Image Metamorphosis Using Snakes and Free-Form Deformations

Seung-Yong Lee†, *Kyung-Yong Chwa, Sung Yong Shin*
Department of Computer Science
Korea Advanced Institute of Science and Technology
Taejon 305-701, Korea

George Wolberg
Department of Computer Science
City College of New York / CUNY
New York, NY 10031

Abstract

This paper presents new solutions to the following three problems in image morphing: feature specification, warp generation, and transition control. To reduce the burden of feature specification, we first adopt a computer vision technique called snakes. We next propose the use of multilevel free-form deformations (MFFD) to achieve C^2-continuous and one-to-one warps among feature point pairs. The resulting technique, based on B-spline approximation, is simpler and faster than previous warp generation methods. Finally, we simplify the MFFD method to construct C^2-continuous surfaces for deriving transition functions to control geometry and color blending.

Keywords: Image metamorphosis, morphing, snakes, multilevel free-form deformation, multilevel B-spline interpolation.

1. Introduction

Image metamorphosis deals with the fluid transformation from one digital image into another. This technique, commonly referred to as *morphing*, has found widespread use in the entertainment industry to achieve stunning visual effects. Smooth transformations are 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.

Given two images, their correspondence is established by an animator with pairs of points or line segments. Each point or line segment specifies an image feature, or landmark. The feature correspondence is then used to compute warps to interpolate the positions of the features across the morph sequence. Once the source and destination images have been warped into alignment for intermediate feature positions, ordinary color interpolation (i.e., cross-dissolve) is performed to generate an inbetween image. Since color interpolation is straightforward, research in image morphing has concentrated on warp generation from the feature correspondence.

In mesh-based techniques, such as mesh warping [19] and the method of Nishita *et al.* [14], nonuniform meshes are used to specify the image features. Spline interpolation or Bézier clipping [15] computes warps from the correspondence of mesh points. Though fast and intuitive, the mesh-based techniques have a drawback in specifying features. A control mesh is always required although the features on an image may have an arbitrary structure.

Field morphing [2] uses a set of line segments to effectively specify the features on an image. A pair of line segments on two images determines a warp from their local coordinate systems. When two or more pairs of line segments are specified, the influences of line segments are blended by their weighted average. This technique suffers from unexpected distortions, referred to as *ghosts*, which prevent an animator from realizing a precise warp in a complex metamorphosis.

If the features on an image are specified by a set of points, the x- and y-components of a warp can be derived by constructing the surfaces that interpolate scattered points. Warp generation by this approach was extensively surveyed in [16, 19], and recently two similar methods were independently proposed using the thin plate surface model [10, 12]. These techniques generate smooth warps that exactly reflect the feature correspondence.

When a warp is applied to an image, the one-to-one property of the warp guarantees that the distorted image does not fold back upon itself. An energy minimization method has been proposed which derives a C^1-continuous and one-to-one warp from a set of feature point pairs [11]. The performance of that method is hampered by its high computational cost. This paper presents a new multilevel free-form deformation (MFFD) technique to derive a C^2-continuous and one-to-one warp that exactly satisfies the feature correspondence. The proposed technique, based on 2D B-spline approximation, is much simpler and faster than the energy minimization method.

Another interesting problem in image morphing is transition control. If transition rates differ from part to part in inbetween images, more interesting animations are possible. In mesh warping [19], a transition curve is assigned to each point of the mesh to control the transition behavior. When complicated meshes are used to specify the features, it is tedious to assign a proper transition curve to every mesh point. Nishita *et al.* mentioned that the transition behavior can be controlled by a Bézier function defined on the mesh [14]. However, the details of the method were not provided.

An effective method for transition control has been proposed in [11]. 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 paper, the MFFD technique for warp generation is simplified and applied to efficiently generate a C^2-continuous surface.

The most tedious aspect of image morphing is feature specification. Despite their usefulness, computer vision techniques remain only marginally utilized for this task [3]. In this paper, we adopt snakes [8], an active contour model made popular in computer vi-

†Current address: Dept. of Computer Science, City College of New York, 138th St. at Convent Ave., Rm. R8/206, New York, NY 10031.
[cssyl | wolberg]@cs-mail.engr.ccny.cuny.edu
[kychwa | syshin]@jupiter.kaist.ac.kr

Appears in: Proc. Siggraph '95.

sion, to reduce the burden of feature specification for animators.

2. Preliminaries

The authors have proposed a general framework for generating an inbetween image from two images, including transition behavior control [9, 11, 19]. The framework is also taken for the new image morphing technique given in this paper.

2.1 Metamorphosis framework

Let F_0 and F_1 be two sets of features specified by an animator on the source and destination 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. The requirement for W_1 is 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 destination 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 destination 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.

$$
\begin{aligned}
\overline{W}_0(p;t) &= (1 - T_0(p;t)) \cdot p + T_0(p;t) \cdot W_0(p) \\
\overline{W}_1(q;t) &= T_1(q;t) \cdot q + (1 - T_1(q;t)) \cdot W_1(q) \\
I_0(p;t) &= \overline{W}_0(p;t) \bullet ((1 - T_0(p;t)) \cdot I_0(p)) \\
I_1(q;t) &= \overline{W}_1(q;t) \bullet (T_1(q;t) \cdot I_1(q)) \\
I(r;t) &= I_0(r;t) + I_1(r;t).
\end{aligned}
$$

Note that $0 \leq T_0, T_1 \leq 1$ and $0 \leq t \leq 1$. Transition rates 0 and 1 imply the source and destination images, respectively. To complete the procedure, a solution is needed to each of the following problems:

- 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.

2.2 The energy minimization method

An energy minimization method has been proposed for deriving warp functions in [11]. That method allows extensive feature specification primitives such as points, polylines, and curves. Feature correspondence is established by converting non-point features to feature points using point sampling. A warp is interpreted as a 2D deformation of a rectangular plate. The feature correspondence assigns a new position to each feature point to derive a warp. 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. It is, however, a bit involved to implement.

Transition functions are obtained by selecting a set of points on a given image and specifying a transition curve for each point. The transition curves determine the transition behavior of the selected points over time. For a given time, transition functions must have the values assigned by the transition curves at the selected points. Considering a transition rate as the vertical distance from a plane, transition functions are reduced to smooth surfaces that interpolate a set of scattered points. The thin plate surface model [18] was employed to obtain C^1-continuous surfaces for transition functions.

2.3 Overview

With the metamorphosis framework described in Section 2.1, we present a more effective technique than the previous energy minimization method. To help an animator specify image features, we use snakes [8], a technique popularized in computer vision. Snakes make it possible to capture the exact position of a feature easily and precisely.

To derive warps from positional constraints, we propose the MFFD as an extension to free-form deformation (FFD) [17]. 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 positional constraints. To guarantee the one-to-one property of a warp, 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. We also present a hybrid approach that combines the two methods.

To obtain smooth surfaces for transition functions, we simplify the MFFD to obtain multilevel B-spline interpolation. This interpolation algorithm efficiently generates a C^2-continuous surface through a set of scattered points.

3. Feature Specification

Features consist of image landmarks, e.g., the profile, eyes, noise, and mouth of a facial image. The position of a feature is usually identified by a boundary curve at edges, where color values change abruptly. We adopt the use of snakes to assist us in the precise positioning of features.

3.1 Snakes

Snakes [8] 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, $v(s) = (x(s), y(s))$, its energy functional can be written as

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

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 paper. We also simplify the spline energy to $E_{spline} = \beta|\frac{d^2v}{ds^2}|^2$, which makes a snake act like a thin plate.

For a gray-scale image I, the gradient ∇I measures the local changes of image values and can be computed by a difference operator or the Sobel operator [1]. The image energy functional can be defined by $E_{image} = -\nabla^2 I = -|\nabla I(x, y)|^2$. 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 and the details of the energy minimization procedure can be found in [8].

3.2 Feature specification primitives

In this paper, the feature specification primitives include points, polylines, and curves as in [11]. However, the positions of features can be derived more effectively by generating snakes from polylines and curves. To specify a feature having large image gradients, a snake is initialized by positioning a polyline or curve near the feature. We then uniformly sample a sequence of points on the polyline or curve, e.g., 20 points per segment. As the snake minimizes its energy, it slithers and finally locks onto the feature by the image force.

To tailor the response of the snake, 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 [8] 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 user-specified primitive.

When 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. We either move f_0 repeatedly or generate a snake from f_0 to identify a feature on I_0. f_1 is then moved to designate the corresponding feature on I_1, and a snake is initiated if necessary.

If f_0 and f_1 are polylines or curves, the correspondence between them is established by their vertices or control points, respectively. The correspondence between two snakes can be derived from the polylines or curves that provide their initial positions. The feature correspondence between two images is internally translated to a set of point pairs sampled on the specified feature primitives.

Fig. 7 shows an example. Fig. 7(a) is the input image. We convert it to a gray-scale image and apply the Sobel operator [1] to compute image gradients. Fig. 7(b) shows the image gradients convolved with a Gaussian filter, where bright intensities denote large gradients. In Fig. 7(c), we place a polyline near the profile of the image. The snake starting from the polyline exactly captures the profile, as in Fig. 7(d). Fig. 7(e) illustrates the specified feature primitives overlaid on the image. The cyan points in Fig. 7(f) represent internally sampled feature points on the primitives. We typically use a uniform sampling rate of 20 points per primitive segment, although only five points per segment are shown in the figure.

4. Warp Generation

Free-form deformation (FFD) was proposed by Sederberg and Parry as a powerful modeling tool for 3D deformable objects [17]. 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 [4] and proposed a technique for animating objects modeled by FFD [5]. Hsu *et al.* employed the FFD method to directly control the shape of an object under complex deformations [7]. 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 generate a C^2-continuous and one-to-one warp from positional constraints. A rectangular plate in the xy-plane is deformed by manipulating a parallelepiped lattice overlaid on it. We take the bivariate cubic B-spline tensor product as the deformation function of FFD because a B-spline has local control. 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.

4.1 Free-form deformation and the 1-to-1 property

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 ij-th control point lies at its initial position, $\phi^0_{ij} = (i, j)$. 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. 1).

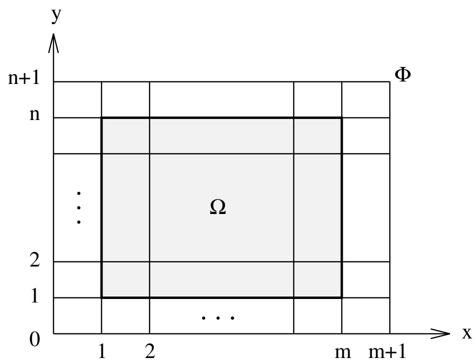


Figure 1: The initial arrangement of the plate and control lattice

Let ϕ_{ij} be the position of the ij-th 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. 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 Eq. (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.

Now, we consider the one-to-one property of the function $\mathbf{w}$ defined in Eq. (1). Function $\mathbf{w}$ 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 [6]. 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.

Theorem 1 *The function* $\mathbf{w}$ *given in Equation (1) is one-to-one if* $(-0.48, -0.48) \le \phi_{ij} - \phi^0_{ij} \le (0.48, 0.48)$ *for all* i, j.

Proof: See Appendix. □

Theorem 1 provides a tight, sufficient, although not necessary, condition. There are examples in which the B-spline surface is not one-to-one even though all control points are displaced by amounts less than 0.5. This means that a B-spline surface may violate the one-to-one property even when control lattice gridlines do not intersect among themselves.

4.2 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 kl-th control points, $k, l = 0, 1, 2, 3$, on lattice Φ determine the deformed position $\mathbf{w}(p)$ of point p. See Fig. 2(a). Let $\Delta q = \mathbf{w}(p) - \mathbf{w}^0(p) = q - p$ be the movement of the point p from its original position. Let $\Delta\phi_{kl} = \phi_{kl} - \phi^0_{kl}$ be the displacement of the kl-th control point from its initial position. From Eqs. (1) and (2), the displacements $\Delta\phi_{kl}$ must satisfy Eq. (3):

$$\Delta q = \sum_{k=0}^{3}\sum_{l=0}^{3} B_k(s)B_l(t)\Delta\phi_{kl}, \tag{3}$$

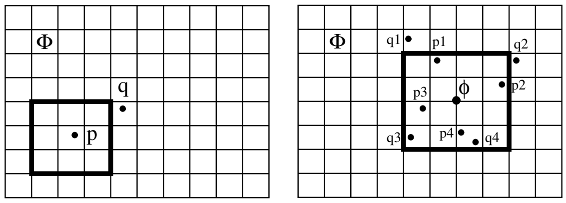
where $s = u - 1$ and $t = v - 1$.

There are many values of $\Delta\phi_{kl}$ that are solutions to Eq. (3). We choose one in the least-squared sense such that

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

where $w_{ab} = B_a(s)B_b(t)$. Among all the solutions to Eq. (3), Hsu *et al.* showed that this minimizes the squared sum of control point displacements [7]. In this solution, the control points near point p get larger displacements than the others because w_{kl} depends on the distance between the kl-th control point and point p. Hence, it generates the deformation $\mathbf{w}$ 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 Eq. (4). However, these displacements



(a) single constraint (b) multiple constraints

Figure 2: Examples of the positional constraints

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 ij-th control point on lattice Φ whose initial position is (i, j), as in Fig. 2(b). The displacement of control point ϕ influences the movement of points in P' when we evaluate the deformation function $\mathbf{w}$. For each point p_c in P', Eq. (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 $w_c\Delta\phi$ 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

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

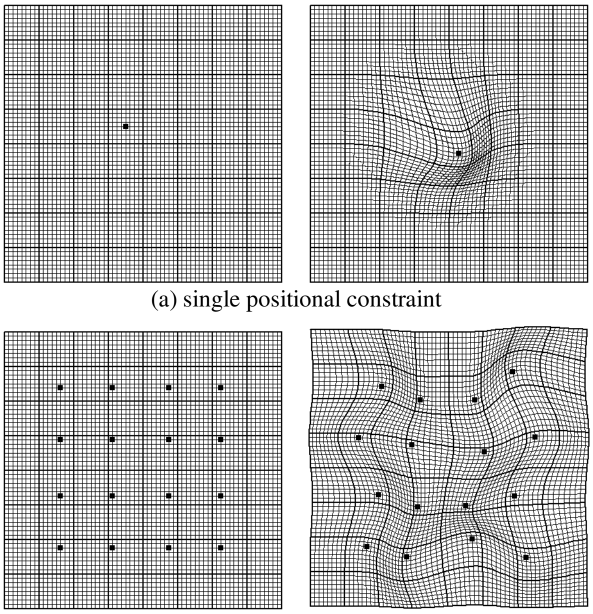
$w_c\Delta\phi$ is the movement of point p_c due to the displacement $\Delta\phi$ of control point ϕ. $w_c\Delta\phi_c$ represents the contribution of control point ϕ, determined by Eq. (4), for moving p_c to its specified position. By differentiating the error with respect to $\Delta\phi$ and equating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

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

When the displacements of the control points on lattice Φ are computed by Eq. (5), the resulting deformation function $\mathbf{w}$ is not guaranteed to be one-to-one. To make the function $\mathbf{w}$ one-to-one, we truncate the displacement $\Delta\phi$ of a control point ϕ so that $(-0.48, -0.48) \le \Delta\phi \le (0.48, 0.48)$. Then, the condition of Theorem 1 holds and the derived function $\mathbf{w}$ is one-to-one.

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 [7]. 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 actual 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. 3 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.



(a) single positional constraint

(b) multiple positional constraints

Figure 3: Examples of the FFD manipulation

4.3 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 4.2 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.

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 simple 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

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

where q_c is the position in Q specified for the 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 0 and 1. 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 FFD manipulation generates a C^2-continuous and one-to-one deformation function. In the MFFD, a deformation function is the composition of several functions derived by FFD manipulations. Hence, the resulting deformation of plate Ω is C^2-continuous and one-to-one. Furthermore, the result is guaranteed to remain one-to-one even when the positional constraints are prone to foldovers. This is achieved by relaxing the requirement to exactly satisfy the positional constraints in order to retain the one-to-one property.

Fig. 4 gives an example in which the MFFD is applied to generate a deformation of plate Ω from positional constraints. Fig. 4(a) shows the selected points in the undeformed shape of the plate. Figs. 4(b) through (e) show a sequence of deformations in which the deformed positions of the selected points gradually approach the specified positions. Fig. 4(f) shows the resulting deformation with the specified positions. In this example, the FFD manipulations are performed no more than twice at each level of the control lattice.

Most of the computation for the MFFD is consumed in evaluating the deformation function $\mathbf{w}$ on plate Ω. When the function $\mathbf{w}$ is evaluated on a 64×64 grid, it takes 0.2 seconds for an SGI Crimson to generate the deformation in Fig. 4(f). When the size of the grid is 512×512, the computation time is 9.2 seconds.

4.4 A hybrid approach

An energy minimization method has been proposed to derive C^1-continuous and one-to-one warps from positional constraints [11]. It generates natural warps but requires much computation if warps are evaluated on large grids. The MFFD can be combined with that method to derive warps more effectively.

Suppose that warps are to be evaluated on an $m \times n$ grid. First, the energy minimization method is used to obtain a warp $\mathbf{w}_e$ on a coarse $(m/k) \times (n/k)$ grid using positional constraints derived

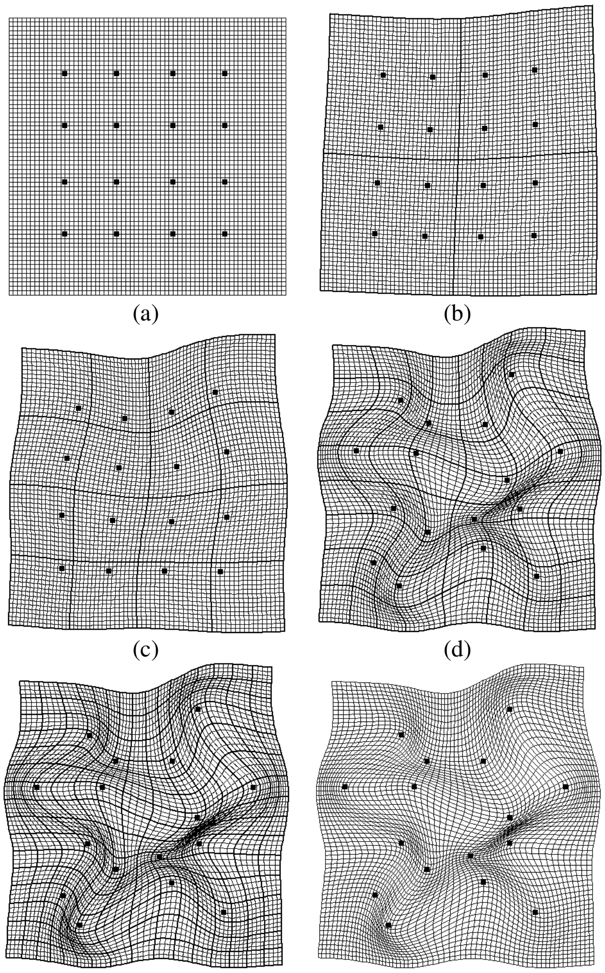


Figure 4: An example of the MFFD

from the original grid by weighted averaging. Then, a warp $\mathbf{w}_0$ is derived on the $m \times n$ grid by constructing a 2D uniform cubic B-spline surface that interpolates the values of the function $\mathbf{w}_e$. The function $\mathbf{w}_0$ is C^2-continuous and one-to-one but does not satisfy the given positional constraints exactly. Finally, the MFFD is applied to handle the remainder of the positional constraints, where $(m/k) \times (n/k)$ is the size of the coarsest control lattice.

In the hybrid approach, the energy minimization method determines the global shape of the generated warp on a coarse grid in a short time. The MFFD on the original grid makes it 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. 5 gives an example. Fig. 5(a) shows the selected points in the undeformed shape of the plate. Figs. 5(b), (c), and (d) 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. 5(b), (c), and (d) on an SGI Crimson are 26.7, 6.4, and 7.5 seconds, respectively.

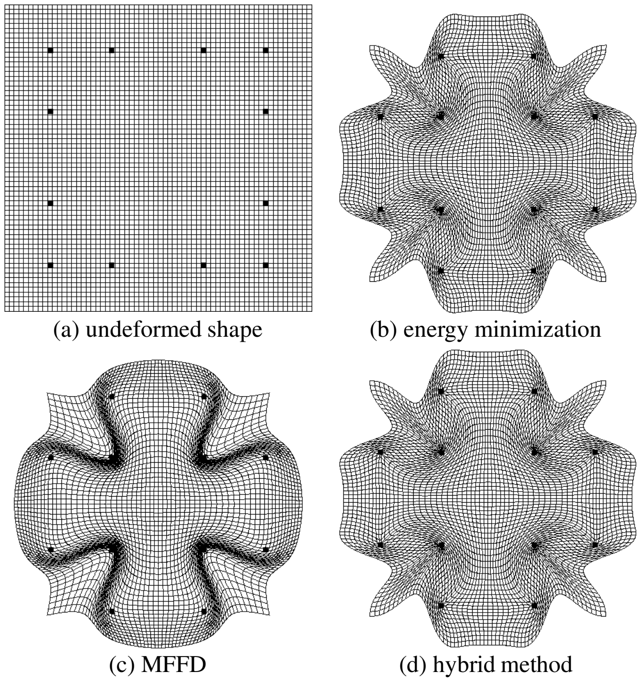


Figure 5: Comparison of the deformed shapes of a plate

5. Transition Control

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. The purpose of deriving this surface is to propagate the transition control information specified at only sparse positions. This information is given by the user with transition curves specified along primitives. Note that these primitives do not necessarily relate to feature primitives. That is, a different set of points, polylines, or curves may be defined to specify transition behavior.

5.1 Manipulation of B-spline surfaces

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 a $(m+2) \times (n+2)$ 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 Eq. (1) for the deformation function $\mathbf{w}$ in Section 4.1.

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 4.2. The required heights of control points from the uv-plane are derived by Eqs. (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.

5.2 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 Eqs. (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 coarsest spacing h_0 determines the area of the resulting surface on which an interpolated point has effect. The finest spacing h_m controls the precision to which the resulting surface interpolates the given points.

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 the maximum difference between the points in P and the final surface f falls below a given threshold. Unlike the MFFD case, the B-spline surface manipulation is applied only once on each control lattice because the heights of the control points are not truncated.

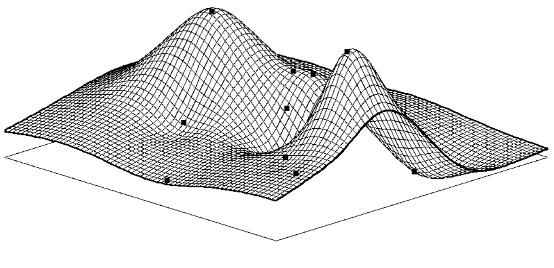


Figure 6: An example of multilevel B-spline interpolation

A surface generated by multilevel B-spline interpolation is C^2-continuous because it is the sum of C^2-continuous B-spline surfaces. Fig. 6 shows an example. Black spots in the figure represent the interpolated points. Most of the computation time is spent by evaluating function f in the region Ω. For a 64×64 grid, it takes 0.1 seconds for an SGI Crimson to generate the surface given in Fig. 6. For a 512×512 grid, the surface is generated in 1.5 seconds.

6. Metamorphosis examples

Fig. 10 gives metamorphosis examples. Figs. 10(a) and (b) show selected frames from a morph sequence between Seung-Yong and George (two of the authors), and Linda. All the images in Fig. 10(a) were generated using the same transition rate everywhere. The transition rates were allowed to vary to generate the images in Fig. 10(b). Fig. 10(c) shows the specified features overlaid on the input images. The bottom two inbetween images in that column demonstrate the effect of a procedural transition function. Fig. 10(d) shows the frames of a morph sequence between Linda and Seung-Yong. Note that transition control is applied to overcome the considerable differences between the hairlines.

One set of transition curves for the source image is sufficient to relate the transition behavior in a morph sequence. Figs. 8(a) and (b) show the primitives upon which transition curves are defined for the inbetween images given in Figs. 10(b) and (d), respectively. Each primitive may have a different transition curve, with all points along a primitive sharing the same transition rate. Figs. 8(c) and (d) depict the transition curves for the respective outer and inner primitives of Figs. 8(a) and (b). For the inbetween images shown in Fig. 10(c), linear functions in y are applied to vary the transformation between the source and destination images. In the first inbetween image, Seung-Yong is changed into George from top to bottom. Similarly, George is transformed into Seung-Yong in the second image.

Fig. 9(a) illustrates the warp function generated for transforming Seung-Yong to George in Fig. 10(a). Dark lines represent the new positions of feature points that have been internally sampled on the source image. Fig. 9(b) shows the surface interpolating through the transition rates that are evaluated from the transition curves along the primitives in Fig. 8.

All images shown in Fig. 10 are 468×480 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. 9.

7. Conclusions

This paper has presented new 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 [8], 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 introduced a new deformation technique, the MFFD, 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. 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 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.

8. Acknowledgements

This work was supported in part by the Korean Ministry of Science and Technology (contract 94-S-05-A-03 of STEP 2000), NSF PYI award IRI-9157260, and PSC-CUNY grant RF-665313.

References

[1] Ballard, Dana H., and Christopher M. Brown. *Computer Vision*. Prentice-Hall, 1982.

[2] Beier, Thaddeus, and Shawn Neely. Feature-Based Image Metamorphosis. *Computer Graphics 26*, 2 (1992), 35–42.

[3] Benson, Philip J. Morph Transformation of the Facial Image. *Image and Vision Computing 12*, 10 (1994), 691–696.

[4] Coquillart, Sabine. Extended Free-Form Deformation: A Sculpturing Tool for 3D Geometric Modeling. *Computer Graphics 24*, 4 (1990), 187–196.

[5] Coquillart, Sabine, and Pierre Jancene. Animated Free-Form Deformation: An Interactive Animation Technique. *Computer Graphics 25*, 4 (1991), 23–26.

[6] Goodman, Tim, and Keith Unsworth. Injective Bivariate Maps. Tech. Rep. CS94/02, Dundee University, U.K., 1994.

[7] Hsu, William M., John F. Hughes, and Henry Kaufman. Direct Manipulation of Free-Form Deformations. *Computer Graphics 26*, 2 (1992), 177–184.

[8] Kass, Michael, Andrew Witkin, and Demetri Terzopoulos. Snakes: Active Contour Models. *International Journal of Computer Vision* (1988), 321–331.

[9] Lee, Seung-Yong. *Image Morphing Using Scattered Feature Interpolations*. PhD thesis, KAIST, Taejon, Korea, February 1995.

[10] Lee, Seung-Yong, Kyung-Yong Chwa, James Hahn, and Sung Yong Shin. Image Morphing Using Deformable Surfaces. In *Proceedings of Computer Animation '94* (Geneva, Switzerland, 1994), IEEE Computer Society Press, pp. 31–39.

[11] Lee, Seung-Yong, Kyung-Yong Chwa, James Hahn, and Sung Yong Shin. Image Morphing Using Deformation Techniques. *The Journal of Visualization and Computer Animation 6*, 3 (1995).

[12] Litwinowicz, Peter, and Lance Williams. Animating Images with Drawings. In *SIGGRAPH 94 Conference Proceedings* (1994), ACM Press, pp. 409–412.

[13] Meisters, G.H., and C. Olech. Locally One-to-one Mappings and a Classical Theorem on Schlicht Functions. *Duke Mathematical Journal 30* (1963), 63–80.

[14] Nishita, Tomoyuki, Toshihisa Fujii, and Eihachiro Nakamae. Metamorphosis Using Bézier Clipping. In *Proceedings of the First Pacific Conference on Computer Graphics and Applications* (Seoul, Korea, 1993), World Scientific Publishing Co., pp. 162–173.

[15] Nishita, Tomoyuki, Thomas Sederberg, and Masanori Kakimoto. Ray Tracing Trimmed Rational Surface Patches. *Computer Graphics 24*, 4 (1990), 337–345.

[16] Ruprecht, Detlef, and Heinrich Müller. Image Warping with Scattered Data Interpolation. *IEEE Computer Graphics and Applications 15*, 2 (1995), 37–43.

[17] Sederberg, Thomas W., and Scott R. Parry. Free-Form Deformation of Solid Geometric Models. *Computer Graphics 20*, 4 (1986), 151–160.

[18] Terzopoulos, Demetri. Multilevel Computational Processes for Visual Surface Reconstruction. *Computer Vision, Graphics, and Image Processing 24* (1983), 52–96.

[19] Wolberg, George. *Digital Image Warping*. IEEE Computer Society Press, Los Alamitos, CA, 1990.

Appendix: Proof of Theorem 1

Let $\Delta\phi_{ij} = \phi_{ij} - \phi_{ij}^0$ and $\mathbf{w} = (x, y)$. Suppose that $\frac{\partial x}{\partial u} > |\frac{\partial x}{\partial v}|$ and $\frac{\partial y}{\partial v} > |\frac{\partial y}{\partial u}|$ at each point on the domain Ω when $(-0.48, -0.48) \leq \Delta\phi_{ij} \leq (0.48, 0.48)$ 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 [13]. Let $\phi_{ij} = (x_{ij}, y_{ij})$, $\phi_{ij}^0 = (x_{ij}^0, y_{ij}^0)$, and $\Delta y_{ij} = y_{ij} - y_{ij}^0$. $\frac{\partial y}{\partial v} > |\frac{\partial y}{\partial u}|$ 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 $-0.48 \leq \Delta y_{ij} \leq 0.48$. 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, $\frac{\partial x}{\partial u} > |\frac{\partial x}{\partial v}|$.

From Eqs. (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,

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

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 $B_k(t) \geq 0$, for $i = 0, 1, 2, 3$, $B_0'(t) \leq 0$, $B_1'(t) \leq 0$, $B_2'(t) \geq 0$, and $B_3'(t) \geq 0$ when $0 \leq t \leq 1$. Therefore, it immediately follows that $c_{20}, c_{30}, c_{21}, c_{31} \leq 0$ and $c_{02}, c_{12}, c_{03}, c_{13} \geq 0$. 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 $\Delta t \geq -s$ and $(4s-3s^2) \geq 0$ when $0 \leq s \leq 1$, we get $(4s - 3s^2)\Delta t \geq -s(4s - 3s^2) \geq -1$, which implies $c_{10} \leq 0$. Similarly, it can be proved that $c_{32} \leq 0$, $c_{01} \geq 0$, and $c_{23} \geq 0$ when $0 \leq s, t \leq 1$.

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 $s \geq t$, then $c_{00}, c_{33} \geq 0$ and if $s < t$, then $c_{00}, c_{33} \leq 0$. 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 $0 \leq s, t \leq 1$, which implies that f has a global minimum at $(1, 1)$. Because $f(1, 1) = 0$, $c_{11} \geq 0$ if $s \geq t$ and $c_{11} < 0$ if $s < t$. Similarly, it can be shown that $c_{22} \geq 0$ if $s \geq t$ and $c_{22} < 0$ if $s < t$.

In summary, $c_{kl} \geq 0$ if $k < l$ and $c_{kl} \leq 0$ if $k > l$ when $0 \leq s, t \leq 1$. Also, if $s \geq t$, then $c_{kk} \geq 0$, and if $s < t$, then $c_{kk} \leq 0$. We consider the case when $s \geq t$. 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 $0 \leq s, t \leq 1$. From the condition that $-0.48 \leq \Delta y_{ij} \leq 0.48$ and the properties of the values of c_{kl}, it holds that

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

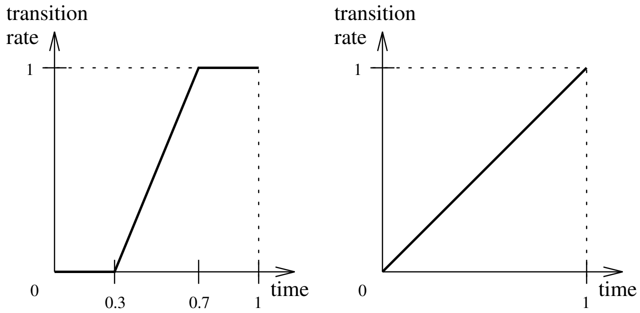
$$
\begin{aligned}
&\geq \quad 1 + 0.48\left(\sum_{k=0}^{3}\sum_{l=0}^{k-1} c_{kl} - \sum_{k=0}^{3}\sum_{l=k}^{3} c_{kl}\right) \\
&= \quad 1 + 0.48C.
\end{aligned}
$$

To derive a lower bound of C, we partition the domain $0 \leq s, t \leq 1$ 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 $0 \leq \Delta s, \Delta t < \Delta d$. 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 $\varepsilon = \frac{4230}{36}$.

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

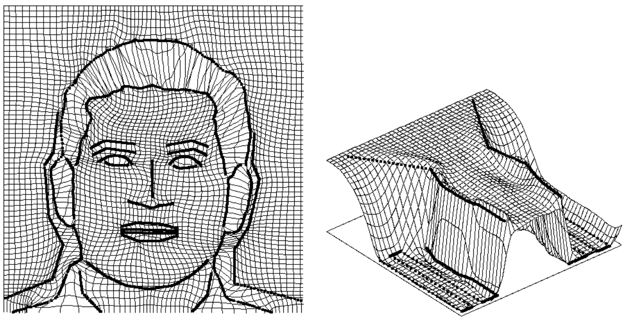
Figure 7: Feature specification; (a)-(f) are shown from left-to-right and top-to-bottom.

(a) (b)



(c) (d)

Figure 8: Primitives with transition curves



(a) (b)

Figure 9: Warp function and surface

(a) (b) (c) (d)

Figure 10: Metamorphosis examples