A Work Minimization Approach to Image Morphing

Peisheng Gao

Thomas W. Sederberg
tom@cs.byu.edu

A Work Minimization Approach to Image Morphing

Peisheng Gao
Thomas W. Sederberg
Department of Computer Science, Brigham Young University
Provo, Utah 84602

Abstract

An algorithm is presented for morphing two images, often with little or no user interaction. For two similar images (such as different faces against a neutral background) the algorithm generally can create a pleasing morph completely automatically.

The algorithm seeks the minimum work to deform one image into the other, where work is a function of the amount of warping and re-coloration. A hierarchical method for finding a minimal work solution is invoked. Anchor point constraints are satisfied by imposing penalties on deformations that disobey these constraints. Good results can be obtained in less than ten seconds for 256 × 256 images.

Key Words: Morphing, image registration, shape blending.

1 Introduction

The term *morph*, short for *metamorphosis*, has been applied to various computer graphics methods for smoothly transforming between geometric models or images. In this paper, morph refers to image metamorphosis—the eye-catching visual technique wherein, given two images I_1 and I_2, a succession of intermediate images is created that depicts a continuous transformation from I_1 to I_2, as illustrated in Figure 1.

Morphing is accomplished by simultaneously warping (deforming) I_1 and I_2 and cross-dissolving. Those general concepts, and other considerations involved in image morphing, are well documented elsewhere (see, for example, Wolberg 1990; Beier and Neely 1992; Lee et al. 1995; Beier et al. 1997) and this paper assumes familiarity with that background.

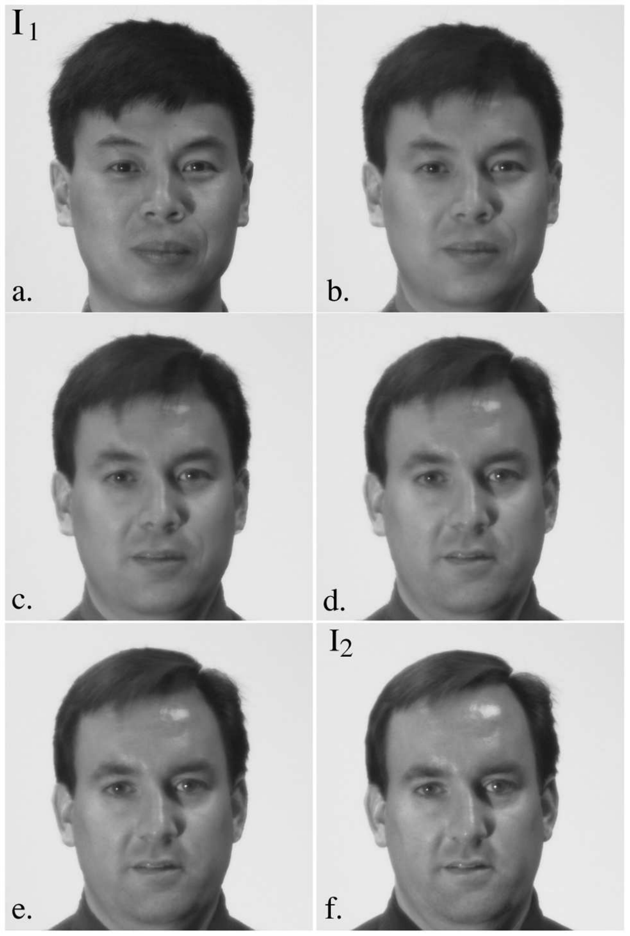

Figure 1: Morph $(1-t)I_1 + tI_2$. b) $t = 0.2$, c) $t = 0.4$, d) $t = 0.6$, e) $t = 0.8$.

The best-looking morphs result from warps that successfully align corresponding features. The manual specification of such warps can be time consuming since a user must input pairs of points, lines, or curves that define correspondences between I_1 and I_2 (Beier and Neely 1992; Burns 1994; Lee et al. 1995; Beier et al. 1997).

This paper presents an algorithm that can determine a warp suitable for morphing two images with little or no human guidance. The algorithm is based on a work minimization strategy that derives its cost directly from the images, not from user-specified constraints. The warp can be computed reasonably quickly: the example in Figure 1 was generated in nine seconds with no user interaction on an HP 9000/780/J282 workstation.

Section 2 briefly reviews related research. Section 3 discusses the warp definition. Section 4 presents the function used to assess warp work. Section 5 explains the approach taken to optimize the work. Section 6 suggests some ways for a user to guide the algorithm. Section 7 presents several examples that help substantiate the merit of this algorithm.

2 Prior research

Morphing originated as a cinematic technique as early as 1904 (Bordwell and Thompson 1997). Digital image morphing—warp plus cross-dissolve—traces back to Tom Brigham at NYIT in the early 1980s (Beier and Neely 1992). The technique evolved in special effects studios, such as Industrial Light and Magic. Wolberg's detailed treatment of image warping includes a morphing algorithm in which the image warp is performed using a two-dimensional spline function (Wolberg 1990). In this setting, the user adjusts the warp by manipulating the spline coefficients. Beier and Neely introduced a user-friendly warp function and interface for specifying correspondences by means of lines drawn on the two key images (Beier and Neely 1992).

Energy minimization methods have previously been brought to bear on the morphing problem, but primarily for the purpose of constraint matching. For example, (Lee et al. 1996) uses a thin-plate model in obtaining a C^1 warp that satisfies feature specification from points, polylines, and curves. (Lee et al. 1995)uses snakes (Kass et al. 1988) —energy-minimizing splines—to expedite feature specification; those features are then constrained using a hierarchical free-form deformation.

The central problem studied in this paper—how to warp one image so that it "most closely" resembles a second image—is referred to in other settings as image registration or stereo matching. For example, medical imaging (Bookstein 1991) applies such algorithms to register images of parallel slices through organs, or to align geometric models of entire organs with a canonical model. In stereo matching (Brown 1992; Dhond and Aggarwal 1989; Weng et al. 1992), the two images to be registered are a stereo pair; registration enables parallax to be inferred and depth to be estimated. Stereo matching is a more straightforward problem than registration for image morphing, since it takes advantage of the epipolar constraint and the two images are slightly different views of the same scene.

The automatic image morphing algorithm presented in this paper found its motivation primarily in the solution to the polygon shape blending problem presented in (Sederberg and Greenwood 1992) (extended to shape blending of B-spline curves in (Sederberg and Greenwood 1995)). The effect of polygon shape blending is similar to image morphing.

In Figure 2, polygon 1 (a cow) and polygon 6 (a deer) are given. The shape blending algorithm in (Sederberg and Greenwood 1992) automatically computes the intermediate polygons 2–5, without requiring the user to specify any matching features. That algorithm models each given polygon as if it

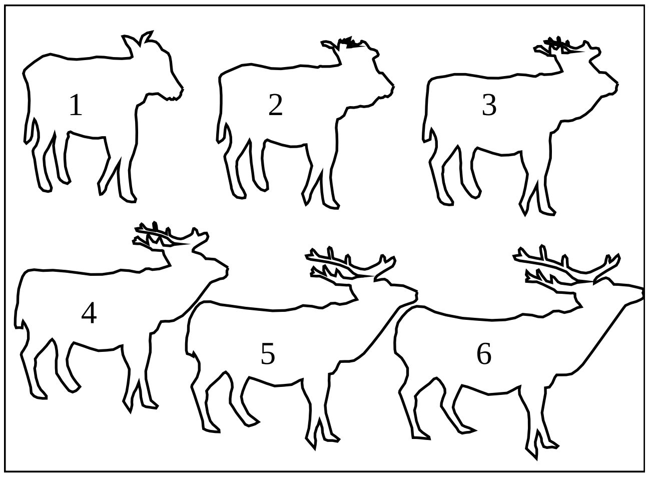

Figure 2: Cow-to-deer polygon shape blend.

were formed by bending a piece of wire. A legal shape blend is taken to be one that can be represented with the bending and stretching of wire—no cutting or splicing allowed—such that each vertex on one key polygon maps to a vertex on the other key polygon, and vice versa. The algorithm then selects the "best" shape blend as the one requiring the least amount of "work" to bend and stretch one polygon into the other.

The polygon shape blending algorithm must address two independent questions: What is the most suitable measure of work, and how can the least-work solution be found? In (Sederberg and Greenwood 1992), work equations are modeled after the work required to deform a piece of wire. Bending work and stretching work are computed independently. As in elastic stretching and bending of physical wire, the stretching work in the shape blend algorithm is proportional to the square of the change of length of each polygon edge, and bending work is proportional to the square of the angle change at each vertex.

For the polygon shape blend problem, the number of all legal shape blends is exponential in the number of vertices, which might make the task of finding a global minimal-work solution seem hopeless. Fortunately, a dynamic programming solution exists which can find the least-cost shape blend in $O(n^2 \log n)$ time, where n is the largest number of vertices on either key polygon. Unfortunately, this solution does not adapt to the image morphing problem, so a heuristic optimization method is presented in Section 5

3 Image Warping

The image warping phase of the image morphing process is typically performed using smooth functions: *mesh warping* techniques involve C^2 tensor-product cubic B-splines (Lee et al. 1995; Wolberg 1990) or C^∞ Bézier maps (Nishita et al. 1993), and *field morphing* methods (Beier and Neely 1992) are also generally C^∞.

We have chosen to use C^0 bi-linear uniform B-splines. Although the algorithm can work using B-splines of higher order (or other types of piecewise maps), we implemented a piecewise bi-linear function for the sake of speed. The morph algorithm spends the majority of its time sampling the color of mapped pixels, so the speed of the algorithm is roughly proportional to the speed of computing the warped location of a pixel. For our application, the use of bi-quadratic B-splines would be about five times slower than bi-linear. Furthermore, the goal of image warping is to align features as closely as possible. Many features—for example, a hair line or the silhouette of an article of clothing—are not particularly smooth. One can argue that a C^0 pixel-resolution map can be just as suitable in such instances as a C^2 map.

For the purposes of our discussion, we will assume that I_1 and I_2 are both square images of resolution $R \times R$. The generalization to the images being different sized rectangles is straightforward, but would needlessly clutter our notation.

The warp is defined hierarchically as follows. Impose an (x, y) coordinate system on I_1 with $(0, 0)$ in the lower left corner and $(1, 1)$ in the upper right corner. Denote by $F_i : \binom{x}{y} \to \binom{\tilde{x}}{\tilde{y}}$ a warp of I_1 using a $2^i \times 2^i$ grid of bi-linear B-spline patches:

$$F_i(x, y) = \sum_{j=0}^{2^i} \sum_{k=0}^{2^i} B_j(x) B_k(y) \mathbf{P}^i_{jk}, \tag{1}$$

where

$$B_j(x) = \begin{cases} 2^i x - j & \text{for } x \in [\frac{j}{2^i}, \frac{j+1}{2^i}] \\ j + 2 - 2^i x & \text{for } x \in [\frac{j+1}{2^i}, \frac{j+2}{2^i}] \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

and likewise for $B_k(y)$.

There are two reasons for using a hierarchical warp: it plays a crucial roll in the optimization method in Section 5, and it offers a simple control over the time/quality tradeoff, as discussed in Section 7.

F_i is onto if $\mathbf{P}^i_{00} = (0, 0)$, $\mathbf{P}^i_{2^i,0} = (1, 0)$, $\mathbf{P}^i_{0,2^i} = (0, 1)$ and $\mathbf{P}^i_{2^i,2^i} = (1, 1)$, and none of the other edge control points is moved from their assigned edge. F_i is one-to-one if all quadrilaterals $\mathbf{Q}^i_{jk} = \mathbf{P}^i_{j,k}$–$\mathbf{P}^i_{j+1,k}$–$\mathbf{P}^i_{j+1,k+1}$–$\mathbf{P}^i_{j,k+1}$, $j = 0, \ldots, 2^i - 1$, $k = 0, \ldots, 2^i - 1$ are convex and oriented counterclockwise. Figure 3 shows an example of a one-to-one F_2 warp.

The undisplaced, lattice positions of the control points are

$$\mathbf{P}^i_{jk} = (\frac{j}{2^i}, \frac{k}{2^i}). \tag{3}$$

The identity warp is obtained when each control point $\mathbf{P}^i_{jk}$ is in its lattice position (3).

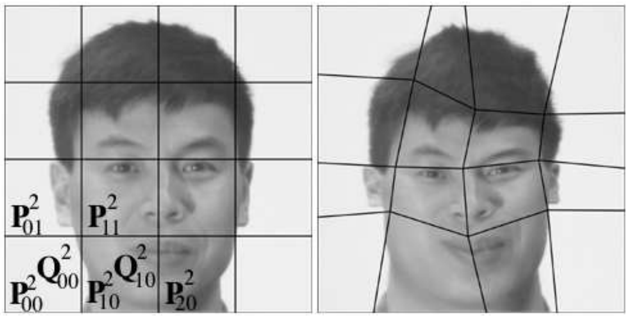

Figure 3: An F_2 warp with vertices $\mathbf{P}^2_{jk}$ and quadrilaterals $\mathbf{Q}^2_{jk}$.

An F_{i+1} warp (with control points $\mathbf{P}^{i+1}_{jk}$) is equivalent to an F_i warp (with control points $\mathbf{P}^i_{jk}$) if

$$\begin{aligned}\mathbf{P}^{i+1}_{jk} &= \frac{1}{4}\mathbf{P}^i_{\lfloor j/2\rfloor,\lfloor k/2\rfloor} + \frac{1}{4}\mathbf{P}^i_{\lfloor (j+1)/2\rfloor,\lfloor k/2\rfloor} + \\ &\quad \frac{1}{4}\mathbf{P}^i_{\lfloor j/2\rfloor,\lfloor (k+1/2)\rfloor} + \frac{1}{4}\mathbf{P}^i_{\lfloor (j+1/2)\rfloor,\lfloor (k+1/2)\rfloor}\end{aligned} \tag{4}$$

for all $j, k \in [0, \ldots 2^{i+1}]$.

Denote by $c_1(x, y)$ the pixel color at location (x, y) in I_1. If (x, y) lies on a boundary between two pixels, use the color of pixel to the North of a horizontal boundary, or to the East of a vertical boundary. Denote by $c_2(F_i(x, y))$ the pixel color on I_2 of a point (x, y) mapped by F_i. Denote by $F_i \circ I_1$ the result of applying warp F_i on I_1, without resampling.

4 Work equations

Motivated by the polygon shape blending algorithm (Sederberg and Greenwood 1992) reviewed in Section 2, our hope is to find some measure of "work" that can determine how easily a morph can be performed. We will use the terms *cost* and *work* interchangeably.

Model I_1 as a physical painting of discrete pixels on a deformable medium. Consistent with the choice of C^0 bi-linear B-splines for the warp equation, we might think of the deformable medium as a grid of $2^i \times 2^i$ distinct squares $\mathbf{Q}^i_{jk}$ made of an idealized elastic material. Each square $\mathbf{Q}^i_{jk}$ has vertices $\mathbf{P}^i_{jk}$,$\mathbf{P}^i_{j+1,k}$, $\mathbf{P}^i_{j+1,k+1}$, $\mathbf{P}^i_{j,k+1}$. The deformed square is denoted $F_i \circ \mathbf{Q}^i_{jk}$, as shown in Figure 3.

The process whose work we want to measure has two parts: warp I_1 so that it looks like I_2 as much as possible, then modify the colors of $F_i \circ I_1$ so that it is identical to I_2.

In order to physically impose a warp F_i on this grid, one must deform each square by moving each of its four corners to their assigned locations, and then perhaps stick pins through the corners to hold the deformed squares in place. The "rubber" is thick enough that it won't buckle under compression, so it takes work to shrink a square as well as to stretch it.

This warp of I_1 should ideally align its features quite well with I_2, but generally $F_i \circ I_1$ will not be identical to I_2. To make them the same, we need to visit each pixel of I_2 and possibly repaint it to some degree. The more different the colors, the more work is needed to bring them into agreement. Since we have not resampled $F_i \circ I_1$, a pixel of I_2 might be covered by a mosaic of pieces of mapped pixels from $F_i \circ I_1$, and each segment of that mosaic involves a different amount of re-coloration work. Since those mosaics can be complicated, we estimate the re-coloration cost by sampling (see Section 4.1).

The work to transform I_1 into I_2 using a given warp F_i is computed by summing the warping and re-coloration work for each rubber-like square:

$$W_{total} = \sum_{j=0}^{2^i-1}\sum_{k=0}^{2^i-1} c_c\left(\frac{256}{R}\right)^2 W^c_{jk} + c_s\frac{400}{256^2}W^s_{jk} + c_a\frac{600}{256^2}W^a_{jk} \tag{5}$$

where W^a_{jk} is work due to angle-change, W^s_{jk} is work due to stretching, and W^c_{jk} is the re-coloration work for the square $\mathbf{Q}^i_{jk}$. The coefficients c_a, c_s, and c_c control the relative influence that these components exert on the total cost. The constants $\left(\frac{256}{R}\right)^2$, $\frac{400}{256^2}$, and $\frac{600}{256^2}$ are, admittedly, unattractive; they were derived empirically so that, in most cases, choosing c_c, c_s, and c_a the all be one (or, in general, to vary between zero and ten) will yield good results. Division by R in $\left(\frac{256}{R}\right)^2$ assures that the coefficient c_c is independent of image resolution; otherwise, the morph result for a pair of images at one resolution would differ from the result of the same two images at different resolution. Section 7 discusses appropriate values for those coefficients. One additional work component, used for optional anchor points, is discussed in Section 6.1.

We compute the values of W^c, W^s, and W^a as follows.

4.1 Re-coloration cost, W^c

The most important cost component is the re-coloration cost. It also consumes a large majority of the total algorithm time, so it is crucial that this computation be fast. We have experimented with several different re-coloration cost functions and have found that the following simple formula generally works well:

$$W^c_{jk} = \sum_{\bar{\mathbf{P}} \in \mathbf{Q}^i_{jk}} d(c_1(\bar{\mathbf{P}}), c_2(F_i(\bar{\mathbf{P}}))), \qquad (6)$$

with $\bar{\mathbf{P}}$ being the center of a pixel, and d being a function that computes the "distance" between two colors. The examples in this paper were computed taking d to be the Cartesian distance between the two colors in RGB space. In words, we obtain W^c_{jk} by sampling each pixel in I_1 whose center $\bar{\mathbf{P}}$ lies within $\mathbf{Q}^i_{jk}$. The distance d is found between that pixel's color, and the color of the pixel in I_2 that $\bar{\mathbf{P}}$ maps to. W^c_{jk} is the sum of those distances.

Taking d to be Cartesian distance in RGB space has yielded good results in the cases we have tested, although those cases have generally involved fairly similar colors. We suspect that closer study might suggest a more intelligent distance metric, especially for situations where hues differ.

There could be room for improving this method of approximating the re-coloration cost, both in terms of speed and accuracy. For instance, the simple method in (6) undersamples $\mathbf{Q}^2_{jk}$ when the area of the deformed $\mathbf{Q}^i_{jk}$ is larger than its undeformed area. Again, our experience to date suggests that the quality of morphs generated using this sampling frequency is generally as good as that obtained using more dense sampling. Also, it might be made faster in regions of uniform color where less dense sampling would suffice. However, sampling in regions of uniform color can efficiently be sped up in the optimization phase (see Section 5).

4.2 Stretching cost, W^s

The angle and stretching costs provide a simple approximation to the work of deforming the rubber squares. We use the equation

$$W^s_{jk} = \delta_1^2 + \delta_2^2 + \delta_3^2 + \delta_4^2 \qquad (7)$$

where the four δ values are the length changes that the four edges of $\mathbf{Q}^i_{jk}$ experience when warped.

4.3 Angle cost, W^a

The angle cost is taken to be

$$W^a_{jk} = (\theta_1^2 + \theta_2^2 + \theta_3^2 + \theta_4^2)/2^i \qquad (8)$$

where the four θ values are the angle *changes*, in radians, that the four angles of $\mathbf{Q}^i_{jk}$ experience during warping. Division by 2^i assures that the two maps $F_i \equiv F_{i+1}$ will have the same cost.

5 Optimization

Assuming the existence of a cost function with the nice property that a lower cost indicates a more pleasing morph, our problem of creating an automatic morph algorithm reduces to that of finding the global minimum cost function from among all possible warps. This optimization problem involves 2^{2i+1} degrees of freedom (the (x, y) control point coordinates) and countless local minima. The standard solutions to such an optimization problem include genetic algorithms and simulated annealing. We experimented with a genetic algorithm, but it was much slower (at least, our implementation of it) than the following method, which we used to create the examples in this paper. While it makes no pretense at finding a global minimum, it is simple and fast, and produces surprisingly good results.

Denote by $\hat{F}_i$ the current least-cost warp, and by $W(\hat{F}_i)$ the work associated with that warp from (5). $\tilde{F}$ is the final warp computed by the algorithm.

```
Initialize $\hat{F}_1$ to the identity transformation (3).
**for** i=1 to $i_{\max}${
    **for** each control point $\mathbf{P}^i_{jk}$ {
        **for** $n = 0$ to $n_i${
            **for** each displacement $\delta \in \Delta^n_i${
                Move $\mathbf{P}^i_{jk}$ to $\hat{F}_i(\mathbf{P}^i_{jk} + s_{i,n}\delta)$.
                Denote the resulting warp by $F'_i$.
                **if** $W(F'_i) < W(\hat{F}_i)$, replace $\hat{F}_i$ with $F'_i$.
            }
        }
    }
    Determine the warp $\hat{F}_{i+1}$ equivalent to $\hat{F}_i$,
    as in equation (4).
} $\tilde{F} = \tilde{F}_{i_{\max}}$
```

In words, at each refinement level i, the algorithm visits each vertex $\mathbf{P}^i_{jk}$ and adaptively refines its position. The set of perturbations is defined by an array of offset vectors Δ^n_i. $\mathbf{P}^i_{jk}$ is moved to the best of those candidate locations, and from this (possibly) new location, it is perturbed again using a (possibly different) set of offsets Δ^{n+1}_i. This adaptive refinement of each vertex continues n_i times. For the examples in this paper, we used $n_1 = 5$, $n_2 = 3$, $n_3 = 2$, and $n_4 = n_5 = n_6 = 1$.

The array of offset vectors Δ^n_i changes with i and n. Brief experimentation has shown that good results can be obtained fast if Δ^1_2 and Δ^1_3 are as illustrated in Figure 4 and if all other offset positions are as illustrated in Figure 5.

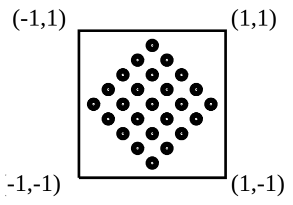

Figure 4: Offset points Δ^1_2 and Δ^1_3.

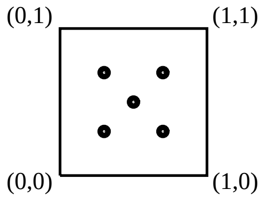

Figure 5: Other Δ^n_i.

The offset vectors in Δ^n_i need to be scaled depending on i and n. The degree of scaling is specified by $s_{i,n}$. For example, for the offset vectors just described, we used $s_{1,1} = 0.5$, $s_{1,2} = 0.1$, $s_{1,3} = 0.05$, etc.

Corner vertices are not moved, while a *legal* displacement for an edge vertex keeps the vertex on its assigned edge. (A *legal* displacement for any vertex is one that does not violate the one-to-one map.)

The algorithm spends 95% of its time computing $W(\tilde{F}_i)$ as the respective control points are moved to new candidate positions. The movement of a single control point $\mathbf{P}_{jk}$ alters at most four domain squares: $\mathbf{Q}^i_{jk}$, $\mathbf{Q}^i_{j-1,k}$, $\mathbf{Q}^i_{j,k-1}$, and $\mathbf{Q}^i_{j-1,k-1}$. After $\mathbf{P}^i_{jk}$ is perturbed, $W(\hat{F}_i)$ can be found most quickly by just computing the change in work for $\mathbf{Q}^i_{jk}$, $\mathbf{Q}^i_{j-1,k}$, $\mathbf{Q}^i_{j,k-1}$, and $\mathbf{Q}^i_{j-1,k-1}$.

The optimization algorithm is illustrated in Figure 6. Here, I_1 from Figure 1 is being warped to match I_2. The left column shows the original I_1 and warps F_1, F_2 and F_3. The right column shows the difference between the images on the left and I_2. White means no difference and black means maximum difference.

The following table lists the total work $W(\hat{F}_i)$ at the end of each optimization level for the morph in Figures 1 and 6. If no warping is performed, the re-coloration work is 3384.5 units.

i	W^s	W^b	W^c	$W(\hat{F}_i)$
0	0	0	3384.5	3384.5
1	4.8	16.3	3121.4	3142.5
2	106.5	98.4	1994.8	2199.8
3	158.1	160.2	1069.6	1387.9
4	184.7	121.9	562.0	868.6
5	180.7	94.3	459.8	734.8
6	182.4	92.9	413.4	688.7

6 User Guidance

With no user guidance, the algorithm may produce unacceptable results, possibly because the optimizer may get stuck on a local minimum. However, even if we went to the expense of computing the globally minimum-cost warp, we would not always be happy with the result. For example, Figure 26 in (Sederberg and Greenwood 1992) shows a polygon shape blending result that can be proven impossible to attain solely using work minimization (unless one uses different work equations for different portions of the polygon). Surely, such examples abound in image morphing.

Figure 15 shows a morph between two key frames of a swinging-arm animation. This morph was computed with help from some user input, as described in Sections 6.1 and 6.2. Figure 7 shows two frames of a morph from the same two key frames, created without user intervention. The work computed for the bad morph is actually less than the work for the good morph in Figure 15, meaning that the work equations used did not achieve the goal of “the lower cost the better morph”.

However, even in cases where user input is needed, the work-minimization algorithm appears to require far less user help than manual morph al-

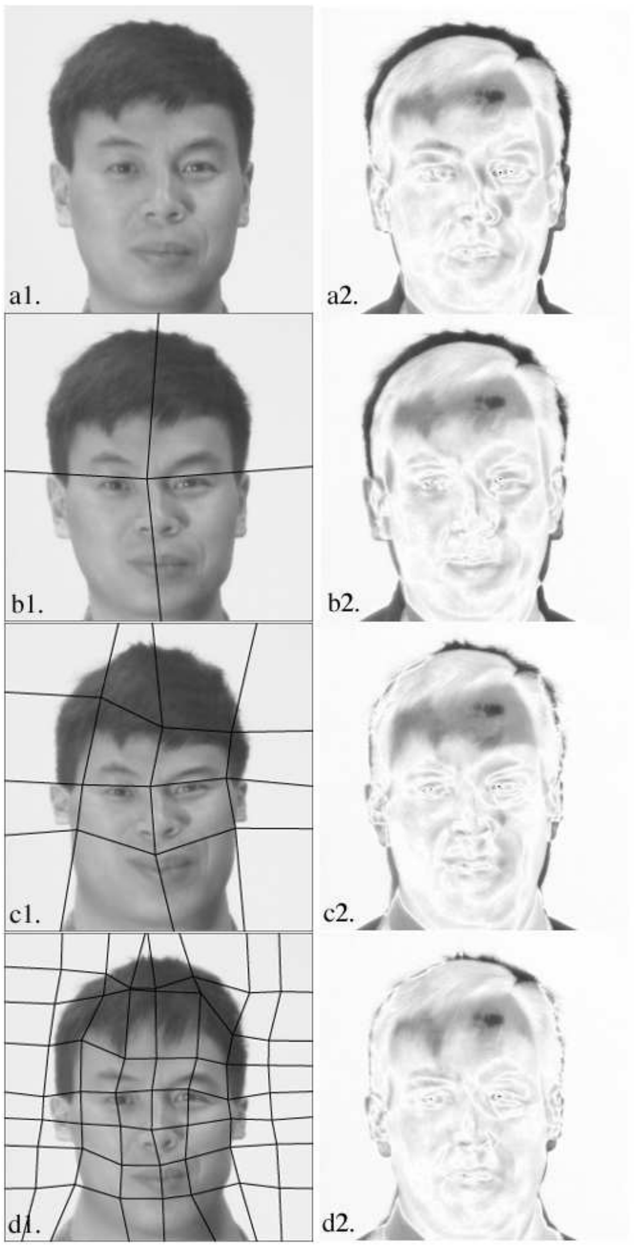

Figure 6: a1) I_1; b1) $\hat{F}_1 \circ I_1$; c1)$\hat{F}_2 \circ I_1$; d1) $\hat{F}_3 \circ I_1$. Right column shows difference between left column and I_2.

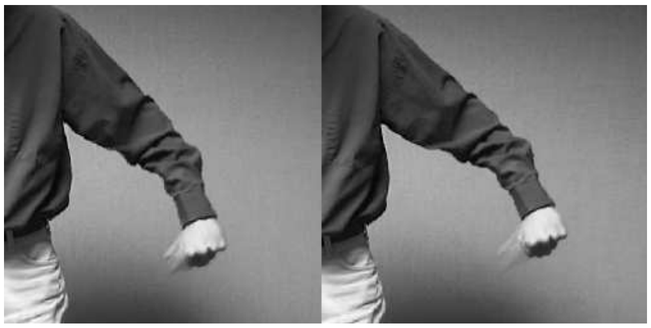

Figure 7: Unacceptable automatic morph.

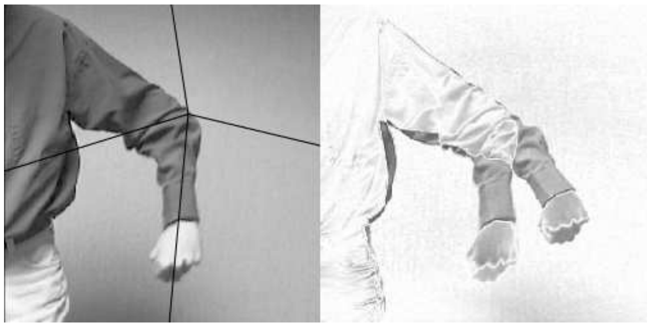

Figure 8: $\hat{F}_1 \circ I_1$ and color difference $I_2 - \hat{F}_1 \circ I_1$.

gorithms. The next two subsections describe two methods for user interaction: anchor points and initial warp.

6.1 Anchor points

Anchor points (called key points in some commercial software) come in pairs, (a_1, a_2). The user positions a_1 on I_1 and a_2 on I_2. The algorithm then assures that $\tilde{F}(a_1) = a_2$.

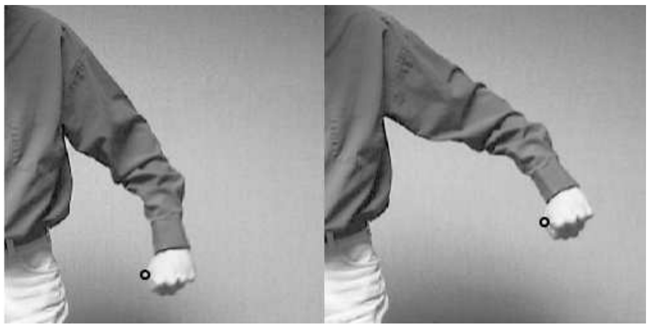

Figure 9: Anchor point specification.

The constraints imposed by anchor points can be incorporated into the work equations by adding a penalty function that encourages the warp to align each pair of anchor points:

$$W^a = \begin{cases} c_1 \frac{2d}{\delta} & \text{if } d \leq \delta/2 \\ \frac{c_1(\delta - d) + c_2(d - \delta/2)}{\delta/2} & \text{if } d > \delta/2 \end{cases} \quad (9)$$

where

$$\delta = \frac{R}{2^{2^i}},$$

$c_2 = 3\delta^2$, $c_1 = c_2/8$, and $d = ||\tilde{F}_i(a_1) - a_2||$, the distance from $\tilde{F}_i(a_1)$ to a_2. Recall that R is the image resolution and i is as in (1).

The warp in Figure 8 was created by specifying a single pair of anchor points, shown in Figure 9.

6.2 Specify the Initial Warp F_1

Another approach to guiding the warp is to invite the user to adjust the control points. If the grid for F_1 in Figure 10 is provided, the algorithm successfully converges to the warp in Figure 8.

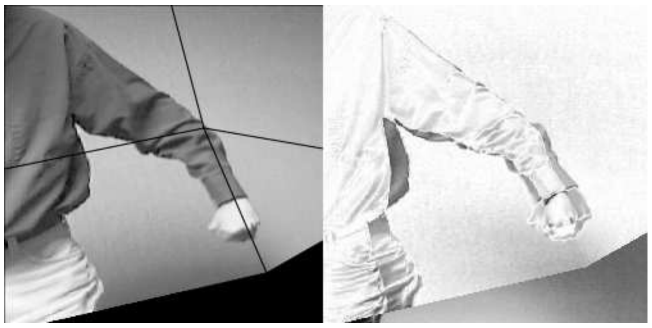

Figure 10: Manual specification of F_1.

Notice that this F_1 warp is not onto. One can argue that the onto requirement is not essential for well-defined image morphs. However, in our implementation, we allow a user-specified F_1 to pull away from the edges, but force F_3 to be onto.

7 Examples and Discussion

The early optimization levels ($i = 1, 2, 3$) run faster than the later ones. It is possible to terminate the optimization after any level and observe the current morph. Figure 11 shows the $t = 0.5$ image from the morph sequence in Figure 14, terminated after $i = 2$, $i = 3$, $i = 4$, and $i = 5$. The following table lists the execution time and total morph-work for each of the four images in Figure 11. The morph sequence in Figure 14 was computed using $i = 6$.

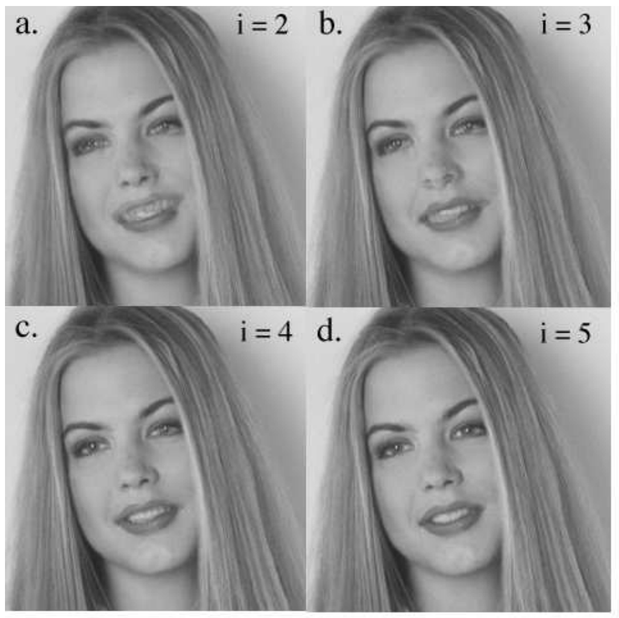

Figure 11: a) result after $i = 2$, b) after $i = 3$, c) after $i = 4$, and d) after $i = 5$.

i	cpu seconds	$W(\hat{F}_i)$
2	3.3	3822
3	4.6	2856
4	5.8	2056
5	7.26	1660
6	10.2	1512

Times came from runs on an HP 9000/780/J282 workstation. The resolution is 256×256. The time for 512×512 is about four times longer.

The coefficients c_c, c_s, and c_a in (5) have significant impact on the quality of the morph. The morph in Figure 1 was computed automatically using coefficients $c_c = c_s = c_a = 1$, as was the morph in Figure 11.

There is a small degree of intuition behind the choice of coefficients. For example, if $c_s = c_a = 0$, the warp will tend to be more fluid. Figure 12 is the middle frame of a morph that was generated using $c_s = c_a = 0$. While morphs can sometimes be improved by adjusting the work coefficients, the most efficient strategy is probably to set $c_c = c_s = c_a = 1$ and add anchor points as needed.

This method clearly works best when the two images are relatively similar. Figure 13 shows a morph between the two authors. The image in the middle was generated by the algorithm with no assistance. The only flaw is a very slight "ghosting" on the right ear lobe. The warp in Figure 14 was

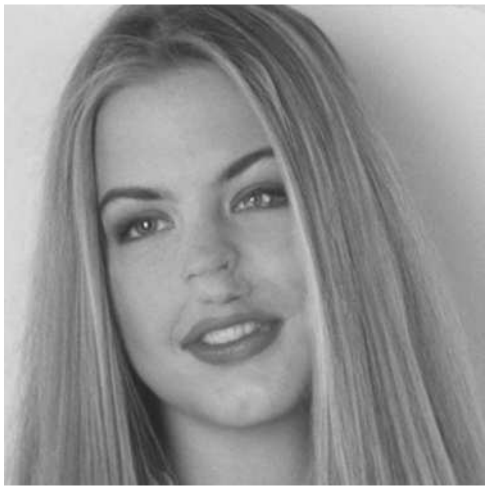

Figure 12: Example with $c_s = c_a = 0$.

also created with no user assistance. The warp in Figure 15 needed one anchor point. The colors in Figure 16 differ enough that the algorithm required eight anchor points. The algorithm succeeds nicely in mapping the outline of the birds in Figure 17, but six anchor points were needed to align the internal details.

8 Conclusions

We have presented an algorithm that is capable of automatically creating good image morphs in many cases where the two images are sufficiently similar. In cases where user guidance is needed, we believe that this approach can significantly ease the burden placed on the artist. It appears to be a time saving, work-minimizing tool.

These results can be extended in several directions. Currently a single work equation applies to the entire image. It seems reasonable to allow different equations for different regions of a scene. For example, the background should be permitted to move freely to accommodate motion in the foreground. Perhaps the stretching and angle coefficients should be zero for background, and the color work should be zero for background mapping to background, but relatively high for background mapping to non-background.

The use of different color spaces for computing re-coloration work should be studied.

9 Acknowledgements

This paper benefited from several discussions with Bill Barrett, Brian Morse, Kirk Duffin, and Eric Mortenson. Alyn Rockwood shared some valuable insights on hierarchical optimization.

Bibliography

Beier T, Neely S (1992) Feature-based image metamorphosis. In: Catmull EE (ed) *Computer Graphics (SIGGRAPH '92 Proceedings)*, 26:35–42

Beier T, Costa B, Gomes J, Velho L (1997) Warping and morphing of graphical objects. Course notes #34, SIGGRAPH'97.

Bookstein FL (1991) *Morphometric tools for landmark data.* Cambridge University Press, New York

Bordwell D, Thompson, K (1997) *The power of Mise-en-scene in Film Art, an introduction.* McGraw Hill

Brown LG (1992) A Survey of Image Registration Techniques. ACM Computing Surveys, 24:325–276.

Burns P (1994) *CD-Morph.* Addison-Wesley

Dhond UR, Aggarwal, JK (1989) Structure from Stereo: A Review. IEEE Transactions on Systems, Man and Cybernetics, 19:1289–1510

Kass M, Witkin A, Terzopoulos D (1988) Snakes: Active contour models. *International Journal of Computer Vision*, 1:321–331

Lee SY, Chwa KY, Shin SY, Wolberg G (1995) Image metamorphosis using snakes and free-form deformations. In Cook R (ed) *SIGGRAPH 95 Conference Proceedings*, Annual Conference Series, pages 439–448. ACM SIGGRAPH, Addison Wesley

Lee SY, Chwa KY, Hahn J, Chin SY (1996) Image morphing using deformation techniques. *Journal of Visualization and Computer Animation*, 7:3–24

Nishita T, Fujii T, Nakamae E (1993) Metamorphosis using Bézier clipping. In *First Pacific Conference on Computer Graphics and Applications.* Korean Information Science Society, Korean Computer Graphics Society.

Sederberg TW, Greenwood E (1992) A physically based approach to 2D shape blending. In Catmull EE (ed), *Computer Graphics (SIGGRAPH '92 Proceedings)*, 26:25–34

Sederberg TW, Greenwood E (1995) Shape blending of 2–D piecewise curves. In Daehlen M, Lyche T, Schumaker L (eds) *Mathematical methods for curves and surfaces*, 497–506. Vanderbilt University Press.

Weng J, Ahuja N , Huang TS (1992) Matching Two Perspective Views. PAMI,14:806–825

Wolberg G (1990) *Digital image warping.* IEEE Computer Science Press

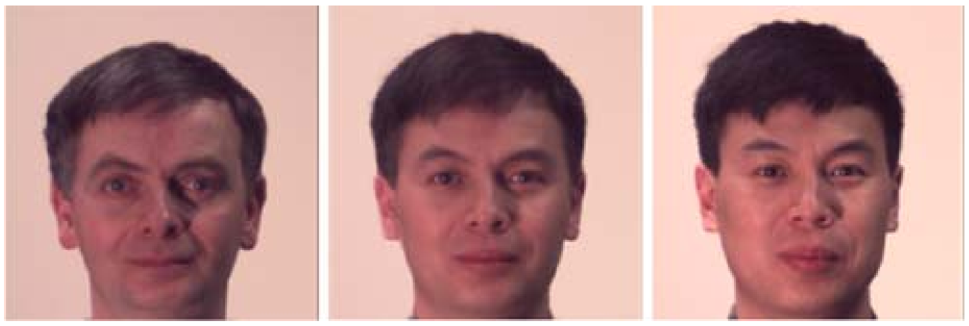

Figure 13: The two authors and the automatically-generated mid-way morph

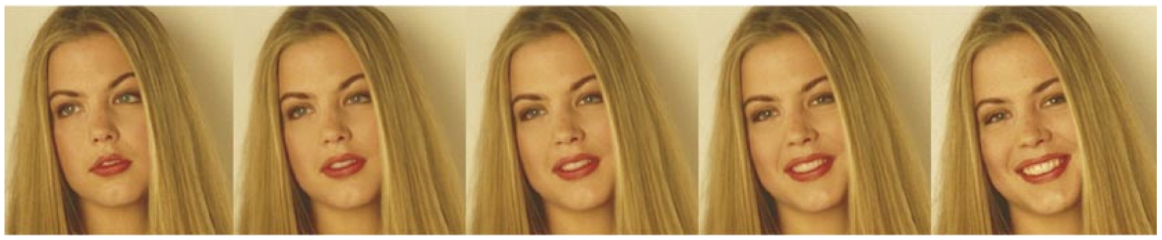

Figure 14: Automatic image morph

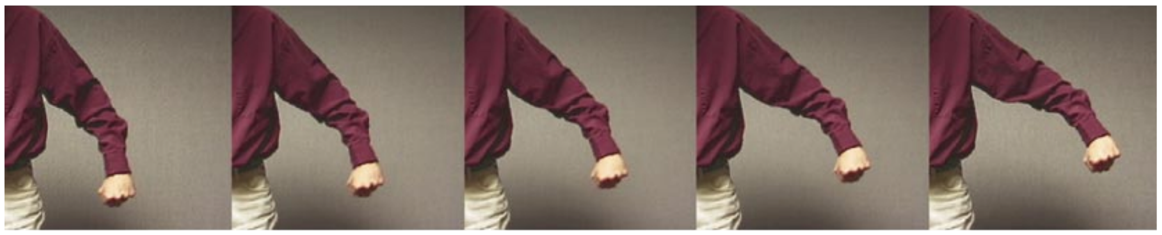

Figure 15: Morph computed with one anchor point, or by setting the initial 2X2 grid

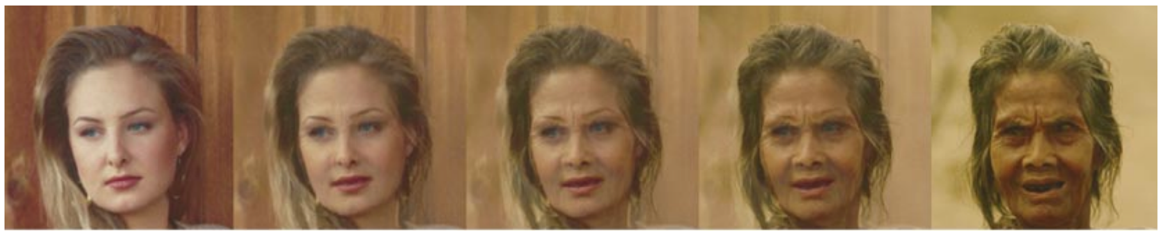

Figure 16: Morph computed with six anchor points

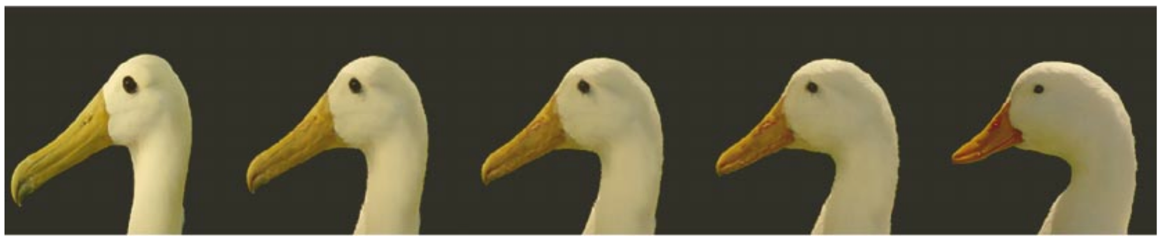

Figure 17: Morph computed with five anchor points