Image Morphing Using Deformation Techniques

Seung-Yong Lee*, Kyung-Yong Chwa*, James Hahn**,
and Sung Yong Shin*

*Department of Computer Science
Korea Advanced Institute of Science and Technology
373-1 Kusong-dong Yusong-gu Taejon, 305-701, Korea

**Department of EE & CS
The George Washington University
801 22nd Street NW, Washington, DC 20052, U.S.A.

SUMMARY

This paper presents a new image morphing method using a two-dimensional deformation technique which provides an intuitive model for a warp. The deformation technique derives a C^1-continuous and one-to-one warp from a set of point pairs overlaid on two images. The resulting inbetween image precisely reflects the correspondence of features specified by an animator. We also control the transition behavior in a metamorphosis sequence by taking another deformable surface model, which is simpler and thus more efficient than the deformation technique for a warp. The proposed method separates transition control from feature interpolation and is easier to use than the previous techniques. The multigrid relaxation method is employed to solve a linear system in deriving a warp or transition rates. This method makes our image morphing technique fast enough for an interactive environment.

Keywords: Image morphing, Deformation technique, Energy minimization method, Variational principle, Multigrid relaxation method

1 Introduction

Image morphing deals with the metamorphosis of an image to another image. The metamorphosis generates a sequence of inbetween images in which an image gradually changes into another image over time. Image morphing techniques have been widely used in creating special effects for television commercials, music videos such as Michael Jackson's *Black or White*[1], and movies such as *Willow* and *Indiana Jones and the Last Crusade*[2].

The problem of image morphing is basically how an inbetween image is effectively generated from two given images[1]. When two face images are given, for example, a middle image may look like a third face resembling the given faces. An inbetween image can be derived from two images by properly interpolating the positions of corresponding features and their shapes and colors. A feature of an image is its characterizing part such as the profile of a face and eyes and usually identified by a boundary curve at which colors change abruptly.

A warp is a two-dimensional geometric transformation and generates a distorted image when it is applied to an image. When two images are given, an image morphing method first establishes the feature correspondence between them. The correspondence is then used to compute warps that distort the images to align the positions of features and their shapes. A cross-dissolve of colors at each corresponding pair of pixels in the distorted images finally gives an inbetween image.

The most difficult part of image morphing is to derive warps which distort images to align their features. The features on an image are usually specified by an animator with a set of points or line segments overlaid on the image. A warp is then computed from the correspondence between the features on two images. Therefore, an image morphing technique must be convenient in specifying features and show a predictable distortion which reflects the feature correspondence.

In mesh warping[2], features are specified by a nonuniform control mesh, and a warp is computed by a spline interpolation. Nishita *et al.*[3] also used a nonuniform control mesh to specify features and computed a warp using a two-dimensional free-form deformation and Bézier clipping. Field morphing[1] specifies features with a set of line segments and computes a warp by taking the weighted average of the influences of line segments.

Mesh warping and the method of Nishita *et al.* show good distortion behaviors but have a drawback in specifying features. A control mesh is always required while the features on an image can have an arbitrary structure. Field morphing gives an easy-to-use and expressive method in specifying features. However, it suffers from unexpected distortions referred to as ghosts, which prevent an animator from realizing precise warps as shown in Section 7. The time for computing a warp is proportional to the number of line segments. This is disadvantageous when a complicated feature set must be specified.

These drawbacks can be overcome by a physically-based approach which provides an intuitive model for a warp. Consider an image printed on a sheet of rubber. When selected points on the sheet are moved, the sheet deformation thus obtained makes the image appear distorted. The distorted image conforms to the displacement of each selected point and shows a proper distortion over the entire image. If a set of point pairs specifies the feature correspondence between two images, we can derive a necessary warp from the sheet deformation which moves each feature point to its correspondent. There have been a number of results[4, 5, 6] in flexible object modeling that give concrete theory and techniques for supporting this approach.

This paper takes the rubber sheet model and presents a new two-dimensional deformation technique for deriving warps. The technique efficiently generates C^1-continuous and one-to-one deformations from positional constraints. This approach does not restrict a feature set to have any structure such as a mesh, allowing more freedom in designing a warp. The resulting warps show natural distortions which precisely reflect the

feature correspondence between images.

Another interesting but not yet fully investigated problem of image morphing is the control of transition behavior in a metamorphosis sequence. In generating an inbetween image, the rate of transition is usually applied uniformly over all points on the image. This results in an animation in which the entire image changes synchronously to another image. If we control the transition rates on different parts of an inbetween image independently, a more interesting animation can be obtained.

Mesh warping[2] assigns a transition curve for each point of the mesh, and these curves determine the transition rate when the positions of features are interpolated. 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.*[3] mentioned that the speed of transition can be specified by a Bézier function defined on the mesh. However, the details of the method were not provided except only one example.

This paper uses a deformable surface model to control the transition behavior, by assigning the transition curves for selected points on an image. These points are not necessarily the same as those used for specifying features. The transition rates on an inbetween image are derived from the curves by constructing a deformable surface. This approach separates transition control from feature interpolation and thus is much easier to use than the previous techniques.

Section 2 explains the steps for generating an inbetween image and defines the problems to be solved for completing the steps. The following two sections concentrate on deformation techniques to give the solutions of the problems. Section 5 introduces the multigrid relaxation method used for solving a linear system in deriving a warp and transition rates. Section 6 presents the extensions of the basic technique employed for the new image morphing method. Section 7 compares the presented method to the previous ones in warp generation and gives metamorphosis examples. Section 8 summarizes the contributions of this paper.

2 Problems in Image Morphing

2.1 Application of a warp to an image

An image I can be represented by a function from a bounded two-dimensional region Ω to a color space. A warp W is a function from Ω to Ω, which specifies a new position for each point on I. When W is applied to I, each pixel on I is copied onto the distorted image I' at the position determined by W.

The four-corner mapping paradigm[2] considers each pixel on I as a square and transforms it into a quadrilateral on I'. The quadrilateral often straddles several pixels on I' or lies in the interior of one pixel. A partial contribution is handled by scaling the intensity of the pixel on I in proportion to the fractional part of the pixel on I'. This technique generates a distorted image without holes and properly resolves the collapsed pixels.

To implement the four-corner mapping, we should evaluate a warp function W at each corner of the pixels on an image I. Hence, when the domain of W is discretized for a numerical solution, the size of a grid is chosen as the resolution of I. Once W has been computed on the grid, the four-corner mapping can be performed by the blending hardware of a SGI machine[7] in a short time.

2.2 Inbetween image generation

When two images I_0 and I_1 are given, the image morphing problem is to generate a sequence of inbetween images $I(t)$ such that $I(0) = I_0$ and $I(1) = I_1$. We assume that time t varies from 0 to 1 when the source image I_0 continuously changes to the destination image I_1.

Let W_0 be the warp function which specifies the corresponding point on I_1 to each point on I_0. When it is applied to I_0, W_0 has to distort I_0 to match I_1 in the positions of features and their shapes. Let W_1 be the warp function from I_1 to I_0. The requirement for W_1 is to map the features on I_1 to the features on I_0 when it distorts I_1.

To generate an inbetween image $I(t)$, we derive two warp functions $W_0(t)$ and $W_1(t)$ from W_0 and W_1 by linear interpolation in time t. I_0 and I_1 are then distorted by $W_0(t)$ and $W_1(t)$, resulting in intermediate images $I_0(t)$ and $I_1(t)$, respectively. The corresponding features on I_0 and I_1 have the same positions and shapes on $I_0(t)$ and $I_1(t)$. Finally, $I(t)$ is obtained by cross-dissolving the colors between $I_0(t)$ and $I_1(t)$. That is,

$$
\begin{aligned}
W_0(t) &= (1-t)\cdot R + t\cdot W_0 && (1)\\
W_1(t) &= t\cdot R + (1-t)\cdot W_1 && (2)\\
I_0(t) &= W_0(t) \bullet I_0 && (3)\\
I_1(t) &= W_1(t) \bullet I_1 && (4)\\
I(t) &= (1-t)\cdot I_0(t) + t\cdot I_1(t), && (5)
\end{aligned}
$$

where R denotes the identity warp function, and $W \bullet I$ denotes the application of a warp W to an image I.

In the above procedure, time plays the role of transition rate which determines the relative influences of the source and destination images on an inbetween image. A transition rate is a value between 0 and 1. With a transition rate near zero, an inbetween image looks more similar to the source image. Transition rates near one imply that inbetween images should be much like the destination image.

With the formulae (1), (2), and (5), the same transition rate t is applied to all points on the inbetween image $I(t)$. Therefore, the characteristics of the source and destination images are reflected in the same ratio all over an inbetween image. The rate of transition can be made different from point to point to derive a more interesting inbetween image. We introduce a transition function to facilitate the control of transition behavior in generating an inbetween image. A transition function T specifies the rate of transition for each point on an image over time.

Let T_0 be a transition function defined on the source image I_0. In generating $I(t)$, $T_0(t)$ determines how fast each point on I_0 moves to the corresponding point on the destination image I_1. $T_0(t)$ also determines how much the color of each point on I_0 is reflected on the corresponding point on $I(t)$. Let T_1 be the transition function defined on the destination image I_1, which specifies the same transition behavior with T_0. T_1 can be derived from T_0 with the correspondence of points between I_1 and I_0. To each point on I_1, $T_1(t)$ should assign the transition rate which $T_0(t)$ gives to the corresponding point on I_0.

To control the movement of each point on an inbetween image $I(t)$, we replace time t in formulae (1) and (2) with $T_0(t)$ and $T_1(t)$, respectively. For the color transformation, however, time t in formula (5) cannot be simply replaced by $T_0(t)$ and $T_1(t)$. It is because the transition functions T_0 and T_1 are not defined on the distorted images $I_0(t)$ and $I_1(t)$ but the given images I_0 and I_1, respectively. Hence, we rearrange formulae (3), (4), and (5) so that $T_0(t)$ and $T_1(t)$ are used to attenuate the color intensities of I_0 and I_1 before applying warp functions. That is,

$$
\begin{aligned}
W_0(t) &= (1-T_0(t))\cdot R + T_0(t)\cdot W_0\\
W_1(t) &= T_1(t)\cdot R + (1-T_1(t))\cdot W_1\\
I_0(t) &= W_0(t) \bullet ((1-T_0(t))\cdot I_0)\\
I_1(t) &= W_1(t) \bullet (T_1(t)\cdot I_1)
\end{aligned}
$$

$$I(t) \quad = \quad I_0(t) + I_1(t).$$

The transformation of positions and colors can be independently handled by specifying different transition functions.

2.3 Problems

To complete the above procedure for image morphing, the following two problems need further investigating:

- how to get the warp functions W_0 and W_1, and
- how to get the transition functions T_0 and T_1.

3 Warp Function Generation

This section presents a deformation technique for deriving warp functions and explains how to obtain the warp functions W_0 and W_1 with the technique.

3.1 The deformation model

Deformation techniques based on variational principles have been widely used in computer graphics to model flexible objects in three dimensions[4, 5, 8, 9]. In these techniques, the requirements for a deformation such as smoothness are represented by energy functionals, and the desired shape of an object is derived by minimizing the sum of energy functionals. The energy minimization problems are then transformed to partial differential equations, which are usually solved by numerical methods.

A warp can be considered as a deformation of a rectangular sheet in two-dimensional space. Previous deformation techniques cannot be directly applicable to obtain warps because the deformations of rectangular sheets are confined on two dimensions. In this paper, we present a new two-dimensional deformation technique which efficiently generates a C^1-continuous and one-to-one warp with variational principles.

Let Ω be a rectangular thin plate and $p = (u, v)$ a point on Ω. If every point on the plate is placed on the xy-plane, a shape of the plate can be represented by a vector-valued function, $\mathbf{w}(p) = (x(p), y(p))$. The function $\mathbf{w}$ specifies the position of each point p on the plate, lying in the xy-plane. The natural undeformed shape of the plate is a rectangle on the xy-plane and represented by the identity function, $\mathbf{r}(p) = p$.

Suppose that the selected points on the plate are required to move to the given positions on the xy-plane. The constraints can be forced by minimizing the position energy,

$$E_P(\mathbf{w}) \quad = \quad \beta \sum_k \left\| \mathbf{w}(p_k) - q_k \right\|^2,$$

where q_k is the new position specified for a point p_k on Ω. The parameter β controls the tightness of the positional constraints.

The spline energy of a function $\mathbf{w}$,

$$E_S(\mathbf{w}) \quad = \quad \iint_\Omega \left[\left\| \frac{\partial^2 \mathbf{w}}{\partial u^2} \right\|^2 + 2 \left\| \frac{\partial^2 \mathbf{w}}{\partial u \partial v} \right\|^2 + \left\| \frac{\partial^2 \mathbf{w}}{\partial v^2} \right\|^2 \right] du dv,$$

integrates the curvature variations of $\mathbf{w}$ over the domain Ω. Among the functions satisfying the positional constraints, a smooth function $\mathbf{w}$ can be obtained by minimizing the spline energy. The resulting function $\mathbf{w}$ has continuous first partial derivatives, $\partial\mathbf{w}/\partial u$ and $\partial\mathbf{w}/\partial v$[10].

In addition to C^1-continuity, one-to-one correspondence of a function $\mathbf{w}$ can be obtained by minimizing the Jacobian energy,

$$E_J(\mathbf{w}) = \alpha \iint_\Omega (J-1)^2 du dv,$$

where

$$J = \frac{\partial x}{\partial u}\frac{\partial y}{\partial v} - \frac{\partial x}{\partial v}\frac{\partial y}{\partial u}.$$

The function $\mathbf{w}$ is one-to-one on Ω if the Jacobian J is not zero in the interior of Ω and if $\mathbf{w}$ is one-to-one on the boundary of Ω[11]. Minimizing the Jacobian energy fulfills the first condition because it tries to make J one at each point on Ω. On the boundary of Ω, $\mathbf{w}$ will be made one-to-one by the boundary conditions used for the numerical solution in Section 3.2. For a shape $\mathbf{w}$ of the plate Ω, the Jacobian J determines the infinitesimal area at a point on Ω[12, 13]. It is easy to see that the Jacobian J is one at every point on Ω when the plate is in its undeformed shape $\mathbf{r}$. Hence, the Jacobian energy integrates the area variations of the shape $\mathbf{w}$ from the undeformed shape $\mathbf{r}$ over the plate. The parameter α controls the resistance of the plate to area variation from the undeformed shape.

Consequently, the desired function $\mathbf{w}$ can be derived by minimizing the energy functional,

$$ED(\mathbf{w}) = \frac{1}{2}(E_S(\mathbf{w}) + E_J(\mathbf{w}) + E_P(\mathbf{w})).$$

If a function $\mathbf{w}$ minimizes the energy functional $ED(\mathbf{w})$, the first variational derivative of $ED(\mathbf{w})$ must vanish all over the domain Ω[14]. The condition can be represented by the vector expression,

$$\frac{\delta ED}{\delta \mathbf{w}} = \frac{1}{2}\left(\frac{\delta E_S}{\delta \mathbf{w}} + \frac{\delta E_J}{\delta \mathbf{w}} + \frac{\delta E_P}{\delta \mathbf{w}}\right) = \mathbf{0}, \tag{6}$$

where

$$\begin{aligned}
\frac{\delta E_S}{\delta \mathbf{w}} &= 2\left(\frac{\partial^4 \mathbf{w}}{\partial u^4} + 2\frac{\partial^4 \mathbf{w}}{\partial u^2 \partial v^2} + \frac{\partial^4 \mathbf{w}}{\partial v^4}\right), \\
\frac{\delta E_J}{\delta \mathbf{w}} &= 2\alpha\left(\frac{\partial}{\partial u}\left(J\frac{\partial \mathbf{w}^\perp}{\partial v}\right) - \frac{\partial}{\partial v}\left(J\frac{\partial \mathbf{w}^\perp}{\partial u}\right)\right), \\
\frac{\delta E_P}{\delta \mathbf{w}} &= 2\beta\left(\mathbf{w}(p_k) - q_k\right).
\end{aligned}$$

Here, $\mathbf{w}^\perp$ denotes the vector $(-y, x)$ which is perpendicular to the vector $\mathbf{w} = (x, y)$. The position force $\delta E_P/\delta \mathbf{w}$ appears only at a point p_k on Ω for which its position q_k is specified.

The partial differential equation given in Equation (6) is called the Euler-Lagrange equation. Unfortunately, it is in general very difficult to obtain an analytic solution for the Euler-Lagrange equation. This suggests a numerical method applied to a discrete version of the equation.

3.2 Numerical solution

We discretize the domain Ω to an $M \times N$ regular grid and represent the function $\mathbf{w}$ by its values at the nodes on the grid. The positional constraints are converted to the constraints on the values of the nodal variables. The standard finite difference approximation[15] transforms the differential equation given in Equation (6) into a system of equations which consists of MN unknown vectors and MN vector equations. If the nodal variables comprising the function $\mathbf{w}$ are collected into an MN dimensional vector, the system can be written in a matrix form,

$$\mathbf{A}\mathbf{w} + \alpha\mathbf{j}(\mathbf{w}) + \beta(\mathbf{I}'\mathbf{w} - \mathbf{q}) \quad = \quad \mathbf{0}. \tag{7}$$

$\mathbf{A}$ is an $MN \times MN$ matrix which contains the coefficients of the nodal variables resulting from the spline force $\delta E_S/\delta\mathbf{w}$. $\mathbf{j}(\mathbf{w})$ is an MN dimensional vector which approximates the Jacobian force $\delta E_J/\delta\mathbf{w}$ on the nodal variables. $\mathbf{I}'$ is an $MN \times MN$ diagonal matrix in which an element is one only if the positional constraint is assigned to the corresponding nodal variable. The MN dimensional vector $\mathbf{q}$ contains the positional constraints on the nodal variables. We use the boundary conditions $\partial\mathbf{w}/\partial u = \partial\mathbf{r}/\partial u$, $\partial\mathbf{w}/\partial v = \partial\mathbf{r}/\partial v$, and $\partial^2\mathbf{w}/\partial u^2 = \partial^2\mathbf{w}/\partial v^2 = 0$ in deriving the matrix $\mathbf{A}$ and the vector $\mathbf{j}(\mathbf{w})$.

To solve Equation (7), we rewrite the equation as a diffusion equation,

$$\frac{\partial\mathbf{w}}{\partial t} \quad = \quad \mathbf{A}\mathbf{w} + \alpha\mathbf{j}(\mathbf{w}) + \beta(\mathbf{I}'\mathbf{w} - \mathbf{q}). \tag{8}$$

An initial distribution $\mathbf{w}$ relaxes to an equilibrium solution as $t \to \infty$. At the equilibrium, all time derivatives vanish and hence $\mathbf{w}$ is the solution of Equation (7). When differencing Equation (8) with respect to time, we evaluate the right-hand side at time t rather than time $t-1$, which results in the implicit Euler scheme. For computing the y- and x-components of $\mathbf{j}(\mathbf{w})$, x- and y-components of $\mathbf{w}$ are assumed constant during a time step, respectively. The assumption makes the nonlinear term $\mathbf{j}(\mathbf{w})$ linear with respect to $\mathbf{w}$. The resulting equations are

$$-\gamma(\mathbf{x}_t - \mathbf{x}_{t-1}) \quad = \quad \mathbf{A}\mathbf{x}_t + \alpha\mathbf{B}(\mathbf{y}_{t-1})\mathbf{x}_t + \beta(\mathbf{I}'\mathbf{x}_t - \mathbf{x}_\mathbf{q}) \tag{9}$$

$$-\gamma(\mathbf{y}_t - \mathbf{y}_{t-1}) \quad = \quad \mathbf{A}\mathbf{y}_t + \alpha\mathbf{B}(\mathbf{x}_{t-1})\mathbf{y}_t + \beta(\mathbf{I}'\mathbf{y}_t - \mathbf{y}_\mathbf{q}). \tag{10}$$

$\mathbf{x}_t$ and $\mathbf{y}_t$ are the x- and y-component vectors of the function $\mathbf{w}$ at time t. $\mathbf{B}(\mathbf{y}_{t-1})$ and $\mathbf{B}(\mathbf{x}_{t-1})$ are $MN \times MN$ matrices which contain the coefficients of $\mathbf{x}_t$ and $\mathbf{y}_t$ in the linear approximation of $\mathbf{j}(\mathbf{w})$, respectively. $\mathbf{x}_\mathbf{q}$ and $\mathbf{y}_\mathbf{q}$ denote the positional constraints on $\mathbf{x}$ and $\mathbf{y}$. The parameter γ controls the step size in time.

Equations (9) and (10) can be arranged in the forms,

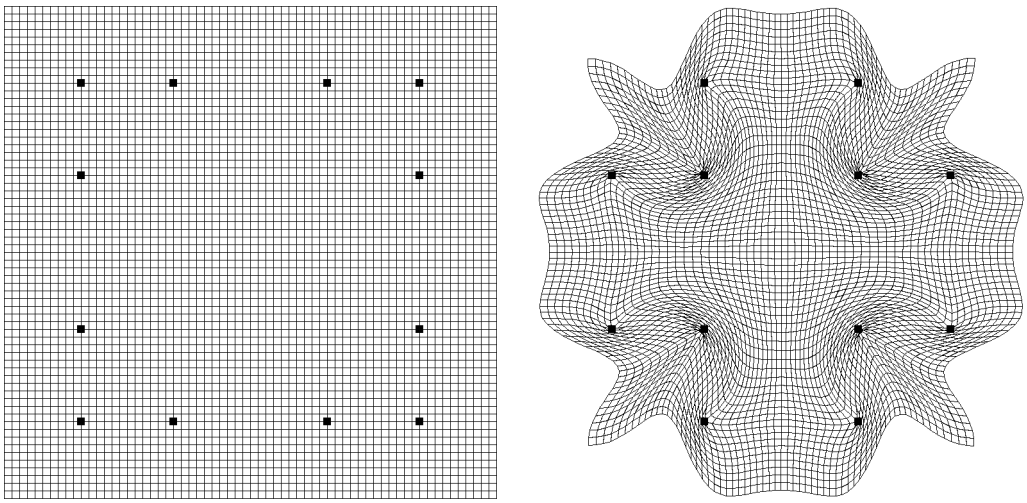
$$(\mathbf{A} + \alpha\mathbf{B}(\mathbf{y}_{t-1}) + \gamma\mathbf{I} + \beta\mathbf{I}')\mathbf{x}_t \quad = \quad \gamma\mathbf{I}\mathbf{x}_{t-1} + \beta\mathbf{x}_\mathbf{q} \tag{11}$$

$$(\mathbf{A} + \alpha\mathbf{B}(\mathbf{x}_{t-1}) + \gamma\mathbf{I} + \beta\mathbf{I}')\mathbf{y}_t \quad = \quad \gamma\mathbf{I}\mathbf{y}_{t-1} + \beta\mathbf{y}_\mathbf{q}, \tag{12}$$

in which $\mathbf{x}_t$ and $\mathbf{y}_t$ can be calculated from $\mathbf{x}_{t-1}$ and $\mathbf{y}_{t-1}$. The multigrid relaxation method in Section 5 efficiently solves Equations (11) and (12) by exploiting the bandedness of the matrices on the left-hand side.

This method for solving Equation (8) takes the implicit Euler scheme for the spline and position forces and the semi-implicit Euler scheme for the Jacobian force. Hence, the solution of the equation can be found very robustly and rapidly with a big time step. The initial shape $\mathbf{x}_0$ and $\mathbf{y}_0$ for Equations (11) and (12) is obtained from an approximate equilibrium solution computed on a hierarchy of coarse grids. With the initial shape, the equilibrium solution can be derived by solving Equations (11) and (12) in several times.

Figure 1 shows a deformation example in which the grid size is 64×64. In the figures, black spots represent the positions of selected points to which positional constraints are assigned. It takes 1.6 seconds to derive the deformation on a SGI Crimson. When the size of the grid is 512×512, the computation time increases to 26.7 seconds. The values of parameters α, β, and γ are 10.0, 2500000.0, and 0.0001, respectively. This example verifies that the proposed method generates a desired deformation very effectively.



(a) The undeformed shape (b) A deformation of the plate

Figure 1: A deformation example

3.3 Generation of warp functions W_0 and W_1

When two images are given, an animator specifies a set of point pairs on the images which represents the correspondence of features. Let P be a set of point pairs (p_i, q_i), where p_i and q_i are points on the source and destination images, I_0 and I_1, respectively. The warp function W_0 has to distort the image I_0 so that each point p_i matches the corresponding point q_i in their positions. The requirement for W_1 is to map each point q_i to the corresponding point p_i when distorting the image I_1 toward I_0. Then, the warp functions are reduced to deformations of a rectangular plate which place the specified points at the given positions.

There are several methods for deriving a warp function from the positional constraints assigned to the points on an image. In the methods, the x- and y-components of a warp function are derived by constructing smooth surfaces which interpolate scattered points. The warp generation in this approach was extensively surveyed in [2, 16]. In addition, Bookstein used the thin-plate surface model and derived a solution by decomposing a surface into a linear part and independent nonlinear deformations of progressively smaller geometric scales[17]. Two similar methods were independently proposed which employ the multigrid relaxation method to compute numerical solutions of the thin-plate surfaces[18, 19]. However, any of these methods does not guarantee that the resulting warp functions have the one-to-one property.

The deformation model in Section 3.1 generates C^1-continuous and one-to-one warp functions from the positional constraints. When a warp function is applied to an image, the one-to-one property guarantees that the distorted image does not fold back upon itself. In generating a warp function, the grid size is chosen

as the resolution of the given image. A large value is usually used for the parameter β so that the resulting warp function exactly moves features points to their correspondents. For the parameter α, a small value is sufficient to provide an one-to-one warp function.

4 Transition Function Generation

This section gives the deformable surface model for a transition function and the way for deriving the transition functions T_0 and T_1 with the surface model.

4.1 The surface model

At a given time, the transition rates of points on an image can be specified by a real-valued function defined on a rectangular region. If we consider a function value as the height from the region, the function can be represented by a surface deformed only in the vertical direction. The deformation technique given in Section 3 is inappropriate for deriving the deformable surface because the technique generates a deformation in two dimensions. Hence, we reduce the deformation model described in Section 3.1 to the thin plate surface model[14], which is simpler and enables a more efficient numerical method. The thin plate surface model has been used in computer vision to solve the visual surface reconstruction problem[10, 20, 21].

Let Ω be a rectangular thin plate on the uv-plane and $p = (u, v)$ a point on Ω. If the plate is allowed to be deformed only in the direction perpendicular to the uv-plane, a shape of the plate can be represented by a function, $f(p)$. The function f specifies a real value for each point on the plate.

Suppose that the function f should have the given values at selected points on the plate. A smooth function f which satisfies the constraints can be derived by minimizing the energy functional,

$$ES(f) = \frac{1}{2}\left(\int\int_{\Omega}\left[\left(\frac{\partial^2 f}{\partial u^2}\right)^2 + 2\left(\frac{\partial^2 f}{\partial u \partial v}\right)^2 + \left(\frac{\partial^2 f}{\partial v^2}\right)^2\right] du dv + \beta \sum_k \left(f(p_k) - t_k\right)^2\right).$$

Here, t_k is the value specified for a point p_k on Ω. If a function f minimizes the energy functional $ES(f)$, the first variational derivative of $ES(f)$ must vanish all over the domain Ω[14]. The condition can be represented by the expression,

$$\frac{\delta ES}{\delta f} = \frac{\partial^4 f}{\partial u^4} + 2\frac{\partial^4 f}{\partial u^2 \partial v^2} + \frac{\partial^4 f}{\partial v^4} + \beta\left(f(p_k) - t_k\right) = 0. \tag{13}$$

The last term containing β appears only at a point p_k on Ω for which a value t_k is specified.

4.2 Numerical solution

We discretize the domain Ω to an $M \times N$ regular grid and represent a function f by its values at the nodes on the grid. The standard finite difference approximation transforms Equation (13) into a system of linear equations which contains MN unknowns and MN equations. If the nodal variables comprising the function f are collected into the MN dimensional vector $\mathbf{f}$, the system may be written in a matrix form,

$$(\mathbf{A} + \beta \mathbf{I}')\mathbf{f} = \beta \mathbf{t}, \tag{14}$$

where $\mathbf{t}$ is an MN dimensional vector which contains the constraints on the values of $\mathbf{f}$. Equation (14) can be efficiently solved by the multigrid relaxation method given in Section 5.

Figure 2 shows a surface example in which the grid size is 64×64. In the figure, black spots represent the interpolated values. It takes 0.4 seconds on a SGI Crimson to generate the surface by the multigrid relaxation method. When the size of the grid is 512×512, its computation time is 5.0 seconds. This shows that the multigrid relaxation method is efficient enough for interactive use.

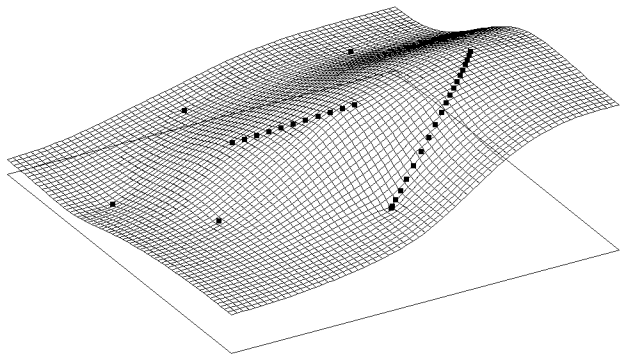


Figure 2: A surface example

4.3 Generation of transition functions T_0 and T_1

To control the transition behavior in a metamorphosis, an animator selects a set of points on an image and specify a transition curve for each point. The point set is not necessarily the same as the point set used for deriving warp functions. A transition curve gives the transition behavior of a point over time as shown in Figure 3.

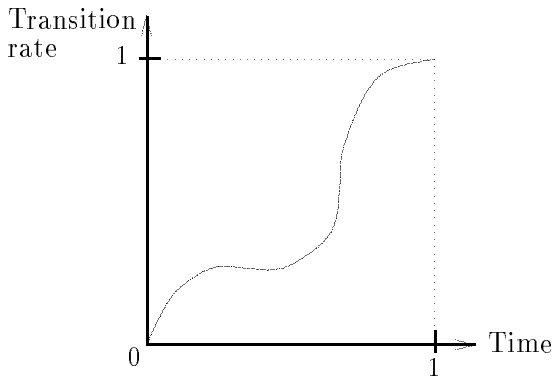


Figure 3: A transition curve

Let P be a set of points on the source image I_0 for which transition curves are specified. Let $C(p_k)$ be the transition curve for a point p_k in P. For a given time t, the transition function $T_0(t)$ should have the transition rate $C(p_k; t)$ at each point p_k in P. With the set of $C(p_k; t)$ as the constraints on values, such a transition function can be derived by the surface model described in Section 4.1. The resulting transition function is C^1-continuous and properly propagates the specified transition curves all over the image I_0.

The transition function T_1 is specified on the destination image I_1 and should give the transition behavior which is the same with T_0. If a point p on I_0 corresponds to a point q on I_1, the transition rate $T_1(q;t)$ should be the same with $T_0(p;t)$ for each time t. Hence, the transition function $T_1(t)$ can be derived by sampling $T_0(t)$ with the warp function W_1, that is, $T_1(q;t) = T_0(W_1(q);t)$.

When a sequence of inbetween images is generated with transition functions, a surface should be constructed for determining the function $T_0(t)$ at each time t. In this case, the solution for a surface is used for the initial solution of the surface at the next time step. Because the surfaces change smoothly with time, this approach provides a good initial solution and reduces the computation time.

5 Multigrid Relaxation Method

Equations (11), (12), and (14) are linear systems in the form,

$$\mathbf{A}\mathbf{f} \;=\; \mathbf{b},$$

where $\mathbf{A}$ is an $MN \times MN$ matrix, $\mathbf{f}$ is an MN dimensional unknown vector, and $\mathbf{b}$ is an MN dimensional vector. Due to the local nature of a finite difference discretization, $\mathbf{A}$ has the nice computational properties such as sparseness and bandedness.

Many types of algorithms have been developed for solving a sparse linear system. Relaxation algorithms such as Jacobi, Gauss-Seidel, or successive-overrelaxation methods exploit the sparseness and bandedness of the matrix $\mathbf{A}$ to efficiently solve the system of equations[15]. In a relaxation, the value of each node is updated with a local computation to satisfy the equation for that node. The iteration of relaxations generates a sequence of approximate solutions which converge asymptotically to the exact solution.

A major drawback of a relaxation scheme is that it converges slowly in general. The multigrid approach was developed to overcome the drawback and has been actively researched by the numerical analysis community[22, 23]. Terzopoulos first applied the multigrid approach to derive a thin-plate surface for solving the visual surface reconstruction problem in computer vision[21].

The multigrid approach applies the ideas of nested iteration and coarse grid correction to a hierarchy of grids[22]. The nested iteration is the way to improve a relaxation scheme with a good initial guess. To obtain an improved initial guess, the nested iteration performs preliminary iterations on a coarse grid and then uses the resulting approximation as an initial guess on a fine grid. Relaxations on a coarser grid are less expensive since there are fewer unknowns to be updated.

The coarse grid correction accelerates the speed of convergence based on an analysis of the error reduction behavior. The analysis shows that the high frequency components of an error are short-lived while its low frequency components persist through many iterations. The important point is that a low frequency on a fine grid may turn into a high frequency on a coarse grid. When a relaxation begins to stall, signalling the predominance of low frequency errors, the coarse grid correction moves the relaxation to a coarser grid, on which those errors appear more oscillatory, and thus the relaxation will be more effective.

If $\mathbf{g}$ is an approximation to the exact solution $\mathbf{f}$, then the error $\mathbf{e} = \mathbf{f} - \mathbf{g}$ satisfies the residual equation,

$$\mathbf{A}\mathbf{e} = \mathbf{r} = \mathbf{b} - \mathbf{A}\mathbf{g}.$$

Once we get the error $\mathbf{e}$, the exact solution $\mathbf{f}$ can be immediately derived by $\mathbf{f} = \mathbf{g} + \mathbf{e}$. The following recursive algorithm incorporates the idea of coarse grid correction with the residual equation by relaxing the error on a hierarchy of grids. In the algorithm, the superscript h denotes the inter-node spacing of a grid. The

matrix $\mathbf{A}^h$ on a grid Ω^h is the approximation of the matrix $\mathbf{A}$ on the finest grid. The vectors $\mathbf{f}^h$, $\mathbf{g}^h$, and $\mathbf{b}^h$ comprise the corresponding nodal variables on the grid Ω^h. I_h^{2h} denotes the decimation of a vector from a finer grid Ω^h to a coarser grid Ω^{2h} while I_{2h}^h is the interpolation in the opposite direction. ν_1 and ν_2 are the parameters for controlling the number of relaxations.

V-Cycle Algorithm
$\mathbf{g}^h \leftarrow MV^h(\mathbf{g}^h, \mathbf{b}^h)$
1. Relax ν_1 times on $\mathbf{A}^h\mathbf{f}^h = \mathbf{b}^h$ with a given initial guess $\mathbf{g}^h$.
2. If Ω^h is the coarsest grid, then go to 4.
   Else $\mathbf{b}^{2h} \leftarrow I_h^{2h}(\mathbf{b}^h - \mathbf{A}^h\mathbf{g}^h)$
   $\mathbf{g}^{2h} \leftarrow 0$
   $\mathbf{g}^{2h} \leftarrow MV^{2h}(\mathbf{g}^{2h}, \mathbf{b}^{2h})$.
3. Correct $\mathbf{g}^h \leftarrow \mathbf{g}^h + I_{2h}^h\mathbf{g}^{2h}$.
4. Relax ν_2 times on $\mathbf{A}^h\mathbf{f}^h = \mathbf{b}^h$ with initial guess $\mathbf{g}^h$.

The algorithm telescopes down to the coarsest grid and then walks its way back to the finest grid. Figure 4(a) shows the schedule for the grids in the order in which they are visited. Because of the pattern of this diagram, this algorithm is called the V-Cycle.

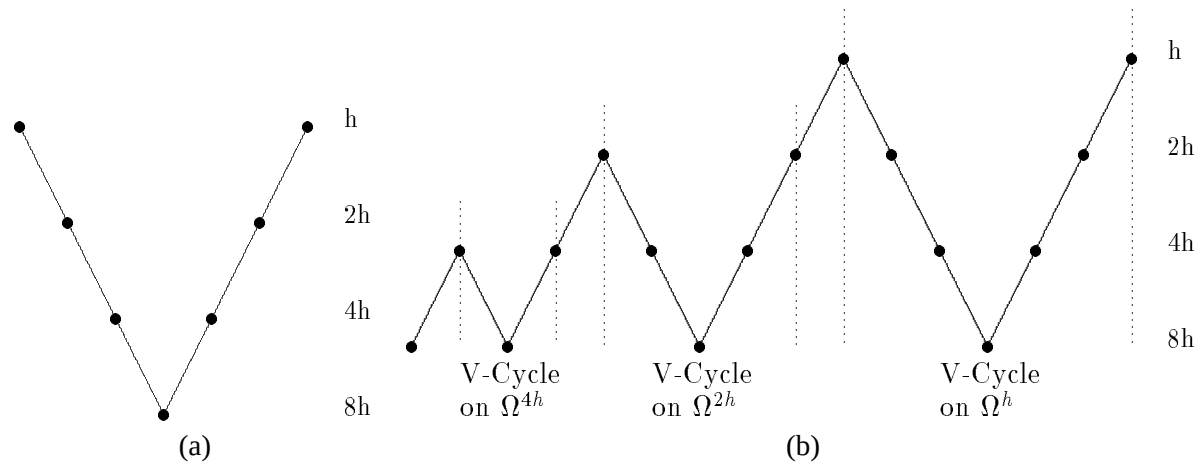


Figure 4: Relaxation schedules for (a) V-Cycle (b) Full Multigrid V-Cycle, all on four levels

The idea of nested iteration can enhance the V-Cycle algorithm by using a hierarchy of grids to provide an improved initial guess on the finest grid. The following recursive algorithm shows the final multigrid relaxation scheme which incorporates the ideas of nested iteration and coarse grid correction on a hierarchy of grid.

Full Multigrid V-Cycle Algorithm
$\mathbf{g}^h \leftarrow FMV^h(\mathbf{g}^h, \mathbf{b}^h)$
1. If Ω^h is the coarsest grid, then go to 3.
   Else $\mathbf{b}^{2h} \leftarrow I_h^{2h}(\mathbf{b}^h - \mathbf{A}^h\mathbf{g}^h)$
   $\mathbf{g}^{2h} \leftarrow 0$

$\mathbf{g}^{2h} \leftarrow FMV^{2h}(\mathbf{g}^{2h}, \mathbf{b}^{2h})$.
2. Correct $\mathbf{g}^h \leftarrow \mathbf{g}^h + I_{2h}^h \mathbf{g}^{2h}$.
3. $\mathbf{g}^h \leftarrow MV^h(\mathbf{g}^h, \mathbf{b}^h)$.

The Full Multigrid V-Cycle algorithm starts at the finest grid. Figure 4(b) shows the scheduling of relaxations on grids. Each V-Cycle is preceded by a smaller V-Cycle designed to provide a better initial guess.

Schemes for relaxation and interpolation are required to implement the Full Multigrid V-Cycle algorithm. The Gauss-Seidel method is always twice superior to Jacobi method in the speed of convergence. The successive-overrelaxation method shows unstable approximations in the first few iterations though it is faster than the others. Because we take small number of relaxations in the V-Cycle algorithm, the Gauss-Seidel method is chosen as the relaxation scheme.

When a vector $\mathbf{v}$ is interpolated from a coarser grid Ω^{2h} to a finer grid Ω^h, we use the Catmull-Rom spline interpolation[24] which guarantees C^1-continuity. The decimation of vector $\mathbf{v}$ from Ω^h to Ω^{2h} is done by weighted averaging the values of neighborhood nodes. That is,

$$\begin{aligned} v_{i,j}^{2h} &= \tfrac{1}{16}[v_{2i-1,2j-1}^h + v_{2i-1,2j+1}^h + v_{2i+1,2j-1}^h + v_{2i+1,2j+1}^h \\ &\quad + 2(v_{2i,2j-1}^h + v_{2i,2j+1}^h + v_{2i-1,2j}^h + v_{2i+1,2j}^h) + 4v_{2i,2j}^h], \end{aligned} \tag{15}$$

where $v_{i,j}$ denotes the value of the vector $\mathbf{v}$ at the node (i, j).

When the relaxation is performed on a coarse grid Ω^{2h}, the matrix $\mathbf{A}^{2h}$ should be approximated from the matrix $\mathbf{A}^h$ on the fine grid Ω^h. A row in the matrix $\mathbf{A}^h$ contains the coefficients of an equation which is solved for a nodal variable on Ω^h in a relaxation. For a node on Ω^{2h}, we derive the row in the matrix $\mathbf{A}^{2h}$ by taking the weighted average of the related rows in the matrix $\mathbf{A}^h$. Each coefficient for a node on Ω^{2h} is computed by applying formula (15) to the corresponding coefficients for its neighborhood nodes on Ω^h.

In the general multigrid approach, the convergence rate and error analysis are performed to determine the number of relaxations on a grid[23]. The relaxation on a grid moves to a coarser grid whenever the convergence rate slows down and ends as soon as the estimated error is less than a given bound. It has been theoretically proven that the multigrid approach requires $O(MN)$ operations to reduce the error to the truncation error level[22].

We implemented the Full Multigrid V-Cycle algorithm in which the parameters ν_1 and ν_2 control the number of relaxations. The computational efforts can be calculated in terms of the work unit W which is defined as the amount of computation required for one relaxation on the finest grid. The necessary computational effort is less than $7/2(\nu_1 + \nu_2)W$[22]. Because small ν_1 and ν_2 are usually sufficient for deriving a satisfactory solution, this is a great enhancement compared to the conventional relaxation schemes.

6 Extensions

6.1 Fast approximation of a warp function

In this paper, a warp function $\mathbf{w}$ is derived by numerically solving Equation (7). When the parameter α is zero, Equation (7) reduces to the linear system,

$$(\mathbf{A} + \beta \mathbf{I}')\mathbf{w} = \beta \mathbf{q}. \tag{16}$$

Equation (16) can be rapidly solved by deriving the x- and y-components of $\mathbf{w}$ with the multigrid relaxation method given in Section 5. When images are not heavily distorted, Equation (16) can be used for generating

smooth warp functions which exactly reflect the feature correspondences[18]. However, the warp functions may not be one-to-one especially when they contain large local distortions as in Figure 1.

6.2 A warp function with a fixed boundary

When a warp function is applied to an image, there may be holes near a boundary of the distorted image. These holes are generated if the boundary of the original image is mapped to the inside of the distorted image. The problem may be overcome by separating background from objects. However, if an object is attached to a boundary, the boundary need to be frozen over the metamorphosis.

A frozen boundary can be obtained by adjusting the boundary condition in computing a warp function. For the horizontal boundaries, we make the y-component of a warp function $\mathbf{w}$ equal to that of the undeformed shape $\mathbf{r}$. Then, the points on the boundaries can move only in the horizontal directions. The vertical boundaries can be handled similarly.

6.3 General feature control primitives

The multigrid relaxation method is used to solve a linear system in generating a warp function. The number of positional constraints hardly affects the computation time required in the multigrid relaxation method. Hence, the total computation time remains nearly constant regardless of the number of feature points. This strong merit makes it possible to easily extend the feature control primitives to include line segments and curves.

When a pair of line segments are specified to establish the correspondence of features between images, a discretization of the line segments generates a set of corresponding point pairs. When curves are used to control feature correspondences, the Catmull-Rom spline curves[25] are adopted to interpolate the points specified by an animator. By properly discretizing the parameter space and computing the points on the curves, we get a set of corresponding points lying on the matching curves. Theses generalized primitives can also be used for controlling transition functions.

6.4 Procedural transition functions

The presented image morphing technique always generates an inbetween image on which the feature point pairs on two images match their positions regardless of transition functions. Therefore, procedural transition functions can be used to generate various interesting inbetween images. For example, let the transition function T_0 be defined by

$$T_0(u, v; t) = \begin{cases} 2t(1 - u/u_{max}), & \text{if } 0 \leq t \leq \frac{1}{2}, \\ 1 - 2(1 - t)u/u_{max}, & \text{if } \frac{1}{2} < t \leq 1. \end{cases}$$

T_0 generates a sequence of inbetween images in which the source image gradually changes to the destination image from left to right. The corresponding transition function T_1 is derived by sampling T_0 with the warp function W_1.

7 Experimental Results

7.1 Comparison with field morphing

Mesh warping[2] provides a fast and intuitive technique for deriving warps and can be easily supported by hardwares. The method of Nishita *et al.*[3] can produce various types of warps which are smooth up to the desired degree. However, these mesh-based methods have a drawback in specifying the features on an image. The disadvantages of mesh warping are fully detailed in [1], most of which are also applied to the method of Nishita *et al*.

Field morphing[1] is comparable to the warp generation technique in this paper in that the features can be effectively specified by line segments. In this section, examples of warps are provided which show the advantages of the proposed technique over field morphing. To generate a warp with the proposed technique, the points on line segments are sampled as mentioned in Section 6.3.

Figure 5(a) is the original image in which a letter 'F' lies on a mesh. We overlay line segments on the image and move them to obtain distorted images. Figure 5(b) is generated when the warp is computed by the field morphing technique. In the image, the lower bar in 'F' does not shrink in the amount specified by the movements of line segments. The right end of the upper bar shows a distortion while the line segment on it is fixed. These abnormal distortions result from that the effects of two or more line segments are blended by simple weighted averaging. In Figure 5(c), the warp is computed by the technique in this paper. Figure 5(c) exactly reflects the movements of line segments and shows proper distortions over the entire image.

In Figures 5(d) and 5(e), the letter 'F' in Figure 5(a) is distorted to obtain the letter 'T'. Field morphing generates the image in Figure 5(d), which does not show the desired distortion. In the image, the influences of line segments crumble each other, and the displacement of any line segment is not properly reflected. In contrast, the proposed technique gives the exact distortion as shown in Figure 5(e). Figures 5(f) and 5(g) show the images obtained when the image in Figure 5(a) is distorted to the letter 'P'.

An image in Figure 5 is of size 460×460 and it takes 19.1 seconds for the proposed technique to generate a distorted image on a SGI Crimson. The field morphing technique requires 51.95 seconds to generate a distorted image. When the number of line segments gets larger, the computation time increases in field morphing while the time remains nearly constant in the proposed technique.

7.2 Metamorphosis examples

Figure 6 shows a metamorphosis example. Figure 6(a) is a face image of the first author of this paper, and Figure 6(b) is an image of a cat. Figures 6(c) and 6(d) show the features specified on the images. Figure 6(e) is the middle image in which the same transition rate is applied to all parts. Figure 6(f) is an inbetween image in which transition rates are different from part to part. The eyes, nose, and mouth of the image look more like the human face than the remaining parts. Figures 6(g) and 6(h) are examples of applying procedural transition functions explained in Section 6.4. $T_0(u, v; t) = u/u_{max}$ and $T_0(u, v; t) = (\sin(4\pi u/u_{max}) + 1)/2$ are used for Figures 6(g) and 6(h), respectively.

In Figure 7, an inconsistency of features between images is overcome by controlling the transition behavior. Figures 7(a) and 7(b) show face images which are considerably different near the ears. We specify the correspondence of features in Figures 7(c) and 7(d). Without transition control, the ears and hair are jumbled up in the middle image as shown in Figure 7(e). To obtain a better inbetween image, we select the parts near the ears in Figure 7(f) and assign the transition curve in Figure 7(i). The transition curve in Figure 7(j) is specified for the line segments in the middle of Figure 7(f). Figure 7(g) shows the inbetween image at time 0.47 where the transition rates are computed from the specified transition curves. In the image, the parts near

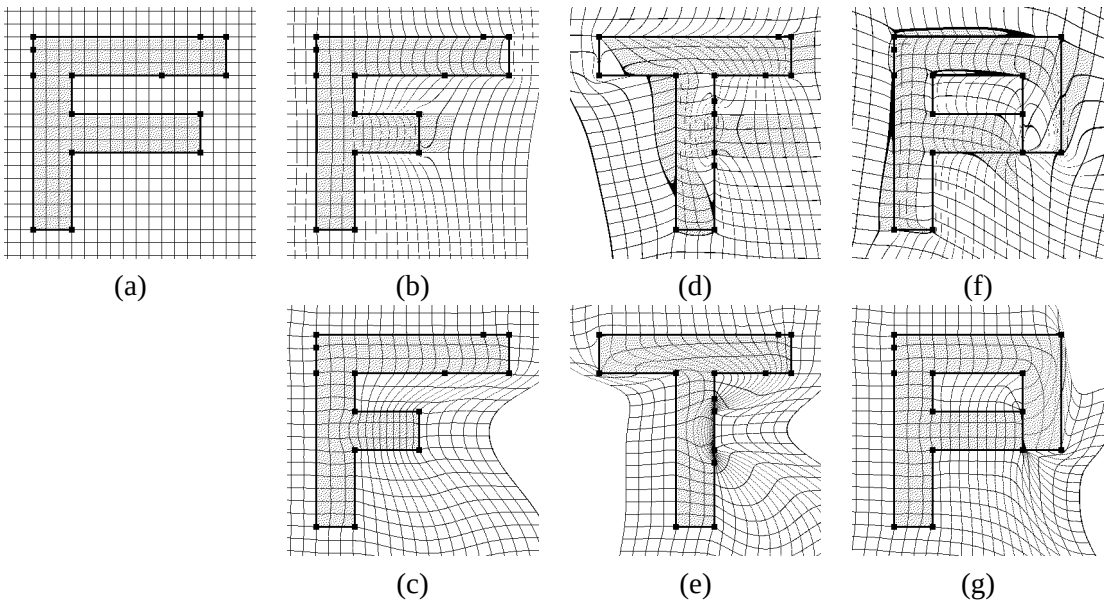


Figure 5: Comparison of warps: (a) is the original image; (b), (d), and (f) are from field morphing; (c), (e), and (g) are from the proposed technique

the ears resemble the first image. Figure 7(h) is the inbetween image at time 0.53 on which the second image dominates the parts near the ears.

In Figure 8, two different facial expressions of a person are interpolated to obtain a facial animation. The second images in the upper and lower rows are the source and destination images, respectively. The first images in the rows show the specified feature correspondence. The others are generated inbetween images. They demonstrate that the features are nicely controlled by the proposed technique. For example, the motion of the mouth looks natural in the inbetween images.

7.3 Performance

We use a workstation SGI Crimson(R4400) to generate the examples in this paper. The resolution of an image in Figure 6 is 475×435. It takes 19.2 seconds to derive a warp on a 475×435 grid. Hence, about 38.4 seconds are taken to generate the image in Figure 6(e). When the transition rates are different from part to part in an inbetween image, a deformable surface should be constructed to compute transition functions. It takes 3.7 seconds to generate the surface on a 475×435 grid. Then, about 42.1 seconds are taken to obtain the image in Figure 6(f). Except the evaluation of procedural transition functions, the computation required for the images in Figures 6(g) and 6(h) is the same as that for the image in Figure 6(e).

An image in Figure 7 is 516×387, and it takes 18.1 seconds and 3.7 seconds to derive a warp and surface, respectively. Each image in Figure 8 is 449×423, and it takes 17.2 seconds to generate a warp. Once the warps are derived, an inbetween image can be obtained in less than one second.

All metamorphosis examples in this section are directly derived from the given images. An inbetween image is generated without a masking process which extracts objects from the background. No additional manipulations are taken to enhance the generated inbetween images. To prevent holes near a boundary of a

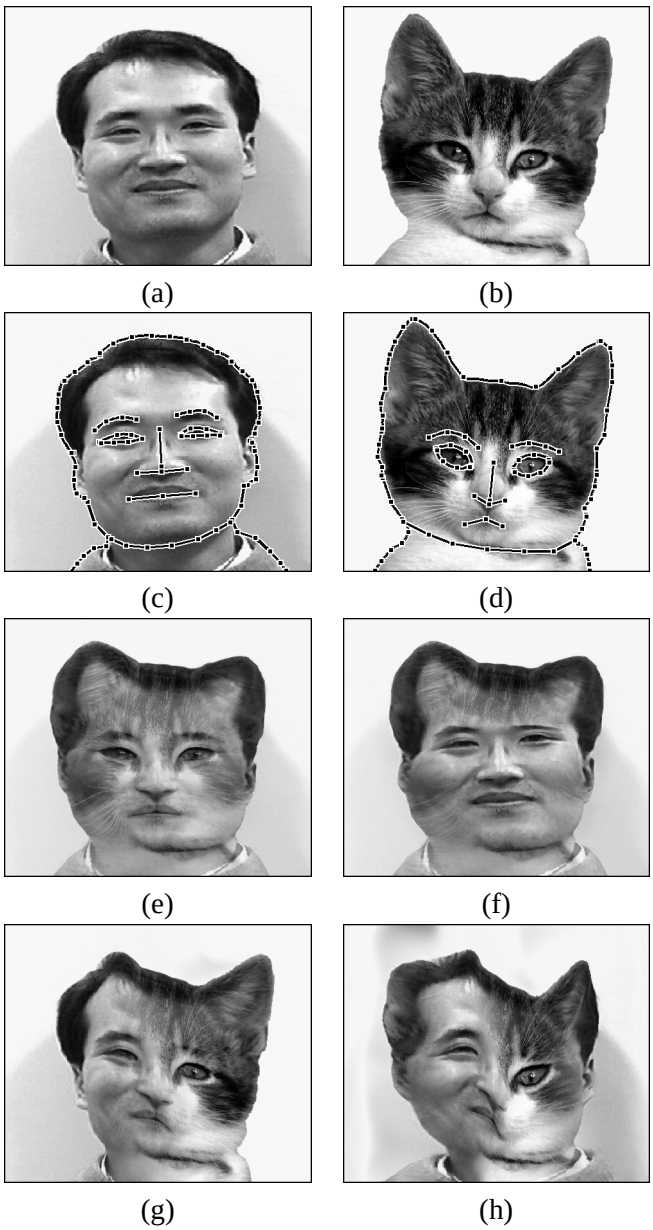


Figure 6: A metamorphosis example: from a person to a cat

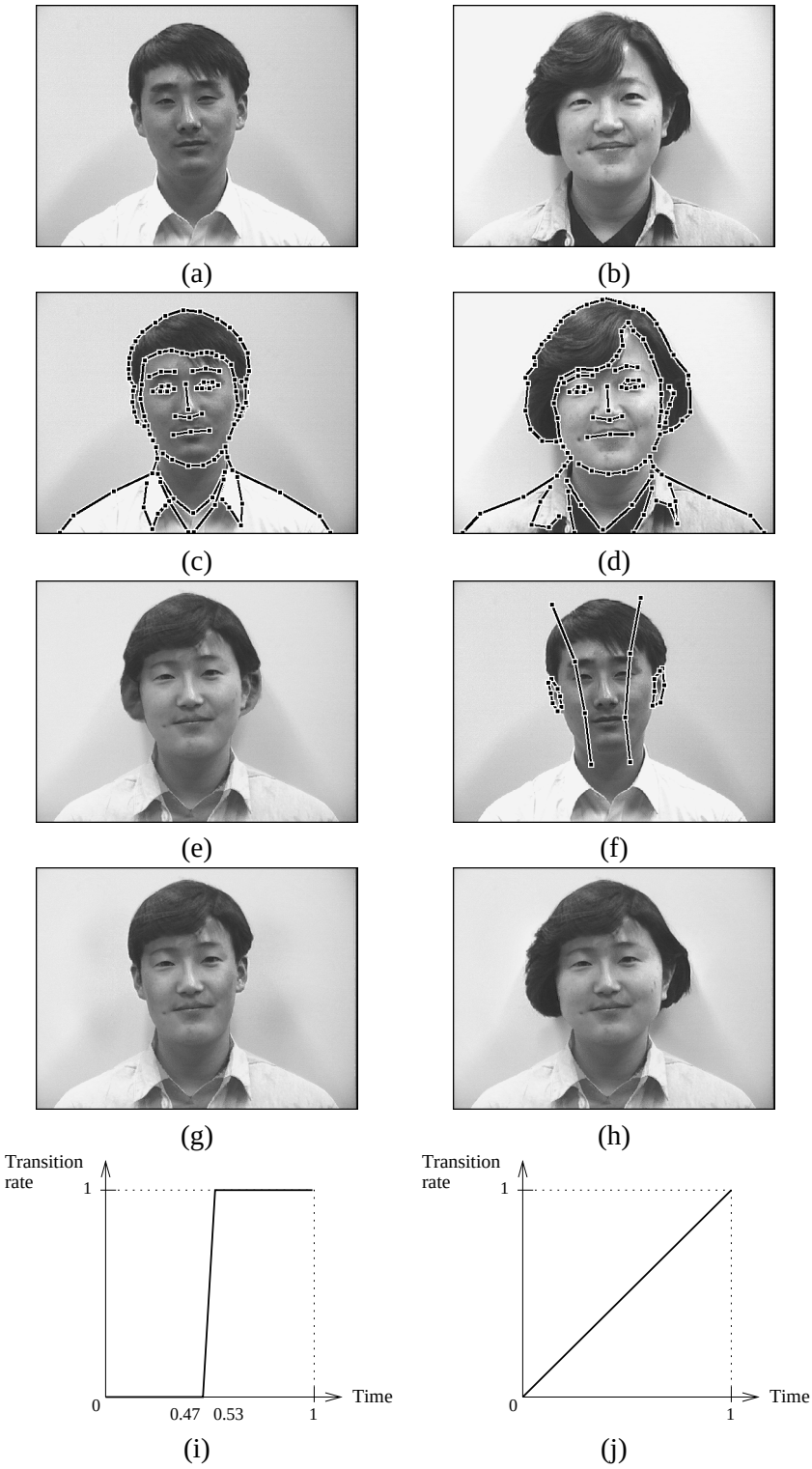


Figure 7: Transition control for overcoming an inconsistency between features

Figure 8: A facial animation

distorted image, a warp function is computed by freezing the boundary as described in Section 6.2.

General primitives of lines and curves mentioned in Section 6.3 enable an animator to efficiently specify the feature correspondence between two images. In deriving an inbetween image, the positions of features are repeatedly adjusted until the desired metamorphosis is obtained. Because the warp computed in this paper precisely reflects the specified feature correspondence, the iterative process can be successfully completed in a few trials. Each metamorphosis example in this section was generated from the given images in less than one hour.

8 Conclusions

This paper presents a new approach to image morphing in deriving a warp and controlling transition behavior. We develop a two dimensional deformation technique to generate a warp from a set of feature point pairs overlaid on two images. The resulting warp is C^1-continuous and one-to-one and precisely reflects the feature correspondence between the images. Because any structure such as a mesh is not necessary for feature specification, an animator enjoys freedom in designing a metamorphosis. The freedom together with good warps make it possible to obtain a desired inbetween image very effectively.

We separate the transition behavior control from the feature interpolation in generating a metamorphosis sequence. The separation results in a method which is much easier to use and more effective than the previous techniques. A transition scenario is realized by specifying the transition curves for selected points on an image. In addition, more interesting transition behaviors can be derived by procedural transition functions.

The multigrid relaxation method is taken to solve a linear system in deriving a warp or transition rates. This method shows a great enhancement in computation time compared to the conventional relaxation schemes. With the stable numerical method for a differential equation and the multigrid relaxation method, the presented image morphing technique is fast enough for an interactive environment.

The most tedious part of image morphing is to establish the correspondence of features between images by an animator. Techniques of computer vision may be employed to automate this task. An edge detection algorithm can provide important features on images, and an image analysis technique may be used to find the correspondence between detected features. One of the most challenging problem in image morphing is to develop an efficient method for specifying features and their correspondence, especially in the morphing between two image sequences.

References

[1] T. Beier and S. Neely. Feature-based image metamorphosis. *Computer Graphics*, 26(2):35–42, 1992.

[2] G. Wolberg. *Digital Image Warping*. IEEE Computer Society Press, 1990.

[3] T. Nishita, T. Fujii, and E. Nakamae. Metamorphosis using Bézier clipping. In *Proceedings of the First Pacific Conference on Computer Graphics and Applications*, pages 162–173, Seoul, Korea, 1993. World Scientific Publishing Co.

[4] D. Terzopoulos, J. Platt, A. Barr, and K. Fleischer. Elastically deformable models. *Computer Graphics*, 21(4):205–214, 1987.

[5] D. Terzopoulos and K. Fleischer. Modeling inelastic deformation: Viscoelasticity, plasticity, fracture. *Computer Graphics*, 22(4):269–278, 1988.

[6] J. C. Platt and A. H. Barr. Constraint methods for flexible models. *Computer Graphics*, 22(4):279–288, 1988.

[7] Silicon Graphics Inc. *Graphics Library Programming Guide*.

[8] G. Celniker and D. Gossard. Deformable curve and surface finite-elements for free-form shape design. *Computer Graphics*, 25(4):257–266, 1991.

[9] W. Welch and A. Witkin. Variational surface modeling. *Computer Graphics*, 26(2):157–166, 1992.

[10] D. Terzopoulos. Regularization of inverse visual problems involving discontinuities. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, PAMI-8(4):413–424, 1986.

[11] G. Meisters and C. Olech. Locally one-to-one mappings and a classical theorem on schlicht functions. *Duke Mathematical Journal*, 30:63–80, 1963.

[12] R. C. Buck. *Advanced Calculus*. McGraw-Hill, third edition, 1978.

[13] M. P. do Carmo. *Differential Geometry of Curves and Surfaces*. Academic Press, second edition, 1990.

[14] I. Gelfand and S. Fomin. *Calculus of Variations*. Prentice-Hall, 1963.

[15] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes in C*. Cambridge University Press, second edition, 1992.

[16] D. Ruprecht and H. Müller. Image warping with scattered data interpolation methods. Research Report 443, Fachbereich Informatik der Universität Dortmund, 44221 Dortmund, Germany, 1992.

[17] F. L. Bookstein. Principal warps: Thin-plate splines and the decomposition of deformations. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 11(6):567–585, 1989.

[18] S. Y. Lee, K. Y. Chwa, J. Hahn, and S. Y. Shin. Image morphing using deformable surfaces. In *Proceedings of Computer Animation '94*, pages 31–39, Geneva, Switzerland, 1994. IEEE Computer Society Press.

[19] P. Litwinowicz and L. Williams. Animating images with drawings. In *SIGGRAPH 94 Conference Proceedings*, pages 409–412. ACM Press, 1994.

[20] W. Grimson. An implementation of a computational theory of visual surface interpolation. *Computer Vision, Graphics, and Image Processing*, 22:39–69, 1983.

[21] D. Terzopoulos. Multilevel computational processes for visual surface reconstruction. *Computer Vision, Graphics, and Image Processing*, 24:52–96, 1983.

[22] W. L. Briggs. *A Multigrid Tutorial*. SIAM, Lancaster Press, Lancaster, PA, 1987.

[23] A. Brandt. Multi-level adaptive solutions to boundary-value problems. *Mathematics of Computation*, 31(138):333–390, 1977.

[24] D. H. Kochanek and R. H. Bartels. Interpolating splines with local tension, continuity, and bias control. *Computer Graphics*, 18(3):33–41, 1984.

[25] G. Farin. *Curves and Surfaces for Computer Aided Geometric Design*. Academic Press, second edition, 1990.