

Automating Image Morphing Using Structural Similarity on a Halfway Domain

JING LIAO

Hong Kong UST and Zhejiang University

RODOLFO S. LIMA and DIEGO NEHAB

Instituto Nacional de Matemática Pura e Aplicada (IMPA)

HUGUES HOPPE

Microsoft Research

PEDRO V. SANDER

Hong Kong UST

and

JINHUI YU

Zhejiang University

The main challenge in achieving good image morphs is to create a map that aligns corresponding image elements. Our aim is to help automate this often tedious task. We compute the map by optimizing the compatibility of corresponding warped image neighborhoods using an adaptation of structural similarity. The optimization is regularized by a thin-plate spline and may be guided by a few user-drawn points. We parameterize the map over a halfway domain and show that this representation offers many benefits. The map is able to treat the image pair symmetrically, model simple occlusions continuously, span partially overlapping images, and define extrapolated correspondences. Moreover, it enables direct evaluation of the morph in a pixel shader without mesh rasterization. We improve the morphs by optimizing quadratic motion paths and by seamlessly extending content beyond the image boundaries. We parallelize the algorithm on a GPU to achieve a responsive interface and demonstrate challenging morphs obtained with little effort.

Categories and Subject Descriptors: I.3.0 [Computer Graphics]: General

NVIDIA Corporation has generously donated the GPUs used in this project. The project was partly supported by Hong Kong GRF grants nos. 619509 and 618513.

Authors' addresses: J. Liao, Hong Kong UST (University of Science and Technology), Clear Water Bay, Hong Kong; Zhejiang University, 38 Zheda Rd, Xihu, Hangzhou, Zhejiang, China 310027; R. S. Lima and D. Nehab, Instituto Nacional de Matemática Pura e Aplicada (IMPA), Estr. Dona Castorina, 110-Jardim Botânico, Rio de Janeiro-RJ, 22460-320, Brazil; H. Hoppe, Microsoft Research; P. V. Sander (corresponding author), Hong Kong UST (University of Science and Technology), Clear Water Bay, Hong Kong; email: psander@cse.ust.hk; J. Yu, Zhejiang University, 38 Zheda Rd, Xihu, Hangzhou, Zhejiang, China 310027.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

© 2014 ACM 0730-0301/2014/08-ART168 \$15.00

DOI: <http://dx.doi.org/10.1145/2629494>

General Terms: Algorithms

Additional Key Words and Phrases: Image interpolation, warping, parameterization, motion paths, correspondences, GPU, Poisson extension

ACM Reference Format:

Jing Liao, Rodolfo S. Lima, Diego Nehab, Hugues Hoppe, Pedro V. Sander, and Jinhui Yu. 2014. Automating image morphing using structural similarity on a halfway domain. *ACM Trans. Graph.* 33, 5. Article 168 (August 2014). 12 pages.

DOI: <http://dx.doi.org/10.1145/2629494>

1. INTRODUCTION

Morphing between images is a long-studied problem addressed in numerous publications including surveys and books [Wolberg 1990, 1998; Gomes et al. 1999]. It usually involves a sequence of steps: determining corresponding feature points in the images, defining the motion paths followed by these features, interpolating a mapping in the space between the features, warping the images into alignment, and finally blending the warped images to produce an animation.

The most challenging step is to establish a good correspondence map between the images, as this is crucial to prevent ghosting artifacts during blending. Creating this map usually involves significant interaction, using tagged features such as points, lines, curves, or grids. Such careful registration of intricate silhouettes can be tedious.

Our aim is to develop an optimization framework that simplifies map creation by automatically aligning image structures including irregular object boundaries. In many cases, an effective morph requires a semantic understanding of the image content, which is beyond the scope of this article. Instead, we let the user draw a small number of point correspondences to guide the process at a high level. The key is that this guidance should be simple and sparse.

There are numerous stereo and optical-flow techniques for automatically estimating image correspondences. However, these techniques assume the images are closely related, for example, multiple views of a consistent scene under similar lighting conditions. In contrast, morphing often considers images of different objects.

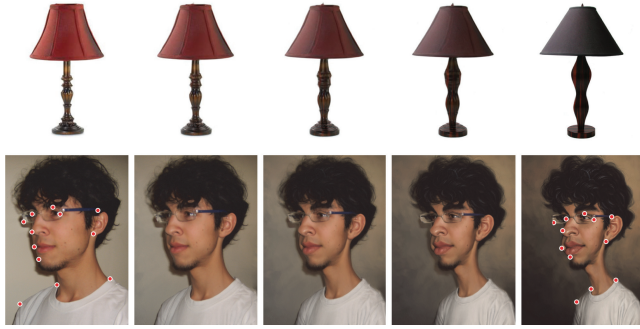


Fig. 1. Our optimization creates effective morphs between dissimilar images by automatically aligning structural image features, requiring little or no user guidance (here 0 and 7 point correspondences). Caricature by ©Court Jones.

Therefore, morphing approaches must allow inexact matching of image neighborhoods (objects with different shapes and colors) and, as a consequence of this increased generality, they perform poorly for problems like optical flow where the images are known to be similar.

As reviewed in Section 2, other work has also considered the problem of optimization-guided image morphing [Gao and Sederberg 1998; Matusik et al. 2005; Wu and Liu 2013]. Our approach is designed to allow greater dissimilarities between images. Rather than comparing colors or gradients at corresponding points, we compare compatibility of corresponding warped image neighborhoods. To this end, we adapt the structural similarity (SSIM) measure introduced for perceptual image comparison by Wang et al. [2004].

We make the resulting optimization framework more tractable by representing the inter-image map using a *halfway parameterization*. This representation offers a number of benefits.

- Symmetry*. The parameterization does not favor either image.
- Continuity*. The map can represent simple occlusion discontinuities using a continuous function, thus simplifying regularization.
- Inclusivity*. When extended at its boundaries, the parametric domain spans the content from both images.

We show that the other steps in morphing also greatly benefit from the halfway parameterization.

- Nonlinear paths*. The map can concisely represent quadratic motion paths to reduce deformation during the morph.
- Evaluation*. The inverse map supports direct evaluation, so the morphed image is computed within a simple pixel shader, without rasterization of a fine triangle mesh.
- Extrapolation*. Correspondences are defined beyond the image boundaries, enabling a joint-image Poisson extension scheme to reduce boundary artifacts.

2. RELATED WORK

Algorithms for automatically finding image correspondences are typically based on registration, optical flow, or stereo matching (for surveys, see Zitova and Flusser [2003], Baker et al. [2011], and Scharstein and Szeliski [2002]). These methods assume structural relationships between source and destination images that are not present in general image morphing. The same is true for view interpolation in image-based rendering (see the survey in Shum

et al. [2003]) and for temporal upsampling of video sequences (e.g., Kang et al. [2003], Mahajan et al. [2009], and Yang et al. [2011]). Nevertheless, for specialized image domains (such as faces or periodic textures), correspondences can be automatically found [Bichsel 1996; Beymer 1996; Covell 1996; Liu et al. 2002; Yang et al. 2012]. Since matching features is in general an ambiguous and subjective problem, artistic control is fundamental. Even when assisted by snapping tools [Kass et al. 1988; Mortensen and Barrett 1995], this interaction can be tedious.

Several methods relate to ours in that they also establish dense correspondences between images by optimizing particular choices of similarity, smoothness, and distortion terms. The methods of Gao and Sederberg [1998] and Wu and Liu [2013] measure image similarity based on color or gradients and therefore assume the images to be related. Matusik et al. [2005] derive a similarity metric based on the intensity of the *compass operator* [Ruzon and Tomasi 2001]. Their warp optimization is aimed at texture morphing and does not offer artistic control. In contrast to these methods, our algorithm accounts for map warping when comparing corresponding image neighborhoods.

Yücer et al. [2012] present an approach to automatically transfer edits across images of the same subject under strongly varying viewpoints and changes in illumination. To this end, their method creates a mapping between the two images. Unlike our approach, their algorithm is designed specifically for images of the same subject, with related colors and gradients.

Our halfway parameterization is related to the *cyclopean coordinates* introduced by Belhumeur and Mumford [1992] for solving the stereo correspondences problem, if one considers moving the camera along the stereo baseline. Mahajan et al. [2009] generalize this concept by allowing each pixel to have a different transition point, an idea adopted by Wu and Liu [2013] as well.

Recent work has explored entirely different approaches to image morphing that lead to interesting effects. Shechtman et al. [2010] applied patch-based texture synthesis to achieve interesting morph-like effects in a fully automated way. Darabi et al. [2012] further improve the quality of patch-based synthesis for scenarios where the two input images have very different colors and structures. Rather than morphing between a source and target image, photobios search among a large collection of images of the same person to produce a plausible transition between two input faces [Kemelmacher-Shlizerman et al. 2011].

There is relatively little work on controlling the motion paths of features during the morph animation. The common approach of linear interpolation does not reproduce rotations, as it leads to unexpected nonmonotonic deformations. One possibility is to offer artistic control at each feature, as proposed by Nishita et al. [1993]. An alternative is to factor rotations out of local transformations and independently interpolate the components [Zhang 1996; Rossignac and Vinacua 2011]. We describe a technique to improve motion paths using a quadratic path optimization at pixel resolution.

3. THE HALFWAY PARAMETERIZATION

The first step in traditional approaches for creating a morph between a pair of images I_0 and I_1 is to define a map ϕ from I_0 to I_1 . One challenge is that each image may contain “disoccluded” regions that are invisible in the other image. Within disoccluded regions of I_0 , map ϕ is typically left undefined and, across disoccluded regions of I_1 , it must be discontinuous. To obtain a more complete representation, one can introduce a second map from I_1 back to I_0 , but then maintaining consistency between the two maps during an optimization process becomes quite expensive.

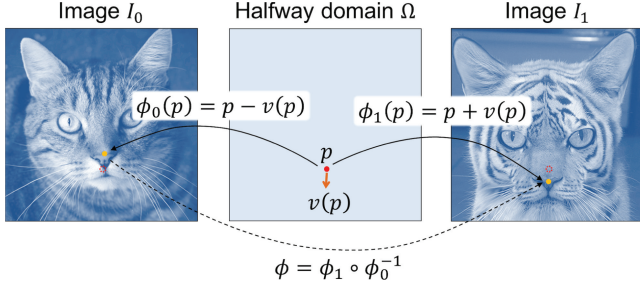


Fig. 2. The inter-image map ϕ is the composition of two maps defined using a common vector field v on a halfway domain.

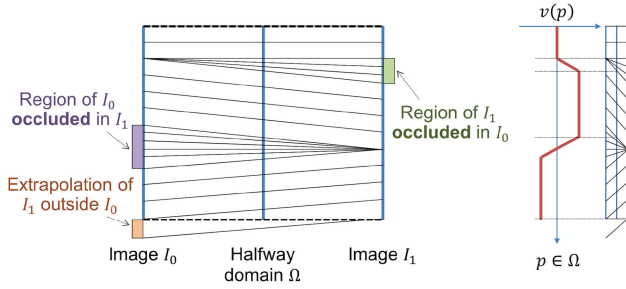


Fig. 3. With a halfway parameterization, discontinuities due to simple occlusions are represented using a continuous vector field v . Here the two images are represented as 1D vertical segments and the associated correspondence field v is plotted as a graph on the right.

Inspired by earlier work (e.g., Mahajan et al. [2009]), we address these issues by forming the map using a halfway domain. Specifically, we define a single 2D vector field v over a domain Ω halfway between the two images and use this field to create two maps (see Figure 2):

$$\phi_0(p) = p - v(p) \quad \text{and} \quad \phi_1(p) = p + v(p). \quad (1)$$

The inter-image map is the composition $\phi = \phi_1 \circ \phi_0^{-1}$. This parameterization is nicely symmetric, because an exchange of the two images simply corresponds to a negation of the vector field. As illustrated in the 1D example of Figure 3, the vector field v is continuous even in the presence of simple occlusions. For instance, the purple region in image I_0 collapses to a point in image I_1 , yet the vector field v defined over the halfway domain remains continuous. Although the parametric domain area corresponding to the purple region is shrunk, it is not degenerate. As shown with the orange region, the map can also be used to extrapolate correspondences when one image extends beyond the boundary of the other. In fact, a simple constant extension of the vector field v (using clamp-to-border sampling) yields useful correspondences beyond the boundaries, as explored in Section 7.

4. CORRESPONDENCE OPTIMIZATION

To compute the halfway parameterization that best aligns the two input images, we develop a multiresolution coarse-to-fine optimization algorithm that operates on a regular grid of vector field samples $v(p)$ defined over the halfway domain. At the finest level of detail, the grid resolution matches that of the input images. We first present the energy function that the algorithm minimizes to find the best pixel neighborhood correspondences, then we describe the multiresolution algorithm and our GPU implementation.

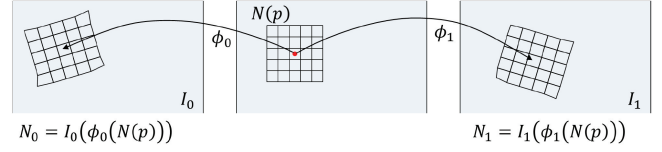


Fig. 4. The SIM energy term measures structural similarity of corresponding warped neighborhoods N_0, N_1 of the two input images.

4.1 Energy Function

At each level in the coarse-to-fine optimization process, we optimize the grid of vector values $v(p)$ to minimize an objective energy E defined over the halfway image domain Ω :

$$E = \sum_{p \in \Omega} E(p), \quad \text{with} \quad (2)$$

$$E(p) = E_{SIM}(p) + \lambda E_{TPS}(p) + \gamma E_{UI}(p). \quad (3)$$

We next describe each of the three energy terms. (We use fixed weights $\lambda = 0.001$ and $\gamma = 100$ for all results in this article.)

Similarity energy. To measure whether corresponding neighborhoods in the two input images have similar edge structure, we use a modified version of the structural similarity index (SSIM) introduced by Wang et al. [2004].

For each gridpoint p in the halfway domain Ω , the corresponding image neighborhoods are obtained using the current mappings $\phi_0(N(p))$ and $\phi_1(N(p))$ of a 5×5 neighborhood $N(p)$ of p (see Figure 4). This allows severely warped neighborhoods to be matched. For each halfway point p , we use bilinear interpolation to gather the corresponding warped neighborhoods from images I_0 and I_1 . Then, we evaluate

$$E_{SIM}(p) = -\frac{1}{wh} \text{SIM}(N_0, N_1), \quad \text{with} \quad (4)$$

$$N_0 = I_0(\phi_0(N(p))) \quad \text{and} \quad N_1 = I_1(\phi_1(N(p))). \quad (5)$$

(Normalization by wh makes the term resolution independent, while the negation turns a high similarity into a low energy.)

Just as in SSIM, the function $\text{SIM}(N_0, N_1)$ is defined in terms of the means, variances, and covariances of the pixel values in the two neighborhoods. However, there are two key differences. First, we omit the luminance term, because we want to match regions with similar edge structure, regardless of luminance. And second, we take the absolute value of the covariance term $\sigma_{N_0 N_1}$, because swapping colors across an edge should not affect the energy. Formally,

$$\text{SIM}(N_0, N_1) = c(N_0, N_1) \cdot s(N_0, N_1), \quad \text{with} \quad (6)$$

$$c(N_0, N_1) = \frac{2\sigma_{N_0 N_1} + C_2}{\sigma_{N_0}^2 + \sigma_{N_1}^2 + C_2} \quad \text{and} \quad (7)$$

$$s(N_0, N_1) = \frac{|\sigma_{N_0 N_1}| + C_3}{\sigma_{N_0 N_1} + C_3}. \quad (8)$$

Here, $C_2 = 58.5$ and $C_3 = 29.3$ are the same constants as in SSIM.

Smoothness energy. In the absence of other constraints, we want the optimization to favor affine functions. This is accomplished by minimizing the thin-plate spline (TPS) energy independently on

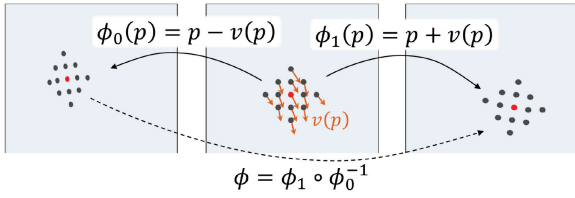


Fig. 5. The Thin-Plate Spline (TPS) term encourages smoothness of the correspondence vector field v defined over the halfway domain.

components v_x and v_y of vector field v :

$$E_{TPS}(p) = TPS(v_x(p)) + TPS(v_y(p)), \quad \text{where} \quad (9)$$

$$TPS(v_x(p)) = \left(\frac{\partial^2 v_x(p)}{\partial p_x^2} \right)^2 + 2 \left(\frac{\partial^2 v_x(p)}{\partial p_x \partial p_y} \right)^2 + \left(\frac{\partial^2 v_x(p)}{\partial p_y^2} \right)^2, \quad (10)$$

and analogously for $TPS(v_y(p))$ (see Figure 5). The TPS energy is defined over the halfway domain grid and discretized as in Finch et al. [2011].

UI energy. To resolve ambiguities and provide artistic control, we allow the user to manually specify pairs of corresponding points u_i^0 and u_i^1 in images I_0 and I_1 . Let $\bar{u}_i = (u_i^0 + u_i^1)/2$ and $v_{u_i} = (u_i^1 - u_i^0)/2$. The correspondence between u_i^0 and u_i^1 forces the halfway vector $v(\bar{u}_i)$ to equal v_{u_i} .

However, \bar{u}_i does not in general lie exactly at a gridpoint $p \in \Omega$. Thus, we enforce soft constraints on the four nearest neighbors $p_{i1}, p_{i2}, p_{i3}, p_{i4}$ of \bar{u}_i . Each soft constraint is weighted by the corresponding bilinear weight $b(p_{ij}, \bar{u}_i)$ such that

$$\sum_{j=1}^4 b(p_{ij}, \bar{u}_i) p_{ij} = \bar{u}_i. \quad (11)$$

Therefore, the UI energy is

$$E_{UI}(p) = \frac{1}{wh} \sum_{i,j \mid p_{ij}=p} b(p_{ij}, \bar{u}_i) \|v(p_{ij}) - v_{u_i}\|^2. \quad (12)$$

Thus, by definition, $E_{UI}(p)$ is zero for gridpoints that are not in the neighborhood of any constraint point \bar{u}_i .

4.2 Optimization

We use a coarse-to-fine minimization approach to accelerate the optimization of the vector field v (and hopefully avoid local minima).

Multiresolution solver. We first build image pyramids where the coarsest level has a resolution of 8×8 or 16×16 pixels. Proceeding coarse-to-fine, we run an iterative relaxation on progressively finer halfway domain grids. After each level is brought to convergence, the solved vector field v is upsampled and used as starting point for the next-finer level. We continue until reaching the desired resolution for the halfway domain grid.

Iterative optimization at each level. At each gridpoint p in the current level, we estimate the direction of the energy gradient relative to $v(p)$ using finite differences. We then use golden section search [Kiefer 1953] to find the new optimal value for $v(p)$ along this search direction. To prevent foldovers, we restrict the search interval so that $\phi_0(p)$ and $\phi_1(p)$ remain inside their respective 1-rings; the maximum displacement is computed as shown in Figure 6. We repeat the optimization for all points p until convergence.

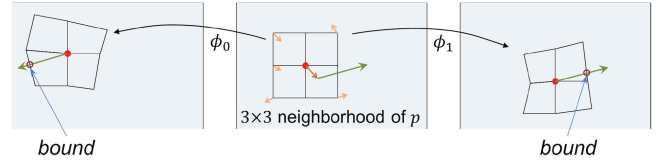


Fig. 6. During line search optimization of $v(p)$, we compute bounds on the search interval to constrain maps ϕ_0 and ϕ_1 to remain bijections.

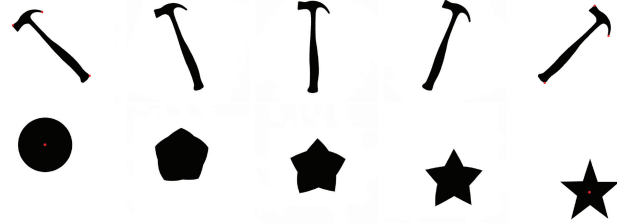


Fig. 7. Minimizing the combination of SIM, TPS, and UI energy terms handles rotation, translation, and boundary deformation. For these two examples, we specified four and one user correspondence points, respectively.

For the optimization at the coarsest level, we omit the SIM energy term; this initial coarse solution is based solely on the user correspondence points and TPS energy.

As shown in Figure 7, even with very few UI correspondence points, the optimization is able to robustly find good correspondences in the presence of affine transformations and shape deformations.

4.3 GPU Implementation Details

To benefit from the huge processing power of modern GPUs, the optimization process should be made as parallel as possible. Unfortunately, not all gridpoints can be independently optimized. The foldover prevention strategy shown in Figure 6 requires that values in a 1-ring around p remain constant while $v(p)$ changes. Worse still, since neighborhoods N_0 and N_1 in the SIM function (4) depend on a 2-ring of values around p , all of these must be held constant while moving $v(p)$. We have found that, although we must indeed respect the parallelism constraint imposed by the foldover prevention, we can loosen the SIM restriction. In theory, this prevents us from guaranteeing that the energy is reduced at each parallel iteration. However, in practice, the optimization converges and the added parallelism speeds up the process by 15% to 130% in our tests, depending on grid resolution.

We therefore divide the input grid into blocks with 64×16 gridpoints, each to be processed by a CUDA block with 32×8 threads. Within each block, we partition the points into the four groups with 32×8 points that satisfy the aforementioned 1-ring minimum separation. The points in each group are optimized in parallel according to the algorithm in Section 4.2. Since there can be no synchronization between independent CUDA blocks, the 64×16 gridpoint blocks are separated by gaps (four gridpoints wide) to avoid conflicts arising from the SIM restriction. Points in these gaps are processed by subsequent invocations that use gaps at different offsets. (Not all points are optimized the same number of times, but we found that this does not significantly affect accuracy.)

Whenever $v(p)$ is modified, it is necessary to update $E_{SIM}(q)$ for all q in its 5×5 neighborhood. Updating these 25 values from scratch would involve as many as $(5^2 - 1) \times 5^2 = 600$ memory accesses

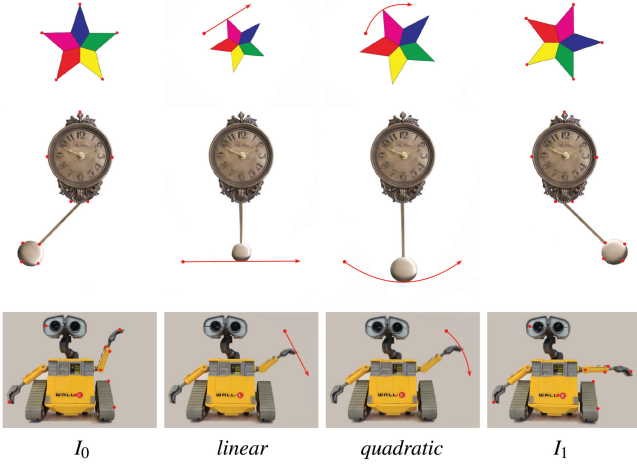


Fig. 8. These examples show that rotating regions shrink when using linear motion paths. With quadratic paths, the shape is better preserved throughout the morph. The red arrows show the path followed by a selected point.

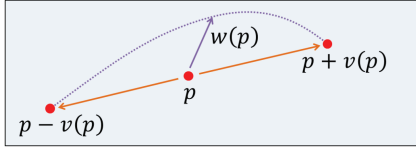


Fig. 9. An additional vector $w(p)$ at each halfway pixel defines a quadratic motion path to reduce map deformation during the morph.

per thread. We instead incrementally update these. The key is to keep the means, variances, and covariance terms for each point as separate value terms. When $v(p)$ is modified, the responsible thread uses fast atomic operations to incrementally update the affected terms for all neighbors q . After a fast intra-block synchronization, each thread recomputes its own $E_{SIM}(p)$ from up-to-date terms.

We optimize the coarsest level (64 to 256 gridpoints) on the CPU using a direct solver. For each finer level, the GPU relaxation solver iterates until no more $v(p)$ changes. Each golden section search usually terminates in fewer than 10 iterations (given a search interval of $\varepsilon = 0.001$).

5. QUADRATIC MOTION PATHS

Given the results of the optimization, the morphing sequence moves each point p from its position $q_0 = \phi_0(p)$ in image I_0 to its position $q_1 = \phi_1(p)$ in image I_1 , as a parameter α spans the time interval $[0, 1]$. The typical approach is to use linear interpolation.

$$q_\alpha = p + (2\alpha - 1)v(p) \quad (13)$$

Unfortunately, when considered as a whole, linear motion paths can lead to undesirable (often nonmonotonic) deformations, even when a global rotation exists (see Figure 8). To reduce this problem, we instead define quadratic motion paths.

At each point p , we solve for an additional vector $w(p)$ that defines the control point $b_{1/2} = p + 2w(p)$ for a quadratic Bézier path:

$$q_\alpha = (1 - \alpha)((1 - \alpha)q_0 + \alpha b_{1/2}) + \alpha((1 - \alpha)b_{1/2} + \alpha q_1) \quad (14)$$

$$= p + (2\alpha - 1)v(p) + 4\alpha(1 - \alpha)w(p). \quad (15)$$

The idea is to use these additional degrees of freedom to improve the morph quality. The path still goes through q_0 and q_1 and it interpolates $q_{1/2} = p + w(p)$.

Computing $w(p)$. We select $w(p)$ by minimizing the sum of two energy terms. The first term, $E_D(w)$, guides the local deformations halfway through the animation to match the halfway rotation and scale between the source and target images. The second term, $E_R(w)$, encourages points that are at rest (i.e., for which $v(p) \approx 0$) to remain at rest during the animation.

Let points p_i, p_j be a pair of (either horizontal or vertical) neighbors in the halfway domain Ω . Define the vectors connecting them in images I_0 and I_1 :

$$d_0(p_i, p_j) = \phi_0(p_j) - \phi_0(p_i) \quad (16)$$

$$= p_j - p_i - (v(p_j) - v(p_i)), \quad (17)$$

$$d_1(p_i, p_j) = p_j - p_i + (v(p_j) - v(p_i)). \quad (18)$$

Halfway through the animation, we want the corresponding vectors to match the halfway rotation and exponential scaling between vectors d_0 and d_1 :

$$\tilde{d}_{1/2}(p_i, p_j) = \sqrt{\|d_0(p_i, p_j)\| \|d_1(p_i, p_j)\|} \hat{d}_s(p_i, p_j), \quad (19)$$

$$\text{with } \hat{d}_s(p_i, p_j) = \hat{d}_1(p_i, p_j) + \hat{d}_2(p_i, p_j) \text{ and } \hat{u} = u/\|u\|. \quad (20)$$

The halfway vector we actually obtain from the quadratic path is

$$d_{1/2}(p_i, p_j) = p_j - p_i + (w(p_j) - w(p_i)). \quad (21)$$

Therefore we set the deformation energy term to

$$E_D(w) = \sum_{p_i, p_j} \|d_{1/2}(p_i, p_j) - \tilde{d}_{1/2}(p_i, p_j)\|^2. \quad (22)$$

For the resting energy term, we use

$$E_R(w) = \sum_{p_i \text{ s.t. } \|v(p_i)\| < 1} (1 - \|v(p_i)\|) \|w(p_i)\|^2. \quad (23)$$

Thus we compute the vectors $w(p)$ that minimize the energy

$$E(w) = E_D(w) + \beta E_R(w). \quad (24)$$

This linear least-squares energy has a unique minimum if there are any resting correspondences. Otherwise, we simply set the mean $\bar{w} = \mathbf{0}$. We use $\beta = 1$ for all examples in this article.

As shown in the examples in Figure 8, using quadratic motion paths that minimize neighborhood deformation in the halfway image helps preserve shape throughout the morph, whereas linear paths often shrink then grow any rotating region.

6. DIRECT PIXEL EVALUATION FOR RENDERING

Inspired by the work of Yang et al. [2011], we propose a simple and efficient backward-mapping algorithm to render intermediate frames within a morph. Our method does not involve mesh rasterization. Instead, each pixel in an intermediate image is independently evaluated. Moreover, unlike mesh-based approaches, our method is able to produce images that cover the entire domain of the input images, as shown in Figure 10.

To simplify the presentation, we begin by considering the case of linear paths. The goal is to invert Eq. (13) so that, given a point q_α in the intermediate image I_α , we find the corresponding point p in the halfway domain Ω . Having obtained p , we use the functions

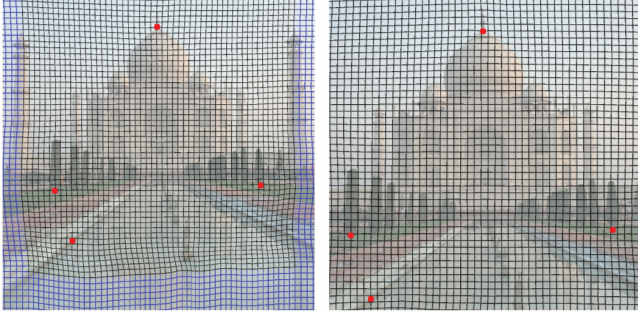


Fig. 10. Visualization of the halfway domain grid overlaid on the two input images. With traditional rasterization, image pixels beyond the domain boundary are left uncovered (as shown in blue). Our iterative pixel evaluation approach fills these pixels with plausible values from the input images.



Fig. 11. In the presence of large rotations, omitting damping in the direct iterative solver may lead to rendering artifacts; the original images are not reproduced at the morph extremes because the solver fails to converge to the correct halfway point. (Compare with Figure 8.)

ϕ_0 and ϕ_1 to perform sampling lookups into images I_0 and I_1 . The algorithm can be summarized as follows:

For each morph image I_α , we do the following.

- (1) We render a single quadrilateral over the whole image domain.
- (2) In the pixel shader, for pixel q_α , we perform an iterative search to find the corresponding point p .
- (3) We sample I_0 at $\phi_0(p)$ and I_1 at $\phi_1(p)$ using bicubic filtering.
- (4) We blend $I_\alpha(p) = (1 - \alpha)I_0(\phi_0(p)) + (\alpha)I_1(\phi_1(p))$.

The main challenge is to find point p in step (2). It is obtained using the following iterative search:

$$p^{(0)} = q_\alpha, \quad (25)$$

$$p^{(i+1)} = q_\alpha - (2\alpha - 1) v(p^{(i)}). \quad (26)$$

The same approach works with quadratic motion paths, except that we instead invert Eq. (15):

$$p^{(0)} = q_\alpha, \quad (27)$$

$$p^{(i+1)} = q_\alpha - (2\alpha - 1) v(p^{(i)}) - 4\alpha(1 - \alpha) w(p^{(i)}). \quad (28)$$

Because the map is smooth, the iterative search generally converges after just a few (e.g., 3–5) iterations in both the linear and quadratic cases. One difficulty we have encountered is that, when the map includes large rotations, the iterative search may cycle and thus fail to converge (Figure 11). We avoid this by damping the vector values during the iterative process:

$$p^{(0)} = q_\alpha, \quad (29)$$

$$v^{(0)} = v(p^{(0)}), \quad (30)$$

$$p^{(i+1)} = q_\alpha - (2\alpha - 1) v^{(i)}, \quad (31)$$

$$v^{(i+1)} = (\eta)v(p^{(i+1)}) + (1 - \eta)v^{(i)}. \quad (32)$$

The same modification also applies to the quadratic path case. We set the exponential smoothing factor $\eta = 0.8$ in all results.

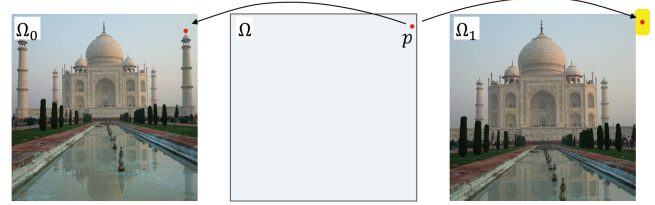


Fig. 12. The fact that a halfway point p may map beyond the extent of one image lets us extrapolate outside the image boundary.

The direct pixel evaluation algorithm is implemented on the GPU and is extremely efficient, as reported in Section 9. One exciting benefit of the direct evaluation approach is that it can be used to immediately render a morphing image over an arbitrary surface. Unlike conventional approaches, this immediate rendering offers high-quality filtering and does not require rasterization into a temporary texture buffer.

7. POISSON-EXTENDED IMAGE BOUNDARIES

Often, the optimized mapping computed in Section 4.2 appropriately maps some regions of I_0 outside the domain of I_1 (or vice versa). For example, in Figure 12 the red point in I_0 is mapped outside I_1 . When morphing, the color sampled from I_1 is therefore undefined. To address this, we extend each image by transferring content from the other image using the halfway parameterization, then seamlessly complete and stitch the resulting extended image using a gradient-domain least-squares optimization (i.e., a Poisson equation).

Pixel transport. We construct for image I_0 an extended version \tilde{I}_0 , for instance with 10% extra pixels in each direction. The interior (in-bounds) region of \tilde{I}_0 is copied directly from I_0 and is never modified. To transport content into the extended region of \tilde{I}_0 , we evaluate a pixel shader with the following algorithm. For each pixel $q_0 \in \tilde{\Omega}_0$, we seek its domain point p (i.e., satisfying $\phi_0(p) = q_0$) to find its corresponding point $q_1 = \phi_1(p) \in \Omega_1$. The search for p is solved using almost the same iterative algorithm as in the direct pixel evaluation of Section 6. The new difficulty is that q_0 lies in the extended region, so after the initialization $p^{(0)} = q_0$, the value $v(p^{(0)})$ is undefined. Nonetheless, we run the iterative algorithm where $v(\cdot)$ is evaluated using clamped-to-border sampling that corresponds to a piecewise constant extension of v . After the iterative search, if the point $q_1(p) = p + v(p)$ lies outside image I_1 , which is not uncommon, we assign pixel q_0 an undefined value. Otherwise, we sample the color in I_1 .

Completion using gradient-domain optimization. In the resulting extended region, for each pair of adjacent pixels, we compute their finite difference. This difference is set to zero if either pixel is undefined, or if one of the pixels is on the perimeter of I_0 . We optimize the pixel colors in the extended region of \tilde{I}_0 to match these desired differences while constraining the colors on the boundary of I_0 (as Dirichlet constraints) [Pérez et al. 2003].

We apply a similar process to seamlessly extend image I_1 to \tilde{I}_1 using colors transported from I_0 . The effect is that the extended region of each image captures detail from the other image while seamlessly matching the appearance of its own inner region. Figure 13 shows an example result. Because the right image is contained within the left image, its extension receives detail transported from the left image. When there is no correspondence, as is the case for the extended

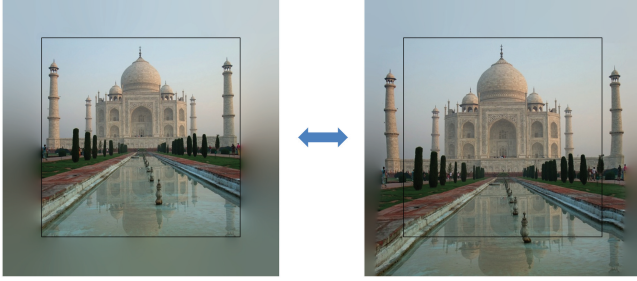


Fig. 13. We extend the image domains beyond their original boundaries (shown by inset rectangles), by transporting pixel values across the images using the extrapolated map and solving a gradient-domain optimization to obtain a seamless reconstruction.

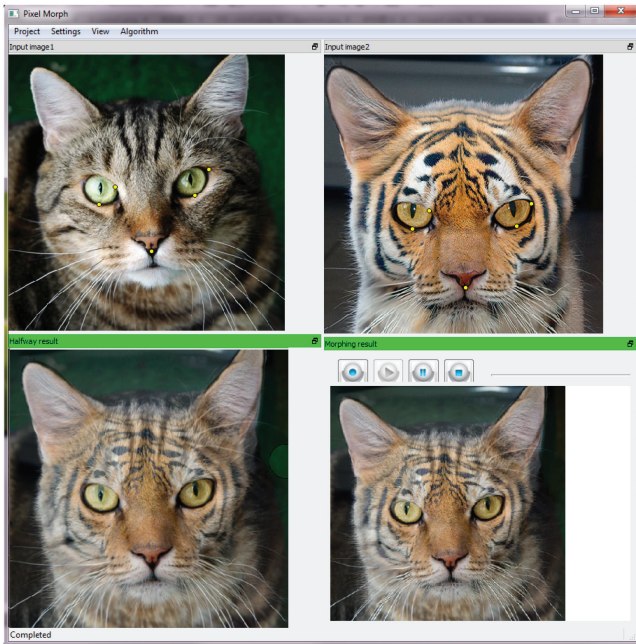


Fig. 14. The interactive morphing system shows the input images, the halfway morph, and an oscillating morph animation.

region of the left image, the gradient-domain optimization yields a diffuse result.

Finally, we perform morphing using the extended images, without any special treatment for boundaries (except clamp-to-border in the unlikely case that the maps ϕ_0 or ϕ_1 were to reach outside the extended images).

8. USER INTERFACE

Our prototype system provides a small set of views and UI controls that allow the user to interactively guide the optimization towards the desired morph effect. A few point correspondence points are typically sufficient to properly align important features to each other. The correspondences between all remaining pixels are automatically found.

Figure 14 shows our interface with an example image pair. The two panes at the top show the input images with overlaid correspondence points. Mouse operations allow the user to quickly add, move,

and remove correspondence points on these images. The bottom-left pane shows the current halfway image. Generally, if this image has no visible ghosting near corresponding object boundaries, the optimizer has achieved a satisfactory result. The user is also able to select correspondence points directly on the intermediate image by clicking the first image position and dragging it to the second image position. Please refer to the accompanying video (accessed in the ACM Digital Library) for a demonstration. Finally, the bottom-right pane displays a morphing animation. The correspondence optimization runs in the background, while the bottom two panes are updated interactively as the optimization progresses. The coarse-to-fine optimization is immediately restarted when the user manipulates the correspondence points. This makes the process of producing convincing morphs fast and intuitive.

9. RESULTS

All our results use the same default parameters in the optimization objective. There is no user intervention beyond the specified pairs of correspondence points, shown in red in all the figures.

Figure 15 compares our approach to several representative morphing techniques using examples extracted from the corresponding papers. Input images with overlaid correspondence points and annotations required by the previous work are shown on the far left and far right. The second image in each row shows the result of the related work, while the third image shows our results. Some traditional techniques require significantly more user guidance than our method to achieve nearly comparable results in these examples [Smythe 1990; Lee et al. 1996; Choi and Hwang 2011; Wu and Liu 2013]. Other techniques are targeted at more specific problems and would be unable to properly handle most of the general examples shown in Figure 16. These include temporal upsampling of similar frames [Mahajan et al. 2009], texture-synthesis-based discontinuous morphs [Shechtman et al. 2010], and methods specific to human faces [Yang et al. 2012]. Gao and Sederberg [1998] explore a similar semi-automatic method, but aim at images with similar colors. Our approach gives excellent results in these cases and our similarity energy lets us handle more generic morphs of scenes with different colors.

Figure 16 shows several results of our approach. The input images are shown on the far left and far right along with overlaid correspondence points used during the morphing process. Note that the top two examples require no correspondence points, providing an accurate result right away. In each row, the three central images show the progression of the morph. These results demonstrate the algorithm's ability to handle different challenging scenarios.

Table I reports the grid resolution and number of correspondence points for each image in Figures 15 and 16. These grid resolutions suffice to achieve an accurate morph. The table also shows a timing breakdown of the different stages of our GPU optimization algorithm. The correspondence optimization (*Solve*) requires about one to seven seconds, depending on the dataset. Even when using dense grids with over half a million points, the optimization takes under seven seconds, which is reasonable for our interactive system. Because the system shows mapping results as the optimization progresses, the user need not wait until the optimization completes to see the effect of newly added correspondence points. The generation of quadratic paths (*Path*) and the computation of the Poisson-extended image boundaries (*Bndry*) take approximately a couple of seconds. Since these do not change the direct pixel correspondences, they are not as important during correspondence point manipulation. Nonetheless, if the system is idle and waiting for

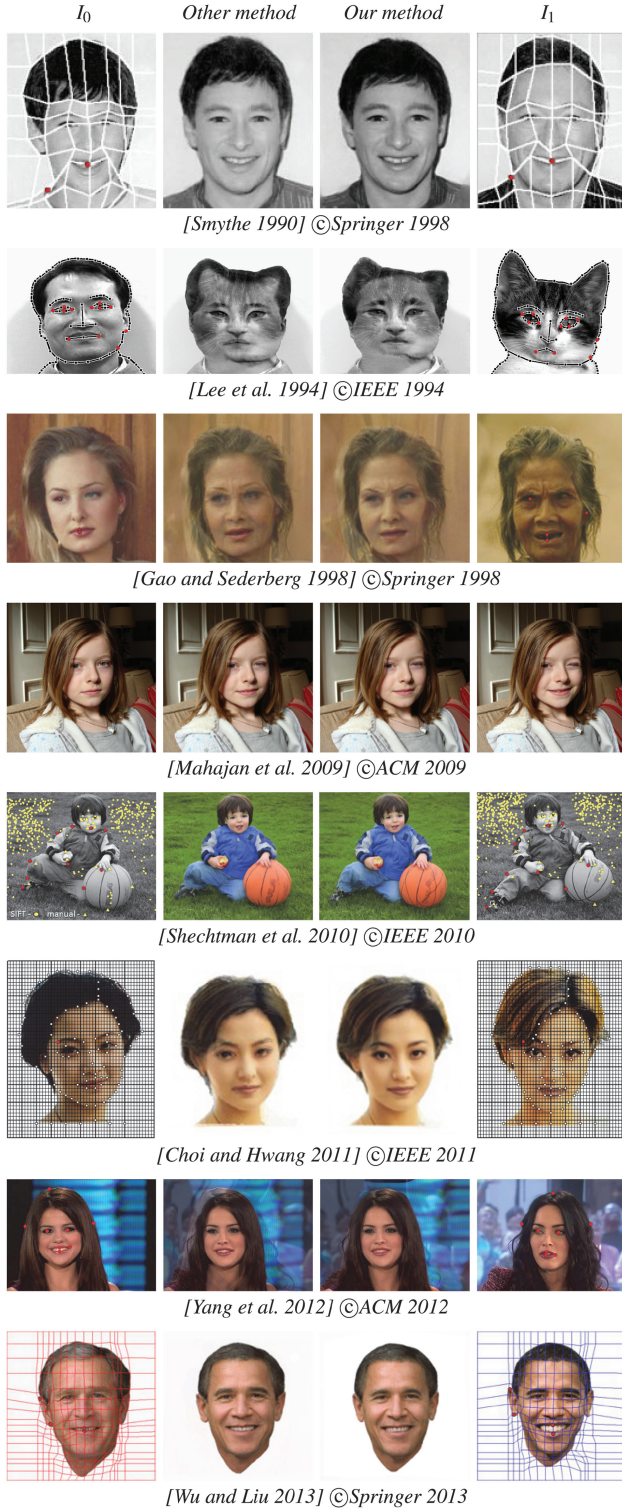


Fig. 15. Comparison with morphs presented in prior methods. Input images I_0 , I_1 are overlaid with the additional input (e.g., grids, points) used in each method, as well as the sparse correspondence points (in red) used in our method.

Table I. Statistics for the Morphs in Figures 15 and 16

Fig	Image	Dataset		Processing times* (sec)			
		Resolution	Points	Solve	Path	Bndry	Total
15	boy	600×480	2	2.5	0.6	0.6	3.7
	man-cat	475×400	12	6.6	2.0	2.0	10.6
	woman	160×160	4	1.4	0.7	0.6	2.7
	girl	270×270	0	1.8	1.0	1.0	3.8
	ball	306×262	10	3.2	0.9	1.0	5.1
	model	97×123	1	2.5	0.4	0.5	3.4
	tv	528×396	8	3.3	1.7	1.9	6.9
	bush-obama	154×208	4	1.8	0.7	0.8	3.3
16	butterfly	306×248	0	2.0	1.2	1.1	4.3
	flower	600×480	0	2.3	2.2	2.4	6.9
	cat-lion	524×800	11	4.6	2.7	4.3	11.6
	monalisa	440×440	2	2.6	1.2	1.8	5.6
	man-lizard	720×540	13	6.3	2.3	2.9	11.5
	caricature	400×448	3	4.7	1.8	2.0	8.5
	lake	462×342	11	4.9	1.5	1.5	7.9

*Hardware: NVIDIA GTX 680 on Intel Core i7 @ 3.4GHz with 8GB RAM.

user input after correspondence minimization, it proceeds to perform these computations to provide a complete result.

Figure 17 shows how the gradual insertion of additional correspondence points progressively improves the quality of the morphing result in complex scenarios.

Our approach does not optimally handle cases where the ideal correspondence is discontinuous due to occlusion. Figure 18 shows such an example, where disoccluded regions on the hair of the subject result in blending artifacts with the background.

10. EXTENSIONS

A limitation of our method is that some complex occlusion cases that involve motion parallax cannot be handled properly with the continuous halfway parameterization that we propose. Some examples are shown in Figures 1 and 2 of our supplementary material. We introduce extensions that allow our method to address these particularly challenging examples and further increase the range of supported morphing effects.

Ignore mask. Our optimization tries to find the best correspondences across the entire input image domains. The mapping between regions that are not important for the desired morph effect (e.g., the background) may have a bad influence on regions that are key (e.g., the foreground). To mitigate this problem, we allow users to delineate regions that will be ignored by the similarity energy term. Pixels flagged by the *ignore mask* do not enter into the computation of the mean, variance, and covariance between neighborhoods in Eq. (6). Figure 19 shows an example with this issue, as well as the improvement achieved with the relaxed constraints due to the ignore mask.

Layer segmentation. We extend our framework to handle a more general class of image transformations, such as complex disocclusions and motion with parallax (see Figure 20). To this end, our interface allows the user to quickly tag different layers using intelligent scissors [Mortensen and Barrett 1995]. These layers are processed independently and then composited (respecting their depth order). When processing each layer, pixels beyond its boundary are also flagged with an ignore mask. Intermediate image pixels that do not overlap with any of the layers are filled in using the same gradient-domain optimization process used for the Poisson-extended image boundaries. In the example of Figure 20, independent vector fields are computed for the ball,



Fig. 16. Example morphing sequences. User-drawn correspondence points are shown in red over the input images I_0 (left) and I_1 (right). Caricature by ©Court Jones.

goalkeeper, and background, leading to improved results. The impact on performance is small since the expensive similarity term need not be computed for masked regions.

Extrapolating caricatures using quadratic motion paths. We can use our technique to extrapolate the morphing sequence

past the original input images in either direction. This is particularly appealing when morphing from a photograph to a caricature of that photograph, as shown in the two rightmost images of Figure 21. Note that, by using our novel quadratic motion paths that minimize deformations during the morph, local distortions in the resulting image are greatly reduced when compared to linear motion paths.

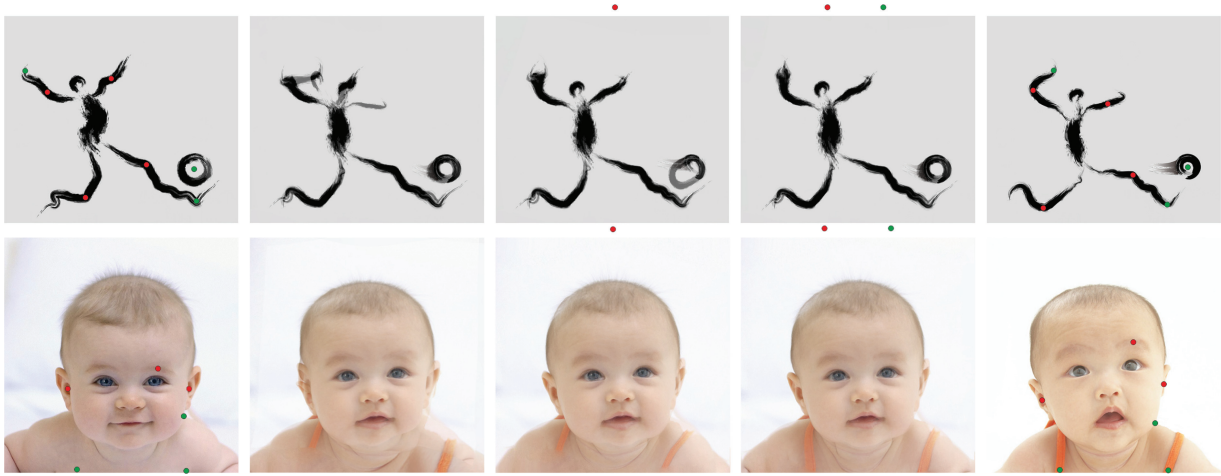


Fig. 17. Given the input images at left and right, the three middle images show halfway morphs obtained with various subsets of control points (no control points, red points only, and both red and green points). Ghosting due to misalignment is progressively reduced.



Fig. 18. Blending artifacts may occur on disoccluded regions, as shown in this rotating head example.

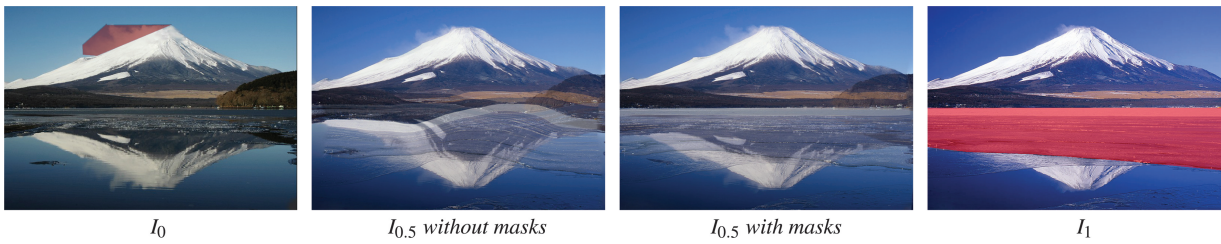


Fig. 19. By introducing a set of “ignore” masks (shown in red), we can avoid undesirable distortions caused by false matching of dissimilar regions.

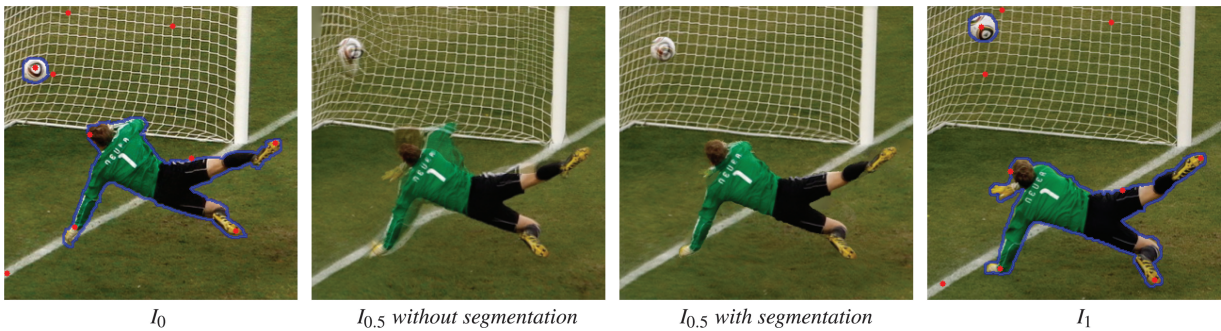


Fig. 20. Segmenting a morph into layers (ball, player, and background) improves results in cases of disocclusion and motion parallax.

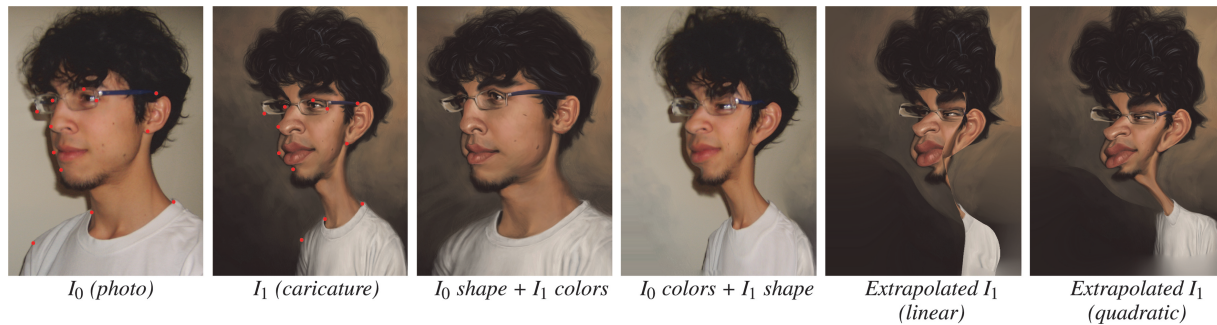


Fig. 21. Examples of decoupling shape and color interpolation during a morph and extrapolating the motion paths. Caricature by ©Court Jones.

Decoupling shape and color transitions. The fine-grained matching provided by our approach allows us to effectively combine colors from one image with the shape of the other image without significant artifacts. The two middle images in Figure 21 show the combinations of the shape from a photograph with painted detail from the caricature and vice versa.

11. CONCLUSIONS AND FUTURE WORK

In this work, we presented a variety of contributions to the well-studied problem of generating high-quality morphs between two images. The use of a halfway parameterization for establishing correspondences provides a way to handle simple occlusions without compromising the optimization process. Handling more complex occlusions and rotations remains an area of future work. Our similarity metric allows neighborhoods with the same structure to be matched, even those with distinct color distributions and geometric distortions. An interactive user interface, powered by an efficient GPU implementation of a coarse-to-fine optimization algorithm, allows artists to quickly specify desired point correspondences to achieve the desired morphs. A quadratic path optimization reduces the distortions that arise from linear paths during the animation sequence. Finally, we introduced a fast, per-pixel iterative search algorithm for inverting the mapping. It does not require mesh rasterization and, in conjunction with our joint Poisson boundary extension, enables out-of-bounds matches when input image domains do not exactly correspond.

As future work, in addition to handling more complex occlusions, we would like to explore the entire space of color and motion paths, perhaps even allowing explicit user control with a stroke-based interface. We would also like to consider extending this technique to morphing between video sequences.

ACKNOWLEDGMENTS

The caricatures in Figures 1, 16, and 21 are courtesy of Court Jones (<http://www.courtjones.com/>), who retains their copyright.

REFERENCES

- S. Baker, D. Scharstein, J. P. Lewis, S. Roth, M. J. Black, and R. Szeliski. 2011. A database and evaluation methodology for optical flow. *Int. J. Comput. Vis.* 92, 1, 1–31.
- P. N. Belhumeur and D. Mumford. 1992. A bayesian treatment of the stereo correspondence problem using half-occluded regions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 506–512.
- D. Beymer. 1996. Feature correspondences by interleaving shape and texture computations. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 921–928.
- M. Bichsel. 1996. Automatic interpolation and recognition of face images by morphing. In *Proceedings of the International Conference on Automatic Face and Gesture Recognition*. 128–135.
- M. Covell. 1996. Eigen-points: Control-point location using principal component analyses. In *Proceedings of the International Conference on Automatic Face and Gesture Recognition*. 122–127.
- S. Darabi, E. Shechtman, C. Barnes, D. B. Goldman, and P. Sen. 2012. Image melding: Combining inconsistent images using patch-based synthesis. *ACM Trans. Graph.* 31, 4, 82.
- M. Finch, J. Snyder, and H. Hoppe. 2011. Freeform vector graphics with controlled thin-plate splines. *ACM Trans. Graph.* 30, 6, 166.
- P. Gao and T. W. Sederberg. 1998. A work minimization approach to image morphing. *The Visual Comput.* 14, 8–9, 390–400.
- J. Gomes, L. Darsa, B. Costa, and L. Velho. 1999. *Warping and Morphing of Graphical Objects*. Morgan Kaufmann, San Francisco.
- S. B. Kang, M. Uyttendaele, S. Winder, and R. Szeliski. 2003. High dynamic range video. *ACM Trans. Graph.* 22, 3, 319–325.
- M. Kass, A. Witkin, and D. Terzopoulos. 1988. Snakes: Active contour models. *Int. J. Comput. Vis.* 1, 4, 321–331.
- I. Kemelmacher-Shlizerman, E. Shechtman, R. Garg, and S. M. Seitz. 2011. Exploring photobios. *ACM Trans. Graph.* 30, 4, 61.
- J. Kiefer. 1953. Sequential minimax search for a maximum. *Proc. Amer. Math. Soc.* 4, 3, 502–506.
- S. Lee, K. Y. Chwa, J. K. Hahn, and S. Y. Shin. 1994. Image morphing using deformable surfaces. In *Proceedings of the Conference on Computer Animation*. 31–39.
- S. Lee, K. Y. Chwa, J. K. Hahn, and S. Y. Shin. 1996. Image morphing using deformation techniques. *J. Visual. Comput. Animat.* 7, 1, 3–23.
- Z. Liu, C. Liu, H. Shum, and Y. Yu. 2002. Pattern-based texture metamorphosis. In *Proceedings of the Pacific Conference on Computer Graphics and Applications*. 184–191.
- D. Mahajan, F.-C. Huang, W. Matusik, R. Ramamoorthi, and P. Belhumeur. 2009. Moving gradients: A path-based method for plausible image interpolation. *ACM Trans. Graph.* 28, 3, 42.
- W. Matusik, M. Zwicker, and F. Durand. 2005. Texture design using a simplicial complex of morphable textures. *ACM Trans. Graph.* 24, 3, 787–794.
- E. Mortensen and W. Barrett. 1995. Intelligent scissors for image composition. In *Proceedings of the Annual ACM SIGGRAPH Conference on Computer Graphics and Interactive Techniques*. 191–198.

- T. Nishita, T. Fujii, and E. Nakamae. 1993. Metamorphosis using Bezier clipping. In *Proceedings of the Pacific Conference on Computer Graphics and Applications*. 162–173.
- P. Perez, M. Gangnet, and A. Blake. 2003. Poisson image editing. *ACM Trans. Graph.* 22, 3, 313–318.
- J. Rossignac and A. Vinacua. 2011. Steady affine motions and morphs. *ACM Trans. Graph.* 30, 5, 116.
- M. Ruzon and C. Tomasi. 2001. Edge, junction, and corner detection using color distributions. *IEEE Trans. Pattern Anal. Mach. Intell.* 23, 11, 1281–1295.
- D. Scharstein and R. Szeliski. 2002. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *Int. J. Comput. Vis.* 47, 1, 7–42.
- E. Shechtman, A. Rav-Acha, M. Irani, and S. M. Seitz. 2010. Regenerative morphing. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 615–622.
- H. Shum, S. Kang, and S. Chan. 2003. Survey of image-based representations and compression techniques. *IEEE Trans. Circ. Syst. Video Technol.* 13, 11, 1020–1037.
- D. B. Smythe. 1990. A two-pass mesh warping algorithm for object transformation and image interpolation. Tech. rep. 1030, ILM.
- Z. Wang, A. Bovik, H. Sheikh, and E. Simoncelli. 2004. Image quality assessment: From error visibility to structural similarity. *IEEE Trans. Image Process.* 13, 4, 600–612.
- G. Wolberg. 1990. *Digital Image Warping*. IEEE Computer Society Press.
- G. Wolberg. 1998. Image morphing: A survey. *The Visual Comput.* 14, 8, 360–372.
- E. Wu and F. Liu. 2013. Robust image metamorphosis immune from ghost and blur. *The Visual Comput.* 29, 4, 311–321.
- F. Yang, E. Shechtman, J. Wang, L. Bourdev, and D. Metaxas. 2012. Face morphing using 3D-aware appearance optimization. In *Proceedings of the Graphics Interface Conference*. 93–99.
- L. Yang, Y.-C. Tse, P. V. Sander, J. D. Lawrence, D. Nehab, H. Hoppe, and C. L. Wilkins. 2011. Image-based bidirectional scene reprojection. *ACM Trans. Graph.* 30, 6, 150.
- K. Yucer, A. Jacobson, A. Hornung, and O. Sorkine. 2012. Transfusive image manipulation. *ACM Trans. Graph.* 31, 6, 176.
- Y. Zhang. 1996. A fuzzy approach to digital image warping. *IEEE Comput. Graph. Appl.* 16, 4, 34–41.
- B. Zitova and J. Flusser. 2003. Image registration methods: A survey. *Image Vis. Comput.* 21, 11, 977–1000.

Received July 2013; revised December 2013; accepted February 2014