Motion-Aware Gradient Domain Video Composition

Tao Chen[1] Jun-Yan Zhu[1] Ariel Shamir[2] Shi-Min Hu[1]
[1]TNList, Department of Computer Science and Technology, Tsinghua University
[2]The Interdisciplinary Center

Abstract—Gradient domain composition methods like Poisson blending offer practical solutions for uncertain object boundaries and differences in illumination conditions. However, adapting Poisson image blending to videos faces new challenges due to the addition of the temporal dimension. In videos, the human eye is sensitive to small changes in the blending boundaries across frames, and slight differences in the motion of the source patch and the target video. We present a novel video blending approach that tackles these problems by merging the gradient of source and target video and optimizing consistent blending boundary according to a user provided *blending trimap* for the source video. We extend the mean-value coordinates interpolation to support hybrid blending with dynamic boundary while maintaining interactive performance. We also provide a user interface and source object positioning method that can efficiently deal with complex video sequences beyond the capability of alpha blending.

Index Terms—gradient domain, video editing, mean-value coordinates, Poisson equation, seamless cloning.

1 Introduction

Video compositions are very useful in film, television and entertainment industry. Such compositions take a part of a source video, usually a foreground object (called "patch"), and paste it into the frames of a target video. This process involves two challenges: extracting the patch from a video, and composing it with the target frames. In both cases a user is typically involved to choose a desirable source patch and a composition location in space and time. Therefore, the main focus of effort in video composition techniques has been to lower the amount of manual effort in this process.

Many works related to video matting and composition have been proposed in computer graphics (see [1], [2], [3] and [4]), and recent work can deal with transparent and refractive objects [5]. These techniques use alpha-blending composition and focus mainly on how to cutout the alpha-matte of the video-patch from the source video. Composing using alpha-blending provides good results in limited scenarios where the source and target videos have similar illumination conditions, the object is easily separable from its background, and the videos do not differ too much in terms of motions, including both global camera motion and local object/texture motion. Uncertain object boundaries, due to heavy motion blur, smoke or dust, and varying illuminations conditions make it difficult to achieve high quality composition using alpha matting.

On images, gradient domain blending aims to solve such problems by transferring the gradient of the source image patch to the target image while maintaining seamless blending boundary. Conventionally, this can be done by solving a Poisson equation. More recently, an interpolation based method using mean-value coordinates (MVC) achieves interactive rates for composition [6].

The main challenges in gradient domain video blending, as opposed to image blending, come from *motion*. First, even if the blending results of each individual frame is satisfactory, the blending boundaries in adjacent frames may not be consistent, and "popping" artifacts may appear. Second, the motions of the source and target gradient fields are typically different and can cause motion artifacts even if the blending boundaries are consistent through time. Third, often there are camera motion differences between the source and the target videos, and there is a need to align them as well.

We present a motion-aware gradient domain video blending technique that addresses the above issues. Our key idea is that instead of using hard boundaries between the foreground and background for composition, we define soft boundaries for the blending region. In each frame, the real boundary of the source patch will lie loosely inside the blending region, but its exact location can be adjusted dynamically through time. This allows greater flexibility to determine *dynamic boundaries* for complex moving objects, and allows fast updates for user interaction such as while the user changes the position of the source object. Moreover, we use a novel method for combining the gradient fields of the source and target videos inside the blending region. We reconstruct a *mixing-gradient field* based on the consistency, and the gradient saliency of the source and target videos. The mixing gradient gradually scatters the possible motion differences on the blending boundary between source and target videos into the whole blending region, producing smoother and more coherent blending results and reducing motion artifacts.

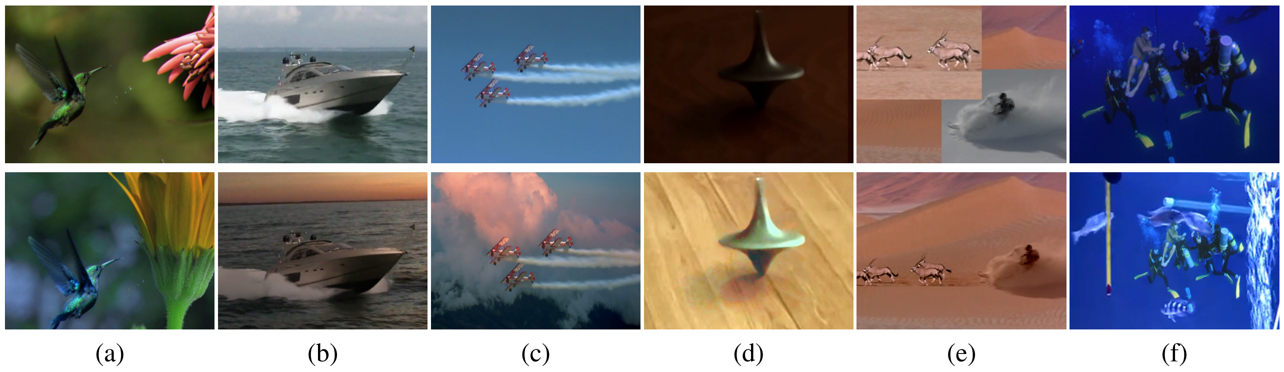


Fig. 1. Challenges for existing video composition schemes (top row, showing source videos): uncertain object boundaries due to shadows, smoke and dust, motion blur etc., and complex motions of both camera and objects. We propose a robust gradient domain video composition method that is capable of dealing with these scenarios (bottom row, showing composed videos). Please see the accompanying video.

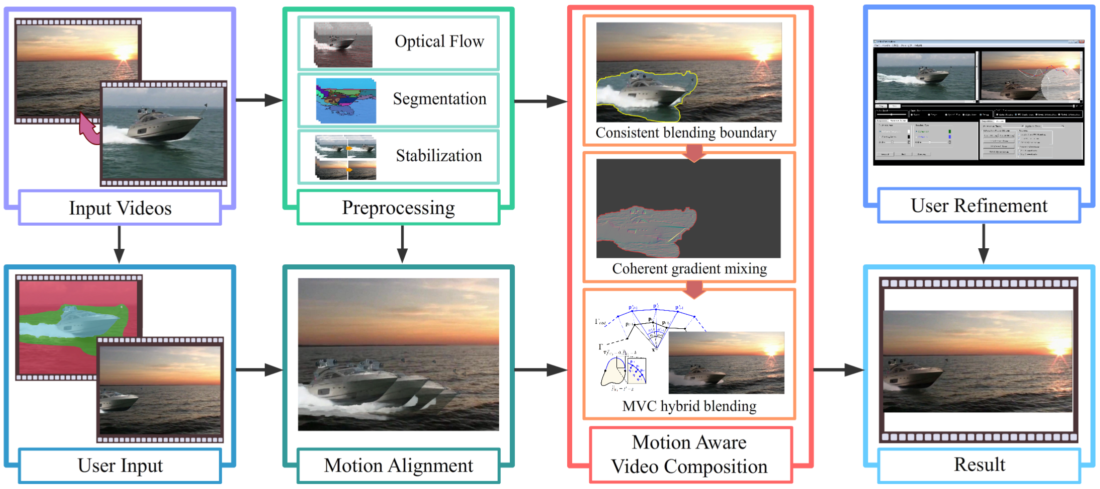


Fig. 2. Pipeline of motion aware video composition method. Given an input source and target video sequences, several preprocessing steps are first applied. These include calculating the optical flow, performing video segmentation and stabilization. The user then inputs blending trimaps on the source video and positions the source object on the first frame. The system will compose an output video by motion aware gradient composition. Consequently, the user can interactively refine the position on chosen frames.

Our solution builds on several previous work including video segmentation [7], motion estimation [8], video stabilization [9], grab-cut [10] and graph-cut [11]. We extend MVC cloning [6] with hybrid blending boundary conditions similar to [12] and use a mixing-gradient field [13]. Our contribution lies not only in creating a full interactive solution for video composition, but also in developing a motion-aware technique for gradient domain video editing.

2 Related Work

Video matting and composition is a well studied problem in computer graphics and computer vision. In 2002, Chuang et al. [1] use bi-direction optical flow to propagate the matting trimaps across video frames. In Li et al. 2005 [2], Wang et al. 2005 [3] and Armstrong et al. 2007 [14], graph-cut based video segmentation is proposed to obtain segmentation. In 2009, Bai et al. [4] present a video cutout system that achieved better segmentation by the collaboration of a set of local classifiers. All these video matting approaches rely on the clear object boundary of the source video. Gradient domain image composition avoids solving difficult matting problem for fuzzy boundaries, and can also deal with large illumination difference of source and target images. Burt and Adelson 1983 [15] uses a multiresolution spline technique to blend two images, while this method is concise and fast, the data incorporation from distant source and destination pixels may generate undesirable result. Pérez et al. 2003 [16] improves it by solving Poisson equation on blending region. Jia et al. 2006 [17] eliminates the blending artifacts by optimizing the blending boundary and collaborating with alpha matting. Chen et al. 2009 [12] further improves the composition quality by using a mixed boundary condition for solving Poisson equation. Bhat et al. 2008 [18] efficiently solves the poisson equation by using Fourier analysis. To achieve real-time performance, Farbman et al. 2009 [6] uses mean-value coordinates interpolation to approximate the process of solving the Laplacian equation in gradient domain composition. Tao et al. 2010 [19] presents an error-tolerant gradient-domain image compositing technique. By defining the boundary gradients and applying a weighted integration scheme to the gradient field, it tackles the color bleeding problem and improves the composition quality. However, motion blur, camera shaking and low quality object boundary due to the complexity of the scene or video quality still limit the applicability of these works.

Our work extends the range of possible video compositions by providing motion aware blending. Several works attempted to extend the gradient domain scheme to video composition. Wang et al. 2004 [20] proposes a 3D video integration algorithm which solves a 3D Poisson equation. Xie et al. 2010 [21] adapts the mean-value coordinates to efficiently solve the video composition. However, these methods did not take spatio-temporal and motion inconsistencies of the source and target gradients into account. This reduces the composition quality and narrows the application range.

Other related works include illumination estimation from images and videos. Since gradient domain composition does not consider the real illumination condition of the source and target, it can provide over-blending effects (pasted objects are too dark or over saturated). Lalonde et al. 2007 [22] use illumination clues to choose illumination consistent images for their Photo Clip Art. Lalonde et al. 2009 [23] further estimate the illumination details from a single outdoor image, which can help to prevent over-blending. GradientShop Bhat et al. 2010 [13] uses a general optimization framework for gradient domain image and video processing, which also inspired our work. Gradient domain composition also suffers from different image noises, which are addressed lately by Image Harmonization techniques proposed in Sunkavalli et al. 2010 [24].

3 Overview

Figure 2 shows the pipeline of our method. First, the user provides blending trimaps on the source video. In the "user input" step of Figure 2, The blue and red region respectively covers the definite foreground and background. The green region in between is a soft boundary for the blending region, and usually covers the uncertain area around the foreground source object such as shadows, dust, smoke or waves (see Figure 3 and the trimaps shown in Figure 10(a)(c)). Both the hybrid blending boundary and the mixing-gradient field will be calculated inside this soft boundary. The boundary between red and green region, denoted as Γ_{out}, is obtained by a user-drawn closed loop on the first frame. The inner boundary Γ_{in}, between the green and the blue regions, is generated by applying a refinement step of grab-cut [10] to another user-drawn closed loop roughly along the object boundary on the first frame. The trimaps on subsequent frames are generated by propagating the user inputs on the first frame (details in Section 5.3).

To compose the new video, the user places the source objects on the first frame of the target video. Using an intuitive user interface, the source and target videos can be aligned. The position and scale of the source object will be automatically calculated on subsequent frames according to feature points registration and optical flow estimation. To generate desired motion path (i.e. the object moving path across video frames) or fix inaccurate automatic alignment, the user can drag and resize the source object on some *motion keyframes*. Our method allows *interactive* composition results to be shown to the user while he or she adjusts the alignment. The details of this user interface will be presented in Section 6.

After the video source objects are positioned, the key challenge for composition is to overcome large appearance and motion differences between the source and target video sequences. We first find a blending boundary in each frame, which minimizes the spatio-temporal differences and motion differences along the boundary. Then, we reconstruct a mixing-gradient field inside the composition region according to optical flow, gradient saliency and continuity. The gradient domain blending also takes into account inter-frame consistency. We efficiently calculate per frame blending results subject to a temporal restriction along the flow vectors of the source and target sequences. These steps are described in Section 5.

Lastly, we propose an approximate calculation for mean-value coordinates, to achieve interactive real-time update when dynamic blending boundary is used. In Section 4 we describe this real-time hybrid blending method for a single frame.

4 Real-time Single Frame Hybrid Blending

Regular Poisson image blending involves solving a large linear system which is time-consuming for use in video blending. Farbman et al. [6] introduce an alternative, coordinate-based, approach to perform the cloning process of normal Poisson blending with Dirichlet boundary conditions. The interpolated value at each interior pixel is given by a weighted combination of values along the boundary based on mean-value coordinates (MVC). The use of coordinates is advantageous in terms of speed, ease of implementation, small memory footprint, and parallelizability. However, since this solution deals with a fixed blending boundary with the same boundary condition as regular Poisson blending, it also carries the same limitations, especially when there are large texture or color differences.

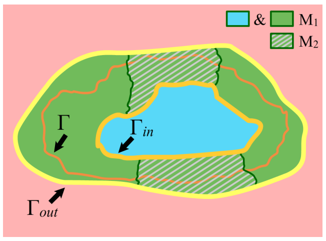
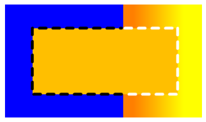
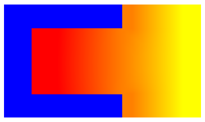
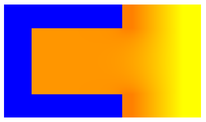
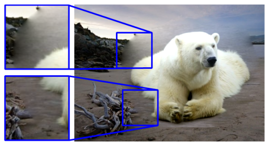


Fig. 3. The definitions of regions and boundaries in this paper.


(a)


(b)


(c)


(d)

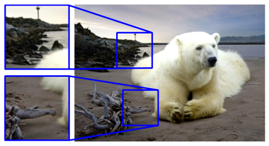

(e)

Fig. 4. Comparison between MVC cloning Farbman et al. 2009 [6] and our MVC hybrid blending. (a) Blending of the rectangle region inside the dashed line and the background. The black dash line designates the inconsistent boundary that should be avoided for blending. (b) MVC cloning with *selective boundary suppression* at the black boundary generates undesirable (red) blending result due to extrapolation of MVC. (c) MVC *hybrid blending* generates plausible blending result by setting Neumann boundary conditions at the black boundary. Next, two highlight regions show, on a real image, the advantage for using mixed boundary conditions (e, top) and optimized blending boundary (e, bottom) in MVC hybrid blending (e) compared to MVC cloning (d).

Farbman et al. [6] propose to use *selective boundary suppression* that removes some boundary points for interpolation to reduce "smudging" artifact. This fails when the removed boundary points are extruded, as shown in Figure 4(a)-(c). Jia et al. [17] propose to optimize the blending boundary to achieve minimal color variation of source object, while Lalonde et al. further improved this to deal with large texture or color differences in [22] and [12]. Here, we improve MVC method by searching for optimized boundary and using mixed boundary conditions similar to the hybrid blending approach suggested in Chen et al. 2009 [12].

Let Ω denote the patch of source frame f^s to be composed onto the target frame f^t, and let Γ be its boundary. As illustrated in Figure 3, Chen et al. [12] classify a band region around the blending boundary Γ into two types: M_1 consists of pixels where texture and color between the source and target are consistent, and M_2 consists of the all other pixels. Texture and color consistencies are measured respectively by the difference of Gabor feature vectors and the difference of the UV color components. Conventional Poisson blending can be safely applied to the M_1 region, but it can cause artifacts (e.g. "smudging" and discoloration) within M_2. In M_2, Chen et al. apply matting to separate the foreground f^s_f and background f^s_b layers of the source image f^s, and use the foreground layer f^s_f for blending to the target image f^t. This technique creates a mixed guidance vector field $\mathbf{G}$ in Ω and a mixed boundary condition on Γ:

$$\begin{gathered}\triangle f' = \mathrm{div}(\mathbf{G}) \text{ over } \Omega, \text{with} \qquad (1)\\ \mathbf{G}|_{M_1} = \nabla f^s \quad \text{and} \quad \mathbf{G}|_{M_2} = \nabla f^s_f \\ f'|_{\Gamma_1} = f^t \quad \text{and} \quad \nabla f'|_{\Gamma_2} = \nabla f^s_f \\ \text{where } \Gamma_i = \Gamma \cap M_i, i = 1,2\end{gathered}$$

In M_2 regions, f' is treated as intermediate result, which is then combined with f^t using alpha blending to obtain the final result f.

In our work, we do not solve the Poisson equation but instead use mean-value coordinates as in Farbman et al. [6] due to its efficiency. To this end, we have to address two challenges. First, the blending boundary Γ in hybrid blending is optimized according to the pixel differences (mismatch). As the source and target alignment changes, Γ is changed, and hence MVC weights can no longer be pre-computed. This reduces the effectiveness of MVC in terms of computational savings. Second, in hybrid blending, the boundary points on Γ_2 are with Neumann boundary conditions, and therefore cannot be used directly for membrane interpolation. We solve the first challenge by an approximation to MVC coordinates, and address the second challenge by first solving the boundary values that will be interpolated.

4.1 MVC approximation

In Farbman 2009 [6], the membrane value at each interior pixel of seamless cloning is given by a weighted combination of values along the boundary. The process includes three steps: region triangulation, mean-value coordinates computation and interpolation. If the blending boundary is fixed, the first two steps can be pre-computed similar to Farbman 2009 [6]. Then, multiple cloning positions of the source patch can be considered efficiently. However, using a fixed, user-drawn, boundary is not optimal for seamless blending. Dynamic blending boundaries are optimized for different blending positions. To deal with changing boundary we propose a new method to approximate the mean-value coordinates.

We observe that the most time-consuming part in MVC computation is calculating the angles from the inner points $\mathbf{x} \in \Omega$ to the points on the blending boundary $\mathbf{p}_i, \mathbf{p}_{i+1} \in \Gamma$. The tangent of these angles $\sphericalangle \mathbf{p}_i, \mathbf{x}, \mathbf{p}_{i+1}$ are used as weights in the MVC calculation (see Figure 5). To bypass computing these angles every time the boundary Γ is changed, we approximate these angles using pre-computed values that are close to the actual angle values.

To achieve this we require a fixed boundary that encloses all possible dynamic blending boundaries. The user-drawn boundary Γ_{out} naturally fits this role, since all the blending boundary optimizations are applied inside it. We construct an adaptive triangular mesh inside Γ_{out} by using a similar method as in Farbman 2009 [6], but using a constant vertex density in the region between Γ_{in} and Γ_{out}, that is identical as the vertex density on Γ_{out}. For each inner vertex $\mathbf{x}$, we calculate the angles to all vertices on Γ_{out}: $\alpha'_{i-1} = \sphericalangle \mathbf{p'}_{i-1}, \mathbf{x}, \mathbf{p'}_i$ and $\alpha'_i = \sphericalangle \mathbf{p'}_i, \mathbf{x}, \mathbf{p'}_{i+1}$ (see Figure 5).

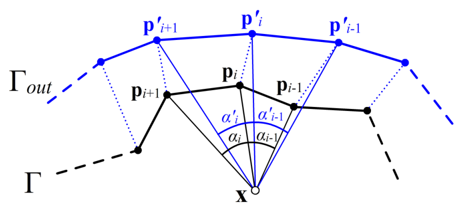


Fig. 5. Angle approximation for mean-value coordinates using a fixed enclosed boundary Γ_{out} (blue).

The points **p'** are sampled on Γ_{out} by the hierarchical boundary sampling described in Farbman 2009 [6]. The tangents of these angles (divided by 2) are saved for all vertices. Then, each time a blending boundary Γ is optimized for a given position of the source patch, we compute distinct correspondence points on Γ_{out} for each point on Γ. To do so, we first link each point on Γ to their nearest point on Γ_{out}. Then, if multiple points on Γ are linked to a same point on Γ_{out}, we choose the one with shortest distance as the "key-correspondence". All other correspondences between the key-correspondences are uniformly distributed between them (see dotted lines in Figure 5). Now, the jth component (before normalization) of our approximated MVC for a vertex $\mathbf{x}$ is given by:

$$w^j = \frac{\tan(\alpha'_{j-1}/2) + \tan(\alpha'_j/2)}{||\mathbf{p}_j - \mathbf{x}||} \tag{2}$$

Since the tangents are pre-computed, the approximated MVC can be computed in real-time. Although this approximation is different from the true MVC, especially when Γ is close to Γ_{in}, in practice the approximated interpolation is visually indistinguishable with the MVC interpolation.

4.2 Boundary value solving

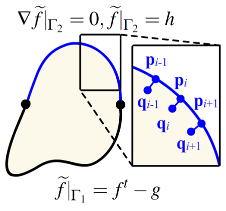
When the hybrid blending boundary contains Neumann boundary conditions, it cannot be directly approximated by MVC interpolation, since some boundary values for the interpolation is unknown. Thus we solve these values before applying our approximated MVC interpolation.

We first convert the Poisson equation to Laplacian equation. The mixed boundary condition is also modified appropriately. To obtain its Laplacian form, we define the correction function $\widetilde{f}$ on Ω such that $f' = g + \widetilde{f}$, where g is the image that provides the source gradient field **G** (see Section 5.2). Hence, the Laplacian form of hybrid blending becomes:

$$\triangle \widetilde{f} = 0 \text{ over } \Omega, \text{with} \tag{3}$$

$$\widetilde{f}|_{\Gamma_1} = f^t - g \quad \text{and} \quad \nabla \widetilde{f}|_{\Gamma_2} = 0$$

For the boundary points on Γ_2 with Neumann boundary conditions we only know the gradient. Hence, we assume their values fit a function $\widetilde{f}|_{\Gamma_2} = h$ (illustrated on the right). Then, by considering the coordinate-based interpolation, we can form a


$\nabla\widetilde{f}|_{\Gamma_2} = 0, \widetilde{f}|_{\Gamma_2} = h$
$\mathbf{p}_{i-1}$
$\mathbf{p}_i$
$\mathbf{p}_{i+1}$
$\mathbf{q}_{i-1}$
$\mathbf{q}_i$
$\mathbf{q}_{i+1}$
$\widetilde{f}|_{\Gamma_1} = f^t - g$

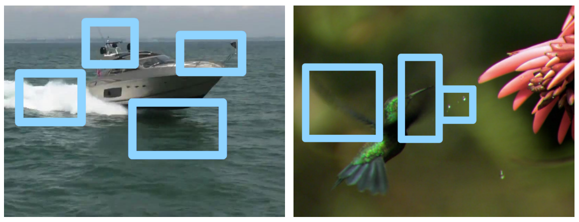


Fig. 6. Challenges for video composition. Motion blur, uncertain boundaries, shadows and reflections are demonstrated inside the blue frames.

small linear system by the pixel values around the Neumann boundary and the known gradient values $\nabla \widetilde{f}|_{\Gamma_2} = 0$.

$$h(\mathbf{p}_i) - \sum_{\mathbf{p}_j \in \Gamma_1} w_i^j (f^t(\mathbf{p}_j) - g(\mathbf{p}_j)) - \sum_{\mathbf{p}_k \in \Gamma_2} w_i^k (h(\mathbf{p}_k)) = 0, \tag{4}$$

where w_i^j and w_i^k are mean-value coordinates for $\mathbf{q}_i$ (neighboring points of $\mathbf{p}_i$) in Γ_1 and Γ_2 respectively. We solve this linear system, which usually contains hundreds of unknowns $h(\mathbf{p}_i)$ by LU factorization. Then, we apply our approximated MVC interpolation (discussed in previous section) to the solved boundary values $h(\mathbf{p}_i)$.

Figure 4(b)-(e) shows the comparison between Farbman et al.'s MVC cloning [6] and our MVC hybrid blending. In practice we have found that the difference between the hybrid blending based on Poison and hybrid blending based on MVC interpolation are almost indistinguishable. On the other hand, after pre-computation, the MVC interpolation is about two orders of magnitudes faster. This is the key technique that enables efficient gradient domain blending on video and interactive editing.

5 Motion Aware Video Composition

The image blending method described in the previous section can produce plausible composition results even for foreground objects that are difficult to extract using conventional alpha-matte cut-out methods (e.g., see the regions inside the blue boxes in Figure 6).

However, if we compose each frame (e.g., the boat in Figure 6(a)) independently without considering inter-frame consistency, the results will suffer from boundary flickering, inconsistent color and texture motion around the composed source object. These artifacts are caused by three reasons:

Motion differences. There may be differences in the color and texture motion between the source patch and the target surrounding. These differences can cause a visible seam to appear on the blending boundary in the video, not only because of foreground object movement, but also because of surrounding movements (e.g., smoke, haze, water). As an example, the water flow motion around the boat in the source patch will not be consistent with the water flow motion of the rest of the target frame.

Position mismatch. Due to our use of dynamic composition boundary, it can change in consecutive frames and cause regions to pop or disappear.

Temporal Fluctuations of Color mismatch. The differences in color between the source and the target on the blending boundary may also change between the frames. The color mismatch is used as the boundary conditions in our blending method and therefore affects the color inside the blending region. This means that changes in the mismatch in consecutive frames can cause color flickering of the entire blending region.

To address these issues, we first create coherent blending boundaries that changes smoothly through the frames, and then construct a novel mixing-gradient field to guide the blending by considering both the source and target gradient fields throughout the entire video.

5.1 Optimizing blending boundary

We first minimize the boundary color mismatch between the source patch and target surroundings on the blending boundary. The minimization will effectively suppress the variation of the appearance of source object in the blended video. We extend the boundary optimization method of hybrid blending (Section 4) from images to video, taking temporal coherency into account. We first apply video segmentation [7] to both the source and target videos to create super-voxels. Then, we calculate the texture and color differences for the corresponding super-voxels of the two videos in the soft boundaries to classify them to either M_1 or M_2 regions as described in Section 4. The blending boundaries of M_2 are generated by coherent matting [4], and we only need to optimize the position of the blending boundary in M_1.

Boundary optimization can be effectively performed using dynamic programming [17]. Extending this to video should not only preserve the preceding function, but also minimize the artifacts caused by the above three reasons. While directly optimizing the boundary on the 3D volume can solve the position mismatch problem, it cannot resolve motion differences or color mismatch. Another drawback of optimizing on the 3D volume is that neighboring pixels along the time axis are usually not continuous due to motion. Moreover, a continuous blending boundary in 3D space is over-restrictive and leads to non-optimal results on each frame. As discussed in [25], [4], [13], a better solution is to optimize the consistency of motion-compensated neighbors instead of the direct neighbors. Bhat et al. [13] also show that this can be done by following a mostly (50% to 60%) accurate optical flow, and by taking confidence values from the motion vectors into account. We follow a similar approach in this work.

We define a cost function for the blending boundary Γ of each frame, using additional terms to control the temporal consistency according to the motion vectors:

$$E(\Gamma,k) = \sum_{\mathbf{p}\in\Gamma} \frac{1}{D_{\mathbf{p}}} \{(k_{\mathbf{p}}-k)^2 + (k_{\mathbf{p}}-\overline{kt}_{\mathbf{p}})^2 + ||\mathbf{v}^t_{\mathbf{p}} - \mathbf{v}^s_{\mathbf{p}}||^2\} \tag{5}$$

There are three terms here for each boundary point $\mathbf{p}$. In the first term, $k_{\mathbf{p}}$ is the color mismatch of the target and source on $\mathbf{p}$, $k_{\mathbf{p}} = (f^t_{\mathbf{p}} - f^s_{\mathbf{p}})$, and k is the average mismatch in the current frame. This term minimizes the color mismatch on a single frame, and is similar to Jia et al. 2006 [17]. In the second term, $\overline{kt}_{\mathbf{p}}$ is the average mismatch between $\mathbf{p}$ and the nearest boundary points on the two adjacent frames. This term minimizes the color mismatch between consecutive frames. In the third term, $\mathbf{v}^t_{\mathbf{p}}$ and $\mathbf{v}^s_{\mathbf{p}}$ are target and source optical flow vector from $\mathbf{p}$ to its corresponding point in the next frames on the target and source respectively. This term seeks a boundary with similar optical flows so that *motion differences* are reduced. $D_{\mathbf{p}}$ is a penalty term that minimizes boundary point distances across frames, which will be discussed shortly. Note that $D_{\mathbf{p}}$ is not a hard constraint and noticeable boundary offsets may still occur after optimization. Our scheme gives higher priority to color mismatch, and we rely on the gradient mixing in the next subsection to remove popping artifacts caused by the position mismatch.

We iteratively minimize the cost function in Equation (5). Initially, k is set to be the average mismatch on the user-provided outer boundaries Γ_{out}, $D_{\mathbf{p}}$ is set to be 1, and the second term $(k_{\mathbf{p}} - \overline{kt}_{\mathbf{p}})^2$ is set to be 0. In each iteration a new boundary is calculated using dynamic programming, where the cost value for each pixel is set to be the sum of the three terms in Equation (5). Next, we project the boundary points to their motion-compensated neighbors in adjacent frames, in both forward and backward directions, using optical flow vectors (Figure 7). Note that the target and source optical flow vectors can generate two different projections for a single boundary point $\mathbf{p}$ on an adjacent frame. We consider the nearest distance from $\mathbf{p}$ to its projections on each adjacent frame, and set $D_{\mathbf{p}}$ to be the average of the two nearest distances to the two adjacent frames. Accordingly, $\overline{kt}_{\mathbf{p}}$ is set to be the average of the color mismatch between $\mathbf{p}$ and the two nearest projected pixels on those two frames. In short, in each iteration we first calculate the new boundaries and then update k, $D_{\mathbf{p}}$ and $\overline{kt}_{\mathbf{p}}$. The iteration terminates when the boundaries converge or a maximum number of iterations is reached (we use 20).

5.2 Gradient mixing

To deal with the remaining artifacts caused by position mismatch and motion differences, we propose to mix the source and target gradients coherently in the soft boundary region.

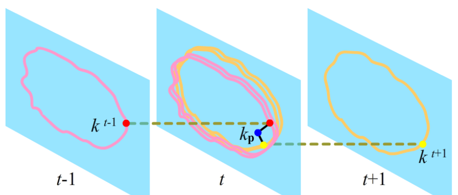


Fig. 7. Pixels used to calculate the temporal cost in blending boundary optimization.

Mixing of gradients for image cloning was proposed by Pérez et al. [16] to preserve salient contents in both source and target images. We extend this approach to videos in a spatio-temporally coherent manner. This gradient mixing is applied on the band region between Γ_{in} and blending boundary Γ. The key idea is to create a gradually mixed gradient field in the spatial domain to reduce the motion inconsistency artifacts. If $\mathbf{G}_i^s$ and $\mathbf{G}_i^t$ are the source and target gradients respectively, then in its naive form, mixing could be obtained for i-th pixel of this region as a linear combination of the source and target gradients:

$$\mathbf{G}_i = \alpha_i \cdot (\mathbf{G}_i^s) + [1-\alpha_i] \cdot (\mathbf{G}_i^t), \tag{6}$$

However, our approach is based on the observation that different situations may need different mixing rules to obtain the desired results by preserving content either from the source or the target video. For example, motions with greater gradient values are more noticeable, and they usually depict the structure of the source object. Hence, larger gradients should be preserved as much as possible. Moreover, to generate temporally coherent results, the preservation of the gradient field should be temporally consistent in motion-compensated neighbors. We use user-defined mixing parameters that address the specific situation of the composition (see Table 1).

The use of MVC cloning demands that the mixed gradient field should be conservative. We reconstruct a conservative mixing gradient field as the guidance vector field according to the following: 1. the gradient magnitude of source and target, 2. the spatio-temporal coherence of the source and target gradients. We define $\mathscr{F}(\mathbf{G})$ as a filtering function governed by the gradient mixing parameters whose values are set by the user. These values can be different for the source and the target filtering. We also define $\mathbf{a}_i = (\alpha_i^x, \alpha_i^y)$ as the mixing weighting factor. Note that to generate a conservative gradient field, α_i^x and α_i^y can be different. For i-th pixel of this region, the reconstructed gradient value is defined as a linear combination of the filtered source and target gradients:

$$\mathbf{G}_i = \mathbf{a}_i \cdot \mathscr{F}(\mathbf{G}_i^s) + [(1,1) - \mathbf{a}_i] \cdot \mathscr{F}(\mathbf{G}_i^t), \tag{7}$$

In the following, we show how to define the mixing parameters and how to use them to generate the gradient filtering functions $\mathscr{F}$, and the initial values for $\mathbf{a}_i^{init} = (\alpha_i^{init}, \alpha_i^{init})$.

Mixing parameters. We provide three gradient mixing parameters, namely "salient", "base", and "distinct". Each one could be toggled either on or off for both the source or the target gradient field. The "salient" toggle decides whether to preserve gradients that are more salient (large) in the source and target videos. The "base" toggle decides whether to preserve the low frequency appearance of the videos. These toggles affect the filtering function $\mathscr{F}$ of the gradient field $\mathbf{G}$ (either source or target) as follows:

$$\mathscr{F}(\mathbf{G}) = \begin{cases} \mathbf{G} - K * \mathbf{G} & \text{if only "salient" is on,} \\ K * \mathbf{G} & \text{if only "base" is on,} \\ \mathbf{G} & \text{if both are on,} \\ 0 & \text{if both are off} \end{cases}$$

here $K = N(\mathbf{p}_i; \sigma^2)$ is a Gaussian filter centered at the i-th pixel (we set $\sigma = 5$), and $*$ is the convolution operator. The "salient" toggle is effective for preserving edges and sharp textures, while the "base" preserves shadows, reflections and smooth textures.

The "distinct" parameter decides whether to preserve regions with distinct motion, including motion that differs from its surrounding area and motion with low optical flow confidence (which usually means unpredictable motion). The user can toggle this setting to affect the initial mixing weight α_i^{init}:

$$\alpha_i^{init} = \frac{||\mathbf{G}_i^s||^2}{||\mathbf{G}_i^s||^2 + ||\mathbf{G}_i^t||^2} + (d^s \mathscr{D}_i^s - d^t \mathscr{D}_i^t), \tag{8}$$

where $||\mathbf{G}_i^s||$ and $||\mathbf{G}_i^t||$ are the gradient magnitudes of the source and target gradient respectively at the i-th pixel. The first term favors the preservation of larger gradient magnitudes. d^s or d^t are equal to 1, if the "distinct" toggle for source or target is on respectively, and 0 otherwise. $\mathscr{D}_i^s$ and $\mathscr{D}_i^t$ are the "distinct" measures for the source and target respectively. When only considering mono-directional optical flow, the measures are defined as :

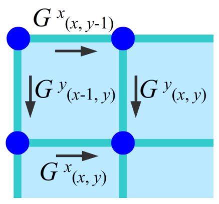
$$\mathscr{D}_i = \begin{cases} \frac{||(\mathbf{DoG} * \mathbf{V})_i||^2}{\psi} & \text{if} \quad wv_i > 0.5, \\ 1 - wv_i & \text{otherwise,} \end{cases} \tag{9}$$

where $\mathbf{DoG} * \mathbf{V}$ is the convolution of $\mathbf{DoG}$ (difference of Gaussian) and the optical flow vector field. The two bandwidths of the $\mathbf{DoG}$ are 12 and 3. The scale factor ψ is set to 50. wv_i is the confidence of optical flow on the i-th pixel. To take the optical flow fields of both directions into account, we use the average value of the two $\mathscr{D}_i$ generated by both optical flow fields. The value of α_i^{init} is clamped to $[0,1]$. Toggling "distinct" directly changes the initial mixing weight of gradient mixing, thus changing the gradient mixing result. This parameter emphasizes motion that are distinct from their background inside the blending region. Note that this region usually contains scene objects that move together or affected by the foreground object, such as motion blur, shadow, wave, smoke, dust, etc. All user-defined mixing parameter values for examples of this paper are shown in Table 1.

After the initial mixing weights and filtered gradients are determined, we define a cost function for each $\mathbf{a}_i$ to ensure feature preservation and smoothness:

$$C(\mathbf{a}_i) = ||\mathbf{a}_i - \mathbf{a}_i^{init}||^2 + \\ + w_1 \sum_{j \in \mathbf{N}_{tem}(i)} ||\mathbf{a}_i - \mathbf{a}_j||^2 + w_2 \sum_{k \in \mathbf{N}_{spa}(i)} ||\mathbf{a}_i - \mathbf{a}_k||^2 \tag{10}$$

The first term requires that the mixing weights are similar as the initial mixing weight $\mathbf{a}_i^{init}$. The second and third terms are respectively penalize temporal and spatial variations. $\mathbf{N}_{tem}(i)$ is the set of indices of the motion-compensated neighbors of the i-th pixel, if the corresponding optical flow confidence is higher than 0.5. Since we may have up to four optical flow vectors (source and target in both forward and backward directions) for each pixel, $\mathbf{N}_{tem}(i)$ contains at most 4 elements. $\mathbf{N}_{spa}(i)$ is the set of indices of the 4-connected neighbors of the i-th pixel on a single frame. We require that the mixing weights $\mathbf{a}_i$ are all 0 on Γ and 1 on Γ_{in}. The third term effectively drives the mixing weights to gradually increase from Γ to Γ_{in}, which smooths the motion differences between the source and target inside the blending region. We set w_1 and w_2 to 0.5 and 0.3 in all our results.



Assuming that the coordinate of i-th pixel on its frame is (x,y), as illustrated on the right, any $\mathbf{G}_i = (G^x_{(x,y)}, G^y_{(x,y)})$ must satisfy the following equation to produce conservative mixed gradients:

$$G^x_{(x,y-1)} + G^y_{(x,y)} = G^y_{(x-1,y)} + G^x_{(x,y)} \tag{11}$$

Substituting Equation (7) into Equation (11), we get a linear equation involving $\alpha^x_{(x,y-1)}, \alpha^y_{(x,y)}, \alpha^y_{(x-1,y)}$ and $\alpha^x_{(x,y)}$. Combining these equations for all pixels yields a set of linear constraints for all $\mathbf{a}_i = (\alpha^x_i, \alpha^y_i)$. The cost function in Equation (10) is minimized with this set of linear constraints, which ensures that the mixing of the gradients is still conservative. The quadratic programming problem can be transformed to a large sparse linear system by using lagrange multipliers. We iteratively solve the system by successive over-relaxation with relaxation factor $\omega = 1.8$. In practice, we found that after five iterations, even with no convergence, the resulting mixing weights already ensure the smoothness of gradient mixing. We then replace the source gradient field in the soft boundary region by the mixing-gradient field and compute blending using the composition technique discussed in Section 4.

Figure 8 shows the gradient fields before and after mixing, the gradient alpha map, and the blending results with and without gradient mixing. Even in a single frame composition, gradient blending produces better results than simple blending. For video blending, it further reduces the motion artifacts described above (please see the accompanying video).

5.3 Coherent MVC cloning

As described in Section 4 the final composition on each frame is achieved using MVC hybrid blending. The triangulation and angle tangents for MVC are calculated using Γ_{out}. Hence, to reduce computation cost we would like Γ_{out} to remain as consistent as possible through frames. After the trimap is provided on the first frame, the boundaries are propagated to subsequent frames. The inner boundaries Γ_{in}

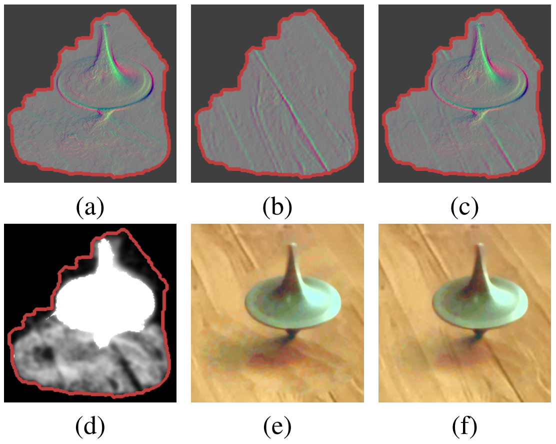


Fig. 8. Gradient mixing example: the red line is the blending boundary Γ. (a) source gradient. (b) target gradient. (c) mixing-gradient. (d) gradient alpha map, illustrated by the norm of alpha vector. (e) composition result without mixing. Note the evident seam between the textures in this case. On videos, such seams also appear when there are motion differences. (f) composition result using our mixing-gradient field.

are propagated using the Bai et al.'s method [4], which relies on overlapping local classifiers and local window propagation. The outer boundaries Γ_{out} are sampled and their correspondence points found on Γ_{in} similar to the method explained in Section 4. Γ_{out} are propagated to subsequent frames by copying the translations of the corresponding points on Γ_{in}. Whenever there is a large deviation from the desired boundary the user can refine it manually on a specific frame. This affects the propagation to later frames. Next, we project Γ^i_{out} to Γ^{i+1}_{out} by a similarity transformation. If the union of projected boundary and Γ^{i+1}_{out} does not deviate too much (10%) from both boundaries, we use the union as the new Γ_{out} on both frames. In such a way, Γ_{out} will be updated and clustered to several groups over time. In each group, the only differences of Γ_{out} are within a similarity transformation, and the triangulation and angle tangents will be computed only once for each group.

To further reduce flickering, we smooth the membrane value of MVC interpolation at the mesh vertices temporally. Our membrane may move together with the source object, which is different from the fixed membrane mentioned in Farbman 2009 [6]. Therefore, instead of smoothing temporal neighbors in their paper, we smooth the motion-compensated neighbors and modify the smoothing weights of older frames accordingly. The smoothed vertex membrane value at frame i is calculated as follow:

$$\frac{1}{W}[m_i + 2^{-d}(\sum_{j=i-5}^{i-1}(i-j)^{-0.75}(||\mathbf{a}||m^s_j\prod_{k=j}^{i-1}wv^s_k + ||(1,1)-\mathbf{a}||m^t_j\prod_{k=j}^{i-1}wv^t_k)],$$

where m_i is the vertex membrane value before smoothing, m^s_j and m^t_j are membrane values of the motion-compensated neighbors on frame j according to source and target optical flow respectively. $\mathbf{a}$ is the gradient mixing weight. wv^s_k and

wv_k^t are the optical flow confidences of the source and target optical flow respectively on frame k. d is the normalized distance to boundary Γ. W is the weight sum for all involved m.

6 Interactive Positioning

Matching a source video object to a target video is challenging not only because the scale, angle, position and lighting must match in all frames, but also because the motion along the frames must be consistent with the target video scene. Our blending technique could compose source objects to the background better, since we seamlessly transfer the shadows and other effects around the transplanted object. However, it can still appear unrealistic if we cannot position the object correctly.

We provide interactive positioning to extend the flexibility of our motion-aware composition framework. There are two main goals for the operation: first, to better align the motion of the source and target videos, and more importantly, to allow fine tuning and editing of the motion of the source object on the target video. We define the trajectory of the center of the source object on the target frame as the object's *motion path*. Our user interface system design combines automatic computation with user interaction to lower the user's effort and allow fast update for interactive feedback while editing. This is achieved by constraining the possible user edits only to specific points along the object's motion path, as well as constraining the magnitude and orientation of velocity changes. These changes are propagated automatically to other frames, reducing the user's effort. Moreover, this prevents unnatural motion artifacts to be created, as well as allows efficient local updates. Together with a method to interleave interaction and computation, we create an interactive-feedback system for easy motion refinement and editing.

6.1 Motion alignment

To remove shaky camera motion, we first stabilize the source and target videos according to Zhang et al. 2009 [9]. This helps the subsequent video registration step. Zhang et al. 2007 [26] performs automatic metric reconstruction from long video sequences with varying focal length. We use this method to recover the camera parameters and remove the camera motion for both source and target videos. In our case, we remove the moving source object from the source video to prevent outliers. In our system the user places (the rotation and scale are also allowed) the source object on the first frame, and the system calculates a motion path using alignment automatically. Still, trying to neutralize the camera motion for all types of complex video scenes would be impractical. Hence, we allow the user to correct the position of the object on some frames manually.

6.2 Motion editing

After automatic alignment and user refinement, good results could be achieved in terms of video composition. However, users may still want to speed up, slow down, or change the trajectory of the object. These innovative effects can hardly be achieved by simple composition with aligned path, and we realize them using interactive motion editing.

Allowing arbitrary modification on any point in the object's path will break the intrinsic property of the motion path, which later causes unnatural artifacts. Moreover, each time the path changes, the boundary optimization (Equation (5)) and gradient mixing (Equation (10)) have to be computed again. This becomes a bottleneck for interactive update performance. For our boat example, it only takes 1.04 seconds for MVC hybrid blending, but it takes 6.64 seconds to optimize Equation (5) and Equation (10) per 100 frames. Hence, to prevent unnatural motion artifacts, and for more efficient computation time, we intelligently constrain the editing of the motion path both to very specific points in time and to a preset amount.

The key idea is to explore both the motion properties of the object and the visual feature of the frames to obtain several *control points*, which indicate *motion keyframes*. Only control points can be edited by the user, and their modifications (including translation, rotation and scale) can be propagated to other frames by linear interpolation. These points are also the optimal points for adjustment. The motion path is divided into several segments based on two criteria: (a) the motion in one segment can be approximated by a uniform motion in a straight line, and (b) the motion property and visual feature should be stable on the control points defining the segments.

Motion property is measured by speed, and visual feature is measured by the color mismatch on the blending boundary mentioned in Section 5. These two criteria retain two properties: motions discontinuity at the control point, and stability of the control point itself. We greedily find n control points on the motion path by minimizing the following energy function:

$$E_n = \mu_1 \sum_{i=1}^{n} (k_{t_i} - \frac{k_{t_i-1} + k_{t_i+1}}{2})^2 + \mu_2 \sum_{i=1}^{n} \vec{\mathbf{v}_{t_i}}^2 + \mu_3 \sum_{i=1}^{n-1} \sigma_i^2, \qquad (12)$$

where t_i is the frame number of the i-th control point. k_{t_i} is the average color mismatch on the blending boundary of frame t_i. $\vec{\mathbf{v}_{t_i}}$ is the object velocity on frame t_i. Both $||\vec{\mathbf{v}}||$ and k are normalized to $[0,1]$. σ_i is the variance of $\vec{\mathbf{v}}$ inside the i-th segment of the motion path with the control points as the ends of segment. Initially, the motion path only contains one segment and the first two terms of Equation (12) are equal to zero whereas the last term is large. We greedily find the control points until Equation (12) stops decreasing. We set $\mu_1 = 0.2, \mu_2 = 0.2$ and $\mu_3 = 0.6$ in our experiments.

Using the control points we divide the motion into several segments of approximated uniform motion in a straight line. If the user edits the control point, these uniform motions will be preserved to a large extent. In contrast, if modification is applied on an internal segment point, it will create a new discontinuity. In addition, we impose local modification constrains on the magnitude and orientation of velocity changes for editing. With these restrictions, we bound the area where the user could change the path positions. When the user is editing the k-th control point, assuming θ_i, θ_i' are the old and new orientation of the i-th point's velocity, and v_i, v_i' are its old and new magnitudes, then the restriction is computed as follows:

$$\forall i \in [n_{k-1}, n_{k+1}], \max(\theta_i' - \theta_i)^2 < T_\theta \text{ and } \max(v_i' - v_i)^2 < T_v$$

To maintain the perspective continuity of composition scene, we set $T_\theta = 30^o$ and $T_v = 20$ as threshold values for all our examples. Figure 9 illustrates this control-point-based motion editing.

Another key advantage of using control points lies in the ability to apply only local updates and to interleave interaction with computation time. When the user edits the path, we do not perform global optimization of Equation (5) and Equation (10) for the whole sequence. Instead, we calculate the blending results on segments which were changed previously but are not being edited. If, due to previous calculating, the boundary, mixing-gradient and MVC membrane values are available on the end frames of these segments, they will become hard constraints in the optimization. In summary, we interleave computation with the user's interaction time to compute the results on other, non-edited parts. In our experiment, editing one point requires 1-2 seconds, in which we can compute 20-35 frames based on local optimization. Because of motion stability of the control points, and because the coherent temporal restriction mainly utilizes the information of consecutive frames, we can merge several passes of local optimizations together to approximate the global optimization.

Compared with 3D Poisson blending, our algorithm allows interactive video editing both because of lower computation complexity and because of our control-points optimizations. While editing one particular frame, the user can see the real-time result only on this frame via MVC hybrid blending. The segments that are not calculated, being calculated and have been calculated are visualized by different colors (red, yellow and green respectively) on the motion path. The user can press a "play" button to check the result of videos using local optimization instantly. The user can also lock control points to avoid unnecessary re-calculation of decided segments (please see supplementary video for real examples).

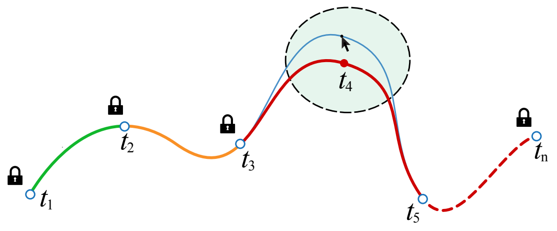


Fig. 9. Interactive motion editing. $t_1, t_2, ...t_n$ are control points on *motion keyframes*. The editing of the control points only affects segments between locked control points.

7 Experimental Results

We tested our video composition system on several challenging examples and demonstrate the results in the paper and accompanying video. Table 1 shows the gradient mixing parameters for each example. Figure 1(a) shows a slow motion hummingbird blended onto a stop-motion flower. The blending trimap of this example is demonstrated in Figure 10(a). Our blending faithfully preserves the rapid motion of the wing of the hummingbird and the water-drops. Figure 1(b) pastes a boat on the sea surface under sunset. The original source and target sequences contain severe camera shaking, and the water flow direction is different. The waves, the reflections, and the flag with the pole which are challenging to previous methods are faithfully preserved, while the motion of the boat is natural due to interactive positioning. Figure 1(c) shows several planes blended into a cloud scene. Our method preserves the clouds between and through the planes and smoke without requiring complicated matting. Its blending trimap is shown in Figure 10(c). Figure 1(d) shows a spinning top cloned to a different table. Our method preserves the shadow and reflection of the spinning top, while adjusting the texture to match the target table. Figure 1(e) is an interesting chasing scene which pastes a snow mobile into a desert scene while chasing some goats. Figure 1(f) pastes a group of diving people into a fish tank, produce an interesting montage scene. The swimming fish in the target video and the bubbles in the source video are both preserved without sudden disappearances thanks to our temporal coherent motion-aware blending technique. Figure 1(a) and (d) demonstrate that our blending method can deal with serious motion blur of source objects, Figure 1(b), (c) and (e) demonstrate that our method is capable of preserving wave, smoke and dust effects in video. Our method also provides more harmonious composition to the target environment.

Performance. All our experiments were performed on a PC with an i7 920 Quad core CPU, 12 GB RAM. Our preprocessing includes dense optical flow calculation, video segmentation for deciding the blending boundary type, and video stabilization (if necessary). Using our un-optimized implementation, these three steps usually take 5, 2 and 2 minutes per 100 frames of a VGA video. Then, the user can interactively draw the blending trimap and set the gradient

	Source			Target		
	salient	base	distinct	salient	base	distinct
bird	√		√	√	√	
boat	√	√	√	√		
planes	√			√	√	
top	√	√	√	√		
snow	√			√		
dive	√		√	√		√

TABLE 1
The gradient mixing parameters for each example.

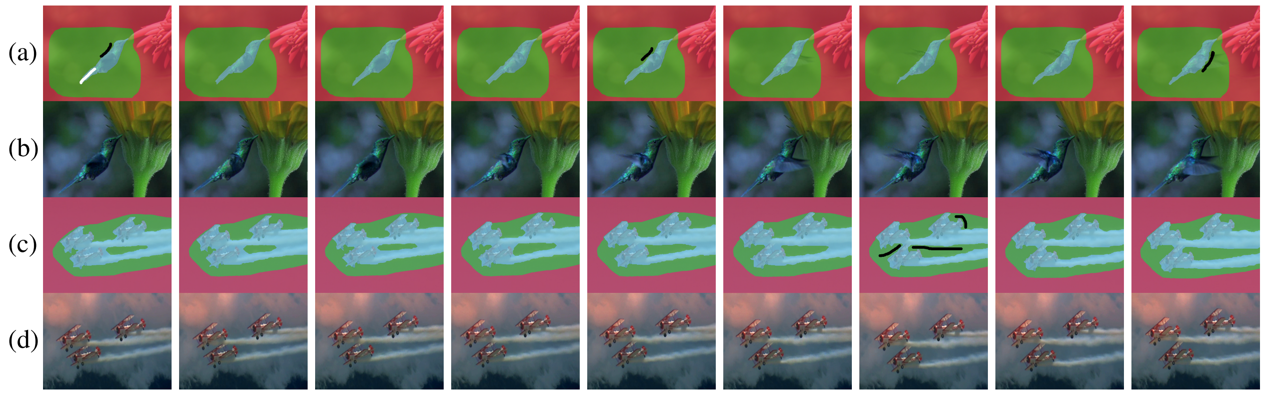


Fig. 10. The input trimap and blending result. The user interaction strokes are shown in black and white. Please see the accompanying video for the results.

mixing parameters. The number of needed user strokes for generating 100 frame trimaps is usually below 10, thanks for the fact that we don't need to extract boundary of the object. Next, the system generates an initial blending result for each frame with the automatic alignment. With this initial blending, we also save the vertex and angle information for the MVC hybrid blending. Then, the user can begin the interaction for motion alignment refinement and motion editing.

The computation time and user interaction amount in each step is shown in Table 2. The interaction time includes the times for generating trimaps and (optional) motion editing. The video blending time indicates initial blending time and blending update time in motion editing. The current frame blending rate is achieved by utilizing two cores of CPU. Although it is approximately three times slower than original MVC cloning due to boundary optimization and distances computation etc., it is still an order of magnitude faster than solving Poisson equation in hybrid blending (by TAUCS sparse linear solver). In practice, we restrict the current frame blending to single thread, and limit its rate to below 24 fps to save the computation power for segment blending optimization. Since our method uses an approximated MVC, we also show the RMS differences of the hybrid blending results. These values indicate that the difference between the resulting images is unnoticeable. Using per segment blending optimization produces additional RMS differences from the global optimization. This could be distinguishable especially on motion keyframes. However, as long as temporal coherence is maintained between these frames, the blending results are still plausible.

Comparison. The supplementary video shows some comparisons of our method to alpha blending using alpha matte generated by Bai et al. 2009 [4], frame-by-frame hybrid blending, 3D Poisson blending [20] and Xie's method [21]. Alpha matting results lose many details around the source object, and the composition is unrealistic due to large illumination differences. Xie's method relying on matting also creates details loss. We further compare the interaction amount to alpha matting in Table 2. It shows that alpha matting usually requires 5-10 times more strokes than our method, even though the obtained alpha mattes on those examples are still not optimal. The results of frame-by-frame blending and Xie's method lack temporal coherency. The flickering on the boat and the wake is due to temporal fluctuations of color mismatch, which can be tackled by our unique blending boundary optimization step. In frame-by-frame blending and 3D Poisson blending, the inconsistent motion along the blending boundary reveals very noticeable seam, this problem is solved by gradient mixing in our result. Our composition results are not affected by the artifacts mentioned above, and are also achieved at an interactive rate.

Limitation. When the blending boundaries of the source and target video sequences are very inconsistent, i.e. their appearance and texture always have large differences, our hybrid blending based method degenerates to video matting. For example, compositing a white rabbit against a black wall. However, such cases are unsuitable for blending in the first place. Inconsistent lighting direction can also create unnatural blending results. This may be solved by illumination estimation and relighting. Another limitation is that our method cannot align video scenes with large camera rotation differences, since we are missing the 3D information. Since our positioning is only applied on image plane, it won't be able to edit complex motion of the object like in 3D real world. For example, our editing tool can't distinguish whether the snow mobile is jumping up or moving away from the screen. However, this can be improved by reconstructing a ground plane of the target scene and add more degrees of freedom for positioning. Our method also relies on various computer vision techniques for pre-processing, and their imperfection could also create artifacts. For example, although our boat result vastly outperforms those generated by existing approaches, one can still notice a blurred region on the deck of the boat, which is caused by the inaccuracy of optical flow and its confidence around the thin handrail of the boat. Adopting more recent and advanced approaches like Liu et al. 2011 [27] may improve the results.

In general, not every pair of video sequences is suitable for composition. Large illumination differences, inconsistent background environments and various motions from source object, camera, and background could all prevent good

	frame number & frame size	average cloned pixels per frame	preprocessing time(s)	interaction time(s)	trimap strokes	matting strokes	video blending time(s)	current frame blending rate	MVC RMS
bird	$115 \times 1280 \times 720$	284,316	814	82	22	98	12.55	37	0.042
boat	$91 \times 1280 \times 720$	182,405	855	127	7	43	6.64	88	0.037
planes	$106 \times 1440 \times 1080$	101,346	1,274	104	5	27	4.10	137	0.024
top	$82 \times 330 \times 300$	37,762	65	25	4	39	0.93	211	0.018
snow	$94 \times 1280 \times 720$	220,818	902	93	8	137	9.56	49	0.029
dive	$297 \times 1008 \times 566$	29,484	2,512	155	26	158	2.81	235	0.025

TABLE 2
Performance statistics for video composition results.

composition results. In fact, an added value of our approach is that we are able to give an approximate evaluation of the composition quality by checking the consistency in different steps. For example, using hybrid blending, we can calculate the composition cost, and after source object positioning, the motion consistency could be easily derived. The accumulated score can suggest if the two video sequences can, in fact, be composed successfully.

8 Conclusion

We presented a novel video blending approach that tackles key challenges in video composition: complex object blending (smoke, water, dust) and motion differences. Using a user provided *blending trimaps* of the source video, we create consistent blending boundaries over time. Then, we mix the gradients of the source and target videos inside the blending region. Our blending is based on an efficient implementation of mean-value coordinates interpolation instead of the traditional Poisson methods. We also provide a user interface to position source objects and refine the results. All these enable us to efficiently deal with complex video sequences beyond the capability of current solutions.

Our current implementation cannot generate the global optimized video composition result in realtime, a possible solution is to use KD-tree to accelerate the boundary optimization and gradient mixing. It is also interesting to extend the method to more challenging results. For example, source objects that are moving towards or away from the camera rather than primarily on a plane perpendicular to the camera. This would require more sophisticated user control to be developed. To more faithfully recover the appearance of the source objects in the target video, better illumination estimation could also provide guidance for gradient domain blending. Lastly, our user interface is still limited in versatility and accessibility, and it could be improved by modern video editing techniques like Chen et al. 2011 [28].

References

[1] Y.-Y. Chuang, A. Agarwala, B. Curless, D. H. Salesin, and R. Szeliski, "Video matting of complex scenes," *ACM Transactions on Graphics*, vol. 21, no. 3, pp. 243–248, July 2002, sepcial Issue of the SIGGRAPH 2002 Proceedings.

[2] Y. Li, J. Sun, and H.-Y. Shum, "Video object cut and paste," *ACM Trans. Graph.*, vol. 24, no. 3, pp. 595–600, 2005.

[3] J. Wang, P. Bhat, R. A. Colburn, M. Agrawala, and M. F. Cohen, "Interactive video cutout," in *SIGGRAPH '05: ACM SIGGRAPH 2005 Papers*. New York, NY, USA: ACM, 2005, pp. 585–594.

[4] X. Bai, J. Wang, D. Simons, and G. Sapiro, "Video snapcut: robust video object cutout using localized classifiers," *ACM Trans. Graph.*, vol. 28, no. 3, pp. 1–11, 2009.

[5] S.-K. Yeung, C.-K. Tang, M. S. Brown, and S. B. Kang, "Matting and compositing of transparent and refractive objects," *ACM Trans. Graph.*, vol. 30, pp. 2:1–2:13, February 2011. [Online]. Available: http://doi.acm.org/10.1145/1899404.1899406

[6] Z. Farbman, G. Hoffer, Y. Lipman, D. Cohen-Or, and D. Lischinski, "Coordinates for instant image cloning," *ACM Trans. on Graph.*, vol. 28, no. 3, p. 67, Aug. 2009.

[7] M. Grundmann, V. Kwatra, M. Han, and I. Essa, "Efficient hierarchical graph based video segmentation," *IEEE CVPR*, 2010.

[8] M. Werlberger, T. Pock, and H. Bischof, "Motion estimation with non-local total variation regularization," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, San Francisco, CA, USA, June 2010.

[9] G. Zhang, W. Hua, X. Qin, Y. Shao, and H. Bao, "Video stabilization based on a 3d perspective camera model," *Vis. Comput.*, vol. 25, pp. 997–1008, October 2009. [Online]. Available: http://portal.acm.org/citation.cfm?id=1668888.1668891

[10] C. Rother, V. Kolmogorov, and A. Blake, ""grabcut": interactive foreground extraction using iterated graph cuts," *ACM Trans. on Graph.*, vol. 23, no. 3, pp. 309–314, 2004.

[11] Y. Boykov and M.-P. Jolly, "Interactive graph cuts for optimal boundary amp; region segmentation of objects in n-d images," in *Eighth IEEE International Conference on Computer Vision*, vol. 1, 2001, pp. 105 –112 vol.1.

[12] T. Chen, M.-M. Cheng, P. Tan, A. Shamir, and S.-M. Hu, "Sketch2photo: internet image montage," *ACM Trans. Graph*, vol. 28, no. 5, pp. 124: 1–10, 2009.

[13] P. Bhat, C. L. Zitnick, M. Cohen, and B. Curless, "Gradientshop: A gradient-domain optimization framework for image and video filtering," *ACM Trans. Graph.*, vol. 29, no. 2, pp. 1–14, 2010.

[14] C. J. Armstrong, B. L. Price, and W. A. Barrett, "Interactive segmentation of image volumes with live surface," *Comput. Graph.*, vol. 31, no. 2, pp. 212–229, 2007.

[15] P. J. Burt and E. H. Adelson, "A multiresolution spline with application to image mosaics," *ACM Trans. Graph.*, vol. 2, pp. 217–236, October 1983. [Online]. Available: http://doi.acm.org/10.1145/245.247

[16] P. Pérez, M. Gangnet, and A. Blake, "Poisson image editing," in *ACM SIGGRAPH 2003 Papers*, ser. SIGGRAPH '03. New York, NY, USA: ACM, 2003, pp. 313–318. [Online]. Available: http://doi.acm.org/10.1145/1201775.882269

[17] J. Jia, J. Sun, C.-K. Tang, and H.-Y. Shum, "Drag-and-drop pasting," *ACM Transactions on Graphics (SIGGRAPH)*, 2006.

[18] P. Bhat, B. Curless, M. Cohen, and L. Zitnick, "Fourier analysis of the 2d screened poisson equation for gradient domain problems," in *ECCV08*, 2008.

[19] M. W. Tao, M. K. Johnson, and S. Paris, “Error-tolerant image compositing,” in *Proceedings of the 11th European conference on Computer vision: Part I*, ser. ECCV’10. Berlin, Heidelberg: Springer-Verlag, 2010, pp. 31–44. [Online]. Available: http://portal.acm.org/citation.cfm?id=1886063.1886067

[20] H. Wang, R. Raskar, and N. Ahuja, “Seamless video editing,” in *ICPR ’04: Proceedings of the Pattern Recognition, 17th International Conference on (ICPR’04) Volume 3*. Washington, DC, USA: IEEE Computer Society, 2004, pp. 858–861.

[21] Z.-F. Xie, Y. Shen, L.-Z. Ma, and Z.-H. Chen, “Seamless video composition using optimized mean-value cloning,” *Vis. Comput.*, vol. 26, no. 6-8, pp. 1123–1134, 2010.

[22] J.-F. Lalonde, D. Hoiem, A. A. Efros, C. Rother, J. Winn, and A. Criminisi, “Photo clip art,” *ACM Trans. on Graph.*, vol. 26, no. 3, pp. 3:1–10, 2007.

[23] J.-F. Lalonde, A. A. Efros, and S. G. Narasimhan, “Estimating natural illumination from a single outdoor image,” in *Proc. of ICCV*, 2009.

[24] K. Sunkavalli, M. K. Johnson, W. Matusik, and H. Pfister, “Multi-scale image harmonization,” *ACM Transactions on Graphics (Proc. ACM SIGGRAPH)*, vol. 29, no. 4, pp. 125:1–125:10, 2010.

[25] A. Bousseau, F. Neyret, J. Thollot, and D. Salesin, “Video watercolorization using bidirectional texture advection,” *ACM Trans. Graph.*, vol. 26, July 2007. [Online]. Available: http://doi.acm.org/10.1145/1276377.1276507

[26] G. Zhang, X. Qin, W. Hua, T.-T. Wong, P.-A. Heng, and H. Bao, “Robust metric reconstruction from challenging video sequences,” in *IEEE Conference on Computer Vision and Pattern Recognition.*, 2007, pp. 1 –8.

[27] F. Liu, M. Gleicher, J. Wang, H. Jin, and A. Agarwala, “Subspace video stabilization,” *ACM Trans. Graph.*, vol. 30, pp. 4:1–4:10, February 2011. [Online]. Available: http://doi.acm.org/10.1145/1899404.1899408

[28] J. Chen, S. Paris, J. Wang, W. Matusik, M. Cohen, and F. Durand, “The video mesh: A data structure for image-based three-dimensional video editing,” in *2011 IEEE International Conference on Computational Photography (ICCP)*, april 2011, pp. 1 –8.