees in computer science from the Korea Advanced Institute of Science and Technology (KAIST) in 1990 and 1995, respectively. He is currently an assistant professor in the Department of Computer Science and Engineering at Pohang University of Science and Technology (POSTECH), Korea. His research interests include computer graphics, computer animation, and image processing.

George Wolberg received his BS and MS degrees in electrical engineering from Cooper Union in 1985, and his PhD degree in computer science from Columbia University in 1990. He is currently an associate professor of computer science at the City College of New York/CUNY. His research interests include image processing, computer graphics, and computer vision. Prof. Wolberg is the recipient of a 1991 U.S. National Science Foundation Presidential Young Investigator Award. He has published more than 20 papers and book chapters and holds two patents. He is the author of *Digital Image Warping* (IEEE Computer Society Press, 1990), the first comprehensive monograph on warping and morphing.

Kyung-Yong Chwa received his BS degree in electrical engineering in 1971 from Seoul National University, Seoul, Korea, and his MS and PhD degrees in electrical engineering and computer science from Northwestern University, in 1977 and 1980, respectively. He is currently a professor in the Department of Computer Science at KAIST and the president of the Korea Information Science Society (KISS). His research interests include graph theory, algorithmic design and analysis, and computational geometry.

Sung Yong Shin received his BS degree in industrial engineering in 1970 from Hanyang University, Seoul, Korea, and his MS and PhD degrees in industrial and operations engineering from the University of Michigan, Ann Arbor, in 1983 and 1986, respectively. He is currently an associate professor in the Department of Computer Science at KAIST. His research interests include computer graphics and computational geometry.