Regenerative Morphing

Eli Shechtman[1,3] Alex Rav-Acha[2] Michal Irani[2] Steve Seitz[3,4]
[1]Adobe Systems [2]Weizmann Institute of Science [3]University of Washington [4]Google

Abstract

We present a new image morphing approach in which the output sequence is regenerated from small pieces of the two source (input) images. The approach does not require manual correspondence, and generates compelling results even when the images are of very different objects (e.g., a cloud and a face). We pose the morphing task as an optimization with the objective of achieving bidirectional similarity of each frame to its neighbors, and also to the source images. The advantages of this approach are 1) it can operate fully automatically, producing effective results for many sequences (but also supports manual correspondences, when available), 2) ghosting artifacts are minimized, and 3) different parts of the scene move at different rates, yielding more interesting (and less robotic) transitions.

1. Introduction

Image morphing is a popular visual effect in which one image is transformed into another. It has been explored since late 80's both in the research community and in the movie industry and today is a common tool with many commercial solutions. Traditional morphing methods [18, 3, 6, 11, 15] combine a geometric warp and an intensity blend. The warping step requires a dense correspondence; this is problematic because of 1) the need for a tedious manual annotation (in most cases), and 2) artifacts are created where the correspondence does not exist (e.g., due to occlusions), or where they are not consistent with the warping model (e.g., complex motion).

In this work we revisit the morphing task and propose a new approach, which we call *regenerative morphing*, in which the output sequence is regenerated from small pieces of the source images. In many cases, we show compelling morphs without the need for any correspondences. Two such examples are shown in (Fig. 1). In the first, a cloud is morphed into a face and in the second one flower bouquet is transformed into another. Note the complexity of these examples as there is barely any visual similarity between the face and the cloud, and the difference in color, layout and quantity of flowers in the second. In both cases, our method manages to *automatically* move, deform and blend locally the visual content of the two images in a compelling way to generate a visually pleasing transform. When user-provided correspondences are available, they can be easily incorporated to provide additional constraints and further improve the transitions.

Figure 1. Regenerative morphing results of a cloud into a face, and one flower bouquet into another. Both are *fully automatic*. In the first example, it is hard to define (even manually) a set of correspondences between the source images, as required by traditional morphing. The resulting morphs evolve smoothly from one source image to the other without loosing their sharp appearance. Note the non-trivial transitions: facial features (eyes, mouth) deform and merge into close-by similar cloud patterns, and similar flowers move towards each other while merging in a seamless way.

One common goal of any morphing method is that the output is *temporally coherent*. However common morphing artifacts ("ghosting" and blurriness due to blending, and fold-overs and "holes" due to warping failure) can be perfectly temporally coherent. What makes these artifact disturbing is their visual appearance - often it is some local region which looks unnatural, i.e., is not representative of the content in the source images. Our key observation is that a good morph should be not only temporally coherent but should also be visually similar to either of the sources at every local region within the morph sequences. Imposing such *source similarity* across the output sequence will suppress generation of artifacts that do not exist in the sources. Moreover, unlike traditional morphing where artifacts in the inner frames are treated *indirectly* by fixing or adding more correspondences (when possible) to the source images, con-

straining local similarity on each inner frame to the sources, *directly* addresses such artifacts.

We pose morphing as an optimization problem with an objective function that captures both *source similarity* and *temporal coherence*. Our optimization framework is inspired by recent example-based methods [17, 10, 16, 2] that showed impressive image and video synthesis results for related problems such as retargeting, reshuffling, hole-filling and texture synthesis. The objective functions used in these methods capture the local similarity of the synthesized output to the input, based on similarities between small overlapping image patches at multiple scales.

In particular we build upon the *bidirectional similarity* (BDS) method [16] that posed the problem of image summarization as a maximization of a bidirectional similarity function between the input and output. This concept is particulary useful for the morphing task since by requiring that each patch at every frame is similar to a source patch (called *"coherence"* in [16]) we get good coverage of all regions in the frames (no "holes"), avoid artifacts and can gracefully handle occlusions (a patch need appear in only one of the sources). By requiring patches in the sources to exist in the output sequence (called *"completeness"*), and by controlling the relative contributions of patches from each source we can control the degree of similarity of the in-between frames to the sources. Temporal coherence is achieved by encouraging bidirectional similarity between each frame and its neighboring frames. In this work we introduce generalizations of bidirectional similarity to multiple sources as well as to *relative similarity* (a frame is α-similar to one source and $(1-\alpha)$-similar to the second). Using these notions we construct an objective function that captures the two properties of a good morph - *source similarity* and *temporal coherence*, and propose an iterative patch-sampling based algorithm to optimize it.

Regenerative Morphing can be applied to various scenarios, ranging from interpolation of views of rigid objects (also called *view interpolation* [6, 15, 9, 8]), to interpolating views of non-rigidly moving objects [14], to morphing between totally different images with no clear correspondences. In many cases of interest, compelling morphs can be generated fully automatically. In other cases, we show that adding a few manual points further improves results, and outperforms prior morphing tools with the same manual correspondences. One nice property of our morphing approach is that different parts of the objects move non-linearly and at varying rates, resulting in more interesting and less "robotic" looking motions.

2. Morphing as an example-based optimization

Morphing problem definition Given two source (input) images, S_1 and S_2, the goal is to generate a sequence of $N-1$ target (in-between) frames - $\{T_n\}_1^{N-1}$ (where we define $T_0 := S_1$ and $T_N := S_2$). Such a sequence should have the following two properties:

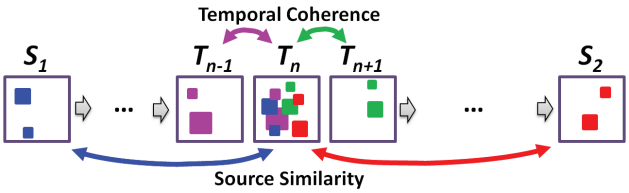

Figure 2. Notations and schematic representation of the morphing objective. We want local similarities between each frame T_n to the two neighboring frames T_{n-1} and T_{n+1} (*Temporal Coherence*), as well as to the two sources S_1 and S_2 (*Source Similarity*).

1. *Temporal Coherence* - the changes across the sequence should be temporally smooth. This is a necessary requirement for any morphing method but not a sufficient one as it does not supply enough constraints on how the visual content should evolve. For example a simple "cross-dissolve" (with the typical "ghosting" artifacts) or a sequence that gets gradually blurry towards the middle frame, can both be considered temporally coherent.

2. *Source Similarity* - every region in every frame should be similar to some region in either of the sources. This will prevent the appearance of the sequence from deviating too much from the source images, thus avoiding artifacts like loss of detail or generation of new content (such as "ghosting"). In addition, we want the similarity to gradually change from one source image to the other.

We transform these two requirements into the following unified objective function:

$$E_{Morph}(T_1,...,T_{N-1}) = E_{SourceSim}(T_1,...,T_{N-1}) + \beta E_{TempCoher}(T_1,...,T_{N-1}) \quad (1)$$

Intuitively, $E_{TempCoher}$ is used to achieve temporal smoothness of appearance as well as smoothness of motion, therefore it should require similarity of all local regions in T_n to close-by regions in T_{n-1} and T_{n+1}. $E_{SourceSim}$ has two goals: First, each frame T_n should be composed of local regions that are similar to regions in either S_1 or S_2 or both, and second, to have a continuous control over the *relative similarity* to each of the sources so it can be changed gradually from S_1 to S_2. These notations and the two terms are schematically shown in Fig. 2. In this work we build upon the Bidirectional Similarity (BDS) method [16] that is based on similar concepts. Therefore we briefly review this method.

2.1. Bidirectional Similarity

The BDS method [16] was originally designed for image and video summarization (retargeting) and was shown to be useful for a variety of other image editing applications. It is based on a bidirectional distance measure between pairs of images - the source image S and a target image T. The measure consists of two terms: (1) The *completeness* term ensures that the output image contains as much visual information from the input as possible, making it a good summary. (2) The *coherence* term ensures that the output is coherent with respect to the input and that new visual structures (artifacts) are penalized. Formally, the distance measure is defined simply as the sum of the average distance of all image patches (e.g., 7x7 pixels) in S to their most similar (nearest-neighbor) patches in T and vice versa:

$$d_{BDS}(S,T) = \overbrace{\frac{1}{N_S}\sum_{s\subset S}\min_{t\subset T} D(s,t)}^{d_{Complete}(S,T)} + \overbrace{\frac{1}{N_T}\sum_{t\subset T}\min_{s\subset S} D(t,s)}^{d_{Coher}(S,T)} \tag{2}$$

where the distance D is the SSD (sum of squared differences) of patch pixel values in L*a*b* color space. This distance measure is minimized by an iterative algorithm to solve for T under various constraints. Starting with some initial guess for the output image, nearest neighbor (NN) correspondences are computed for all overlapping patches in S to all the patches in T (the *completeness* term), and for all patches in T to all the patches in S for (the *coherence* term). Next, "color-voting" is performed to accumulate the pixel colors in T from all overlapping neighbor patches in T and all the patches in S that have their NN overlapping that pixel. Next, the color "votes" are averaged to minimize the sum of square color differences, to generate a new image. These steps are repeated until convergence. See more details in [16].

2.2. Relative similarity with multiple sources

For the *Source Similarity* term we want the target be α-similar to the first source and $(1-\alpha)$-similar to the second. We explore next a few possible generalizations of BDS to two sources, and then apply these generalizations to construct an objective function for the task of generating a morph sequence. It turns out that the *coherence* and *completeness* terms can have different generalizations to multiple sources, so we first present the α-blended generalization of both and then another useful generalization of the *coherence* term.

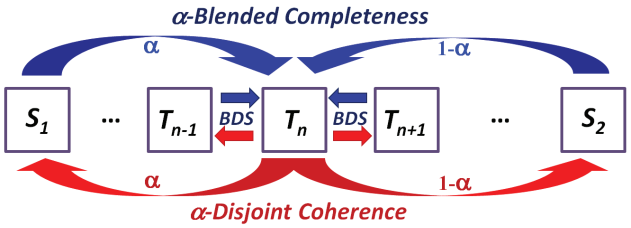

Figure 3. The *Temporal Coherency* and *Source Fidelity* terms. The first is a simple average of the BDS of T_n with its neighbor to the right, and the BDS with its neighbor to the left. The later is a sum of the $\alpha - BlendedCompletness$ and the $\alpha - DisjointCoherence$ of T_n with the two sources.

α-Blended Bidirectional Similarity - One straight forward way to compute the bidirectional distance between a target and two sources with an α-similarity parameter, is by a simple convex summation of the two BDS terms. We call this *α-Blended Bidirectional Similarity*:

$$\begin{aligned} &d_{\alpha BlendBDS}(T,S_1,S_2,\alpha) = \\ &\quad \alpha\cdot d_{BDS}(T,S_1) + (1-\alpha)\cdot d_{BDS}(T,S_2) \end{aligned} \tag{3}$$

Equivalently we can rewrite this as a sum of an *α-Blended Completeness* and an *α-Blended Coherence* terms, where both are interpolated linearly:

$$\begin{aligned} &d_{\alpha BlendCohere}(T,S_1,S_2,\alpha) = \\ &\quad \alpha\cdot d_{Coher}(T,S_1) + (1-\alpha)\cdot d_{Coher}(T,S_2) \\ &d_{\alpha BlendComplete}(T,S_1,S_2,\alpha) = \\ &\quad \alpha\cdot d_{Complete}(T,S_1) + (1-\alpha)\cdot d_{Complete}(T,S_2). \end{aligned} \tag{4}$$

The *α-Blended Completeness* term, can be rewritten as follows from Eqs. (4), (2):

$$\begin{aligned} &d_{\alpha BlendComplete}(T,S_1,S_2,\alpha) = \\ &\frac{\alpha}{N_{S_1}}\sum_{s_1\subset S_1}\min_{t\subset T} D(s_1,t) + \frac{1-\alpha}{N_{S_2}}\sum_{s_2\subset S_2}\min_{t\subset T} D(s_2,t) \end{aligned} \tag{5}$$

This term contains two separate summations over patches in S_1 and S_2, therefore it can be minimized by having *disjoint* regions in T, where each such region is similar to S_1 or S_2 but not necessarily to both. These disjoint regions in T tend to retain the original sharpness of each of the sources. α gives control over the relative importance of patches from S_1 over patches from S_2 and implicitly controls over the relative size of regions in T that are similar to S_1 over those that are similar S_2.

The *α-Blended Coherence* term can be rewritten using Eqs. (4), (2) as:

$$\begin{aligned} &d_{\alpha BlendCoher}(T,S_1,S_2,\alpha) = \\ &\frac{1}{N_T}\sum_{t\subset T}\left(\alpha\min_{s_1\subset S_1} D(s_1,t) + (1-\alpha)\min_{s_2\subset S_2} D(s_2,t)\right) \end{aligned} \tag{6}$$

This term contains a single summation of a weighted sum of two SSD terms for each pixel in T, thus the minimum is obtained by a weighed average of the color "votes" from S_1 and S_2. Specifically, for each image pixel t in T and a

patch that overlaps it, we find a similar patch in S_1 and a similar patch in S_2 and do a weighted average of their pixel colors at corresponding locations with α and $1-\alpha$ weights. When the sources are different, this blending tends to cause blur (even if all three images T, S_1 and S_2 are sharp), due to inherent geometric and color misalignments. To avoid this undesired artifact, we replace the α-*Blended Coherence* with a non-blurry coherence term - α-*Disjoint Coherence*.

α-Disjoint Coherence - A simple way to obtain disjointedness in the coherence term is by choosing for each patch in T the best patch in either S_1 or S_2, i.e., $\min_{s\subset\{S_1\cup S_2\}} D(s,t)$. However it does not provide any control over the *relative similarity* to S_1 vs. S_2. We obtain this control using the following α-*Disjoint Coherence* term:

$$d_{\alpha DisjCoher}(T,S_1,S_2,\alpha) = \frac{1}{N_T}\sum_{t\subset T}\min\left(\min_{s_1\subset S_1} D(s_1,t),\ \min_{s_2\subset S_2} D(s_2,t)+D_{bias}(\alpha)\right), \tag{7}$$

where $D_{bias}(\alpha)$ is an bias (positive or negative) that allows us to favor selecting patches from one source than from the other. One useful control we can get through $D_{bias}(\alpha)$ is over the relative *area size* in T of "votes" from S_1 over the area of "votes" from S_2. Thus, we set $D_{bias}(\alpha)$ to be such that the portion of patches taken from S_1 will be α times the area of T[1]. By changing α gradually from 0 to 1 we can enforce larger and larger regions in T to be similar to S_1.

Next we will show how to combine these *completeness* and *coherence* terms into a single objective function.

2.3. The morphing objective function

Recall the morphing objective function from Eq. (1): $E_{Morph} = E_{TempCoher} + \beta E_{SourceSim}$. The *Temporal Coherence* term makes each target frame bidirectionally similar to its neighboring frames. Since consecutive frames are expected to be visually similar, we use the α-*Blended Bidirectional Similarity* (Eq. (3)) with $\alpha = 0.5$ over the sequence for this term, i.e. -

$$E_{TempCoher}(\{T_n\}_1^{N-1}) = \sum_{n=1}^{N-1} d_{\alpha BlendBDS}(T_n, T_{n-1}, T_{n+1}, 0.5) \tag{8}$$

where we define $T_0 := S_1$ and $T_{N+1} := S_2$. We found that using only two adjacent neighbors was adequate for the results shown in the paper, but larger temporal neighborhoods can be used as well. For the *Source Similarity* term, since the two source images could be substantially different, we use α-*Disjoint Coherence* (Eq. (7)) and α-*Blended Completeness* term (Eq. (5)), both have the disjointness property

[1]For this term we first compute $D_{bias}(\alpha)$ as the α percentile of the per-pixel differences between distances between T to S_1 and T to S_2. Then for each patch in T we choose the best patch from *either* S_1 *or* S_2, after adding the D_{bias} threshold to the S_2 distances.

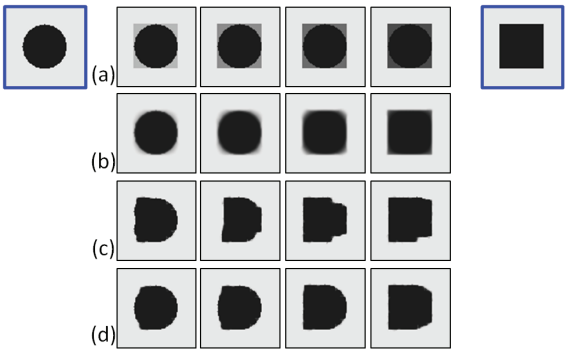

Figure 4. Squaring the circle - morphing the input square into a circle (blue frames). (a) Simple intensity blending generates a "ghosting" effect; (b) Optimizing only *temporal coherence* blurs regions with different structures; (c) Optimizing only *source similarity* gives a sharp but incoherent sequence; (d) Optimizing both terms gives a temporally smooth and sharp sequence.

that retains sharpness, while decreasing linearly α across the sequence (from $1-\frac{1}{N}$ to $\frac{1}{N}$):

$$E_{SourceSim}(\{T_n\}_1^{N-1}) = \sum_{n=1}^{N-1} d_{\alpha BlendComplete}(T_n,S_1,S_2,\tfrac{n}{N}) + d_{\alpha DisjCoher}(T_n,S_1,S_2,\tfrac{n}{N}) \tag{9}$$

These terms are illustrated in Fig. 3 and the effect of each of the terms alone on the output is shown on a toy example in Fig. 4 and on a couple real examples in the supplementary video.

We next describe how to minimize the *morphing objective function* to get the output morph sequence.

3. The optimization algorithm

All terms of the objective function we defined in Eq. (1), are based on the sum of square differences between color values of patches. Therefore the objective can be optimized using an iterative algorithm, similar to [17, 16]. We present here briefly the main steps of the derivation. We start by initializing the morph sequence $\{T_n(0)\}_1^{N-1}$. A simple "cross-dissolve" of the two sources is usually good enough, but we found that linearly interpolating the nearest neighbor offsets from one source to the other, and doing "color-voting" using patches at the interpolated locations usually works better. We then build spatial Gaussian pyramids for the initial sequence, with scale gaps of $\times 1.2$ and start the optimization at the coarsest scale (between 0.2 for very different images to 0.6 for view interpolation). The optimization is done by i global iterations i of looping over all frames n (typically i was 6 at the coarsest scale and decreased gradually to 2). For each frame $T_n(i)$ the minimization of Eq. (1) computes five

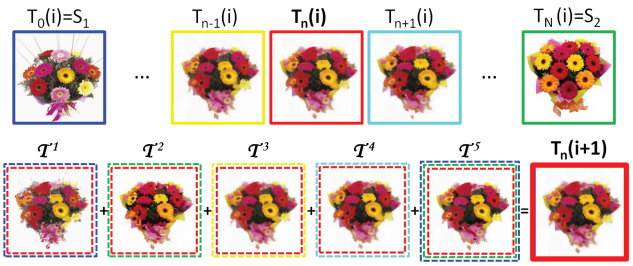

Figure 5. Morphing iteration: at each iteration i, $T_n(i)$ is combined through the various *BDS*, *coherence* and *completeness* terms, with its neighboring frames (yellow, cyan) as well as with the two sources (blue, green). These combinations yield five intermediate images $\mathcal{T}^1 ... \mathcal{T}^5$ (where the colors of their dashed frames correspond to the colors of the sequence frames that participate in their computation). The output of the iteration $T_n(i+1)$ is computed as a weighted average of the intermediate images. Zoom into the PDF to see details.

intermediate images and does a weighted average of them. The first two $\mathcal{T}^1, \mathcal{T}^2$ result from computing a BDS between $T_n(i)$ with its neighboring frames $T_{n-1}(i)$ and $T_{n+1}(i)$ correspondingly (Eq. (2)). The other two images $\mathcal{T}^3$, $\mathcal{T}^4$ are computed with only the *completeness* term between $T_n(i)$ and the two sources S_1 and S_2. Finally, image $\mathcal{T}^5$ is computed similarly using the α-*Disjoint Coherence* term between $T_n(i)$ and the pair S_1 and S_2. The algorithm is summarized next and the five terms are illustrated in Fig. 5:

```
Initialize:  {T_n(0)}_1^{N-1}
 For each scale  s = 1 : S,
  For each iteration  i = 1 : K,
   For each frame  n = 1 : N − 1
      T^1 ← min  d_BDS(T_n(i), T_{n−1}(i))
      T^2 ← min  d_BDS(T_n(i), T_{n+1}(i))
      T^3 ← min  d_Complete(T_n(i), S_1)
      T^4 ← min  d_Complete(T_n(i), S_2)
      T^5 ← min  d_αDisjCohere(T_n(i), S_1, S_2, α)
      T_n(i+1) = (Σ_{j=1}^5 w_j T^j) / (Σ_{j=1}^5 w_j)
  Upscale  s → s + 1
Output:  {T_n(i)}_1^{N-1}
```

where $\alpha = \frac{n}{N}$, $w_1 = w_2 = 0.5$, $w_3 = \beta\alpha$, $w_4 = \beta - w_3$ and $w_5 = \beta$. β controls the relative weight of the *Temporal Coherence* and *Source Similarity* terms and we typically set it to $1.0 - 1.5$.

Motion constraints The original BDS method [16] allowed patches to search their nearest neighbor at any location in the second image. In order to obtain small motion, as required by *Temporal Coherence* we constrain the search space per pixel in the above terms. In all our image morphing examples for different images, we limited the patch searches in consecutive frames to a window of size 0.01-0.03 times the image size around the patch location in the input image, and to a window of size 0.1-0.3 for searches in the source images (where larger offsets are expected). In typical view interpolation scenarios such as in Fig. 7, we expect much more "physically correct" correspondences and interpolation. We thus add reliable feature correspondences as constraints to the objective. We extract SIFT features [13] in both images, and aggressively prune the matches to ensure only the most reliable matches . We use the resulting point correspondences in two ways - first, to do standard morphing (warping+blending) as initialization, and second, to constrain the offsets in a small radius around those locations. For the rest of the pixels we linearly interpolate their offsets and use the search window as mentioned above, around the interpolated locations. In hard cases of wide-baseline views and/or highly non-rigid motions we add also a few manual point correspondences (see more details in Sec. 4).

Run-time We use PatchMatch [2] for the inner loop dense patch searches. Our current Matlab implementation takes several tens of minutes for a 400×400 pair of images on a 2.4GHz single core, 2GB RAM. According to [2], retargeting such a small image using BDS can take a fraction of a second. Our computations per frame are equivalent to computing 4 times the standard BDS for each of the 20 frames with a similar number of synthesis iterations per frame. Therefore, a more optimized implementation should take around a minute for that image size.

4. Results

We strongly recommend viewing the video versions of these results at the project webpage - http://grail.cs.washington.edu/projects/regenmorph/.

Morphing between different images The most unique capability of Regenerative Morphing is its ability to automatically generate transitions between totally different images that have no clear correspondences. We picked 5 pairs of images - two different street views, two pairs of castles and two pairs of totally different images of a face with a cloud and a lion with flowers. The image size was reduced such that the largest dimension was 400, and the sequence length was 20 frames in total. Fig. 9 shows the source images along with a few in-between frames. Our method generates interesting transitions where local structures evolve into similar near-by structures, with minimal "ghosting".

There is no other method that we are aware of that can automatically morph the above pairs. Sparse feature matching methods (e.g., [13]) will fail to find meaningful matches while standard optical flow algorithms (e.g., [4]) fail here due to the large differences in appearance. Although not designed for morphing, SIFT-flow [12] could be applied for morphing different images. Fig. 6 shows a comparison of the flowers sequence to simple blending (top) and SIFT-flow based morphing (middle). Our result is much sharper

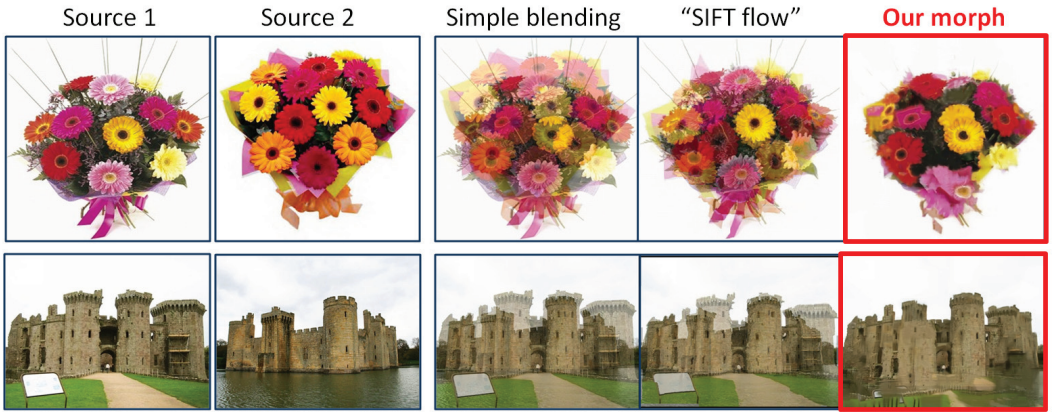

Figure 6. A comparison of automatic morphing of different images (two pairs from Fig. 9) - simple blending, SIFT-flow [12]+blending, and our method (right).

and does not suffer from the "ghosting" artifacts seen in the other two.

View and temporal interpolation Fig. 7 shows a comparison of our method to MovingGradients [14] on the "winking girl" example as well as our results on the other two. Our results are comparable.

Challenging non-rigid interpolation Here we show how Regenerative Morphing can handle even wider baseline views and larger non-rigid motions between the two source images. One such example is shown in Fig. 8 and a few more are shown in the video. In these examples SIFT matches often don't find enough reliable matches on the moving objects and people. Therefore we added a few manual correspondences (such as a few fiducial points on faces, and rough locations of human joints). We show a comparison to a professional morphing software [1] with the same correspondences (second row). One nice property of our view interpolation approach is that by relaxing a bit the accuracy of these correspondences we allow different parts of the sequences to move in a slightly unsynchronized and non-linear way, resulting in more "natural" looking motions (such as independent motions of people, or of different limbs).

Discussion Some of our results may look somewhat jittery in the video. Our method has an inherent tradeoff between motion smoothness and spatial sharpness. The video demonstrates that *temporal coherence* alone leads to temporally smooth but blurry result, while *source similarity* alone leads to a sharp but temporally incoherent video. Intuitively, this is because of the fundamental requirement in our method that every patch in the sequence will look like a patch in the input images. If the two inputs look different, it is likely that some patches will initially look like a patch from the first input and eventually look like a patch from the second input, so some amount of jitteriness (abruptness in motion or appearance or both) is inevitable in order to get sharp frames.

One limitation of our current implementation is in the way we handle color. We use simple color patches which could be insufficient for different images or even in case of view interpolation with changes in illumination and shadows. This could be improved by working in the gradient domain and/or separating intensity from color. Another limitation is the current need for manual correspondences for complex view+time interpolation, such as in the presence of moving people (Fig. 8). One could use computer vision tools (such as a face fiducial points detector [7]) or a human poslet detector [5]) to automate this process.

5. Conclusions

We introduced a new method for image morphing where the traditional warp and blend approach was replaced with a regenerative approach. Following this approach, the morph sequence is directly synthesized from local regions from the two sources, such that it is *temporally coherent* and locally *similar to the sources*. The latter directly minimizes common morph artifacts (such as "ghosting" and blurring). This approach is unique in its capability to automatically generate interesting morphs from visually unrelated pairs of images. Good results are also shown for related tasks such as view and temporal interpolation.

Acknowledgements This research was supported in part by the Israeli Ministry of Science.

References

[1] Abrosoft FantaMorph. http://www.fantamorph.com/. 6, 7

[2] C. Barnes, E. Shechtman, A. Finkelstein, and D. Goldman. PatchMatch: A Randomized Correspondence Algorithm for Structural Image Editing. *ACM SIGGRAPH*, 2009. 2, 5

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

Figure 7. Comparison to MovingGradients [14] - we applied our method on three examples from [14] (one of them shown here and the other in the video). Top row: the two still photos of a winking girl and three frames from *our* result out of 30 (red frame). Bottom row: result from [14] (blue frame). Both interpolation results are comparable in terms of quality.

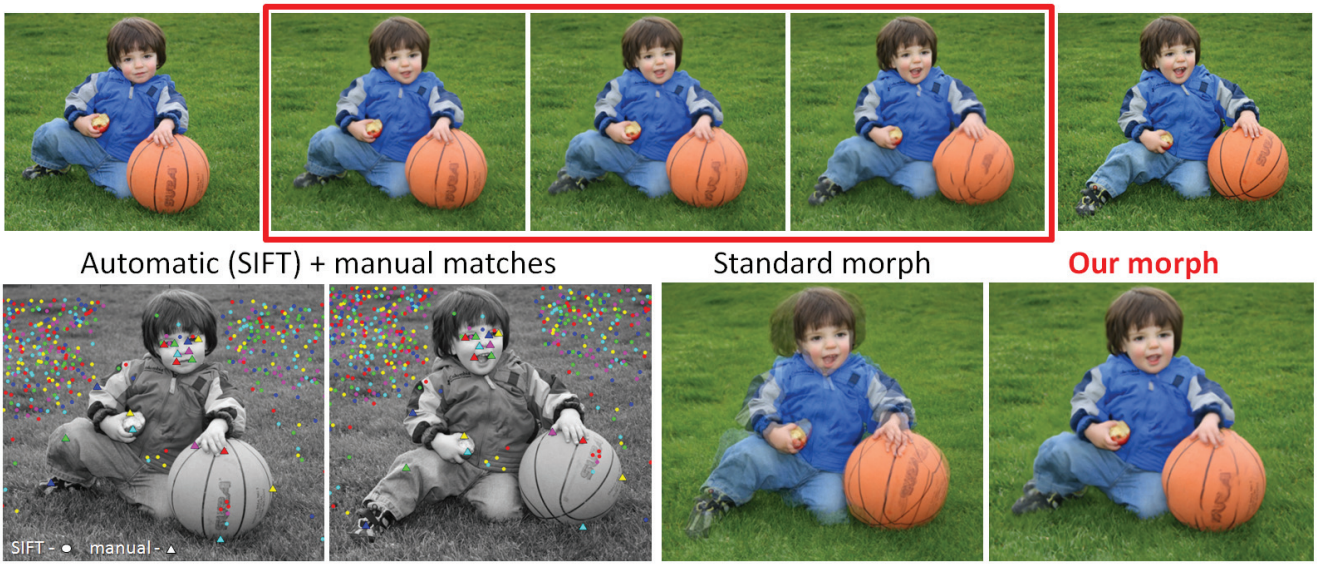

Figure 8. Non-rigid view interpolation and comparison to standard morphing - First row: our result on a boy rolling a basketball. Note the non-rigid motion of the boy's face, limbs and jacket. In this example automatic SIFT correspondence and 17 manual ones were used to improve the result. Second row: the SIFT points (circles) and the manual (triangles) that were used, an output mid frame of a professional morphing software [1] with the same (SIFT+manual) point correspondences, and our result (right). Due to sparsity of points, standard morphing suffers from "ghosting" effects. In our result such effects are minimized.

[4] M. J. Black and P. Anandan. The robust estimation of multiple motions: parametric and piecewise-smooth flow fields. *CVIU*, 63(1), 1996. 5

[5] L. Bourdev and J. Malik. Poselets: Body part detectors trained using 3d human pose annotations. In *ICCV*, 2009. 6

[6] S. E. Chen and L. Williams. View interpolation for image synthesis. In *ACM SIGGRAPH*, 1993. 1, 2

[7] M. Everingham, J. Sivic, and A. Zisserman. "Hello! My name is... Buffy" – automatic naming of characters in TV video. In *BMVC*, 2006. 6

[8] A. Fitzgibbon, Y. Wexler, and A. Zisserman. Image-based rendering using image-based priors. In *ICCV'03*, page 1176, Washington, DC, USA, 2003. IEEE Computer Society. 2

[9] M. Irani, T. Hassner, and P. Anandan. What does the scene look like from a scene point? In *ECCV*, pages 883–897, London, UK, 2002. Springer-Verlag. 2

[10] V. Kwatra, I. Essa, A. Bobick, and N. Kwatra. Texture optimization for example-based synthesis. *ACM SIGGRAPH*, 24, 2005. 2

[11] S. Lee, G. Woberg, K.-Y. Chwa, and S. Y. Shin. Image metamorphosis with scattered feature constraints. *IEEE*

Figure 9. Morphing different images.

TVCG, 2(4):337–354, 1996. 1

[12] C. Liu, J. Yuen, A. Torralba, J. Sivic, and W. T. Freeman. Sift flow: Dense correspondence across different scenes. In *ECCV'08*, pages 28–42. Springer-Verlag, 2008. 5, 6

[13] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 60(2), 2004. 5

[14] D. Mahajan, F.-C. Huang, W. Matusik, R. Ramamoorthi, and P. Belhumeur. Moving gradients: a path-based method for plausible image interpolation. *ACM Trans. Graphics*, 28(3), 2009. 2, 6, 7

[15] S. M. Seitz and C. R. Dyer. View morphing. In *ACM SIGGRAPH*, pages 21–30, New York, 1996. 1, 2

[16] D. Simakov, Y. Caspi, E. Shechtman, and M. Irani. Summarizing visual data using bidirectional similarity. In *CVPR*, Anchorage, AK, USA, 2008. 2, 3, 4, 5

[17] Y. Wexler, E. Shechtman, and M. Irani. Space-time completion of video. *IEEE Trans. PAMI*, 29(3), 2007. 2, 4

[18] G. Wolberg. *Digital Image Warping*. IEEE Computer Society Press, Los Alamitos, CA, 1990. 1