Eurographics Symposium on Rendering 2014
Wojciech Jarosz and Pieter Peers
(Guest Editors)

Volume 33 (2014), Number 4

Semi-Automated Video Morphing

Jing Liao[1], Rodolfo S. Lima[2], Diego Nehab[2], Hugues Hoppe[3] and Pedro V. Sander[1]

[1]The Hong Kong University of Science and Technology
[2]Instituto Nacional de Matemática Pura e Aplicada (IMPA)
[3]Microsoft Research

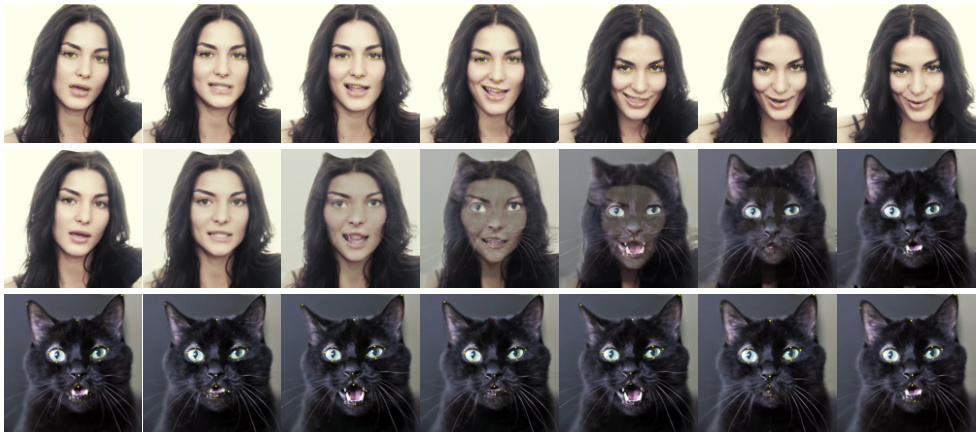

Figure 1: *Given two input videos (top and bottom rows), we create a video morph (middle). Both the alignment of spatial features and the synchronization of local motions are obtained using a spatiotemporal optimization with relatively little user guidance (here 46 mouse clicks).*

Abstract

We explore creating smooth transitions between videos of different scenes. As in traditional image morphing, good spatial correspondence is crucial to prevent ghosting, especially at silhouettes. Video morphing presents added challenges. Because motions are often unsynchronized, temporal alignment is also necessary. Applying morphing to individual frames leads to discontinuities, so temporal coherence must be considered. Our approach is to optimize a full spatiotemporal mapping between the two videos. We reduce tedious interactions by letting the optimization derive the fine-scale map given only sparse user-specified constraints. For robustness, the optimization objective examines structural similarity of the video content. We demonstrate the approach on a variety of videos, obtaining results using few explicit correspondences.

Categories and Subject Descriptors (according to ACM CCS): I.3.0 [Computer Graphics]: General—Video Processing

1. Introduction

Creating smooth transitions (morphs) between images is a well studied problem in computer graphics. In contrast, there is relatively less research on generalizing this concept to videos (Section 2). Because a morph of two images results in a video, a direct generalization would be to morph two videos into a sequence of videos, i.e. a 4D volume. This would be analogous to morphing between two volumetric datasets [LGL95].

Instead, the problem we address is to generate a single output video. The early and late frames in the output video should be similar to those in the corresponding input videos.

In the intermediate frames, objects should start to behave and look like the corresponding objects in the other input video.

Some striking early examples of video morphing include the face transitions in Michael Jackson's "Black or White" music video [PDI91] and the Exxon car-tiger commercial [PDI92]. Such results are obtained with extensive manual interaction, often involving adjustment of many correspondence points on individual frames.

In this paper we explore a semi-automated technique for optimizing a morph between a pair of videos of different scenes. Our main goal is to leverage numerical optimization to align the video content automatically, both spatially and temporally, so that the user need only guide the process at a high level using sparse correspondences. For example, Figure 1 shows a transition between a video of a talking woman and a video of meowing cat, obtained using only 16 correspondence points adjusted in 46 mouse clicks.

Whereas the main difficulty in image morphing is to establish a spatial mapping, video morphing has the additional challenge of temporal synchronization. This issue may be alleviated by creating specially crafted content (e.g., professional dancers timing their moves to a reference choreography). In this work we consider ordinary video content acquired by amateurs. Because we rely on sparse user correspondences, the simple approach of optimizing spatial mappings on each frame individually results in videos that lack proper motion continuity (as shown on the accompanying video).

In addressing these difficulties, our approach is to construct a smooth 3D spatiotemporal map between the videos. The approach incorporates the following elements:

- The inter-video map is represented using a 3D-valued vector field on a 3D spatiotemporal *halfway domain*, and is solved using a coarse-to-fine optimization.
- The objective function considers structural similarity of corresponding pixel neighborhoods of the two videos, and is regularized using a thin-plate-spline smoothness term.
- The optimization is accelerated using GPU parallelism.
- We introduce a *temporal coherence* term to guide the map using estimated optical flow.
- The solution is obtained in two successive stages: temporal synchronization and spatial optimization.

The first three elements adapt and extend concepts from the image morphing work of Liao et al. [LLN*14], as discussed in Section 3. Our novel temporal coherence and synchronization elements let our approach work on a broader set of input videos compared to traditional video morphing techniques.

2. Related work

Image morphing There is extensive research literature on the computation of image morphs, including entire books [Wol90, GDCV99]. Several techniques address the key challenge of establishing an appropriate mapping between images [BN92, LCS95, GS98, MZD05, MHM*09, WL12, LLN*14]. Given the map, trajectories must be computed that slowly transform the position and color of each point in one image to the mapped values in the other image. While these techniques employ some geometric construction or optimization to solve for the mapping, when dealing with morphing of different objects, the computation of correspondences is inherently ambiguous and ill-posed. In those situations, the usual strategy is to obtain sparse correspondences from the user, which are then enforced during the optimization.

Video morphing There is comparatively less research on morphing videos. Early work relies on painstaking annotation of each individual pair of corresponding frames [PDI91, BN92]. Surprisingly, this approach is still commonplace [CSM*13] despite the increase in the computational power available to assist the task. Two papers address this issue. Szewczyk et al. [SFAS97] use feature-tracking methods to ease the burden of designers by transferring manually selected feature correspondences from one frame to the next. They perform video matting and apply morphing separately to each video layer to improve morphing quality. Yu and Wu [YW04] use combinatorial optimization on each frame to automatically align image points with similar color neighborhoods. Their optimization uses a sequence of alternating dynamic programming algorithms in the two dimensions. In both these earlier techniques, it is assumed that the videos are already temporally synchronized.

In contrast, our video morphing scheme aims for full spatiotemporal alignment of two unsynchronized videos. We cast this alignment as a continuous optimization that considers temporal coherence with the underlying optical flow. We use structural similarity for improved matching robustness and exploit GPU parallelism for the costly spatial optimization.

Other video processing techniques Video textures [SSSE00, KSE*03, AZP*05], cliplets [JMD*12], cinemagraphs [TPSK11, BAAR12], and video loops [LJH13] are all techniques for finding seamless transitions from a single video segment to itself. This is achieved by finding compatible video frames or compatible spatiotemporal surfaces within the 3D video volume. Similarly, Rüegg et al. [RWSG13] composite different videos of the same scene by solving for a spatiotemporal transition surface using a graph cut. Poisson blending is often used to improve visual continuity at transitions [MHM*09]. Zitnick et al. [ZKU*04] perform view interpolation between several input videos. Kalantari et al. [KSB*13] combine optical flow and patch-based synthesis to reconstruct high dynamic range video. All these techniques assume video(s) of a single scene rather than a morph between different objects.

Sand et al. [ST04] and Diego et al. [DSL13] propose methods that temporally align video sequences recorded from independent moving cameras. Their approaches automatically find correspondences between frame pairs from the two

input videos and also bring them into spatial alignment. Unlike their methods, our temporal synchronization approach is spatially adaptive, and thus can synchronize motions with different speeds in different regions of the input videos. This is vital for our morphing application which must deal with arbitrary animations of different non-rigid moving subjects.

3. Approach overview

The two input videos V_0, V_1 are color functions $V(x, y, t)$ over 3D spatiotemporal domains. We denote a spatiotemporal point as $p = (p_x, p_y, p_t) = (p_{xy}, p_t)$ where p_{xy} is its 2D spatial position.

To represent a continuous spatiotemporal map between the two input videos, we extend the image morphing framework of Liao et al. [LLN*14]. Let Ω denote the 3D spatiotemporal domain that is conceptually *halfway* between the two videos. Let v be a 3D-valued vector field defined over Ω, which induces two continuous, invertible maps ϕ_0, ϕ_1 from the halfway domain to each of the two input video domains:

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

The inter-video map from V_0 to V_1 is the composition $\phi_1 \circ \phi_0^{-1}$. The construction is nicely symmetric, as swapping the two videos simply corresponds to negating the field v.

The goal is to optimize the vector field v to minimize an objective function:

$$\min_v E \quad \text{with} \quad E = \sum_{p \in \Omega} E(p). \tag{1}$$

Our initial attempt was a direct generalization of the objective energy used by Liao et al. [LLN*14]. The original formulation is as follows:

$$E(p) = E_{SIM}(p) + \lambda\, E_{TPS}(p) + \gamma\, E_{UI}(p), \tag{2}$$

where

- $E_{SIM}(p)$ evaluates the structural similarity of the mapped image neighborhoods $\phi_0(N(p))$ and $\phi_1(N(p))$, in which $N(p)$ denotes the 5×5 pixels nearest to p.
- E_{TPS} computes a thin-plate spline function on the 2D-valued vector field v, thus encouraging it to be smooth.
- $E_{UI}(p)$ introduces soft constraints on $v(p)$ near each halfway point between corresponding 2D image correspondence points specified in the user interface.

The full generalization to video morphing involves

1. extending the neighborhoods N to be spatiotemporal (e.g. $5 \times 5 \times 5$),
2. defining the thin-plate spline (TPS) function on v which is now a 3D-valued function over the 3D domain Ω,
3. generalizing the UI constraints to be spatiotemporal, and
4. performing coarse-to-fine optimization using all three objective terms over the 3D domain Ω.

However, the straightforward generalization proves both challenging and inadequate for a number of reasons. The large spatiotemporal neighborhoods link together many of the unknown coefficients in v, thus making the evaluation of $\partial E/\partial v$ expensive. When the correspondence field v includes a temporal component, constraining injectivity of the functions ϕ_0 and ϕ_1 (i.e. preventing foldovers in the map) is more costly. Also, the mapped neighborhood $\phi_0(N(p))$ of each spatiotemporal point p gets warped in time, thus requiring costly 3D sampling. We find that the 3D-valued TPS function leads to an overly smooth solution that fails to adapt to sudden changes in optical flow present in the input videos. Finally, the full 3D hierarchical optimization requires building video pyramids, and averaging videos in the temporal direction tends to blur their content, which hinders spatial matching at coarse levels.

Instead, we find greater success using a two-stage approach:

- In the first stage, we still optimize the 3D-valued vector field v over the 3D domain Ω, but omit the expensive structural similarity term E_{SIM}. The aim is to solve for a smooth temporal correspondence map between the two videos, using only a crude spatial alignment guided by the sparse user-specified points. We use the temporal component v_t of this correspondence map to temporally resample the input videos, thus synchronizing them without spatially advecting their content.
- In the second stage, given the synchronized videos, we again solve an optimization over the 3D domain Ω but using only a 2D-valued vector field v_{xy} (i.e. with v_t fixed to zero). We introduce a new objective term E_{TC} to measure the temporal coherence of v_{xy} with respect to the estimated optical flows of both videos.

In addition to the advantages outlined earlier, in practice this proposed two level approach is $5\times$ faster than a full generalization of image morphing.

4. User interface and correspondences

Figure 2 shows our system's user interface. It provides operations to add, edit, and remove correspondence points across the two input videos. Red points denote user-specified points, and yellow points denote those propagated automatically to other frames using optical flow. The user may optionally adjust the position of a propagated feature point, thereby adding a new specified point for that feature.

Thus, each feature correspondence i has one or more pairs of user-specified spatiotemporal correspondence points $\hat{u}^a_{i,j}$ (where $a \in \{0, 1\}$ is the input video and j is the pair index), each storing a location (x, y, t) in input video V_a. The spatial location of each feature correspondence i is propagated both backwards and forwards from frames with specified points to every other frame t using precomputed optical flow [WTP*09]. More precisely, we obtain $u^a_i(t)$ for each frame t by finding the forward and backward propagated points from

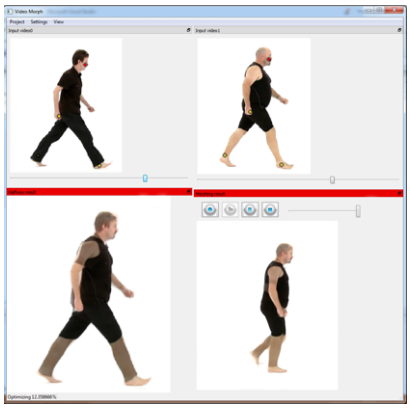

Figure 2: *Our system shows the two input videos with overlaid correspondence points (top). It provides feedback on morph quality using both a halfway morph and a continuously animating video morph that oscillates forward and backward in time (bottom).*

the nearest $\hat{u}^a_{i,j}$ and linearly blending the two points according to the temporal distances of the $\hat{u}^a_{i,j}$ from t.

Each correspondence point also carries a confidence weight $w^a_i(t)$ that determines its impact during the optimization stages. This weight decays from 1 as the point's spatial neighborhood begins to differ from the neighborhoods of the temporally nearest user-specified points $\hat{u}^a_{i,j}$. Specifically, we measure histogram correlation over 7×7 neighborhoods, and again linearly interpolate correlations from the nearest backward and forward $\hat{u}^a_{i,j}$ according to temporal distance.

When specified points are added or modified, the feature propagation is re-evaluated. Because optical flow is precomputed, propagation occurs in real-time, allowing for efficient creation of accurate correspondences. Please see the demo in the accompanying video.

5. Temporal synchronization

In this first stage of optimization (Figure 3), we *temporally* align regions of the input video so that they are in agreement with the specified correspondences. This amounts to a spatially adaptive temporal synchronization of the videos. In the case of two input videos of unequal duration, the duration of the halfway domain is their average duration, and temporal synchronization resamples both videos to this common duration.

Objective As discussed in Section 3, the objective in stage 1 has the form

$$E(p) = \lambda\, E_{TPS}(p) + \gamma\, E_{UI}(p), \tag{3}$$

with constants $\lambda = 0.01$ and $\gamma = 100$ in all our results.

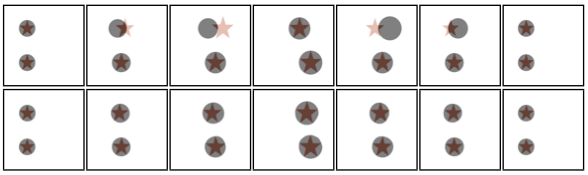

Figure 3: *Example showing two input videos simply blended together (top) and blended after temporal synchronization (bottom).*

Smoothness energy The thin-plate spline energy measures smoothness of the function v over both its spatial and temporal components:

$$E_{TPS}(p) = \mathrm{TPS}\big(v_x(p)\big) + \mathrm{TPS}\big(v_y(p)\big) + \mathrm{TPS}\big(v_t(p)\big),$$

$$\mathrm{TPS}\big(v_x(p)\big) = \Big(\frac{\partial^2 v_x(p)}{\partial {p_x}^2}\Big)^2 + \Big(\frac{\partial^2 v_x(p)}{\partial {p_y}^2}\Big)^2 + \Big(\frac{\partial^2 v_x(p)}{\partial {p_t}^2}\Big)^2 + 2\Big(\frac{\partial^2 v_x(p)}{\partial p_x p_y}\Big)^2 + 2\Big(\frac{\partial^2 v_x(p)}{\partial p_y p_t}\Big)^2 + 2\Big(\frac{\partial^2 v_x(p)}{\partial p_t p_x}\Big)^2,$$

and analogously for $\mathrm{TPS}\big(v_y(p)\big)$ and $\mathrm{TPS}\big(v_t(p)\big)$.

User interface energy The correspondences are enforced through soft constraints in the spatiotemporal domain. Essentially, we desire that the halfway domain grid position that lies halfway between each pair of specified correspondences should map back to those same two correspondence points in the two input videos.

Thus, for each pair of correspondence points $\hat{u}^0_{i,j}, \hat{u}^1_{i,j}$, we compute the spatiotemporal midpoint $\bar{u}_{i,j} = (\hat{u}^0_{i,j} + \hat{u}^1_{i,j})/2$ and the expected vector field value $\hat{v}_{i,j} = (\hat{u}^1_{i,j} - \hat{u}^0_{i,j})/2$. Because $\bar{u}_{i,j}$ does not in general fall on top of a grid point $p \in \Omega$, we enforce soft constraints on the eight nearest spatiotemporal neighbors $\{p^k\}$ of $\bar{u}_{i,j}$. Each soft constraint is weighted by the corresponding trilinear weight $b(p, \bar{u}_{i,j})$:

$$\sum_{k=1}^{8} b(p^k, \bar{u}_{i,j})\, p^k = \bar{u}_{i,j}. \tag{4}$$

The user interface energy is

$$E_{UI}(p) = \frac{1}{A} \sum_{i,j} b(p, \bar{u}_{i,j}) \left\| v(p) - \hat{v}_{i,j} \right\|^2, \tag{5}$$

where A, the area in pixels of the halfway domain, is introduced to attain resolution independence in the coarse-to-fine solver. Thus, $E_{UI}(p) = 0$ at any gridpoint p that is not adjacent to a constraint.

Solver In stage 1, finding the map function v that minimizes E is a sparse linear least squares problem. We solve it using a coarse-to-fine approach similar to that in Liao et al. [LLN*14]. We build a 3D pyramid over the spatiotemporal domain of each input video by averaging both spatially and temporally. We then run a conjugate-gradient solver (implemented with NVIDIA's cuSPARSE library) on progressively finer grids in the halfway domain.

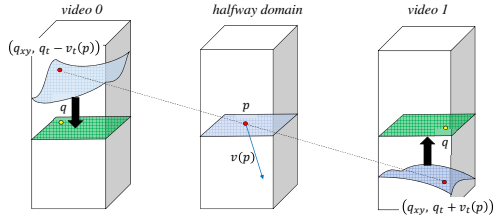

Figure 4: *Temporal resampling for video synchronization.*

Temporal resampling Having obtained the inter-video map, for each input video V_a we create the synchronized video V'_a using the following resampling procedure. For each pixel $q = (q_{xy}, q_t)$ in V'_a, we seek the point $p \in \Omega$ that (1) has the same time q_t and (2) spatially maps to q_{xy}:

$$p = (p_{xy}, q_t) \quad \text{s.t.} \quad p_{xy} + (2a-1)v_{xy}(p) = q_{xy}. \tag{6}$$

Having obtained p, we sample the input video at the original spatial location q_{xy} but using the temporal map defined at p (see Figure 4):

$$V'_a(q) = V_a\big(q_{xy}, q_t + (2a-1)v_t(p)\big). \tag{7}$$

Computing point p_{xy} in Equation (6) involves solving an inverse map. It is achieved efficiently using an iterative search (similar to that in the rendering algorithm of Liao et al. [LLN*14]):

$$p^{(0)}_{xy} = q_{xy}, \tag{8}$$
$$v^{(0)}_{xy} = v_{xy}(p^{(0)}_{xy}), \tag{9}$$
$$p^{(i+1)}_{xy} = q_{xy} - (2a-1)\, v^{(i)}_{xy}, \tag{10}$$
$$v^{(i+1)}_{xy} = (\eta)v_{xy}\big(p^{(i+1)}_{xy}\big) + (1-\eta)v^{(i)}_{xy}. \tag{11}$$

We use an exponential smoothing factor η (0.8 in all results) to improve convergence. Because the vector field v is smooth, the search usually converges in a few iterations.

In the resampling process of Equation (7), the temporal source value $t' = q_t + (2a-1)v_t(p)$ generally requires sampling between two successive frames. To avoid ghosting due to averaging, we perform another iterative search using precomputed optical flow F to find the corresponding points in the two frames and appropriately blend them. Formally, to evaluate $V_a(q_{xy}, t')$, we find the corresponding point $(r_{xy}, \lfloor t' \rfloor)$ in the preceding frame as:

$$r^{(0)}_{xy} = q_{xy}, \tag{12}$$
$$f^{(0)}_{xy} = F(r^{(0)}_{xy}, \lfloor t' \rfloor), \tag{13}$$
$$r^{(i+1)}_{xy} = q_{xy} - (t' - \lfloor t' \rfloor) f^{(i)}_{xy}, \tag{14}$$
$$f^{(i+1)}_{xy} = (\eta)F(r^{(i+1)}_{xy}, \lfloor t' \rfloor) + (1-\eta)f^{(i)}_{xy}. \tag{15}$$

The corresponding point in the succeeding frame is given by $(r_{xy} + F(r_{xy}, \lfloor t' \rfloor), \lfloor t' \rfloor + 1)$. We then interpolate the two colors using t'.

Figure 5: *Result of spatial optimization on the example of Figure 3.*

6. Spatial Optimization

In this second stage, we seek to *spatially* align the frames of the two temporally synchronized videos (Figure 5). The solver iterates on each pair of corresponding frames, employing a sequence of 2D optimizations rather than the single 3D optimization required for temporal synchronization.

Objective For each frame t, we solve

$$\min_v E \quad \text{with} \quad E = \sum_{p \in \Omega_t} E(p), \quad \text{where}$$

$$E(p) = E_{SIM}(p) + \lambda E_{TPS_{xy}}(p) + \gamma E_{UI_{xy}}(p) + \beta E_{TC}(p),$$

with constants λ = 0.005, γ = 100, and β = 0.1 in all our results. Note that v is now a 2D-valued function on the 2D domain Ω_t. (In this section, points p, q also refer to 2D points.)

The similarity energy E_{SIM} encourages matching between 5×5 neighborhoods that have similar edge structure. We refer the reader to Liao et al. [LLN*14] for further details on this energy term.

The smoothness energy $E_{TPS_{xy}}$ operates as in the first stage, but only spatially:

$$E_{TPS_{xy}}(p) = \mathrm{TPS_{xy}}\big(v_x(p)\big) + \mathrm{TPS_{xy}}\big(v_y(p)\big),$$

$$\mathrm{TPS_{xy}}\big(v_x(p)\big) = \Big(\frac{\partial^2 v_x(p)}{\partial {p_x}^2}\Big)^2 + 2\Big(\frac{\partial^2 v_x(p)}{\partial p_x p_y}\Big)^2 + \Big(\frac{\partial^2 v_x(p)}{\partial {p_y}^2}\Big)^2,$$

and analogously for $\mathrm{TPS}\big(v_y(p)\big)$. Figure 6 illustrates the importance of this regularizing smoothness term. Its weight λ is kept small to make it comparable with the other terms.

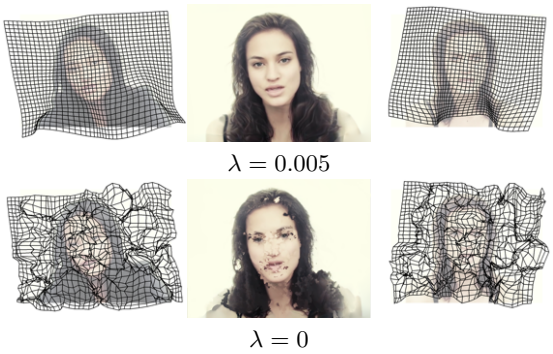

Figure 6: *Importance of the smoothness energy $E_{TPS_{xy}}$. The middle column shows the morphing results; left and right columns show the deformation of the halfway domain grid overlayed on the input video frames for solutions with and without the smoothness energy term.*

The energy $E_{UI_{xy}}$ is also defined just spatially. Whereas E_{UI} in the temporal synchronization stage considered only the specified correspondence points $\hat{u}$, here we include all propagated correspondence points $u_i^a(t)$ (along with their associated weights $w_i^a(t)$) to bring the features into spatial correspondence in all frames:

$$E_{UI_{xy}}(p) = \frac{1}{A}\sum_{i,t'} w_i^0(t')\, w_i^1(t')\, b(p, \bar{u}_i(t'))\, \left\| v(p) - (\hat{v}_i(t'))_{xy} \right\|^2.$$

Temporal coherence The new term E_{TC} replaces the temporal derivatives omitted in $E_{TPS_{xy}}$. Guided by optical flow estimates, it attempts to preserve consistent mappings across the video frames and thus avoid popping artifacts (as shown in our accompanying video). Intuitively, if we were to forward-warp the videos and their respective correspondences from the previous frame to the current frame using optical flow, we would like the correspondences to match those attained in the current frame.

More precisely, $E_{TC}(p)$ is defined as follows (see Figure 7). Let the *halfway optical flow* $F^{1/2}(r)$ of a domain point $r \in \Omega_{t'}$ be the average of the optical flows F^0, F^1 from the input videos V^0, V^1 under the map given by v:

$$F^{1/2}(r, t') = (F^0(r - v(r), t') + F^1(r + v(r), t'))/2.$$

Given the domain pixel p in the current frame t, we first need to find the corresponding domain pixel q in the previous frame $t-1$ whose halfway optical flow maps it to p, i.e. satisfying

$$q + F^{1/2}(q, t-1) = p. \tag{16}$$

To find q, we again resort to iterative search:

$$q^{(0)} = p, \tag{17}$$

$$f^{(0)} = F^{1/2}(q^{(0)}, t-1), \tag{18}$$

$$q^{(i+1)} = p - f^{(i)}, \tag{19}$$

$$f^{(i+1)} = (\eta)\big(F^{1/2}(q^{(i+1)}), t-1\big) + (1-\eta) f^{(i)}. \tag{20}$$

Given q, we advect the corresponding points $q - v(q)$ and $q + v(q)$ to the current frame and measure their differences from points $p - v(p)$ and $p + v(p)$ respectively, as illustrated by the two symmetric red lines in Figure 7. Interestingly, after substitutions with (6,16), the two difference vectors are seen to be exact opposites:

$$\begin{aligned}&(q - v(q)) + F^0(q - v(q), t-1) - (p - v(p))\\ &\quad = -v(q) + \big(F^0(q - v(q), t-1) - F^1(q + v(q), t-1)\big)/2 - v(p),\end{aligned}$$

$$\begin{aligned}&(q + v(q)) + F^1(q + v(q), t-1) - (p + v(p))\\ &\quad = v(q) + \big(F^1(q + v(q), t-1) - F^0(q - v(q), t-1)\big)/2 + v(p).\end{aligned}$$

Thus we define the temporal coherence error as

$$E_{TC}(p) = -E_{SIM}(q) \left\| v(q) - v(p) + q - p + F^1(q + v(q), t-1) \right\|,$$

where $E_{SIM}(q)$ is used as an attenuation factor so that temporal coherence is disregarded if the mapping at q has high similarity error (and is therefore unreliable).

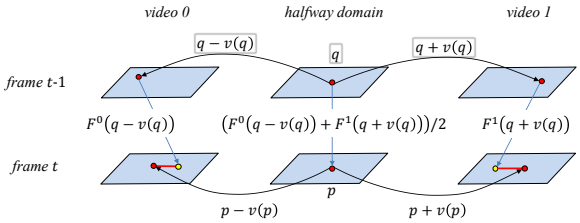

Figure 7: *Computation of temporal coherence energy.*

Solver We again use a coarse-to-fine iterative solver. To avoid ghosting effects during the optimization, when constructing the 3D pyramid we perform point-sampling along the temporal dimension. In each level, successive 2D optimizations are performed on the halfway domain of each pair of matching frames from the two resampled videos. We use an iterative optimization procedure following the approach described in Liao et al. [LLN*14] but using the new nonlinear energy function described above.

7. Results

After generating the mappings for all frames, we use the approach of Liao et al. [LLN*14] to create the final video. This includes the use of quadratic motion paths, Poisson-extended boundaries, and rendering using direct pixel evaluation.

A subset of frames from our video morph results are shown in Figures 1, 8, 10, and 11. Note that the structural similarity term properly aligns the boundaries and the temporal coherence term makes the mapping smooth (see accompanying video). Even on the challenging *woman-cat* example with the twitching cat ears, and the *dog-lion* example with different backgrounds, the method produces a good transition.

Figure 11 focuses on the improvement due to temporal synchronization in two examples. In the input videos (first rows), one can note the unsynchronized motions of legs and arms in the *walk* scene and the wings of the butterflies in the *butterfly* scene. The second row shows the result omitting the temporal synchronization step and applying spatial optimization directly. Note that the approach fails to produce a proper transition. After temporal synchronization, shown on the third row, we can then better spatially align the features and produce the morph shown in the fourth row. Note that we selected the frames that better show the temporal misalignment. Figure 8 shows additional results. Earlier works do not apply any kind of temporal synchronization and do not achieve temporal coherence, resulting in artifacts unless the input videos already have similar motions. Please refer to the accompanying video for a demonstration of results with and without temporal coherence.

Table 1 reports quantitative results for all examples in the paper. The number of correspondences $\{i\}$ and total number of point adjustments $\{i, j\}$ (mouse clicks) vary significantly depending on the similarity of the two input videos. Note that

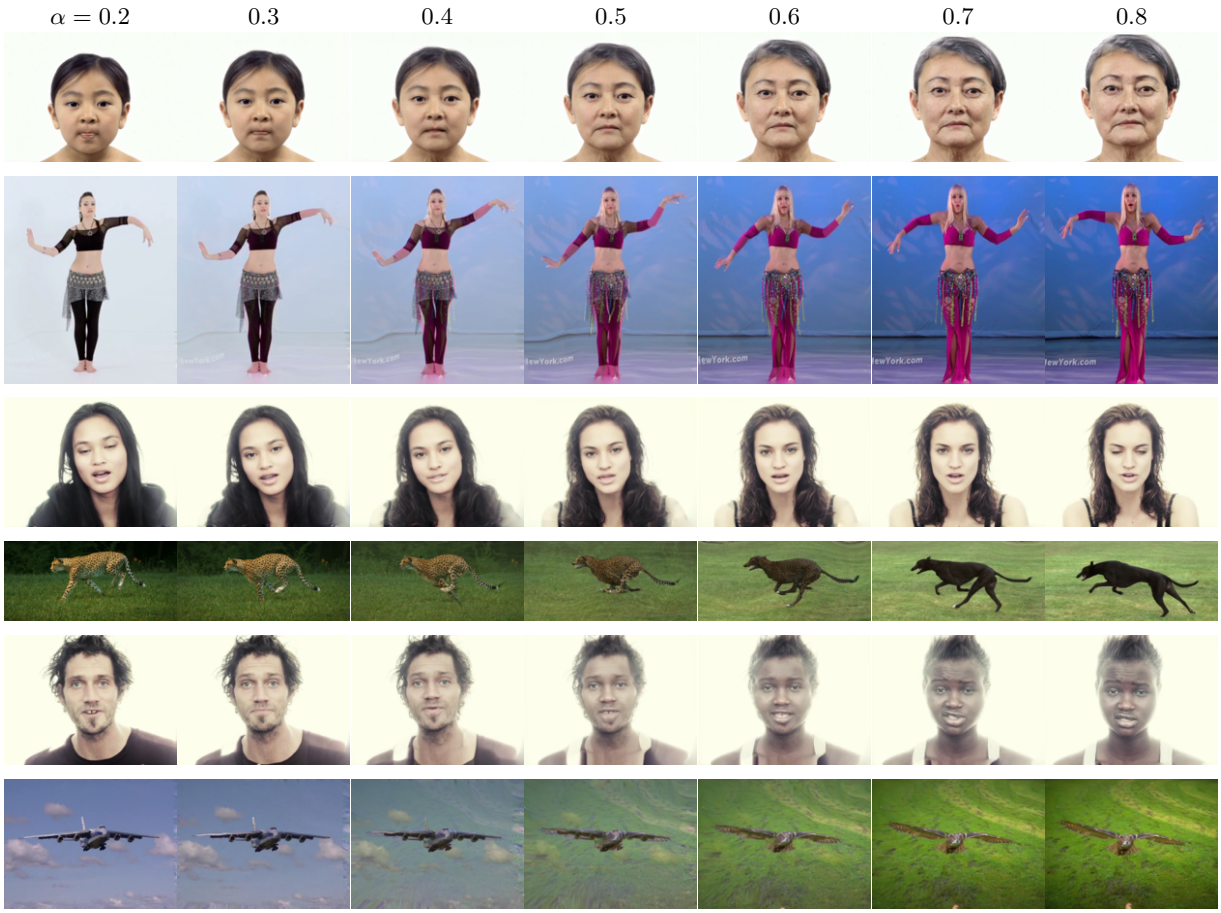

Figure 8: *Additional video morph results.*

the temporal resampling stage takes less than a minute for all examples, and the final spatial optimization takes around 2–4 minutes. It is important to note that when interactively adding correspondence points, the user seldom needs to wait until the entire optimization completes. The partial results from the coarse level usually suffice to get a sense of whether more correspondence points are needed or whether the optimization is converging to a well aligned result.

Limitations Since our approach computes a smooth mapping for the entire frame, it is problematic to blend videos of foreground subjects with shifted background. This is shown in the weightlifting example of Figure 12. Spatially optimizing the foreground subject results in distortions on the background. This can only be addressed by matting the background prior to the morph and processing it separately.

If there are significant topological differences between the features of the videos, a continuous mapping cannot be achieved. Note that in the top row of Figure 9, the man's arm extends beyond his body, unlike in the bottom row. It is therefore impossible to generate a continuous mapping between the two videos without blending the background of the top example with the foreground of the bottom example. The middle row shows the inadequate blended result. To handle such topological changes during animation, one might consider 3D shape proxies [HDK07].

Table 1: *Statistics for all the results in the paper.*

	Dataset				*Processing times* (sec)*			
Fig	Image	Resolution	Points	Clicks	Temp.	Spatial	Flow	Total
1	woman-cat	560 × 560 × 74	16	46	22.3	199.8	32.2	254.3
8	child-lady	640 × 480 × 160	7	28	67.6	233.1	79.3	380.0
	dance	600 × 720 × 80	11	48	40.3	131.4	31.2	202.9
	woman	800 × 600 × 70	7	25	28.6	120.8	34.4	183.8
	leopard-dog	640 × 360 × 80	10	78	35.1	162.3	25.7	223.1
	man-woman	800 × 600 × 80	10	34	23.0	156.0	40.0	219.0
	plane	800 × 600 × 80	8	18	37.1	178.9	40.2	256.2
10	flower	512 × 512 × 80	5	16	23.4	151.8	30.3	205.5
	ball	800 × 600 × 110	8	37	48.5	236.9	58.4	343.8
	dog-lion	800 × 600 × 100	8	32	39.7	204.6	29.8	274.1
11	walk	600 × 720 × 80	6	34	42.5	127.9	37.9	208.3
	butterfly	512 × 512 × 70	6	40	34.8	183.4	25.3	243.5

*Hardware: NVIDIA GTX 780, Intel Core i7 @ 3.6GHz, 16GB RAM.

Figure 9: *Inadequate blending (middle column) caused by topological differences between the features in the input videos (left and right column).*

8. Conclusion

We present a new approach to create smooth transitions between videos of different objects. We leverage some of the machinery developed for image morphing and adapt it to address issues related to video synchronization and temporal coherence of the mapping. We show how our interface and GPU-accelerated algorithms allow a user to efficiently morph between challenging input videos. In future work, we would like to explore the possibilities of allowing discontinuities in our mapping, to address situations in which there is disagreement between foreground and background motions.

Acknowledgements

The source material for the video examples can be found at https://www.youtube.com/playlist?list=PLHRTFTSKn9rF4KcC25VrkiLTb9HvL7cYM.

References

[AZP*05] AGARWALA A., ZHENG K. C., PAL C., AGRAWALA M., COHEN M., CURLESS B., SALESIN D., SZELISKI R.: Panoramic video textures. *ACM Trans. Graphics (Proc. of ACM SIGGRAPH 2005) 24*, 3 (2005). 2

[BAAR12] BAI J., AGARWALA A., AGRAWALA M., RAMAMOORTHI R.: Selectively de-animating video. *ACM Trans. Graphics (Proc. of ACM SIGGRAPH 2012) 31*, 4 (2012). 2

[BN92] BEIER T., NEELY S.: Feature-based image metamorphosis. *Computer Graphics (Proc. of ACM SIGGRAPH 1992) 26*, 2 (1992). 2

[CSM*13] CERNIELLO A., SIRCHIO K., MEIER N., EARLE E., CUDDY G., REVELEY M.: Danielle, 2013. Video clip http://vimeo.com/74033442. 2

[DSL13] DIEGO F., SERRAT J., LOPEZ A. M.: Joint spatio-temporal alignment of sequences. *IEEE Transactions on Multimedia* (2013). 2

[GDCV99] GOMES J., DARSA L., COSTA B., VELHO L.: *Warping and Morphing of Graphical Objects*. Morgan Kaufmann, 1999. 2

[GS98] GAO P., SEDERBERG T. W.: A work minimization approach to image morphing. *The Visual Computer 14*, 8-9 (1998). 2

[HDK07] HORNUNG A., DEKKERS E., KOBBELT L.: Character animation from 2D pictures and 3D motion data. *ACM Trans. Graphics 26*, 1 (2007). 7

[JMD*12] JOSHI N., MEHTA S., DRUCKER S., STOLLNITZ E., HOPPE H., UYTTENDAELE M., COHEN M.: Cliplets: Juxtaposing still and dynamic imagery. *Proc. of UIST* (2012). 2

[KSB*13] KALANTARI N. K., SHECHTMAN E., BARNES C., DARABI S., GOLDMAN D. B., SEN P.: Patch-based high dynamic range video. *ACM Trans. Graphics 32*, 6 (2013). 2

[KSE*03] KWATRA V., SCHÖDL A., ESSA I., TURK G., BOBICK A.: Graphcut textures: image and video synthesis using graph cuts. *ACM Trans. Graphics (Proc. of ACM SIGGRAPH 2003) 22*, 3 (2003). 2

[LCS95] LEE S. Y., CHWA K. Y., SHIN S. Y.: Image metamorphosis using snakes and free-form deformations. In *Proc. of ACM SIGGRAPH 1995* (1995). 2

[LGL95] LERIOS A., G. C. D., LEVOY M.: Feature-based volume metamorphosis. In *Proc. of ACM SIGGRAPH 1995* (1995). 1

[LJH13] LIAO Z., JOSHI N., HOPPE H.: Automated video looping with progressive dynamism. *ACM Trans. Graphics (Proc. of ACM SIGGRAPH 2013) 32*, 4 (2013). 2

[LLN*14] LIAO J., LIMA R. S., NEHAB D., HOPPE H., SANDER P. V., YU J.: Automating image morphing using structural similarity on a halfway domain. *ACM Trans. Graphics* (2014). To appear. 2, 3, 4, 5, 6

[MHM*09] MAHAJAN D., HUANG F.-C., MATUSIK W., RAMAMOORTHI R., BELHUMEUR P.: Moving gradients: A path-based method for plausible image interpolation. *ACM Trans. Graphics (Proc. of ACM SIGGRAPH 2009) 28*, 3 (2009). 2

[MZD05] MATUSIK W., ZWICKER M., DURAND F.: Texture design using a simplicial complex of morphable textures. *ACM Trans. Graphics (Proc. of ACM SIGGRAPH 2005) 24*, 3 (2005). 2

[PDI91] PDI: Black or white, 1991. Video clip http://www.youtube.com/watch?v=F2AitTPI5U0#t=327. 2

[PDI92] PDI: Exxon tiger, 1992. Video clip http://www.youtube.com/watch?v=vZ7RI3Lt0QY#t=19. 2

[RWSG13] RÜEGG J., WANG O., SMOLIC A., GROSS M.: DuctTake: Seam-based compositing. *Computer Graphics Forum (Proc. of Eurographics) 32*, 2 (2013). 2

[SFAS97] SZEWCZYK R., FERENCZ A., ANDREWS H., SMITH B. C.: Motion and feature-based video metamorphosis. In *Proc. of ACM Multimedia* (1997). 2

[SSSE00] SCHÖDL A., SZELISKI R., SALESIN D. H., ESSA I.: Video textures. In *Proc. of ACM SIGGRAPH 2000* (2000). 2

[ST04] SAND P., TELLER S.: Video matching. *ACM Trans. Graphics (Proc. of ACM SIGGRAPH 2004) 23*, 3 (2004). 2

[TPSK11] TOMPKIN J., PECE F., SUBR K., KAUTZ J.: Towards moment images: Automatic cinemagraphs. In *Proc. of CVMP* (2011). 2

[WL12] WU E., LIU F.: Robust image metamorphosis immune from ghost and blur. *The Visual Computer* (2012). 2

[Wol90] WOLBERG G.: *Digital Image Warping*. IEEE Computer Society Press, 1990. 2

[WTP*09] WERLBERGER M., TROBIN W., POCK T., WEDEL A., CREMERS D., BISCHOF H.: Anisotropic Huber-L1 optical flow. In *BMVC* (2009). 3

[YW04] YU Y., WU Q.: Video metamorphosis using dense flow fields. *Computer Animation and Virtual Worlds 15*, 3-4 (2004). 2

[ZKU*04] ZITNICK C. L., KANG S. B., UYTTENDAELE M., WINDER S., SZELISKI R.: High-quality video view interpolation using a layered representation. *ACM Trans. Graphics (Proc. of ACM SIGGRAPH 2004) 23*, 3 (2004). 2

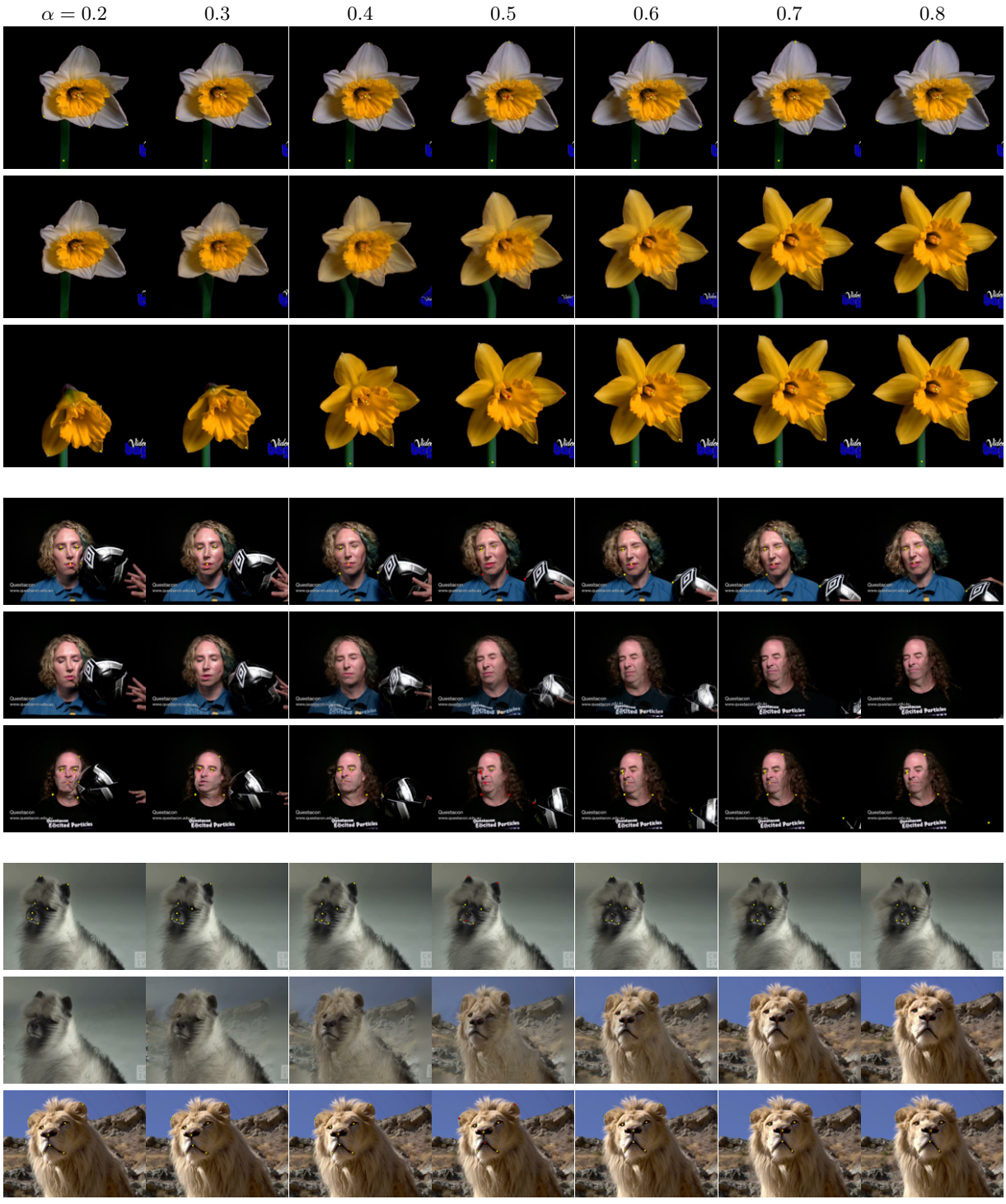

Figure 10: *For each video morph result, the top and bottom rows show frames from the input videos and the middle row shows the resulting morph sequence.*

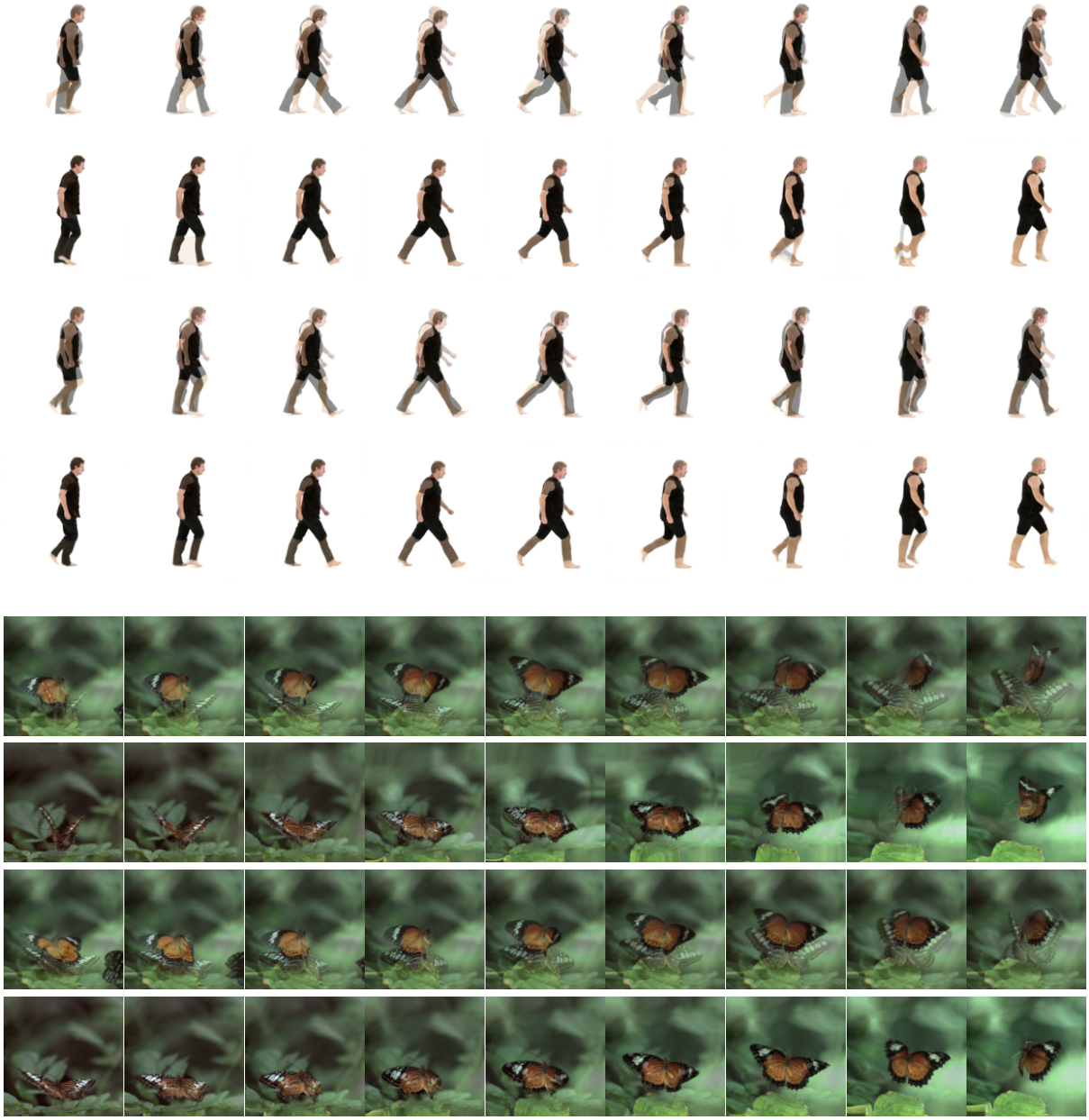

Figure 11: *Benefits of temporal synchronization. In each example, the four rows show (1) blending of the input videos, (2) the result of directly performing spatial optimization without synchronization and the resulting artifacts, (3) blending of the temporally synchronized videos, and (4) the final result that combines both temporal synchronization and spatial optimization.*

Figure 12: *In this video morph result, the shift in the Olympic rings in the background makes it impossible to properly align both the foreground and background elements using the same smooth mapping.*