Motion-Aware Gradient Domain Video Composition

Tao Chen, Jun-Yan Zhu, Ariel Shamir, and Shi-Min Hu*

Abstract—For images, 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 video faces new challenges due to the added temporal dimension. In video, the human eye is sensitive to small changes in blending boundaries across frames, and slight differences in motions of the source patch and target video. We present a novel video blending approach that tackles these problems by merging the gradient of source and target videos and optimizing a consistent blending boundary based on a user provided *blending trimap* for the source video. Our approach extends mean-value coordinates interpolation to support hybrid blending with a 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 capabilities of alpha blending.

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

1 Introduction

Video composition is very useful in the film, television and entertainment industries. Such composition takes a frame sequence from a source video, usually extracting a foreground object (here called a 'patch'), and pastes it onto a frame sequence in a target video. This process involves two challenges: extracting the patch from the video, and composing it with the target frames. In both cases a user must typically be involved in the process, both to choose a desirable source patch, and a composition location in space and time. The main focus of video composition techniques is to lower the amount of manual effort needed in this process.

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

For images, gradient domain blending solves such problems by transferring the gradient of the source image patch to the target image while maintaining a seamless blending boundary and correcting illumination differences. This can be done by solving a Poisson equation, or by interpolation using mean-value coordinates (MVC), the latter providing interactive rates of composition [6].

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

Our motion-aware gradient domain video blending technique addresses the above issues. The key idea is to use soft, rather than hard, boundaries between the foreground and background for composition inside the blending region. In each frame, the real boundary of the source patch lies 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, e.g. if 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 consistency and gradient saliency of the source and target videos. The mixing gradient disperses motion differences across the blending boundary between the source and target videos in-

This work was supported by the National Basic Research Project of China (Project Number 2011CB302205), the Natural Science Foundation of China (Project Number 6113300861103079), National Significant Science and Technology Program (Project Number 2012ZX01039001- 003), National High Technology Research and Development Program of China (Project Number 2013AA013903) and the Israel Science Foundation (grant no. 324/11).

T. Chen, J.-Y. Zhu and S.-M. Hu are with TNList, Department of Computer Science, Tsinghua University, Beijing 100084, China.
A. Shamir is with The Interdisciplinary Center, Israel.
∗ *The corresponding author.*

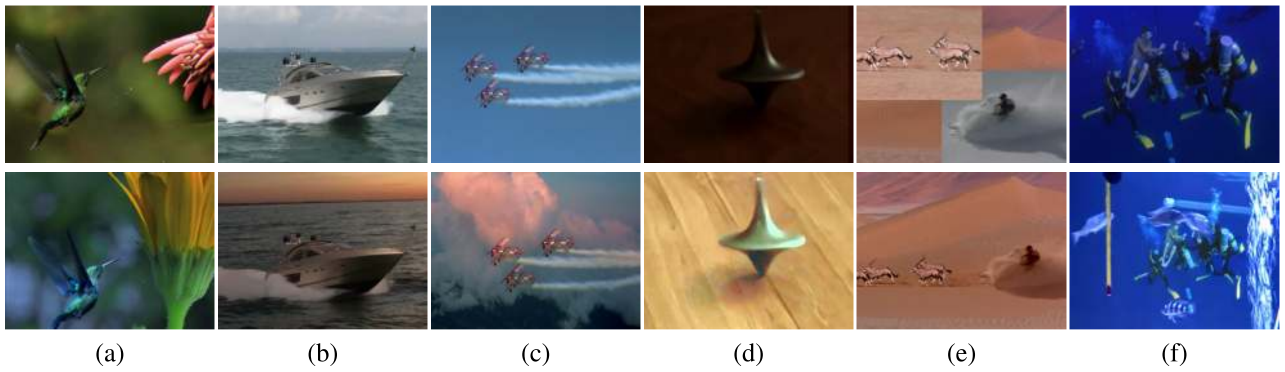


Fig. 1. Challenging problems for existing video composition schemes involving uncertain object boundaries due to shadows, smoke and dust, motion blur etc., and complex motions of both camera and objects. Our robust gradient domain video composition method can cope with such scenarios (top: source video, bottom: composed video). Also see the supplementary materials.

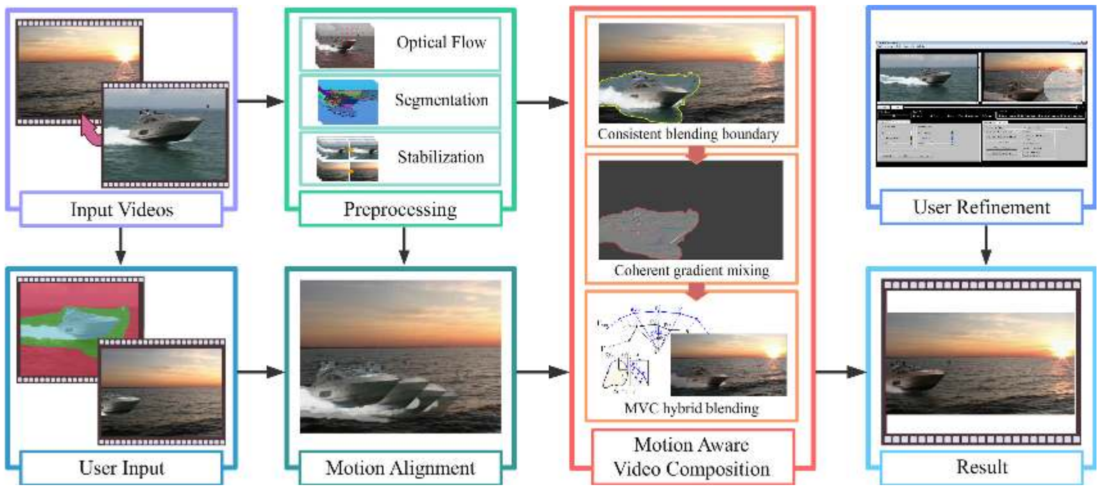


Fig. 2. Pipeline for motion-aware video composition. Given input source and target video sequences, preprocessing steps include calculating optical flow, video segmentation and stabilization. The user then inputs blending trimaps on the source video, and positions the source object on the first frame. An output video is produced using motion-aware gradient composition. The user can subsequently interactively refine object position in chosen frames.

to the whole blending region, producing smoother and more coherent blending results with reduced motion artifacts (see Figure 1 and examples in the supplementary materials).

Our solution builds on several previous areas including video segmentation [7], motion estimation [8], video stabilization [9], grab-cut [10], [11]. We extend MVC cloning [6] with hybrid blending boundary conditions in a similar way to [12] and use a mixing-gradient field [13]. Our contributions are: (i) a fully interactive solution for video composition, (ii) a novel video blending approach that tackles consistency in both motion and appearance by merging the gradients of source and target video and optimizing the dynamic blending boundary, (iii) extension of mean-value coordinates interpolation to support hybrid blending with a dynamic boundary, while maintaining interactive performance, (iv) we provide a user interface and source object positioning method that can efficiently deal with complex video sequences beyond the capability of alpha blending.

2 Related Work

Video matting and composition is a well studied problem in computer graphics and computer vision. In 2002, Chuang et al. [1] used bi-direction optical flow to propagate matting trimaps across video frames. In Li et al. [2], Wang et al. [3] and Armstrong et al. [14], graph-cut based video segmentation is used for segmentation. In 2009, Bai et al. [4] presented a video cutout system that achieved better segmentation by use of a set of local classifiers. Tong et al. [15] designed a novel interface which can efficiently select and cut out video objects as a video plays. Tang et al. [16] proposed a novel video matting method based on opacity propagation. All these video matting approaches rely on having a well-defined object boundary in the source video. Gradient domain image composition avoids solving the difficult matting problem for fuzzy boundaries, and can also deal with large illumination differences between source and target images. Burt and Adelson [17] used a multiresolution spline technique to blend two images; while this method is concise and fast, data incorporation from distant source and destination pixels may generate undesirable results. Pérez et al. [18] improved upon this idea by solving a Poisson equation in the blending region. Jia et al. [19] eliminated blending artifacts by optimizing the blending boundary and use of alpha matting. Chen et al. [12] further improved composition quality by using mixed boundary conditions when solving the Poisson equation. Zhang et al. [20] proposed an environment-sensitive image blending method based on image patches surrounding the source object in the target image.

Bhat et al. [21] efficiently solved the Poisson equation using Fourier analysis. To achieve real-time performance, Farbman et al. [6] used mean-value coordinates interpolation to approximately solve the Laplacian equation used for gradient domain composition. Tao et al. [22] presented an error-tolerant gradient-domain image compositing technique. By defining boundary gradients and applying a weighted integration scheme to the gradient field, it solved the color bleeding problem and improved composition quality. However, motion blur, camera shake and ill-defined object boundaries due to scene complexity or video quality still limit the applicability of these works to video.

Our work extends the range of video composition possible by providing motion-aware blending. Several works have attempted to extend the gradient domain scheme to video composition. Wang et al. [23] proposed a 3D video integration algorithm which solves a 3D Poisson equation. Xie et al. 2010 [24] adapted mean-value coordinates to efficiently perform the video composition. However, these methods do not take spatio-temporal and motion inconsistencies of the source and target gradients into account, lowering composition quality and narrowing 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 result in over-blending effects in which pasted objects are too dark or over-saturated. Lalonde et al. [25] used illumination clues to choose illumination-consistent images for their Photo Clip Art. Lalonde et al. [26] further estimated the illumination from a single outdoor image, which can help prevent over-blending. In GradientShop, Bhat et al. [13] used a general optimization framework for gradient domain image and video processing, which inspired our work. Gradient domain composition suffers from image noise, has been addressed lately by image harmonization techniques proposed in Sunkavalli et al. [27]. The motion patches described in Zhang et al. [28] inspired our gradient mixing approach.

3 Overview

Figure 2 shows the general pipeline of our method. In the user input step, the user provides blending trimaps on the source video: blue and red regions cover the definite foreground and definite background respectively. The green region in between is the blending region where a soft blending boundary is defined; it usually covers uncertain area around the foreground source object including shadows, dust, smoke or waves (see e.g. Figure 3 and the trimaps shown in Figure 10(a,c)). Both the hybrid blending boundary and the mixing-gradient field are determined inside this blending region. The boundary between the red and green regions, denoted by Γ_{out}, is obtained as a user-drawn closed loop on the first frame. The inner boundary Γ_{in}, between the green and blue regions, is generated by applying a refinement step of grab-cut [11] to another user-drawn closed loop roughly around the object boundary in the first frame. Trimaps in subsequent frames are generated by propagating the user inputs from the first frame (see 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 are aligned. The position and scale of the source object is automatically calculated in subsequent frames using feature point registration and optical flow estimation. To generate the desired motion path (i.e. the object's moving path across video frames), or to fix inaccurate automatic alignment, the user can drag and resize the source object in selected *motion keyframes*. Our method allows *interactive* composition results to be shown to the as the alignment is adjusted; details of this user interface are presented in Section 6.

After the source objects have been positioned, the key challenge for composition is to overcome large appearance and motion differences between the source video patch and the target video sequence. We first find a blending boundary in each frame which minimizes the spatio-temporal and motion differences along the boundary. Then, we construct a mixing-gradient field inside the composition region based on optical flow, gradient saliency and continuity. 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, an approximate calculation is used for mean-value coordinates, to achieve interactive real-time updates when a dynamic blending boundary is used; we describe this method for a single frame in Section 4.

4 Real-time Single Frame Hybrid Blending

Done directly, Poisson image blending involves solving a large linear system which is too time-consuming for video blending. Farbman et al. [6] introduced an alternative,

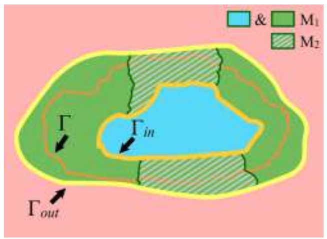


Fig. 3. Definitions of regions and boundaries.

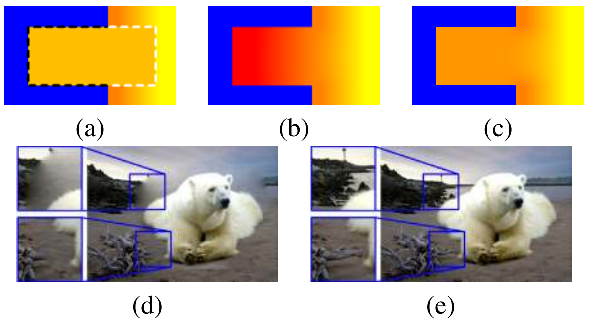


Fig. 4. Comparison between MVC cloning [6] and our MVC hybrid blending. (a) Blending of the rectangular region inside the dashed line and the background. The black dashed line is an inconsistent boundary that should be avoided for blending. (b) MVC cloning with *selective boundary suppression* at this boundary generates an undesirable red blending result due to extrapolation of MVC. (c) MVC *hybrid blending* generates a plausible blending result by setting Neumann boundary conditions at this boundary. MVC hybrid blending's advantages of using (i) mixed boundary conditions are further shown in (e, top), and (ii) an optimized blending boundary in (e, bottom); compare these to to MVC cloning's results in (d).

coordinate-based, approach to perform the cloning process of normal Poisson blending with Dirichlet boundary conditions (which specifies the values a solution needs to take on the boundary of the domain). The interpolated value at each interior pixel is given by a weighted combination of values along the boundary based on mean-value coordinates (MVC). Use of MVC is advantageous in terms of speed, ease of implementation, memory footprint, and parallelization. However, since this solution uses a fixed blending boundary with the same boundary conditions as Poisson blending, it also carries the same limitations, especially in the presence of large texture or color differences.

Farbman et al. [6] use *selective boundary suppression*, which removes some boundary points for interpolation, to reduce smudging artifacts. This fails when the removed boundary points are extruded, as shown in Figure 4(a)–(c). Jia et al. [19] optimize the blending boundary to achieve minimal color variation within the source object, while Lalonde et al. further improved this to deal with large texture or color differences [25], [12]. Here, we improve the MVC method by searching for an optimized boundary and using mixed boundary conditions similar to the hybrid blending approach suggested in Chen et al. [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: the pixels where the texture and color of the source and target are consistent are classified as M_1, and all other pixels are classified as M_2. Texture and color consistencies are measured by the difference of Gabor feature vectors and the difference of the UV color components respectively. The object pixels that are surrounded by the inner boundary Γ_{in} are also classified into M_1. Conventional Poisson blending can be safely applied in the M_1 region, but it can cause artifacts (e.g. smudging and discoloration) within M_2. For M_2 pixels, 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 the target image f^t to the source patch. This technique first solves the following Poisson equation to compute an intermediate blending result f':

$$\triangle f' = \operatorname{div}(\mathbf{G}) \text{ over } \Omega. \quad (1)$$

This is similar to conventional Poisson blending, except that it creates a mixed guidance vector field $\mathbf{G}$ over Ω:

$$\mathbf{G}|_{M_1} = \nabla f^s \quad \text{and} \quad \mathbf{G}|_{M_2} = \nabla f^s_f, \quad (2)$$

with mixed boundary conditions on Γ: Dirichlet boundary conditions on Γ_1, and Neumann boundary conditions (specifying values of the derivative of the solution on the boundary of the domain) on Γ_2:

$$f'|_{\Gamma_1} = f^t \quad \text{and} \quad \nabla f'|_{\Gamma_2} = \nabla f^s_f, \text{ where } \Gamma_i = \Gamma \cap M_i, i = 1,2 \quad (3)$$

For pixels in M_2, f' is combined with f^t using alpha blending to obtain the final result: $f = \alpha f' + (1-\alpha) f^t$, where α is obtained by alpha matting.

In our work, we do not solve the Poisson equation but instead use mean-value coordinates as mentioned in Farbman et al. [6] for efficiency. To this end, we must address two challenges. First, the position of the blending boundary Γ in hybrid blending is optimized according to pixel differences (mismatches). As the source and target alignment changes, Γ also changes, and hence MVC weights cannot pre-computed. This reduces the computational savings provided by MVC . Secondly, in hybrid blending, the boundary points on Γ_2 have Neumann boundary conditions, and therefore cannot be used directly for membrane interpolation in MVC. We solve the first challenge by approximatng MVC coordinates, and the second by first determining the boundary values to be interpolated.

4.1 MVC approximation

In Farbman et al. [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 coordinate computation, and interpolation. If the blending boundary is fixed, the first two steps can be pre-computed. Then, multiple cloning positions of the source patch can be considered efficiently. However, using a fixed, user-drawn boundary is insufficient for seamless blending in our case. Dynamic blending boundaries as suggested in [19] aim to minimize the difference of color mismatches along the boundary, and so are optimized for different blending positions. To deal with such a changing boundary we use a new method to approximate mean-value coordinates.

The most time-consuming part of MVC computation is calculating the angles from the inner points $\mathbf{x} \in \Omega$ to the

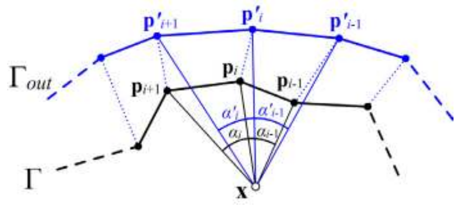


Fig. 5. Angle approximation for mean-value coordinates using a fixed enclosed boundary Γ_{out} (blue).

points on the blending boundary $\mathbf{p}_i, \mathbf{p}_{i+1} \in \Gamma$. The tangents 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 avoid computing these angles every time the boundary Γ is changed, we approximate these angles using pre-computed values close to the actual 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 blending boundary optimizations are applied within this boundary. We construct an adaptive triangular mesh inside Γ_{out} using a similar method to that in Farbman et al. [6], but using a constant vertex density in the region between Γ_{in} and Γ_{out}, identical to 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). The points $\mathbf{p'}$ are sampled on Γ_{out} by hierarchical boundary sampling as described in Farbman et al. [6]. The semi-tangents of these angles are saved for all vertices.

Next, each time the position of a blending boundary Γ is optimized for a given source patch position, we compute corresponding points on Γ for each sampled point on Γ_{out}. First, we uniformly sample Γ with twice the number of points $\mathbf{p'}$ on Γ_{out}. Next, we link each $\mathbf{p'}$ point on Γ_{out} to its nearest sample point on Γ, giving one-to-one correspondences for all pairs. If multiple $\mathbf{p'}$ points are linked to the same point on Γ, we use only the one with the shortest distance. For a point $\mathbf{p'}$ whose correspondence cannot be decided by the above procedure, we find its closest left and right neighboring points $\mathbf{p'}_l, \mathbf{p'}_r$ that already have corresponding points $\mathbf{p}_l, \mathbf{p}_r$ on Γ. We find the interval ratio (by counting pixels along the boundary) of the point from $\mathbf{p'}_l$ and $\mathbf{p'}_r$ and match it to a point on Γ_{out} with the same interval ratio between $\mathbf{p}_l$ and $\mathbf{p}_r$. Now, the jth component (before normalization) of our approximated MVC for a vertex $\mathbf{x}$ is given by:

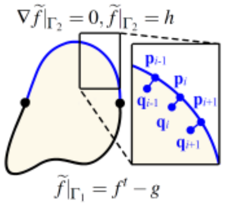
$$w^j = \frac{\tan(\alpha'_{j-1}/2) + \tan(\alpha'_j/2)}{||\mathbf{p}_j - \mathbf{x}||} \tag{4}$$

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, approximate interpolation is visually indistinguishable from MVC interpolation.

4.2 Boundary value solution

MVC are used to approximate the solution of the Laplacian equation in [6]. Thus, a prerequisite for using MVC is to convert the Poisson equation in hybrid blending to a Laplacian equation. The mixed boundary conditions must also be modified appropriately. To obtain the 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 $\mathbf{G}$ (see Section 5.2). By substituting f' from Equation (1), the Laplacian form of hybrid blending becomes:

$$\triangle \widetilde{f} = 0 \text{ over } \Omega, \text{with } \widetilde{f}|_{\Gamma_1} = f^t - g \quad \text{and} \quad \nabla \widetilde{f}|_{\Gamma_2} = 0. \tag{5}$$


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

If the hybrid blending boundary contains parts with Neumann boundary conditions, it cannot be directly approximated by MVC interpolation, as some boundary values for interpolation remain unknown. Thus, we find these values before applying approximate MVC interpolation. At 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 coordinate-based interpolation, we can form a small linear system for the pixel values around the Neumann boundary and the known gradient values $\nabla \widetilde{f}|_{\Gamma_2} = 0$.

$$h(\mathbf{p}_i) - \sum_{\mathbf{p}_j \in \Gamma_1} w_i^j (f^t(\mathbf{p}_j) - g(\mathbf{p}_j)) - \sum_{\mathbf{p}_k \in \Gamma_2} w_i^k (h(\mathbf{p}_k)) = 0, \tag{6}$$

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 approximate MVC interpolation as above to these boundary values $h(\mathbf{p}_i)$.

Figure 4(b)–(e) compares Farbman et al.'s MVC cloning [6] and our MVC hybrid blending. In Figure 4(d), since the manually specified blending boundary intersects with the sky and wood, the results shows serious texture smudging, which is avoided by MVC hybrid blending and blending boundary optimization, as shown in Figure 4(e). In practice, we have found that the difference between hybrid blending based on the Poisson equation and blending based on MVC interpolation are almost indistinguishable. However, after pre-computation, MVC interpolation is about two orders of magnitudes faster. This key technique enables efficient gradient domain blending for video and interactive editing. Since we pre-compute the tangents instead of the entire w^j, our method must compute one more addition and division operation for each vertex. Combining the additional boundary optimization step and boundary value solving step, on average our MVC hybrid blending is about three times slower than conventional MVC.

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, such as the regions inside the blue boxes in Figure 6. However, if

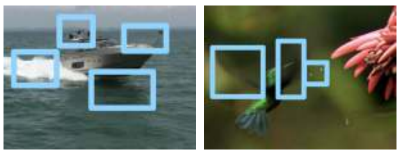


Fig. 6. Challenges for video composition. Motion blur, uncertain boundaries, shadows and reflections occur inside the blue rectangles.

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

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

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

Temporal fluctuations in color mismatch. Differences in color between the source and target across the blending boundary may also change between frames. Color mismatch is used as a boundary condition in our blending method and therefore affects color inside the blending region as well. Changes in this mismatch over consecutive frames can cause color flickering in the entire blending region.

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

5.1 Optimizing the blending boundary

First we minimize the boundary color mismatch between the source patch and target surroundings across the blending boundary. This effectively suppresses variation in appearance of the source object in the blended video. We extend the boundary optimization method of hybrid blending (Section 4) from images to video, taking temporal coherence into account. We apply video segmentation [7] to both source and target videos, using a hierarchical graph-based algorithm to over-segment a volumetric video graph into space-time regions. These regions, or *super-voxels*, are grouped by appearance to ensure color and texture consistency. We also apply the motion estimation method described in [8] to compute optical flow for both source and target videos. Based on non-local total variation regularization, motion estimation integrates the low level image segmentation process so that it can tackle poorly textured regions, occlusions and small scale image structures. Then, we calculate the texture and color differences for the corresponding super-voxels in the two videos in the soft boundaries to classify them as either belonging to M_1 or M_2 as described in Section 4. The blending boundaries of M_2 are generated by coherent matting [4], which adds a temporal alpha prior into the alpha matting optimization process to achieve temporally coherent alpha matting. Therefore, we only need to optimize the position of the blending boundary in M_1.

Boundary optimization can be effectively performed using dynamic programming [19]. Extending this to video must not only preserve the preceding function, but also minimize the artifacts caused by the above three issues. While directly optimizing the boundary on the 3D volume can solve the position mismatch problem, it cannot resolve motion differences or color mismatches. Another drawback of optimizing over 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 for each frame. As discussed in [29], [4], [13], a better solution is to optimize the consistency of motion-compensated neighbors instead of direct neighbors. Bhat et al. [13] show that this can be done using a moderately (50% to 60%) accurate optical flow, and by taking confidence values from the motion vectors into account. We follow a similar approach.

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

$$E(\Gamma,k) = \sum_{\mathbf{p}\in\Gamma} \frac{1}{D_{\mathbf{p}}}\{(k_{\mathbf{p}}-k)^2 + (k_{\mathbf{p}}-\overline{kt}_{\mathbf{p}})^2 + ||\mathbf{v}^t_{\mathbf{p}}-\mathbf{v}^s_{\mathbf{p}}||^2\} \qquad (7)$$

There are three terms for each boundary point $\mathbf{p}$. In the first term, $k_{\mathbf{p}}$ is the color mismatch of the target and source at $\mathbf{p}$, $k_{\mathbf{p}} = (f^t_{\mathbf{p}} - f^s_{\mathbf{p}})$, and k is the average mismatch for the current frame. This term minimizes the color mismatch on a single frame, and is similar to the one used in Jia et al. [19]. In the second term, $\overline{kt}_{\mathbf{p}}$ is the average mismatch between $\mathbf{p}$ and the nearest boundary points on 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 vectors from $\mathbf{p}$ to the corresponding point in the next frames in 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, and 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 mismatches, and we rely on gradient mixing in the next subsection to remove popping artifacts caused by position mismatch.

We iteratively minimize the cost function in Equation (7).

Initially, k is set to the average mismatch on the user-provided outer boundaries Γ_{out}, $D_{\mathbf{p}}$ is set to 1, and the second term $(k_{\mathbf{p}} - \overline{kt}_{\mathbf{p}})^2$ is set to 0. On each iteration, a new boundary is calculated using dynamic programming, where the cost for each pixel is set to the sum of the three terms in Equation (7). 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 $\mathbf{p}$ in an adjacent frame. We consider the nearest distance from $\mathbf{p}$ to its projections in each adjacent frame, and set $D_{\mathbf{p}}$ to the average of the two nearest distances to the two adjacent frames. Accordingly, $\overline{kt}_{\mathbf{p}}$ is set to the average of the color mismatches between $\mathbf{p}$ and the two nearest projected pixels on those two frames. On each iteration we first find the new boundaries and then update k, $D_{\mathbf{p}}$ and $\overline{kt}_{\mathbf{p}}$. Iteration terminates when the boundaries converge or a maximum number of iterations has been reached (we use 20).

5.2 Gradient mixing

To deal with the remaining artifacts caused by position mismatch and motion differences, we mix the source and target gradients coherently in the soft boundary region. Mixing of gradients for image cloning was proposed by Pérez et al. [18] to preserve salient content in both source and target images. We extend this approach to video in a spatio-temporally coherent manner. Gradient mixing is applied to 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 motion inconsistency artifacts. If $\mathbf{G}_i^s$ and $\mathbf{G}_i^t$ are 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:

$$\mathbf{G}_i = \alpha_i \cdot (\mathbf{G}_i^s) + [1 - \alpha_i] \cdot (\mathbf{G}_i^t), \tag{8}$$

However, our approach is based on the observation that different situations may need different mixing rules to selectively preserve content from either 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, preservation of the gradient field should be temporally consistent in motion-compensated neighbors. We use user-defined mixing parameters that address the specific composition problem (see Table 1).

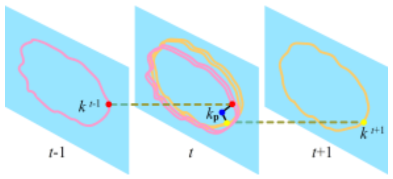


Fig. 7. Pixels used to calculate the temporal cost in blending boundary optimization.

The use of MVC cloning demands that the mixed gradient field should be conservative. We reconstruct a conservative mixing gradient field as a guidance vector field based on (i) the gradient magnitude of source and target, and (ii) 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 with values set by the user; these values can differ for source and the target filtering. We also define $\mathbf{a}_i = (\alpha_i^x, \alpha_i^y)$ as the mixing weighting factor. For a conservative gradient field, α_i^x and α_i^y can be different. At the i-th pixel of this region, the reconstructed gradient value is given by a linear combination of the filtered source and target gradients:

$$\mathbf{G}_i = \mathbf{a}_i \circ \mathscr{F}_s(\mathbf{G}^s)_i + [(1,1) - \mathbf{a}_i] \circ \mathscr{F}_t(\mathbf{G}^t)_i, \tag{9}$$

where $\circ$ is the Hadamard product (or entrywise product) of two vectors. 2D gradient vector $\mathscr{F}_s(\mathbf{G}^s)_i$ and $\mathscr{F}_t(\mathbf{G}^t)_i$ are the i-th elements of filtered source and target gradient field matrices. 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 can be toggled on or off for both the source and target gradient fields. 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 each video. These toggles affect the filtering function $\mathscr{F}$ of the gradient field $\mathbf{G}$ (either source or target) as follows:

$$\mathscr{F}(\mathbf{G}) = \begin{cases} \mathbf{G} - K * \mathbf{G} & \text{if only "salient" is on,} \\ K * \mathbf{G} & \text{if only "base" is on,} \\ \mathbf{G} & \text{if both are on,} \\ 0 & \text{if both are off} \end{cases} \tag{10}$$

where $K = N(\mathbf{p}_i; \sigma^2)$ 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" toggle 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 (typically, unpredictable motion). The user can toggle this setting to affect the initial mixing weight α_i^{init}:

$$\alpha_i^{init} = \frac{||\mathbf{G}_i^s||^2}{||\mathbf{G}_i^s||^2 + ||\mathbf{G}_i^t||^2} + (d^s \mathscr{D}_i^s - d^t \mathscr{D}_i^t), \tag{11}$$

where $||\mathbf{G}_i^s||$ and $||\mathbf{G}_i^t||$ are the gradient magnitudes of the source and target gradient at the i-th pixel respectively. 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, 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 :

$$\mathscr{D}_i = \begin{cases} \frac{||(\mathbf{DoG}*\mathbf{V})_i||^2}{\psi} & \text{if} \quad wv_i > 0.5, \\ 1 - wv_i & \text{otherwise}, \end{cases} \tag{12}$$

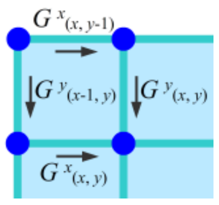
where $\mathbf{DoG} * \mathbf{V}$ is the convolution of $\mathbf{DoG}$ (difference of Gaussian) and the optical flow vector field $\mathbf{V}$. The two bandwidths of the $\mathbf{DoG}$ are 12 and 3. The scale factor ψ is set to 50. wv_i is the confidence in optical flow at the i-th pixel. To take the 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 motions that are distinct from their backgrounds inside the blending region. Note that this region usually contains scene objects that move together or are affected by the foreground object, such as motion blur, shadow, wave, smoke, dust, etc. User-defined mixing parameter values for the examples in this paper are shown in Table 1.

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

$$C(\mathbf{a}_i) = ||\mathbf{a}_i - \mathbf{a}_i^{init}||^2 + \\ + w_1 \sum_{j \in \mathbf{N}_{tem}(i)} ||\mathbf{a}_i - \mathbf{a}_j||^2 + w_2 \sum_{k \in \mathbf{N}_{spa}(i)} ||\mathbf{a}_i - \mathbf{a}_k||^2 \tag{13}$$

The first term requires that the mixing weights are close to the initial mixing weight $\mathbf{a}_i^{init}$. The second and third terms penalize temporal and spatial variations respectively. $\mathbf{N}_{tem}(i)$ 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 in 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 the i-th pixel on its frame is (x,y) (see right), any $\mathbf{G}_i = (G^x_{(x,y)}, G^y_{(x,y)})$ must satisfy the following equation to ensure conservative mixed gradients:



$$G^x_{(x,y-1)} + G^y_{(x,y)} = G^y_{(x-1,y)} + G^x_{(x,y)} \tag{14}$$

Substituting Equation (9) into Equation (14) gives 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 $\mathbf{a}_i = (\alpha_i^x, \alpha_i^y)$. The cost function in Equation (13) is minimized with this set of linear constraints, ensuring that the mixing of the gradients is still conservative. This 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, after five iterations, even without convergence, the resulting mixing weights already ensure smoothness of gradient mixing. We replace the source gradient field in the soft boundary region by the mixing-gradient field and perform blending using the composition technique discussed in Section 4.

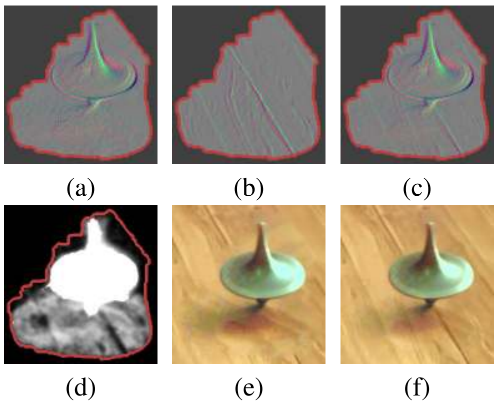


Fig. 8. Gradient mixing: the red line is the blending boundary Γ. (a) source gradient. (b) target gradient. (c) mixing-gradient. (d) gradient alpha map, illustrated by the norm of alpha vector. (e) composition result without mixing. Note the evident seam between the textures in this case. On videos,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

Figure 8 shows gradient fields before and after mixing, the gradient alpha map, and 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 motion artifacts (please see the supplementary materials).

5.3 Coherent MVC cloning

As described in Section 4, the final composition for 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 in the first frame, the boundaries are propagated to subsequent frames. The inner boundaries Γ_{in} are propagated using Bai et al.'s method [4], which relies on overlapping local classifiers and local window propagation. The outer boundaries Γ_{out} are sampled and corresponding points found on Γ_{in} as explained in Section 4. Γ_{out} are propagated to subsequent frames using 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 in a specific frame, determining propagation to later frames. Next, we project Γ_{out}^i to Γ_{out}^{i+1} by a similarity transformation. If the union of projected boundary and Γ_{out}^{i+1} does not deviate too much (10%) from both boundaries, we use the union as the new Γ_{out} for both frames. This way, Γ_{out} is updated and clustered into several groups over time. Within each group, the only differences in Γ_{out} are a similarity transformation; the triangulation and angle tangents are 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, unlike the fixed membrane used in Farbman 2009 [6]. Therefore, instead of smoothing temporal neighbors as in their paper, we smooth motion-compensated neighbors and modify the smoothing weights of older frames accordingly. The smoothed vertex membrane value at frame i is calculated as follow:

$$\frac{1}{W}[m_i + 2^{-d}(\sum_{j=i-5}^{i-1}(i-j)^{-0.75}(||\mathbf{a}||m_j^s\prod_{k=j}^{i-1}wv_k^s + ||(1,1)-\mathbf{a}||m_j^t\prod_{k=j}^{i-1}wv_k^t)], \tag{15}$$

where m_i is the vertex membrane value before smoothing, and m_j^s and m_j^t 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_k^s and wv_k^t are the optical flow confidences for the source and target respectively at frame k. d is the normalized distance to boundary Γ. W is the sum of weights for all 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 can compose source objects with the background better, since it seamlessly transfers the surroundings (e.g. shadows) of the transplanted object. However, it can still appear unrealistic if we do not position the object correctly.

We provide an interactive positioning interface to extend the flexibility of our motion-aware composition framework. It has two main goals: to better align the motions 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 design combines automatic computation with user interaction to reduce user effort and allow fast update with interactive feedback while editing. Towards this end we constrain the possible user edits to specific points along the object's motion path, and also constrain the magnitude and orientation of possible velocity changes. These manual changes are then automatically propagated to other frames. This prevents the creation of unnatural motion artifacts, and allows efficient local updates. Combining interaction and computation provides an interactive-feedback system for easy motion refinement and editing.

6.1 Motion alignment

To remove camera shake, we first stabilize the source and target videos using ideas from Zhang et al. [9]. This method is based on a 3D perspective camera model, and formulates the stabilization problem as a quadratic cost function using smoothness and similarity constraints. Stabilization helps the subsequent video registration step. Zhang et al. [30] perform automatic metric reconstruction from long video sequences with varying focal length. We use their method to recover the camera parameters and remove the camera motion from both source and target videos. We remove the moving source object from the source video to prevent outliers. The user places the source object on the first frame, possibly with some rotation and scaling. The system calculates a motion path for subsequent frames using automatic alignment. Trying to neutralize camera motion for all types of complex video scenes is impractical, so we allow the user to correct the position of the object on certain frames manually.

6.2 Motion editing

After automatic alignment and user refinement, good video composition results can be achieved. However, users may still want to speed up, slow down, or change the trajectory of the source object on the target frame. Such effects cannot be achieved by simple composition with an aligned path: we support them via interactive motion editing.

Allowing arbitrary modifications to any point in the object's path will break the intrinsic property of the motion path, causing artifacts to occur. Moreover, each time the path is changed by the user, boundary optimization (Equation (7)) and gradient mixing (Equation (13)) must be re-computed. This becomes a bottleneck for interactive update performance. In our boat example, MVC hybrid blending takes only 1s, but these other computations take 6.6 seconds per 100 frames. Hence, to prevent unnatural motion artifacts, and for efficiency, we intelligently constrain the editing of the motion path both to very specific points in time and to preset amounts.

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*, 2D points on the motion path that indicate *motion keyframes*. We allow the user to edit only control points, and their modifications (including translation, rotation and scale) are propagated to other frames by linear interpolation. These points are optimal points for adjustment. The motion path is divided into several segments based on two criteria: (i) the motion in one segment can be approximated by a uniform motion in a straight line, and (ii) the motion properties and visual features should be stable at the control points.

Motion is quantified by speed, and visual features are measured by color mismatch across the blending boundary as mentioned in Section 5. These two criteria allow us to determine two properties: motion discontinuity at control points, and stability of the control points themselves. We greedily find n control points on the motion path by minimizing the following energy function:

$$E_n = \mu_1 \sum_{i=1}^{n} (k_{t_i} - \frac{k_{t_i-1} + k_{t_i+1}}{2})^2 + \mu_2 \sum_{i=1}^{n} \vec{\mathbf{v}}_{t_i}^{\,2} + \mu_3 \sum_{i=1}^{n-1} \sigma_i^2, \quad (16)$$

where t_i is the frame containing the i-th control point, k_{t_i} is the average color mismatch on the blending boundary in frame t_i, and $\vec{\mathbf{v}}_{t_i}$ is the object velocity in 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 control points at the ends of the segment. We set $\mu_1 = 0.2, \mu_2 = 0.2$ and $\mu_3 = 0.6$ in our experiments. Initially, the motion path only contains one segment, which means $n = 2$. The first two terms of Equation (16) are set to zero whereas the last term is large. Next, we greedily find new control points until Equation (16) stops decreasing.

Using the control points, we divide the motion into several segments of approximate uniform straight line motion. If the user edits the control point, these uniform motions are largely preserved. In contrast, if modification is applied to an internal segment point, it will create a discontinuity. In addition, we impose local modification constraints on the magnitude and orientation of velocity changes during editing. These restrictions bound the area where the user can change the path positions. When editing the k-th control point, assume θ_i, θ_i' are the old and new orientation of the i-th point's velocity. If v_i, v_i' are its old and new magnitudes, then the restriction is:

$$\forall i \in [n_{k-1}, n_{k+1}], \max(\theta_i' - \theta_i)^2 < T_\theta \text{ and } \max(v_i' - v_i)^2 < T_v \quad (17)$$

To maintain the perspective continuity of the composition, we set $T_\theta = 30^o$ and $T_v = 20$ as threshold values for all our examples. Figure 9 illustrates 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 (7) and Equation (13) for the whole sequence. Instead, we calculate the blending results on segments which were changed previously but are not being edited. If previous calculations have provided the boundary, mixing-gradient and MVC membrane values on the end frames of these segments, they are made hard constraints in the optimization. In summary, we interleave computation with user interaction to compute results for other, non-edited parts. In our experiments, editing one point requires 1-2 seconds, during which we can compute 20-35 frames based on local optimization. Because of motion stability of the control points, and because the temporal coherency restriction mainly utilizes information in consecutive frames, we can merge several passes of local optimization to approximate global optimization.

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

7 Experimental Results

We have 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 for this example is shown in Figure 10(a). Our blending faithfully preserves the rapid motion of the wing of the hummingbird and the waterdrops. Figure 1(b) pastes a boat on the sea surface at sunset. The original source and target sequences contain severe camera shaking, and the water flow directions differ. The waves, reflections, and flag with the pole pose challenges to previous methods, yet are faithfully preserved, and 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 complicated matting. The 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 chase scene which pastes a snowmobile into a desert scene, chasing some goats. Figure 1(f) pastes a group of people diving into a fish tank, producing an interesting montage. The swimming fish in the target video and the bubbles in the source video are

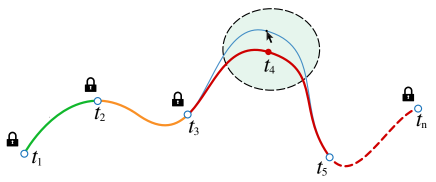


Fig. 9. Interactive motion editing. $t_1, t_2, ...t_n$ are control points on *motion keyframes*. Editing control points only affects segments between locked control points.

TABLE 1
The gradient mixing parameters for each example.

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

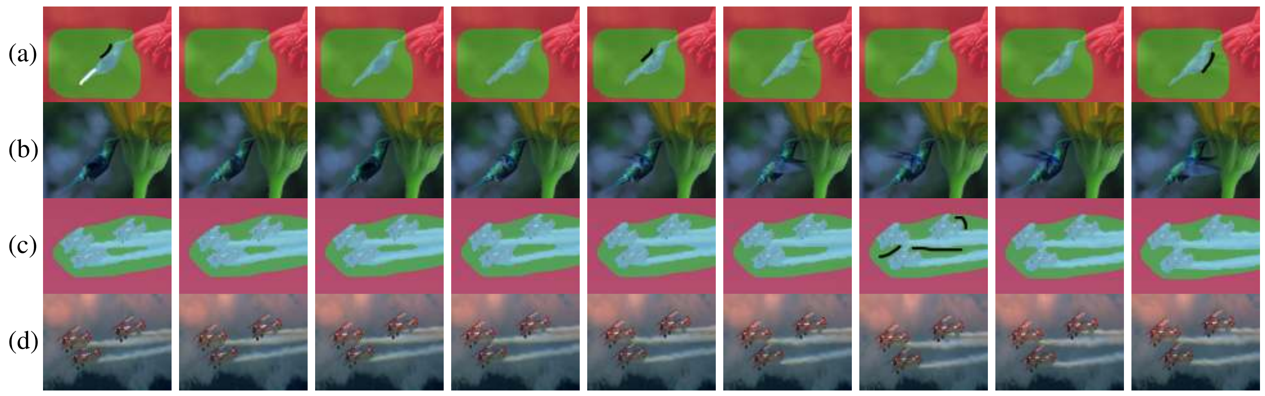


Fig. 10. The input trimap and blending result. The user interaction strokes are shown in black and white. Please see the accompanying video for the results.

both preserved without sudden disappearances thanks to our temporally 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 waves, smoke and dust effects in video. Our method provides harmonious compositions with the target environment.

Performance. All our experiments were performed on a PC with an i7 920 Quad core CPU and 12GB RAM. Preprocessing includes dense optical flow calculation, video segmentation to decide the blending boundary type, and video stabilization (if necessary). Using our un-optimized implementation, these three steps typically take 5, 2 and 2 minutes per 100 frames of VGA video. The user can interactively draw the blending trimap and set the gradient mixing parameters. The number of needed user strokes for generating 100 frame trimaps is usually under 10, as we do not need to extract the boundary of the object. Next, the system generates an initial blending result for each frame with automatic alignment. With this initial blending, we also save the vertex and angle information for MVC hybrid blending. Then, the user can begin to interact for motion alignment refinement and motion editing.

Computation time and user interaction time for each step are shown in Table 2. Interaction time includes time for generating trimaps and (optional) motion editing. 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 the original MVC cloning method due to boundary optimization and distances computation, etc., it is still an order of magnitude faster than solving a Poisson equation as done in hybrid blending (using the TAUCS sparse linear solver). In practice, we restrict current frame blending to a single thread, and limit its rate to below 24 fps to save the computational power for segment blending optimization. Since our method uses approximate MVC, we also show the RMS (root mean square) differences of the hybrid blending results. These values indicate that the differences are unnoticeable. Using per-segment blending optimization produces additional differences from the global optimization, which can be distinguished, especially on motion keyframes. However, as long as temporal coherence is maintained between these frames, 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.'s method [4], frame-by-frame hybrid blending, 3D Poisson blending [23] and Xie's method [24]. Alpha matting loses many details around the source object, and the composition is unrealistic due to large illumination differences. Xie's method also loses details due to use of matting. We further compare the amount of interaction to that needed by alpha matting in Table 2: alpha matting usually requires 5-10 times more strokes than our method, even though the alpha mattes in those examples are still not optimal. Frame-by-frame blending and Xie's method produce results which lack temporal coherence: flickering on the boat and the wake are due to temporal fluctuations in color mismatch, which are overcome by our blending boundary optimization step. In frame-by-frame blending and 3D Poisson blending, inconsistent motion along the blending boundary leads to a very noticeable seam; this problem is solved by gradient mixing in our approach. Our composition results are not affected by the artifacts mentioned above, and are also achieved at an interactive rate.

In Table 2 we summarize statistics and performance for the examples in this paper. For single frame composition, our MVC hybrid blending has higher quality than conventional MVC [6] as evident in less color bleeding and texture smudging, albeit at a cost of greater computation. For video composition, compared to matting based methods such as [4] and [24], our method better preserves motion details such as smoke, blurring and spray; compared to frame-by-frame hybrid blending and [24], our results displays better temporal coherence; compared to 3D Poisson blending [23], our method avoids visual seams around objects, and is more flexible when the target video contains motion-salient objects in the blending region.

Limitation. When the blending boundaries of the source

TABLE 2
Performance of video composition.

	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

and target video sequences are very inconsistent, i.e. their appearances and textures have large differences, our hybrid blending based method degenerates to video matting, e.g. when compositing a white rabbit against a black wall. However, such cases are unsuitable for blending in the first place. Inconsistent lighting directions 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 lacking 3D information. Since positioning is only applied in the image plane, it is unable to produce complex 3D object motion: for example, our editing tool cannot distinguish whether the snow mobile is jumping up or moving away from the screen. However, this could be improved by reconstructing a ground plane of the target scene and adding more degrees of freedom for positioning. Our method also relies on various computer vision techniques for pre-processing, and their imperfections can also lead to 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. This is caused by inaccuracy of optical flow and its confidence values around the thin handrail of the boat. Adopting more recent advanced approaches like those in Liu et al. 2011 [31] may improve the results. Comparison of different composition methods is also limited: there is no reliable way to quantitatively evaluate composition quality such as temporal consistency, apart from examining the results. Currently, such comparisons rely mostly on subjective judgements or user studies.

Not every pair of video sequences is suitable for composition. Large illumination differences, inconsistent background environments and various motions of source object, camera, and background can all prevent good 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, motion consistency could be easily determined. The accumulated score can suggest if the two video sequences can, in fact, be composed successfully.

8 Conclusion

We have presented a novel video blending approach that tackles key challenges in video composition: complex object blending (smoke, water, dust) and motion differences. User-provided *blending trimaps* of the source video allow us to create consistent blending boundaries over time. We mix the gradients of the source and target videos inside the blending region, based on an efficient implementation of mean-value coordinates interpolation instead of 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 globally optimized video composition result in realtime; a possible solution is to use a KD-tree to accelerate boundary optimization and gradient mixing. It would also be 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 controls. 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 those in Chen et al. 2011 [32].

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.

[2] Y. Li, J. Sun, and H.-Y. Shum, “Video object cut and paste,” *ACM Transactions on Graphics*, 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,” *ACM Transactions on Graphics*, vol. 24, no. 3, pp. 585–594, Jul. 2005.

[4] X. Bai, J. Wang, D. Simons, and G. Sapiro, “Video snapcut: robust video object cutout using localized classifiers,” *ACM Transactions on Graphics*, 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 Transactions on Graphics*, vol. 30, pp. 2:1–2:13, February 2011.

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

[7] M. Grundmann, V. Kwatra, M. Han, and I. Essa, “Efficient hierarchical graph-based video segmentation,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2010, pp. 2141–2148.

[8] M. Werlberger, T. Pock, and H. Bischof, “Motion estimation with non-local total variation regularization,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2010, pp. 2464–2471.

[9] G. Zhang, W. Hua, X. Qin, Y. Shao, and H. Bao, "Video stabilization based on a 3d perspective camera model," *The Visual Computer*, vol. 25, pp. 997–1008, October 2009.

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

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

[12] T. Chen, M.-M. Cheng, P. Tan, A. Shamir, and S.-M. Hu, "S-ketch2photo: internet image montage," *ACM Transactions on Graphics*, 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 Transactions on Graphics*, 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," *Computers & Graphics*, vol. 31, no. 2, pp. 212–229, 2007.

[15] R.-F. Tong, Y. Zhang, and M. Ding, "Video brush: A novel interface for efficient video cutout," *Computer Graphics Forum*, vol. 30, no. 7, pp. 2049–2057, 2011.

[16] Z. Tang, Z. Miao, Y. Wan, and D. Zhang, "Video matting via opacity propagation," *The Visual Computer*, vol. 28, pp. 47–61, 2012.

[17] P. J. Burt and E. H. Adelson, "A multiresolution spline with application to image mosaics," *ACM Transactions on Graphics*, vol. 2, pp. 217–236, October 1983.

[18] P. Pérez, M. Gangnet, and A. Blake, "Poisson image editing," *ACM Transactions on Graphics*, vol. 22, no. 3, Jul. 2003.

[19] J. Jia, J. Sun, C.-K. Tang, and H.-Y. Shum, "Drag-and-drop pasting," *ACM Transactions on Graphics*, vol. 25, no. 3, pp. 631–637, Jul. 2006.

[20] Y. Zhang and R. Tong, "Environment-sensitive cloning in images," *The Visual Computer*, vol. 27, pp. 739–748, 2011.

[21] P. Bhat, B. Curless, M. Cohen, and C. Zitnick, "Fourier analysis of the 2d screened poisson equation for gradient domain problems," *Proceedings of European Conference on Computer Vision (ECCV)*, pp. 114–128, 2008.

[22] M. Tao, M. Johnson, and S. Paris, "Error-tolerant image compositing," *Proceedings of European Conference on Computer Vision (ECCV)*, pp. 31–44, 2010.

[23] H. Wang, R. Raskar, and N. Ahuja, "Seamless video editing," in *International Conference on Pattern Recognition (ICPR)*. Washington, DC, USA: IEEE Computer Society, 2004, pp. 858–861.

[24] Z.-F. Xie, Y. Shen, L.-Z. Ma, and Z.-H. Chen, "Seamless video composition using optimized mean-value cloning," *The Visual Computer*, vol. 26, no. 6-8, pp. 1123–1134, 2010.

[25] J.-F. Lalonde, D. Hoiem, A. A. Efros, C. Rother, J. Winn, and A. Criminisi, "Photo clip art," *ACM Transactions on Graphics*, vol. 26, no. 3, pp. 3:1–10, 2007.

[26] J. Lalonde, A. Efros, and S. Narasimhan, "Estimating natural illumination from a single outdoor image," in *IEEE International Conference on Computer Vision*, 2009, pp. 183–190.

[27] K. Sunkavalli, M. K. Johnson, W. Matusik, and H. Pfister, "Multi-scale image harmonization," *ACM Transactions on Graphics*, vol. 29, no. 4, pp. 125:1–125:10, 2010.

[28] S. Zhang, Q. Tong, S. Hu, and R. Martin, "Painting patches: Reducing flicker in painterly re-rendering of video," *Science China Information Sciences*, vol. 54, pp. 2592–2601, 2011.

[29] A. Bousseau, F. Neyret, J. Thollot, and D. Salesin, "Video watercolorization using bidirectional texture advection," *ACM Transactions on Graphics*, vol. 26, July 2007.

[30] 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 (CVPR)*, 2007, pp. 1 –8.

[31] F. Liu, M. Gleicher, J. Wang, H. Jin, and A. Agarwala, "Subspace video stabilization," *ACM Transactions on Graphics*, vol. 30, pp. 4:1–4:10, February 2011.

[32] 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 *IEEE International Conference on Computational Photography (ICCP)*, april 2011, pp. 1 –8.

Tao Chen received the BS degree in Fundamental Science Class and the PhD degree in computer science from Tsinghua University in 2005 and 2011 respectively. He is currently a postdoctoral researcher in Department of Computer Science and Technology, Tsinghua University, Beijing. His research interests include computer graphics, computer vision and image/video composition. He received the Netexplorateur Internet Invention Award of the World in 2009, and China Computer Federation Best Dissertation Award in 2011.

Jun-Yan Zhu received the BE degree with honors in computer science and technology from Tsinghua University in 2012. He is currently a PhD student in the Computer Science Department at Carnegie Mellon University. His research interests include computer vision, computer graphics and computational photography.

Ariel Shamir is an associate Professor at the school of Computer Science at the Interdisciplinary Center in Israel. Prof. Shamir received his Ph.D. in computer science in 2000 from the Hebrew University in Jerusalem. He spent two years at the center for computational visualization at the University of Texas in Austin. During 2006 he held the position of visiting scientist at Mitsubishi Electric Research Labs in Cambridge MA. Prof. Shamir he has numerous publications in journals and international refereed conferences, he has a broad commercial experience working with and consulting numerous companies. He is a member of the ACM SIGGRAPH, IEEE Computer and Eurographics societies.

Shi-Min Hu received the PhD degree from Zhejiang University in 1996. He is currently a professor in the Department of Computer Science and Technology at Tsinghua University, Beijing. His research interests include digital geometry processing, video processing, rendering, computer animation, and computer-aided geometric design. He is associate Editor-in-Chief of The Visual Computer, and on the editorial boards of Computer-Aided Design and Computer & Graphics. He is a member of the IEEE and ACM.