Video Operations in the Gradient Domain

Iddo Drori Tommer Leyvand Shachar Fleishman Daniel Cohen-Or Hezy Yeshurun

Abstract

Fusion of image sequences is a fundamental operation in numerous video applications and usually consists of segmentation, matting and compositing. We present a unified framework for performing these operations on video in the gradient domain. Our approach consists of 3D graph cut computation followed by reconstruction of a new 3D vector field by solving the Poisson equation. We introduce new methods for fusing video smoothly and separating foreground elements from a video background for compositing by defining smooth and sharp transition constraints on a gradient video compositing equation. We demonstrate the applicability of smooth video transitions by fusing pairs for video mosaics, video folding, and video texture synthesis, and demonstrate the applicability of sharp video transitions by video segmentation, video trimap extraction and 3D compositing into a new sequence. Our results demonstrate that our method maintains coherence of the video matte and composite, and avoids temporal artifacts.

Index Terms

Video processing, gradient domain, object segmentation, graph cuts, spline fitting, motion estimation.

I. Introduction

COMPOSITING video and separating foreground video elements from a video background for compositing, as well as background video completion, are operations required in the production of most modern motion pictures. Smoothly fusing parts of video is important for a number of applications: video mosaics, video texture synthesis, and video folding by reusing

parts of video footage. Separating foreground and background is required for creating a video matte which is used for compositing elements into a new background.

Compositing is the process of seamlessly combining multiple image or video regions. It is commonly used in creating mosaics, editing, and texture synthesis. Image and video mosaics are used in applications in which there are multiple sources, each with a limited field of view or from a different origin. The goal is to merge these sources to form a single mosaic. This is performed by first aligning the sources by warping and then smoothly blending the overlapping regions. Image and video editing operations include temporal transitions, placing a figure over a background, and seamlessly combining different image or video regions. Texture synthesis commonly consists of searching for similar patches followed by merging them together. These applications involve compositing as an essential step of the process.

Natural matting is the inverse problem of deriving alpha values around object boundaries by which background and foreground pixels are blended. The operations of compositing and matting are therefore linked by the compositing equation which defines the color of a pixel as a convex combination of foreground and background pixels. The input to video matting is the composite video and a trimap which consists of three grayscale values denoting background, foreground and unknown video regions. Matting is derived from the compositing equation and requires background video completion and foreground estimation as well as a trimap. Background video completion is the process of filling in missing regions in video based on temporal and spatial information. Segmentation of a video into layers, compositing, matting and completion are all inherently linked by their required input-output and usage. Computing a trimap and background video completion, in turn, require rough segmentation of the foreground and compositing of the completed video background.

A. Overview

Our approach is to work in 3D with video volumes in the gradient domain. Therefore, we represent the compositing equation for video volumes and differentiate. Working in the gradient domain allows performing local operations to get a global effect. The differentiation transforms the equation into differential coordinates and results in the *gradient video compositing equation*. Next, we zero out terms in the equation according to the application, whether it requires a smooth transition between videos or sharp transitions between uniform video regions. A new three-dimensional vector field results from removing terms from the gradient video compositing equation. This vector field is reconstructed to get the final result. Frame-by-frame gradient operations give rise to strong temporal artifacts which are noticeable in video, and therefore, we extend the reconstruction equations to three dimensions.

We develop and use several tools in this work. The first is 3D minimum graph cuts and the second is reconstruction from a modified 3D vector field by solving the Poisson equation. The types of cuts and iterations differ for the different applications of video fusion and separation. The new vector fields are defined with respect to the cut, and the result of solving the 3D Poisson equation is a new video or matte according to the application. In addition we use 3D splines to fit positions of tracker data and 2D splines to mark regions around foreground elements in a few key-frames and interpolate between them. Finally, motion estimation is used for the cut computation and in temporal background video completion.

Graph partitioning techniques are commonly used for image and object segmentation, for stitching together aligned images to form mosaics, and for texture synthesis. The graph is partitioned into two disjoint regions by a cut that runs through an error surface defined by pixels in the overlapping region or through graph edges defined between adjacent pixels. Traditionally,

the cost function that determines the degree of dissimilarity between the two pieces is based on intensities and gradients.

In conjunction, a variety of operators such as blending, maximum, and bilinear functions are used for merging image regions. Performing these operations in the spatial, frequency, or gradient domains provides different effects. Inspired by the retinex theory [1] and its extensions [2], reconstructing an image from its modified gradient field has been used for removing shadows from images [3], compressing the dynamic range of images [4], and image editing operations [5]. Operating in the gradient domain is especially suitable for compositing as it focuses on the different types of transitions between regions.

B. Poisson Equation

The Poisson equation allows reconstruction of a scalar function given a vector field and boundary conditions. In our context, modification of the function is performed by modifying the gradient field and the boundary conditions and then reconstruction of the result. The formulation can be viewed as a least squares minimization. Local artifacts are avoided as the least squares solution distributes the error throughout the reconstructed video.

C. Video Compositing

Our approach to video compositing begins by applying a smoothness condition on the gradient video compositing equation. We proceed by first cutting along overlapping sequences in regions where both the motions do not conflict and the colors are similar. Instead of considering only similarity in appearance, our criterion for computing the cut also takes into account motion by computing the error between the motion in overlapping regions and the difference between their colors. Next, the video is reconstructed from a new 3D vector field. This is important

(a) (b) (c) (d)

Fig. 1. (a-b) Input frames from a pair of time-lapse sequences. (c) Cut. (d) Result of gradient video compositing.

for alleviating the problem of fusing together videos with varying appearance as shown in Figure 1 (a),(b). The seam in a 2D slice of a 3D spatial cut between the two sequences shown in (c) is noticeable due to the different scenes and their illumination. Reconstruction of the merged 3D gradients fuses together the regions avoiding the discontinuities as shown in (d). In images, feathering or multi-resolution blending [6] locally smooths the sharp transition. In image sequences temporal effects are enhanced, and therefore 3D gradient compositing is important for maintaining the detail and the temporal coherence of the result.

D. Video Segmentation, Matting and Completion

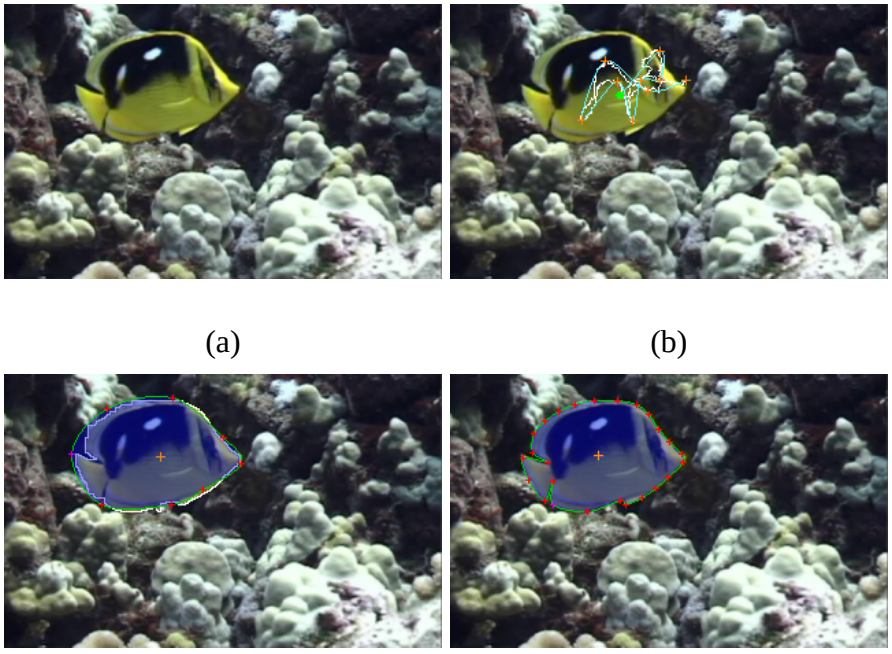
Video segmentation begins by computing a rough binary mask, which is refined to a trimap that consists of three grayscale values, and finally ends with an accurate alpha video matte in [0,1]. For the initial binary mask, the user marks a point on the object and a set of points around the object in a single frame. We then track the object forwards and backwards in time and fit a 3D spline to the tracker data as shown in Figure 2 (b) projected onto a single frame. This is followed by refinement of key-frames using closed 2D shape splines as shown in Figure 2 (c) in green. This serves as input to the next part of our process which requires extracting a more

accurate result. This is computed by iterative 3D graph cuts that snap to the object boundaries. We iterate between steps of dilation and cut computation to find the contour of the video element as shown in (c) overlaid in white. Usually the graph cut results in a discrete representation over which the user has no control, and which is not always our desired final solution. Therefore our approach is to fit a spline to the resulting cut in key-frames. This allows the user to change the control points of the spline as shown in (d) and then perform additional iterations of cut computation.

The resulting contour is dilated once more to form a video trimap which is refined by an accurate computation of a video matte. The video matte is computed by using the 3D Poisson equation. This requires completion of the video background and estimation of the foreground. Our approach to background video completion is using the temporal information present in video by searching forwards and backwards in time spans for background pixels whose motion field is small, and filling in any remaining foreground pixels by diffusion.

II. Related Work

Porter and Duff [7] introduced compositing arithmetic for anti-aliased accumulation of separate elements with alpha into a single image. The associative over operator is used to composite a foreground element over a background. A common approach used for integrating images to form a mosaic is feathering [8], which weighs the contribution of pixels in overlapping regions as a bilinear function of distance from the region boundary or cut. Burt and Adelson [6] introduced the Laplacian pyramid to smoothly merge images according to a binary mask, resulting in a wide overlapping region for low frequencies and a narrow overlap for fine details. The Poisson equation has been widely used in computer vision. The idea of reconstructing an image from its modified gradient field was recently used for recovering reflectance images [9], removing



(a) (b)

(c) (d)

Fig. 2. (a) Input frame. (b) Projected tracker data and 3D spline. (c) Key-frame interpolation and minimum graph cut. (d) Spline fitting to graph cut and its modification.

shadows from images [3], and introduced to computer graphics by Fattal et al. [4] for compressing the dynamic range of images. Subsequently it was used for image editing operations such as seamless blending [5], [10] by mixing image gradients. Perez et al. [5] generalized the framework from rectangular domains to arbitrary patches by requiring Dirichlet boundary conditions, and applied the maximum operator to gradients and suppressed small gradients.

Davis [11] stitched together images to form a mosaic by enforcing a constraint that the error between overlapping regions is minimal. The images are stitched along a minimum cost path computed by Dijkstra's algorithm. The cost function used to determine the cut is the difference between pixel intensities in the overlapping regions normalized by the difference between extreme values. Efros and Freeman [12] applied this method for stitching together texture blocks. The cost function is defined as the difference between pixel intensities in the overlapping regions.

A minimum cost path through the error surface is computed by dynamic programming and is linear in the number of pixels. Kwatra et al. [13] performed texture synthesis by finding the min-cut of a graph using a cost function defined on the edges between adjacent pixels from the overlapping regions. Taking into account the spatial frequencies in the cost function is especially important for perceptual masking. Dividing color by gradient magnitudes gives priority to high gradient regions which provide a snapping effect.

Object segmentation in image sequences is one of the fundamental problems in computer vision. This problem is usually addressed either by discrete representations which are currently manifested by graph partitioning techniques, or by continuous methods typically referred to as active contours. Active contours [14], [15] are used for segmentation by minimizing an energy function which consists of terms that guide the contour towards image features and prefer smooth boundaries. The snake is sensitive to the initial contour and approaches the nearest local minimum. Mortensen and Barrett [16] use dynamic programming for segmenting an object in an image and coined the term intelligent scissors. Xu et al. [17] use iterative graph cuts for segmentation of an object from a background. The image is represented by an adjacency graph, and the user defines a polyline which is dilated to form an initial region around the object. The outer and inner boundaries of this region define the source and sink of the graph and the computed minimum cut is iteratively dilated to form a new boundary region, until converging to a global minimum. Irani et al. [18] and Wang and Adelson [19] use optical flow to separate a video into layers. We apply 3D iterative graph cuts of a video volume to find the initial segmentation of a foreground video element.

Mitsunaga et al. [20] take the derivative of the image compositing equation and assume relatively small changes in the foreground and background regions to derive the relationship

between image gradients and matte gradients. Apostoloff and Fitzgibbon [21] learn the joint distribution between image gradients and alpha gradients from a collection of image sequences. Recently, Jian et al. [22] use the Poisson equation for 2D matting and perform a set of local operations to refine the resulting matte. We present a unified framework for performing different operations on video in the gradient domain, and solve a 3D Poisson equation to extract a video matte avoiding temporal artifacts. Chuang et al. [23] extend Bayesian image matting [24] to video and use optical flow to interpolate a trimap. In contrast, we use an iterative 3D graph-cut approach to segment the foreground and compute a trimap. We show the relationship between completion and compositing, and complete the video background by motion estimation and diffusion.

III. Gradient Video Operations

We present a unified framework for performing operations on video in the gradient domain. We begin by extending a single dot product $c = a \cdot b$ for a single pixel c and vectors a and b to multiple dot-products, $C_p = A_p \cdot B_p = \sum_i A_p^i B_p^i$, one for each coordinate $p = (x, y, z)$ of the video volume C. We then differentiate and work with differential coordinates:

$$\nabla C_p = \sum_i (\nabla A_p^i B_p^i + A_p^i \nabla B_p^i). \tag{1}$$

Finally we create a new vector field by modifying and setting terms to zero according to the desired operation, and reconstructing the result.

A. Gradient Video Compositing Equation

In the spatial domain the *compositing equation* defines the value of each pixel as a linear combination of foreground and background:

$$C = \alpha F + (1 - \alpha)B. \tag{2}$$

In video compositing, C, F, B denote video volumes, α a 3D matte, 1 a volume of ones, and the operations are point-wise multiplication, addition and subtraction between three-dimensional arrays. Differentiating we get the *3D gradient video compositing equation*:

$$\nabla C = \alpha \nabla F + (1 - \alpha)\nabla B + \nabla \alpha (F - B). \tag{3}$$

Our observation is that a smooth transition between foreground and background videos along the 3D matte boundaries yields:

$$\nabla \alpha (F - B) = 0, \tag{4}$$

$$G = \alpha \nabla F + (1 - \alpha)\nabla B, \tag{5}$$

where G is a new vector field. The volume is then reconstructed from the new vector field by solving a 3D Poisson equation. Whereas in the inverse problem, a sharp transition between uniform foreground and background yields:

$$\alpha \nabla F + (1 - \alpha)\nabla B = 0, \tag{6}$$

$$\nabla C = \nabla \alpha (F - B), \tag{7}$$

which defines the relationship between video matte gradients and video gradients for uniform regions.

IV. 3D POISSON

In this work we perform operations on video in the gradient domain. Frame-by-frame gradient compositing and matting gives rise to strong temporal artifacts. We therefore extend the reconstruction equations to three dimensions. Following the notation of [4], we search the space of all 3D potential functions for a function V whose gradients are closest to G in the least squares sense; V should minimize the integral:

$$\int\int\int F(\nabla V, G)dx\,dy\,dz, \tag{8}$$

where

$$\begin{aligned} & F(\nabla V, G) = \|\nabla V - G\|^2 \\ = \; & (\frac{\partial V}{\partial x} - G_x)^2 + (\frac{\partial V}{\partial y} - G_y)^2 + (\frac{\partial V}{\partial z} - G_z)^2. \end{aligned} \tag{9}$$

According to the variational principle, a function V that minimizes the integral in Eq. 8 must satisfy the Euler-Lagrange equation, that for three independent variables generalizes to:

$$\frac{\partial F}{\partial V} - \frac{d}{dx}\frac{\partial F}{\partial V_x} - \frac{d}{dy}\frac{\partial F}{\partial V_y} - \frac{d}{dz}\frac{\partial F}{\partial V_z} = 0, \tag{10}$$

which is a partial differential equation in V. Substituting F results in satisfying the Poisson equation:

$$\nabla^2 V = \nabla \cdot G, \tag{11}$$

where

$$\nabla^2 = \frac{\partial^2 V}{\partial x^2} + \frac{\partial^2 V}{\partial y^2} + \frac{\partial^2 V}{\partial z^2} \tag{12}$$

is the three-dimensional Laplacian operator, and

$$\nabla \cdot G = \frac{\partial G_x}{\partial x} + \frac{\partial G_y}{\partial y} + \frac{\partial G_z}{\partial z} \tag{13}$$

is the divergence of the vector field measuring the extent to which a point in the field is a source or sink.

A. Discrete Solution

To solve for V we consider Eq. 11 as a boundary value problem of the form $L\phi = f$, where L is an operator, ϕ an unknown scalar field, and f a non-homogeneous term. When there is no analytical solution, the domain is a discrete volume grid. Methods for solving the Poisson equation in 3D are well known in scientific computation [25].

The divergence is approximated by the finite backward difference:

$$\begin{aligned} divG &\approx G_x(x,y,z) - G_x(x-1,y,z) \\ &+ G_y(x,y,z) - G_y(x,y-1,z) \\ &+ G_z(x,y,z) - G_z(x,y,z-1). \end{aligned} \tag{14}$$

The intensity gradient ∇V is approximated by the finite forward difference:

$$\begin{aligned} \nabla V(x,y,z) &\approx (V(x+1,y,z) - V(x,y,z), \\ &\quad V(x,y+1,z) - V(x,y,z), \\ &\quad V(x,y,z+1) - V(x,y,z)). \end{aligned} \tag{15}$$

The Laplacian is approximated using finite differences by a Taylor series expansion of each voxel. We considering the six connected neighbors of each voxel and their three directions. Then expand in the positive and negative directions to fourth order, add each pair of equations, solve

for the second derivative, and sum the solutions in each of the coordinates to get:

$$\begin{aligned}\nabla^2 V(x,y,z) &\approx V(x+1,y,z)+V(x-1,y,z)\\ &+ V(x,y+1,z)+V(x,y-1,z)\\ &+ V(x,y,z+1)+V(x,y,z-1)\\ &- 6V(x,y,z). \end{aligned} \tag{16}$$

This results in a set of linear equations that in matrix form contains only seven non-zero elements in each row. For compositing of video pairs the computation is performed on the entire volume and therefore we use a 3D Poisson solver with multi-grid [25] on an entire regular domain. We solve Eq. 11 separately for each channel. For video matting the computation is performed only on video regions corresponding to the gray area of the video matte and therefore we use a direct solver [26] on an iregular grid. The boundary conditions are defined by the known (0 or 1) regions of the trimap. We solve a linear system whose unknowns are voxels only in the gray region.

B. Boundary Conditions

We apply two types of boundary conditions in 3D for the respective applications of smooth and sharp transitions. Neumann boundary conditions specify the normal derivative of a function on a surface, which in our case means that for compositing of video pairs the derivative on the volume boundary is zero, and pad the volume boundaries. Dirichlet boundary conditions specify the value of a function on a surface:

$$V|_{\partial\Omega} = V^*|_{\partial\Omega} \tag{17}$$

where V^* provides the values on the boundary $\partial\Omega$. In our case this means that known matte voxels (0 or 1) constitute boundary conditions for neighbouring unknown voxels in the gray region of the trimap. Dirichlet boundary conditions allow working with arbitrary 3D outlines.

V. Smooth Transition

A. Appearance and Motion Cuts

To compute the vector field in Eq. 5 requires computing a mask which determines the contributions of the gradients from the videos. Therefore, prior to reconstruction, our goal is to find a good criterion for computing a cut between overlapping videos. We would like the cut to pass through areas where both the motions do not conflict and the colors are similar. Therefore, in the overlapping region we look at the error volume between both the colors and the motions. The matching cost between two overlapping regions A and B in each 3D coordinate p is defined as:

$$\textit{cost}(p) = \lambda\|\mathbf{c}_A(p) - \mathbf{c}_B(p)\| + (1-\lambda)\|\mathbf{f}_A(p) - \mathbf{f}_B(p)\|, \tag{18}$$

where **c** denotes color channels, **f** denotes motion field values, and λ a scalar that determines the balance between color and motion.

The cut is followed by reconstructing a function from a 3D vector field. In this respect the motion constraint is more elaborate than directly requiring that the gradients also agree along the cut by the constraint $\|\mathbf{G}_A(p) - \mathbf{G}_B(p)\|$. We use an optical flow technique of Black and Anandan [27] which estimates the flow in a multi-scale fashion, from coarse to fine, to handle large motions. It uses robust statistics to avoid large errors from outliers and allows for flow discontinuities.

B. Efficiency

Working with video, our goal is to limit the computation time of the entire compositing process to be linear in the number of voxels. We have experimented with finding a minimum graph cut in two and three dimensions [28] as shown in Figure 7 (c). However, when compositing pairs of video the cut computation is performed on the entire video volume and therefore computation time is dependent on the number of vertices times the number of edges in the graph, which is quadratic in the number of voxels. A direct extension of image quilting [12] to 3D using dynamic programming would result in a minimum cost 1D path in the volume. Using dynamic programming for each spatial or temporal 2D slice separately would not preserve coherence, resulting in noticeable artifacts.

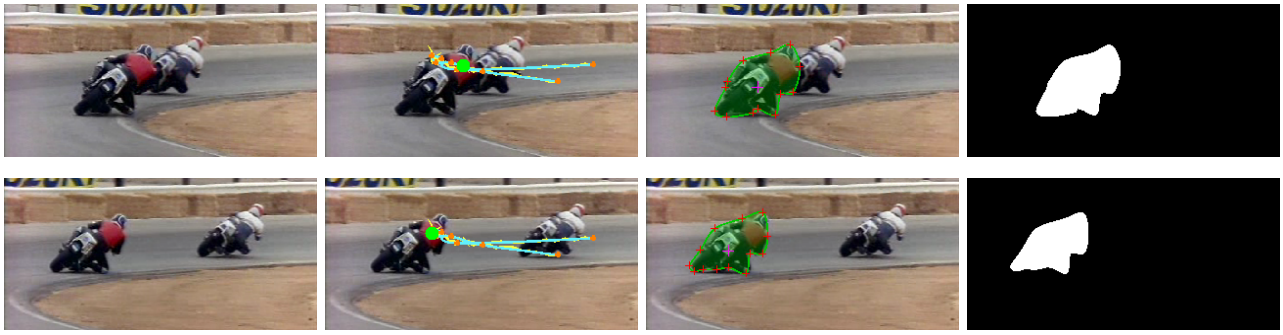
We therefore combine dynamic programming with a greedy heuristic, modifying the cost according to adjacent slices. More specifically, in the initialization phase, the cost of each position is the sum of three functions: (i) a function of appearance and motion as defined in Eq. 18; (ii) a function of the matte which is the distance from the corresponding previous cut maintaining temporal coherence; and (iii) a Gaussian function of position which avoids trivial cuts in the overlapping region by assigning a penalty to voxels near the boundaries. In the update phase, this sum is added to the minimum over the previous positions. We perform this computation for finding both spatial and temporal cuts. Temporal cuts are found by rotating the inputs, performing the computation, and then rotating back the output. We found the results of this approach to be sufficient for our purposes as it avoids extreme changes of the cut, while maintaining linear computation time.

VI. SHARP TRANSITION

Our approach for extracting a foreground video element consists of three stages: (i) key-frame tracking; (ii) iterative graph-cuts; and (iii) matte extraction. First, the user marks a point on the object and a region around the object in a key-frame. The position is tracked throughout the entire sequence and the scale is estimated. Next we fit a 3D spline to the tracker data and define closed 2D splines around the object in each key-frame. The splines are interpolated across the remaining frames resulting in a binary mask. Next, we apply an iterative 3D graph cut algorithm for extracting a trimap from the binary mask, and finally we extract an accurate 3D alpha matte from the trimap by completing the background and estimating the foreground in the gray region, and then solving a 3D Poisson equation. The different stages are characterized by the accuracy of data maintained in each stage: starting from a rough binary mask, onto a more accurate trimap, and finally a fine 3D alpha channel used for 3D compositing.

A. Key-frame Tracking

We would like to extract an object from an input sequence. First, the user marks a point on the object to track and an area around it in a single frame. According to this point we track the object and estimate its location in each frame to get an ellipse that denotes the change in scale in two orthogonal axes. Let T denote the set of tracker positions in each frame. We fit a 3D spline to the tracker position as shown projected onto a single frame in Figure 3 (b). Fitting the 3D spline is performed incrementally using a greedy algorithm. We begin by fitting a 3D spline through all positions $P = T$. Next, in each iteration of the algorithm we find a point q whose removal from the set P results in a new spline minimizing an overall error between the tracker data T and the spline $S_{P \setminus q}$:



(a) (b) (c) (d)

Fig. 3. Tracking: (a) Input frames. (b) Tracker data and 3D spline projected on frame. (c) Closed 2D shape spline. (d) Binary mask.

$$q^* = \arg\min_{q \in P} d(S_{P \setminus q}, T) \tag{19}$$

$$d(S_P, T) = \sum_{t \in T} d(S_P, t), \tag{20}$$

where $d(S_P, t)$ is the distance between the 3D spline curve S_P and a tracker point t according to the L_2 norm. Fitting is performed until reaching a minimum number of key-frames. This result is refined by the user by adding key-frames and refining the 2D shape splines around the object as shown in Figure 3 (c). Each key-frame the user updates is propagated to the following frames. The output defines an initial binary mask as shown in (d) for the iterative graph-cut algorithm which extracts a trimap.

B. Iterative Graph-Cuts

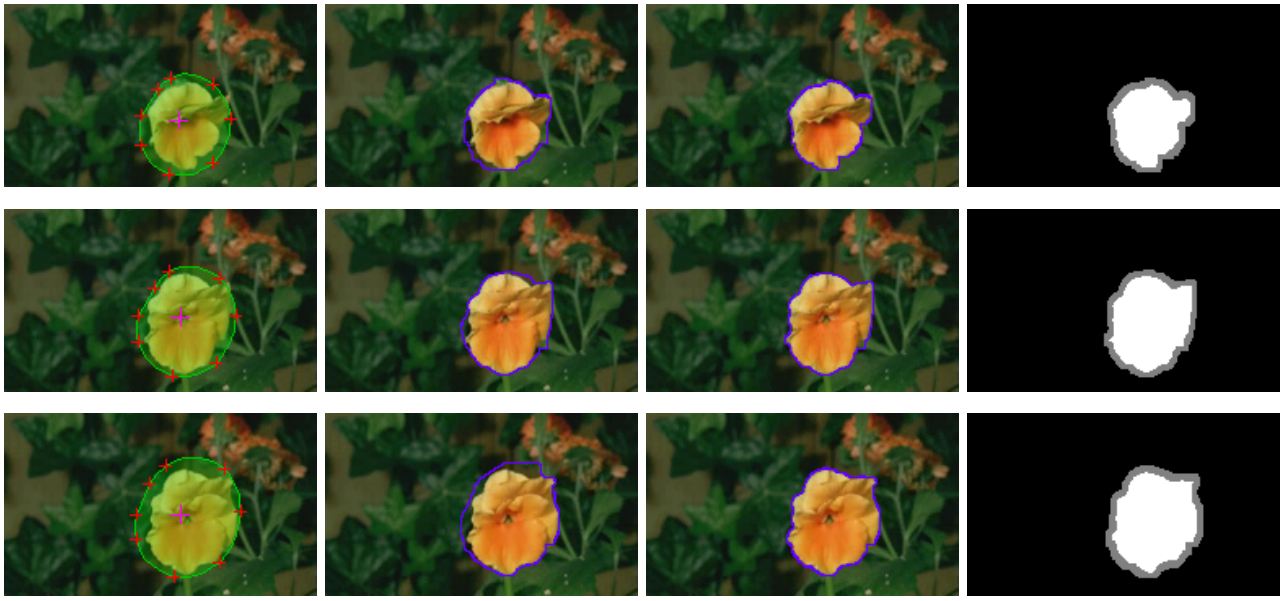
In order to extract the foreground object from a video and composite it onto a new background we need to find a trimap that defines the foreground and background regions and an unknown region (where alpha values are in $[0, 1]$). Given a rough binary mask line around the object, we perform iterations of dilation and computation of the minimum graph cut in the dilated region.

Each iteration defines a more accurate region and the cut typically converges between three to ten iterations depending on the accuracy of the spline and the initial kernel size which is decreased in each iteration. The cut is computed by taking the inner and outer vertices of the dilated region to be source and sink. This provides the cut with an initial estimate of the object boundary which is refined in each iteration. This iterative process defines a contour around the object. Nodes of the graph are created only in the boundary region, resulting in a relatively small graph and efficient computation (rather than a regular graph of the entire volume with n^3 nodes). The weights of vertical, horizontal and temporal edges are set to the absolute of color difference divided by the sum of directional luminance gradient magnitudes.

Figures 4 demonstrates the various stages using our graph cut trimap extraction: The user marks point on the object and a few points around the object in a key-frames. The object is tracked and a 3D spline is computed through the tracker data. In addition 2D shape splines are computed through the point around the object and interpolated between all key-frames as shown in (a). This defines an initial binary video mask which serves as an initial estimate to the 3D iterative graph cut algorithm. The boundary is dilated to form a region in which a 3D minimum cut is computed as shown in (b). Dilation and minimum cut computation are performed iteratively, as described in Figure 5, resulting in a contour around the object shown in (c) which is once more dilated to form a video trimap shown in (d).

C. Video Matting and Completion

The trimap is used to compute an accurate video matte. We extract the matte by assuming that the background and foreground are relatively smooth and setting the corresponding terms in the gradient compositing equation to zero. The 3D gradients of the video are then proportional to the 3D gradients of the matte. The scaling factor for each voxel in the unknown region is the



(a) (b) (c) (d)

Fig. 4. Video trimap extraction: (a) Keyframe spline interpolation. (b) Initial slice of 3D graph cut. (c) Result of iterative 3D graph cut. (d) Graph cut trimap.

difference between the foreground and the background. The resulting matte is used to composite the video element onto a new background or video.

To extract the matte we need to complete the video background B and estimate the foreground

Input: color video, binary foreground video mask

Output: foreground video trimap

Algorithm:

```
do
    set mask boundary to gray
    dilate gray boundary
    compute 3D minimum graph cut in boundary
    update new binary mask
until convergence
```

Fig. 5. Iterative graph-cuts: trimap extraction pseudocode.

F only in the unknown (gray) trimap regions. In this respect using a 3D video volume presents a clear advantage over 2D matting. Temporal information is commonly used for video completion [29]. Approximating camera motion with a global transformation by tracking a sparse set of features allows warping the image sequence to a single reference frame, applying the inverse transformation after completion. We utilize the temporal information present in image sequences by searching in temporal neighborhoods along the missing volume. We search along time spans, forward and backward in time, to find the nearest existing background in the same spatial coordinate whose flow magnitude is near zero. When a foreground object cannot be filled in by pixels from a temporal neighborhood, for example, when background information is unavailable or its temporal distance is too far, we perfom diffusion. We have also found useful the temporal median computed over span points with no or little motion. Background completion is described in pseudocode in Figure 6. The foreground video in the gray trimap region is estimated by diffusion.

Input: color video, video trimap

Output: completed video background in gray trimap region

Algorithm:

```
compute optical flow
for each frame
    for each unknown pixel
        search time span from near to far
            if background pixel and
            flow magnitude near zero
                assign color, update mask, and end search
if unknown pixels remain
    fill in by diffusion
```

Fig. 6. Background video completion pseudocode.

VII. Results

A. Video Compositing

We first demonstrate some 2D results of reconstructing an image from merged gradients of a pair of input images with respect to a 2D minimum graph cut. The top row of Figure 7 (d) shows the reconstructed result is coherent in both texture and color. The lower row of Figure 7 shows a pair of input images with different skies and actors in the scenes (a-b). The third figure (c) shows the result of a 2D minimum graph cut, and the rightmost figure (d) shows the result of gradient image compositing with respect to (c). Gradient image compositing merges the sky seamlessly while maintaining the silhouettes around the shield. Computation time is under a minute for a pair of 720 by 480 input images on a 1.8 GHz P3 processor, for both the quadratic 2D minimum graph-cut computation and for solving the 2D Poisson equation using the full multigrid method.

(a) (b) (c) (d)

Fig. 7. (a-b) Pair of input images. (c) 2D minimum graph cut. (d) Result of gradient image compositing.

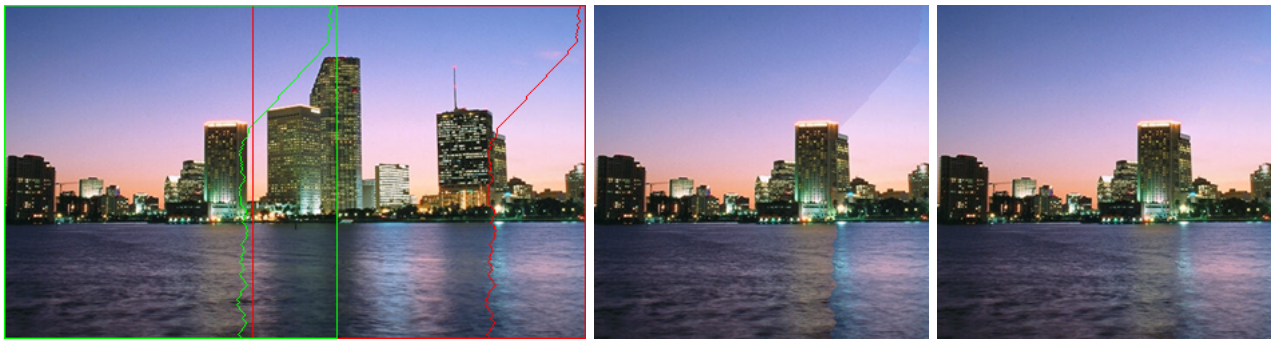
We have experimented with compositing various pairs of sequences with varying geometric and photometric differences from a database of video footage. The subjects of these sequences

(a) (b) (c)

Fig. 8. Video mosaic: (a-b) Input frames from a pair of time-lapse sequences. (c) Result of gradient video compositing.

are varied and include time-lapse, scenery, textures, and more. We set an equal balance between color and motion, $\lambda = 0.5$, in computation of the cut. Total computation time of the *entire video* is between 30 and 60 seconds for input pairs of size between 112^3 and 148^3 on a 1.8 GHz P3 processor with 1GB of RAM. In the following we show the spatial and temporal effects resulting from gradient video compositing (gvc). The accompanying videos demonstrate these and other spatial and temporal effects. We use video pairs to demonstrate the applicability and advantages of our compositing method for mixing, extending and folding videos in space and time. These operations are commonly used in creating video mosaics, in video editing, and video texture synthesis.

1) Video mosaics: Figure 8 shows frames from a pair of time-lapse image sequences from different sources (a-b), and a frame from the result of gradient video compositing is shown in (c). Notice that while the cut is found only in the overlapping region, the sequence is reconstructed from gradients on the entire domain to get a single coherent time-lapse output. We applied our technique for temporally merging pairs of different sequences which is used for extending videos in time and space for creating video mosaics.



(a) (b) (c)

Fig. 9. Image folding: (a) Input image. (b) Cut. (d) Result of gradient image compositing.

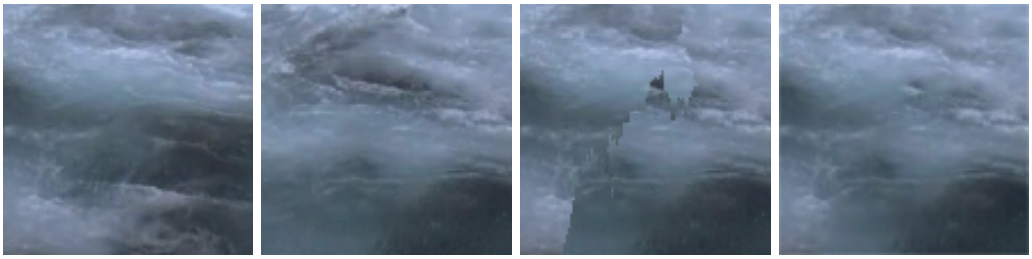
2) Video folding: Figure 9 demonstrates the application of our method to folding an image by placing the left and right portions of the image one over the other, computing a 2D cut in the overlapping region, and then reconstructing the image from the merged gradients. Figure 10 shows the application of our method to spatially folding a video by placing the left and right volumes one over the other, computing a 3D cut in the overlapping volume, and reconstructing the video from a 3D vector field. We demonstrate compositing results for both static and moving cameras, as well as a scene with noisy snowfall background, while maintaining temporal coherence of the output. Figure 11 shows the application of our method to folding image sequences by cutting a temporal band to change the sequence duration. A 2D slice of a 3D temporal cut of the first and last sequence segments is shown in (c), and the result of gradient video compositing in (d). This is used for folding videos in space and time.

3) Video textures: Figure 12 shows frames from different pairs of image sequences (a-b). A 2D slice of 3D spatial and temporal cuts is shown in (c). Our method spatially and temporally composites the clouds, fire and smoke textures, and mixes the water textures seamlessly as shown in (d). This is used to extend videos in space and time for video textures.

In these results we demonstrate the compositing of pairs of sequences. For applications such

(a) (b) (c)

Fig. 10. Spatial video folding: (a) Input frame. (b) Cut. (c) Result of gradient video compositing.



(a) (b) (c) (d)

Fig. 11. Temporal video folding: (a) Input frame. (b) Cut. (c) Result of gradient video compositing.

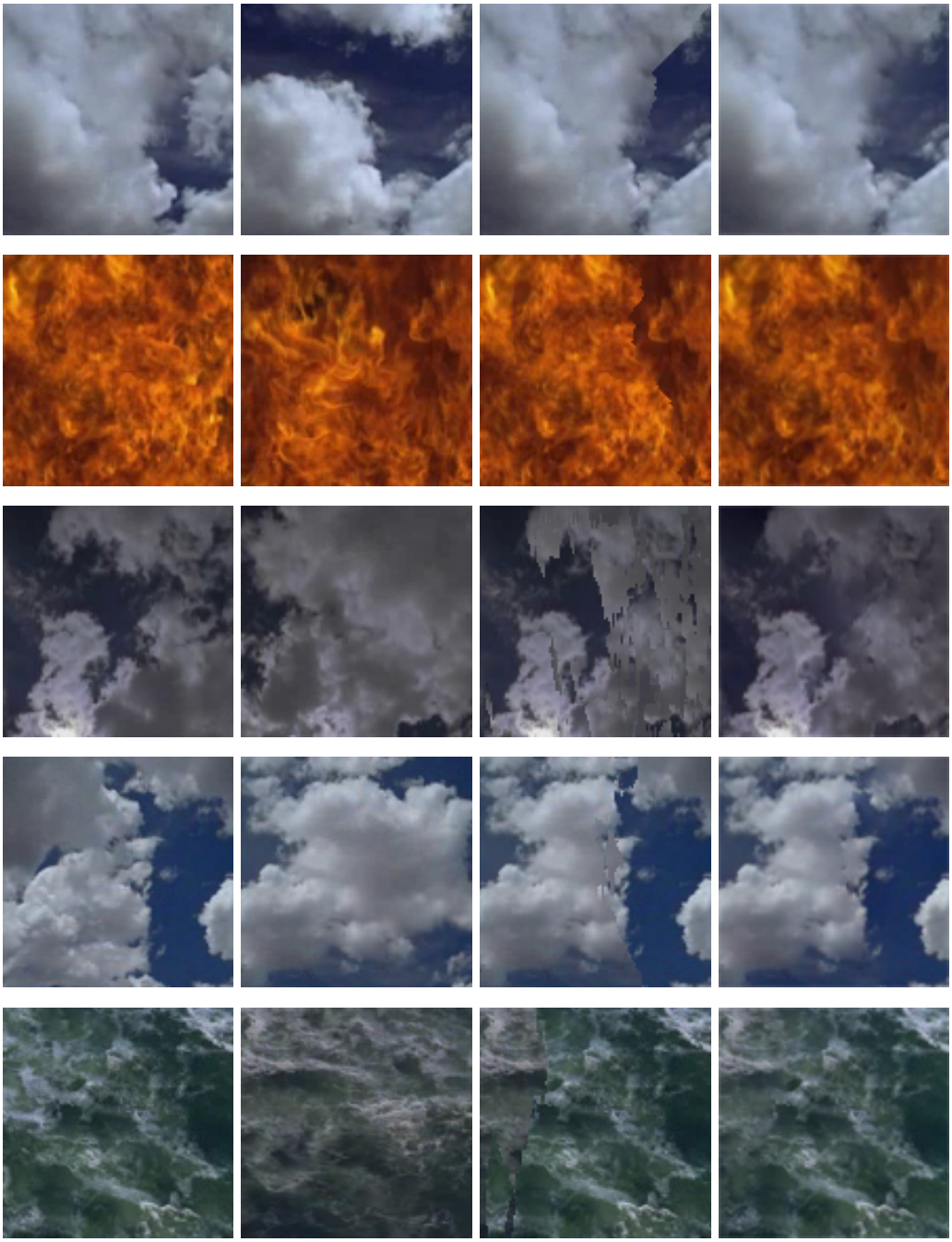
as video mosaics, editing, and texture synthesis this extends to multiple matching sequences. The sequences are sequentially cut, storing each mask, and the resulting 3D gradient field is reconstructed once. This ensures that the spatial and temporal appearance of the entire sequence is coherent.

B. Video Trimap Extraction

We have experimented with our method for video trimap extraction on various videos with uniform, textured and complex backgrounds. Figure 13 shows a frame from the result of video trimap extraction for a time-lapse sequence of a flower opening (shown in the accompanying videos). The figure demonstrates the various steps of our method: (a) keyframe spline interpolation, (b-c) iterative 3D graph cuts, and a frame of the resulting trimap (d).

C. Video Matting and Completion

We first demonstrate 3D gradient video matting on a simple sequence with noticeable transparencies. A frame of a short sequence of a lioness is shown on the left of Figure 14 and a frame of the resulting gradient video matte on the right. Notice the fine detail on the chin and whiskers. Two-dimensional, frame-by-frame, gradient compositing and matting suffers from



(a) (b) (c) (d)

Fig. 12. Spatial and temporal video textures: (a-b) Input frames from sequence pairs. (c) Cut. (d) Result of gradient video compositing.

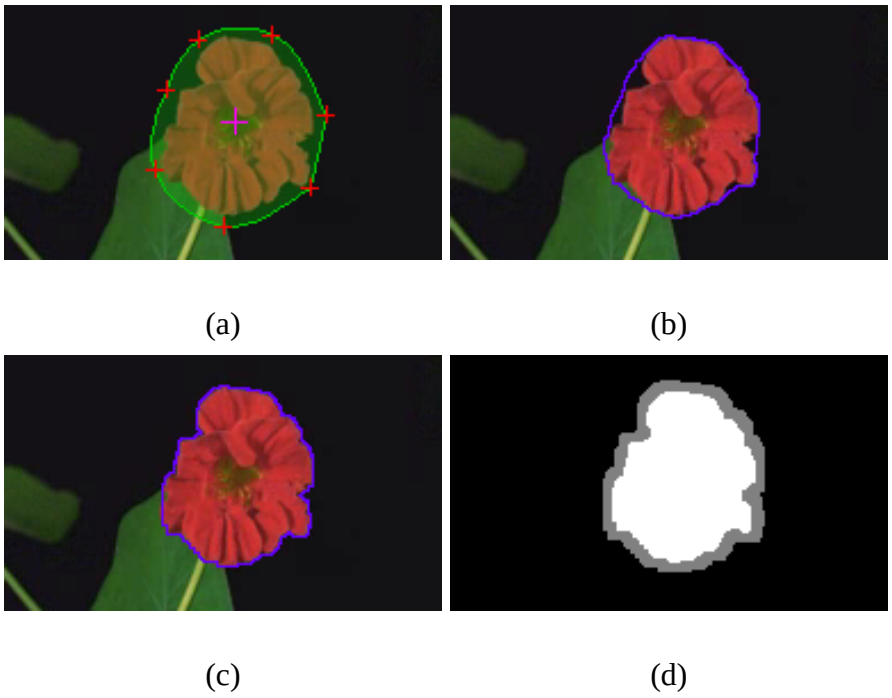


Fig. 13. Video trimap extraction: (a) Keyframe spline interpolation. (b) Initial slice of 3D graph cut. (c) Result of iterative 3D graph cut. (d) Graph cut trimap.

temporal artifacts and the advantage of our video matting approach is that it is performed in 3D and therefore results in a consistent matte which maintains temporal coherence. Figure 15 demonstrates the results of gradient video matting. The sequence contains 192 frames of 192 by 112 resolution, and contains four user specified key-frames. An input frame is shown in (a) and the corresponding graph cut trimap in (b). Foreground estimation in the unknown trimap region is shown in (c) and video background completion of the unknown trimap region in (d). The accurate video matte extracted by our 3D method is shown in (d) and a composite on a new background in (e). Total computation time is 170 seconds, including seven graph cut iterations and solving for 138565 unknowns in the matte computation.

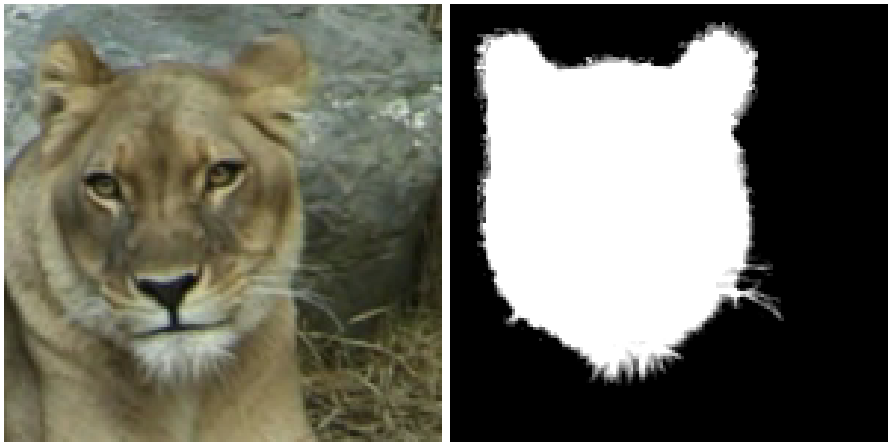


Fig. 14. Video matting: frame from input video (left), and frame from resulting video matte (right).

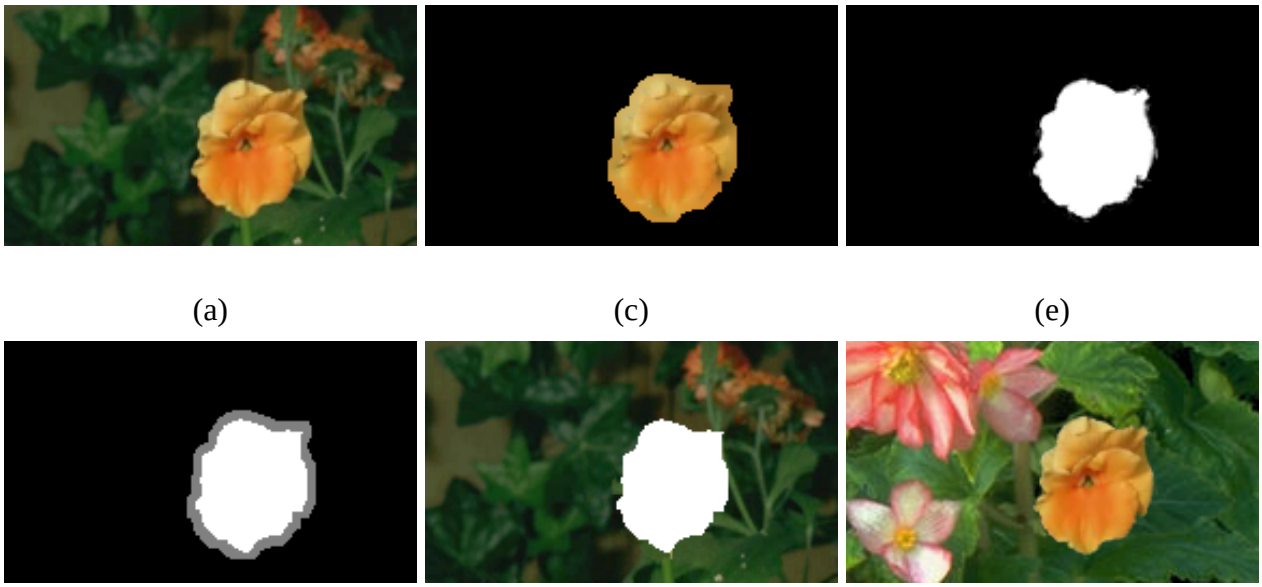


Fig. 15. Video matting and completion: (a) Frame from input video. (b) Graph cut trimap. (c) Foreground estimation in unknown region. (d) Background completion of unknown region. (e) Alpha matte. (f) Composite.

VIII. Conclusions

The contributions of this work include the introduction and use of the gradient video compositing equation for smooth and sharp transitions between videos. This is performed by cutting followed by reconstruction of a 3D vector field, utilizing and maintaining the temporal coherence

present in the video and performing iterative 3D graph cuts for capturing the contours of foreground video elements. It is based on extending the reconstruction of a vector field to 3D using the Poisson equation and considers motion in the cut computation. We provide a 3D formulation which reconstructs a temporally coherent output, avoiding temporal artifacts by spreading the reconstruction error of our least squares solution over the entire video volume. Finally, our results show useful applications for video mosaics, folding, video textures, compositing, video trimap extraction, video matting and background completion, within a unified framework.

REFERENCES

[1] E. H. Land and J. McCann, "Lightness and retinex theory," *Journal of the Optical Society of America*, vol. 1, no. 61, pp. 1–11, 1971.

[2] Z. Rahman, D. J. Jobson, G. A. Woodell, and G. D. Hines, "Multi-sensor fusion and enhancement using the retinex image enhancement algorithm," in *Visual Information Processing XI Proceedings of SPIE*, vol. 4736, 2002.

[3] G. D. Finlayson, S. D. Hordley, and M. S. Drew, "Removing shadows from images," in *European Conference on Computer Vision*, 2002, pp. 823–836.

[4] R. Fattal, D. Lischinski, and M. Werman, "Gradient domain high dynamic range compression," *ACM Transactions on Graphics (TOG)*, vol. 21, no. 3, pp. 249–256, 2002.

[5] P. Perez, M. Gangnet, and A. Blake, "Poisson image editing," *ACM Transactions on Graphics (TOG)*, vol. 22, no. 3, pp. 313–318, 2003.

[6] P. J. Burt and E. H. Adelson, "Merging images through pattern decomposition," *Applications of Digital Image Processing VIII*, vol. 575, pp. 173–181, 1985.

[7] T. Porter and T. Duff, "Compositing digital images," in *Computer Graphics (Proceedings of ACM SIGGRAPH 84)*, 1984, pp. 253–259.

[8] R. Szeliski, "Video mosaics for virtual environments," *IEEE Computer Graphics and Applications*, pp. 22–30, 1996.

[9] Y. Weiss, "Deriving intrinsic images from image sequences," in *Proceedings of International Conference on Computer Vision*, 2001, pp. 68–75.

[10] A. Levin, A. Zomet, S. Peleg, and Y. Weiss, "Seamless image stitching in the gradient domain," in *Proc. of the European Conference on Computer Vision, to appear*, 2004.

[11] J. Davis, “Mosaics of scenes with moving objects,” in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 1998, pp. 354–360.

[12] A. A. Efros and W. T. Freeman, “Image quilting for texture synthesis and transfer,” in *Proceedings of ACM SIGGRAPH 2001*, 2001, pp. 341–346.

[13] V. Kwatra, A. Schodl, I. Essa, G. Turk, and A. Bobick, “Graphcut textures: image and video synthesis using graph cuts,” *ACM Transactions on Graphics (TOG)*, vol. 22, no. 3, pp. 277–286, 2003.

[14] M. Kass, A. Witkin, and D. Trezopoulos, “Snakes: Active contour models,” *International Journal of Computer Vision*, vol. 1, no. 4, pp. 321–331, 1987.

[15] A. Blake and M. Isard, *Active Contours*. Springer-Verlag, 1998.

[16] E. N. Mortensen and W. A. Barrett, “Intelligent scissors for image composition,” in *Proceedings of ACM SIGGRAPH 95*, 1995, pp. 191–198.

[17] N. Xu, R. Bansal, and N. Ahuja, “Object segmentation using graph cuts based active contours,” in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2003, pp. 46–53.

[18] M. Irani, B. Rousso, and S. Peleg, “Computing occluding and transparent motions,” *International Journal of Computer Vision*, vol. 12, pp. 5–16, 1994.

[19] J. Y. A. Wang and E. H. Adelson, “Representing moving images with layers,” *IEEE Transactions on Image Processing*, vol. 3, no. 5, pp. 625–638, 1994.

[20] T. Mitsunaga, T. Yokoyama, and T. Totsuka, “Autokey: Human assisted key extraction,” in *Proceedings of ACM SIGGRAPH 95*, 1995, pp. 265–272.

[21] N. Apostoloff and A. Fitzgibbon, “Bayesian video matting using learnt image priors,” in *BMVA symposium on Spatiotemporal Image Processing*, 2004.

[22] J. Sun, J. Jia, C.-K. Tang, and H.-Y. Shum, “Poisson matting,” *ACM Transactions on Graphics, to appear*, 2004.

[23] Y.-Y. Chuang, A. Agarwala, B. Curless, D. H. Salesin, and R. Szeliski, “Video matting of complex scenes,” *ACM Transactions on Graphics (TOG)*, vol. 21, no. 3, pp. 243–248, 2002.

[24] Y.-Y. Chuang, B. Curless, D. H. Salesin, and R. Szeliski, “A bayesian approach to digital matting,” in *IEEE Conference on Computer Vision and Pattern Recognition*, vol. 2, 2001, pp. 264–271.

[25] A. Brandt, “Multiscale scientific computation: Review 2001,” in *Multiscale and Multiresolution Methods: Theory and Applications*, T. Barth, T. Chan, and R. Haimes, Eds. Springer Verlag, 2001.

[26] TAUCS, “A Library of Sparse Linear Solvers, Tel-Aviv University,” http://www.tau.ac.il/ stoledo/taucs/, 2003.

[27] M. J. Black and P. Anandan, “The robust estimation of multiple motions: Parametric and piecewise-smooth flow fields,”

Computer Vision and Image Understanding, vol. 63, no. 1, pp. 75–104, 1996.

[28] Y. Boykov, O. Veksler, and R. Zabih, “Fast approximate energy minimization via graph cuts,” *IEEE transactions on Pattern Analysis and Machine Intelligence*, vol. 23, no. 11, pp. 1222–1239, 2001.

[29] Pixeldust@, “2d3,” http://www.2d3.com, 2003.