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

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 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 paper. 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 estimating image correspondences automatically. However, these techniques assume the images are closely related, e.g., multiple views of a consistent scene under similar lighting conditions. In contrast, morphing often considers images of different objects. 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 show that the other steps in morphing also greatly benefit from 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; 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 survey in [Shum et al. 2003]) and for temporal upsampling of video sequences [e.g., Kang et al. 2003; Mahajan et al. 2009; Yang et al. 2011]. Nevertheless, for specialized image domains (such as faces or periodic textures), correspondences can be found automatically [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. Matsukis et al. [2005] derives 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.

Yu 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 subject 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 non-monotonic 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 interpolate the components independently [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 \( \phi \) from \( I_0 \) to \( I_1 \). One challenge is that each image may contain “disoccluded” regions
We first present the energy function that the algorithm minimizes to define the correspondence field \( \mathbf{v} \) plotted as a graph on the right.

which are invisible in the other image. Within disoccluded regions of \( I_0 \), map \( \phi \) 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.

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 \( \mathbf{v} \) over a domain \( \Omega \) halfway between the two images, and use this field to create two maps (see figure 2):

\[
\phi_0(p) = p - \mathbf{v}(p) \quad \text{and} \quad \phi_1(p) = p + \mathbf{v}(p).
\]

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 \( \mathbf{v} \) 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 \( \mathbf{v} \) 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 \( \mathbf{v} \) (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 \( \mathbf{v}(p) \) 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.

4.1 Energy Function

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

\[
E = \sum_{p \in \Omega} E(p), \quad \text{with} \quad E(p) = E_{\text{SIM}}(p) + \lambda E_{\text{TPS}}(p) + \gamma E_{\text{UI}}(p).
\]

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

**Similarity energy.** To measure if 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 \( \Omega \), 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 halfpixel point \( p \), we use bilinear interpolation to gather the corresponding warped neighborhoods from images \( I_0 \) and \( I_1 \). Then, we evaluate

\[
E_{\text{SIM}}(p) = \frac{1}{wh} \text{SIM}(N_0(p), N_1(p)), \quad \text{with} \quad N_0 = I_0(\phi_0(N(p))) \quad \text{and} \quad N_1 = I_1(\phi_1(N(p))).
\]

(Normalization by \( wh \) makes the term resolution-independent, and 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 \( \sigma_{N_0N_1} \), because swapping colors across an edge should not affect the energy. Formally:

\[
\text{SIM}(N_0, N_1) = c(N_0, N_1) \cdot s(N_0, N_1), \quad \text{with} \quad c(N_0, N_1) = \frac{2\sigma_{N_0N_1}C_2}{\sigma_{N_0}^2 + \sigma_{N_1}^2 + C_2} \quad \text{and} \quad s(N_0, N_1) = \frac{\sigma_{N_0N_1} + C_1}{\sigma_{N_0}^2 + \sigma_{N_1}^2 + C_3}.
\]

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 components \( v_x \) and \( v_y \) of vector field \( \mathbf{v} \):

\[
E_{\text{TPS}}(p) = \text{TPS}(v_x(p)) + \text{TPS}(v_y(p)), \quad \text{where} \quad \text{TPS}(v_x(p)) = \left( \frac{\partial^2 v_x(p)}{\partial p_x^2} \right)^2 + 2\left( \frac{\partial^2 v_x(p)}{\partial p_x \partial p_y} \right)^2 + \left( \frac{\partial^2 v_x(p)}{\partial p_y^2} \right)^2.
\]

\[
E_{\text{UI}}(p) = \text{UI}(v_x(p)) + \text{UI}(v_y(p)), \quad \text{where} \quad \text{UI}(v_x(p)) = \left( \frac{\partial v_x(p)}{\partial p_x} \right)^2 + \left( \frac{\partial v_x(p)}{\partial p_y} \right)^2 + \left( \frac{\partial^2 v_x(p)}{\partial p_x^2} \right)^2 + 2\left( \frac{\partial^2 v_x(p)}{\partial p_x \partial p_y} \right)^2 + \left( \frac{\partial^2 v_x(p)}{\partial p_y^2} \right)^2.
\]
We repeat the optimization for all points and analogously for \( TPS(v_y(p)) \) (see Figure 5). The TPS energy is defined over the halfway domain grid, and is 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 \( u_1^1 \) and \( u_1^4 \) in images \( I_0 \) and \( I_1 \). Let \( \bar{u}_i = (u_1^1 + u_1^4)/2 \) and \( v_{ui} = (u_1^1 - u_1^4)/2 \). The correspondence between \( u_1^1 \) and \( u_1^4 \) forces the halfway vector \( v(\bar{u}_i) \) to equal \( v_{ui} \).

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

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

Therefore, the UI energy is

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

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

### 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 \( 8 \times 8 \) or \( 16 \times 16 \) 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 in Figure 6. We repeat the optimization for all points \( p \) until convergence.

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.

ACM Transactions on Graphics, Vol. VV, No. N, Article XXX, Publication date: Month YYYY.
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 \( \alpha \) spans the time interval \([0, 1]\). The typical approach is to use linear interpolation:

\[
q_\alpha = p + (2\alpha - 1)v(p). \tag{13}
\]

Unfortunately, when considered as a whole, linear motion paths can lead to undesirable (often non-monotonic) 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:

\[
q_\alpha = (1 - \alpha)((1 - \alpha)q_0 + \alpha b_{1/2}) + \alpha((1 - \alpha)b_{1/2} + \alpha q_1) \tag{14}
\]

\[
= p + (2\alpha - 1)v(p) + 4\alpha(1 - \alpha)w(p). \tag{15}
\]

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 \( \Omega \). Define the vectors connecting them in images \( I_0 \) and \( I_1 \):

\[
d_0(p_i, p_j) = \phi_0(p_j) - \phi_0(p_i) \tag{16}
\]

\[
d_1(p_i, p_j) = p_j - p_i + v(p_j) - v(p_i). \tag{17}
\]

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

\[
\begin{align*}
\tilde{d}_{1/2}(p_i, p_j) &= \sqrt{\|d_0(p_i, p_j)\| \|d_1(p_i, p_j)\|} \hat{d}_1(p_i, p_j), \tag{19}
\end{align*}
\]

with \( \hat{d}_1(p_i, p_j) = \frac{d_1(p_i, p_j) + d_2(p_i, p_j)}{2} \) and \( \hat{u} = u/\|u\| \). \tag{20}

The halfway vector we actually obtain from the quadratic path is

\[
d_{1/2}(p_i, p_j) = p_j - p_i + \left( w(p_j) - w(p_i) \right). \tag{21}
\]

Therefore we set the deformation energy term to

\[
\begin{align*}
E_D(w) = \sum_{p_i, p_j} \|d_{1/2}(p_i, p_j) - \tilde{d}_{1/2}(p_i, p_j)\|^2. \tag{22}
\end{align*}
\]

For the resting energy term, we use

\[
E_R(w) = \sum_{p_i, p_j, \|v(p_i)\| < 1} \left(1 - \|v(p_i)\|\right) \|w(p_j)\|^2. \tag{23}
\]

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

\[
E(w) = E_D(w) + \beta E_R(w). \tag{24}
\]

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

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 evaluated independently. 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 equation (13) so that, given a point \( q_\alpha \) in the intermediate image \( I_\alpha \), we find the corresponding point \( p \).
We instead invert equation (4)

\[ f(y(x)) = x \]  

The same approach works with quadratic motion paths, except that 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:

\[ p^{(0)} = q \alpha, \]
\[ v^{(0)} = v(p^{(0)}), \]
\[ p^{(i+1)} = q \alpha - (2 \alpha - 1) v(p^{(i)}), \]
\[ v^{(i+1)} = (\eta) v(p^{(i+1)}) + (1 - \eta) v^{(i)}. \]

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

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. We 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 \( \tilde{I}_0 \), e.g., 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 \( q_0 \in \Omega_0 \), 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, which 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
Automating Image Morphing using Structural Similarity on a Halfway Domain

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 found automatically.

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 for a demonstration. Finally, the bottom-right pane displays a morphing animation. The correspondence optimization runs in the background, and 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 aimed 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.
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.
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 does not need to 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 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
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.

Table I. Statistics for the morphs in figures 15 and 16.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Processing times* (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fig: 15</td>
<td></td>
</tr>
<tr>
<td>boy</td>
<td>600\times480 2</td>
</tr>
<tr>
<td>man-cat</td>
<td>475\times400 12</td>
</tr>
<tr>
<td>woman</td>
<td>160\times160 4</td>
</tr>
<tr>
<td>girl</td>
<td>270\times270 0</td>
</tr>
<tr>
<td>ball</td>
<td>306\times262 10</td>
</tr>
<tr>
<td>model</td>
<td>97\times123 1</td>
</tr>
<tr>
<td>tv</td>
<td>528\times396 8</td>
</tr>
<tr>
<td>bush-obama</td>
<td>154\times208 4</td>
</tr>
<tr>
<td>Fig: 16</td>
<td></td>
</tr>
<tr>
<td>butterfly</td>
<td>306\times248 0</td>
</tr>
<tr>
<td>flower</td>
<td>600\times480 0</td>
</tr>
<tr>
<td>cat-lion</td>
<td>524\times800 11</td>
</tr>
<tr>
<td>monalisa</td>
<td>440\times440 2</td>
</tr>
<tr>
<td>man-lizard</td>
<td>720\times540 13</td>
</tr>
<tr>
<td>caricature</td>
<td>400\times448 3</td>
</tr>
<tr>
<td>lake</td>
<td>462\times342 11</td>
</tr>
</tbody>
</table>

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

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 equation (6). Figure 19 shows an example with this issue, and 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 that end, our interface allows the user 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, goalkeeper, and background, leading to improved results. The impact on performance is small since the expensive similarity term does not need to 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.
Automating Image Morphing using Structural Similarity on a Halfway Domain

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.

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 that 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 correspond exactly.

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

NVIDIA Corporation has generously donated the GPUs used in this project. The project was partly supported by Hong Kong GRF grants #619509 and #618513. The caricatures in figures 1, 16, and 21 are courtesy of Court Jones (http://www.courtjones.com/), who retains their copyright.

REFERENCES


REFERENCES


Received July 2013; accepted 7 2014