Alpha-Flow for Video Matting

M. Sindeev[1,2], A. Konushin[2], and C. Rother[3]

[1] Keldysh Institute of Applied Mathematics, Moscow, Russia
[2] Lomonosov Moscow State University, Moscow, Russia
[3] Microsoft Research Cambridge, United Kingdom

Abstract. This work addresses the problem of video matting, that is extracting the opacity-layer of a foreground object from a video sequence. We introduce the notion of alpha-flow which corresponds to the flow in the opacity layer. The idea is derived from the process of rotoscoping, where a user-supplied object mask is smoothly interpolated between keyframes while preserving its correspondence with the underlying image. Our key contribution is an algorithm which infers both the opacity masks and the alpha-flow in an efficient and unified manner. We embed our algorithm in an interactive video matting system where the first and last frame of a sequence are given as keyframes, and additional user strokes may be provided in intermediate frames. We show high quality results on various challenging sequences, and give a detailed comparison to competing techniques.

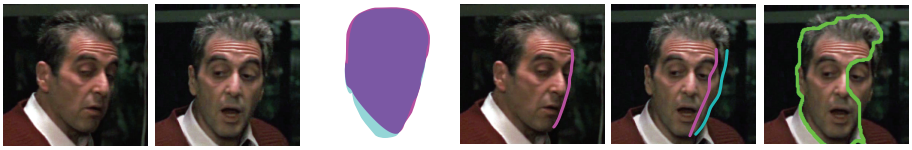
1 Introduction

For many video manipulation tasks a popular first step is to separate a foreground object from its background. Examples are video cut-and-paste or unwrap mosaic [1]. The foreground extraction task is called video matting and is to-date a very hard problem, even when manual user guidance is allowed and state-of-the-art algorithms are employed. As most approaches on video matting we assume a simple image formation process as

$$I = \alpha F + (1 - \alpha)B, \tag{1}$$

where I is the color of a pixel in the source video, F and B are the foreground and background colors respectively, and α is the blending factor, which is in the range $[0, 1]$. The goal is to recover α, F and B given the source video sequence I. A special case is the binary segmentation problem were α is limited to two values, $\{0, 1\}$. Since the problem is severely under-constrained, additional user input is often necessary. This can be in the form of scribbles in the places where α is 0 or 1, or in form of a full alpha-matte for keyframes. The problem of recovering the unknown variables is then often formulated as an energy minimization problem where the energy function consists of a likelihood term, which is the "recomposition error" measured as the squared difference of the left- and right-hand sides of equation (1), and prior terms for the unknown variables.

The main motivation of our work comes from the approach that human rotoscopists take to perform video matting. They animate a spline contour that is smoothly interpolated between the keyframes. This gives in practice good results and hides artifacts since the smooth motion of the contour (based on keyframes only) often matches the real motion of boundaries, or at least approximates them to an extent where imprecisions are unnoticeable to the viewer. We extend this strategy by considering also the underlying motion information from the video.



(a) (b) (c) (d) (e) (f)

Fig. 1. Illustration of the alpha-flow principle. (a, b) Two frames from the movie "Godfather III". **(c)** Overlaid alpha mattes for these frames. Note that difference of the mattes is much smaller than between images. **(d)** Fragment of the contour overlaid onto frame (a). **(e)** Purple contour from (e) as if it was tracked to the frame (b) using the ideal optical flow in RGB. The actual contour of the head is shown in cyan. Note that the space between the two contours is occluded in the first frame. **(f)** Result of Video SnapCut [2], which does not exploit the idea of alpha-flow. As expected, the mistakes are mainly on the right-hand side of the face (because of occlusion).

By replacing the spline contours of a rotoscopist with a full alpha-matte one can view the video-matting problem as a motion guided interpolation between keyframes.

Our key observation is that the optical flow of α is much smoother than the optical flow of the RGB video, as well as that of F and B. We call the optical flow of the alpha channel the **alpha-flow** and illustrate this concept in Fig. 1. Note that alpha-flow, in contrast to RGB flow, does not have occlusions from fore- to background. The only scenario when the alpha-flow is not smooth is at self-occlusions of the foreground which are close to the edge of the silhouette, e.g. tip of noise in Fig. 1(a). Note that in contrast to optical flow, which tries to model the projection of the true motion field, alpha-flow does not directly have a physically meaningful explanation. We discuss this in more detail in the next section. Nevertheless, we show experimentally that alpha-flow is a very useful concept for achieving visually pleasing video matting results with as few user interactions as possible.

Concerning the user input, we assume that there are (at least) two keyframes given, which correspond to the first and last frame of the sequence. Long sequences can also be fragmented by several keyframes into parts. The keyframes are assumed to be "full", i.e. contain the alpha channel for the respective frames. This is done by using an image matting method such as [3]. Usage of two keyframes has the main benefit of preventing the alpha-matte from drifting. This is also further consolidated by the fact that we use a bidirectional optical flow which handles occlusions better. We also allow the user to provide "sparse" keyframes, which consist of "unknown" pixels and a few strokes of fore- and/or background.

1.1 Related Work

In the following, we group methods by their usage of optical flow. Fig. 2 gives a schematic overview on how different methods perform when a temporal tracking error is present. Note that some of the selected methods perform binary segmentation and not matting. These algorithms are still relevant for the comparison, because they deal with the similar problem of label propagation through the sequence and because border matting [4] or guided filter [5] can be applied to the binary mask.

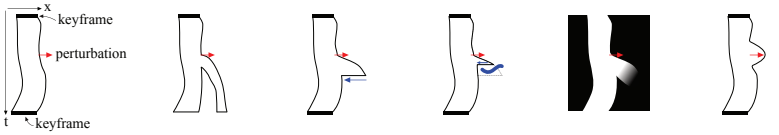


Fig. 2. Illustrating the impact of a tracking (temporal) error on the segmentation for different methods. (a) Temporal slice of keyframes (black bars) and the border of the ground truth segmentation. Red arrow shows a tracking error ("perturbation"). **(b)** Forward tracking methods use only one keyframe and continue tracking the erroneous region forward. **(c)** Video-volume methods (usually using graph cut) often produce flicker (blue arrow), when they suddenly switch back to the correct region. **(d)** When the user applies an additional background brush stroke (schematically shown in blue), the flicker gets smaller (blue arrow) but is still present. The only solution would be to precisely place the brush stroke at the point of perturbation. **(e)** "Spilling" (or "ghosting") artefact that happens in algorithms which treat video as a 3D volume. **(f)** The "bulge" artefact of an automated rotoscoping method.

We first discuss methods that do not use optical flow. Examples are Video Cutout [6] and Video Object Cut-and-Paste [7]. These methods perform an oversegmentation of the source frames and then construct a 3D graph of the obtained segments. Temporal connections are based on colors of the segments only, which leads to the following problem: the boundary of the output mask often snaps to different edges in consecutive frames (especially in case of occlusion). This is hard to fix by adding fore- or background seeds (the user often has to cover large amount of problematic 2D segments). This effect is demonstrated in Fig. 2 (c, d) and also Fig. 3 (a, b), using our own implementation of [7]. Such "flickering" artifacts are known to be visually disturbing, and easily give away the fact that the video was composited.

A different variant of 3D volume-based techniques replace the hard segmentation by a soft one. For example, the methods [8, 9] use a 6-connected 3D volume to propagate the alpha-matte. This usually leads to an opposite problem of flickering, called "spilling". Instead of accurately following the object contour, the matte produces ghosting artifacts of the earlier object position that dissolves in time very slowly. Fig. 2 (e) and Fig. 3 (d) show examples, where we approximated the 3D Laplacian using 2D scanline slices[4], which is acceptable here due to dominantly horizontal motion and vertical lines (this was confirmed by similar results using true 3D Laplacian on much smaller resolution).

To overcome the flickering or spilling artifacts it is sensible to track the actual object position rather than softly constrain alpha between frames. This is often done by assuming that the alpha matte obeys the same motion field as the image itself. There are several methods which use this strategy. All of them precompute optical flow for the whole sequence. They differ by the way they relate the matte and the flow. Opacity Propagation algorithm [10] uses a slanted, flow-aligned 3D window. The algorithm [11] re-weights the 3D kernel according

[4] We used our own implementation, which is a generalization of [8, 9].

to the optical flow. However this is only applicable if the object undergoes a slow motion, i.e. when the motion magnitude does not significantly exceed the kernel size, otherwise occlusions still lead to spilling. Another possibility is to forward propagate the binary segmentation or trimap from the keyframe, e.g. [12] or use the segmentation result as a trimap (soft constraint) for the next frame, as in Video SnapCut method [2]. The problem of forward propagation techniques is shown in Fig. 2 (b). We compare to [2] in our experimental section.

Some methods rely mostly on motion to produce the segmentation. An example is the T-junctions method [13] where the motion is used as the main cue for segmentation. The method [14] formulates the labeling problem on top of the point tracks to propagate the segmentation. Particle Video [15] and Video Mesh [16] algorithms also fall into this category. We compare to [15] in our experiments.

Finally, we discuss those methods that are closest to ours, that is the automated rotoscoping method [17] and the motion-coherent tracking method [18]. These methods produce a hard segmentation only. Both methods relax to some extent the assumption of an image motion model and use a hand-crafted motion model for the mask itself. The rotoscoping method [17] analyzes motion in a narrow band along the contour inside the object. This helps to avoid occlusions. The main feature, which is an inspiration for our work, is that the 3D tube, swept by the contour, is constrained at both ends via keyframes. Therefore, tracking errors will cause perturbations in the tube, but it will still retain the average direction of the motion defined by the keyframes (as well as by reliable frames, i.e. simple frames will guide the difficult ones). We call this artifact a "bulge", which is shown in Fig. 2 (f). The main drawback of this approach is that the contours are limited in their expressive power, i.e. they cannot represent well hair, fuzzy areas and small details. Another difficulty is that the user must maintain the correct number and the correct correspondences for control points, when creating the keyframes.

The tracking method [18] solves jointly for the binary mask and the optical flow. This is similar to our goal. However, one major difference to our work is that they formulate the joint optimization via a large multi-label graph construction. Hence, due to computational limitations, this method can only process the video sequence in a forward-tracking manner, several frames at a time. The problem of such an approach is that the segmentation can

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

Fig. 3. Comparison of video-volume based techniques. (a) Result of [7] for a volume with 20 frames, where first and last keyframe is given. **(b)** A 3-frame volume with first and last keyframe given. A "flickering" artifact of [7] (zoom-in shown), that is a sudden change in shape (red line marks the true boundary). **(c)** Fragment of source frame for (b). Note, the black picture frame on the wall creates a strong edge which misleads the segmentation in (a, b). **(d)** "Spilling" artifact in 3D volume matting. Here we approximated the 3D by 2D scanline slices. Our result shown in Fig. 8 (b) is considerably better.

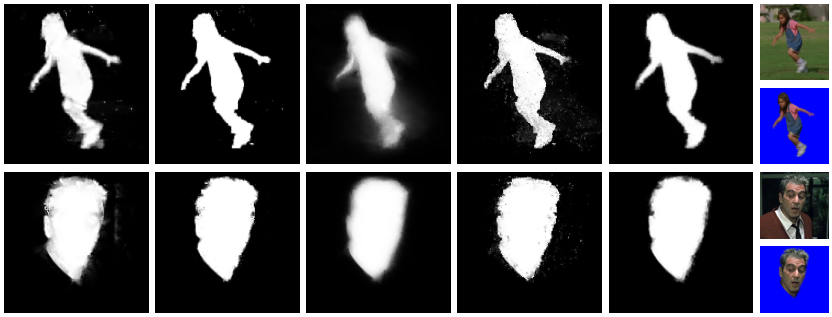


Fig. 4. Effect of different energy terms and optimization strategies. Sequences: top – *girl*, bottom – *godfather*. **(a)** Alpha matting which is regularized by standard RGB optical flow. This is the baseline approach for many methods. In our method this is the first iteration, stopped after step 3. **(b)** Applying the Guided filter to the result in (a). This means we stopped our approach after the first iteration in step 4. Note the various isolated foreground pixels, which cause temporal visual artifacts (see supp. material) **(c)** Final result using alpha-flow with occlusion handling switched off (i.e. all motion vectors are used). **(d)** Using thresholded binary alpha when computing alpha-flow. The final result is worse than (e), which means that continuous alpha is important. **(e)** Final result using alpha-flow. **(f)** Original frame and composite of (e) onto constant color background.

drift, as illustrated in Fig. 2 (b). We will confirm this problem in the experimental section. In our approach drift is avoided due to the use of two keyframes, at the beginning and at the end.

We conclude by summarizing our insights on how the various methods react to the unavoidable tracking errors, as shown in Fig. 2. The figure suggests that (f), the automated rotoscoping method [17], behaves in the most graceful way. Let us point out three important observations for this case (the "bulge" artifact):

- The bulge is smooth. Thus it does not annoy the viewer, and helps to hide the fact that the video was altered.
- The bulge reaches its maximum at the point of the tracking error. Hence it is much easier to fix (the user intuitively fixes the largest errors first), compare to Fig. 2 (c, d) where the maximum is far from the actual point of error.
- The bulge has also the smallest magnitude across all the methods, because it is only affected by the tracking error itself without drifting further.

We would like to achieve similar stability for the video-matting problem, by also using two keyframes at both ends of the sequence.

2 Our approach

Our goal is to optimize jointly the 2D alpha-flow V and the opacity α value for all pixels in the video volume. This amounts to minimizing the following objective function:

$$E(V,\alpha) = \sum_{x,y,t} w_\alpha(\alpha(x,\, y,\, t) - \alpha(x + V_x,\, y + V_y,\, t + 1))^2 + \\ + \lambda(|\nabla V_x|^2 + |\nabla V_y|^2) + J(\alpha) \quad (2)$$

subject to user constraints on alpha (keyframes), where the weighting parameters are typically set to $w_\alpha = 1000$ and $\lambda = 0.02$. Note that in our interactive system the weighting can be adjusted manually by the user to reflect the complexity of the video matting task, i.e. complex motion and simple alpha matte versus simple motion and complex alpha matte.

The first term expresses the alpha constancy assumption, similarly to color constancy assumption used in optical flow. The second term is a classical pairwise term that forces the alpha-flow to be spatially smooth. If only these two terms were to be minimized, the solution would connect the keyframes with a smooth, morphing-like transition. The term which prevents this effect is the last term, $J(\alpha)$, which is the only term that is conditioned on the image. It forces the alpha matte in each frame to obey the so-called matting Laplacian, see details in later section 2.3 and eq. (8).

There are two additional constraints which we enforce during the optimization, but did not include in the overall energy (2). They correspond to a *sparsity constraint on* α, and a robustness constraint of the alpha flow with respect to *occlusions*. We did not include them in the overall energy for two reasons: a) the overall objective function would be rather complex and non-intuitive, and b) due to constraints on computational speed and memory we use fast techniques, such as filtering, which only approximately optimize the underlying energy. Note, to use filtering techniques for e.g. optical flow computation is a common, and practically useful technique [19].

The overall algorithm of our video matting system is as follows:

1. Initialize alpha-flow with RGB optical flow (Sec. 2.1)
2. Occlusion handling: Construct temporal superpixels (Sec. 2.2)
3. Apply Closed Form Matting [20] to temporal superpixels (Sec. 2.3)
4. Apply Guided Filter to increase sparsity in alpha [5] (Sec. 2.4)
5. Compute the alpha-flow (Sec. 2.1)
6. Go to step 2

We terminate the algorithm at step 4 (or 3) after a fixed number of iterations, typically 3. As can be seen, our main strategy is to iterate between the computation of alpha (step 3 and 4) and alpha-flow (step 5). Note that step 1 initializes the alpha-flow with the optical flow. To do this we replace in eq. 2 the alpha-constancy assumption with the standard color-constancy assumption, that is $|I(x,\, y,\, t) - I(x + V_x,\, y + V_y,\, t + 1)|^2$.

Fig. 4 gives an overview on the importance of the different terms in the energy. It also illustrates the effect of using optical flow instead of alpha-flow, which corresponds to terminating our algorithm after one iteration at step 3 or 4.

Let us analyze, again, the difference between optical flow and alpha flow. As discussed above, see fig. 1, alpha-flow is typically more smooth than optical

flow. This is obvious for pixels which truly belong to fore- or background, but does it also hold for fractional alpha? For many videos the answer is yes. To see this let us analyze why fractional alpha values occur in practice: (a) semi-transparent objects, (b) spatial discretization artifacts (anti-aliasing), (c) temporal discretization artifacts (motion blur), (d) camera's Point Spread Function (PSF) caused by optical effects. We argue that for many videos the practically dominating effect is (d), see also [21]. Note that this is an effect of the camera and not of the motion in the physical world. Hence, if we correctly estimate the motion of the underlying binary (infinite-resolution) silhouette, before applying camera effects, then the alpha-constancy assumption is perfectly valid for all pixels. An extreme example is a spinning vase, captured with a stationary, out-of focus camera, where the binary silhouette, before applying camera effects, is identical in all images. Note, that cases (a–c) are not modeled explicitly with our flow field, but these cases are considered in our per frame matting constraint.

Note that we also experimented with an extension of eq. 2, where we included an additional color-constancy constraint. For some video matting tasks this gave minor benefits for the regularization of the flow field, however, for simplicity we omitted it in our overall system.

2.1 Flow computation

For a fixed α matte, equation (2) becomes the standard cost function for flow estimation, consisting of a data and smoothness term:

$$E(V) = \sum_{x,y,t} (\alpha(x,\, y,\, t) - \alpha(x + V_x,\, y + V_y,\, t + 1))^2 + \lambda(|\nabla V_x|^2 + |\nabla V_y|^2), \quad (3)$$

where λ is smoothness factor.

Before describing our method, let us mention that we started by utilizing a state-of-the-art technique [22] for alpha-flow computation. We have seen experimentally that sub-pixel accuracy is, however, not crucial and our requirements on the precision of alpha-flow can be relaxed. To be precise, to ensure temporal coherency of opacity values, the exact value of the flow does not matter as long as the flow vectors connect foreground to foreground and background to background. Because our algorithm iterates alpha estimation and alpha-flow estimation, the computational speed is more important then precision. To achieve decent processing speed we combined the existing approaches [23] and [24].

The quadratic decoupling technique proposed in [23] alternates "direct search" and "smoothing steps" with mathematically reasoned weights that lead to convergence to a reliable local optimum. Thus data and smoothness terms are decoupled and linked via a soft constraint. This allows for a separate optimization of these terms. The weight of the soft constraint is increased over several iterations to provide convergence. For the smoothing step we use a conjugate gradient solver with a limited number of iterations.

For the direct-search step (which optimizes the data term) we use ideas from the PatchMatch algorithm [24]. Instead of trying all possible displacements in range $[-M, M] \times [-M, M]$, where M is the maximum allowed motion, we generate a random motion vector for each pixel from the same range. Then during

a two-pass procedure the values of data term plus coupling term are compared for the neighboring pixels, and the winning vector is stored.

Compared to algorithms with very high accuracy, e.g. mexOpticalFlow library [22], our algorithm takes about 2 seconds per frame, while mexOpticalFlow requires 1-3 minutes per frame. While the flow becomes more accurate with mexOpticalFlow, we did not observe any significant improvements in the final matting result.

As mentioned above, in the first iteration of our algorithm we compute standard RGB optical flow by replacing the alpha-constancy constraints with the color-constancy constraint. Note that pixels in semitransparent areas may not have a distinct optical flow. Hence, in the first iteration of our algorithm they will most likely be matched to pixels of the same color. We assume that these matched pixels often have the same alpha value (similar assumption is made in [10] when they match image windows using SSD metric). Note, this assumption is not always true. For instance, pixels may move to a different background and change color, or pixels of different colors over different backgrounds may have the same resulting color. Luckily, in real-world videos colors are typically smooth enough for this condition to hold locally. This provides enough information for the matting process to work properly.

Finally, for better occlusion handling, which will be discussed later, we compute both a forward and backward flow.

2.2 Occlusion handling

The 3D video volume is overconstrained, each pixel has spatial connections via the matting Laplacian and at least two outgoing temporal connections (forward and backward flow vectors).[5] Some of the temporal constraints are outliers, due to occlusions. Since speed is important for us, we do not want to compute an "occlusion-aware" flow. Instead, we consider the occlusion handling task as a general outlier-detection problem, defined via a set of constraints. Note that we always keep the spatial constraints, but want to drop some of the temporal ones, i.e. the ones that are not likely to be satisfied.

Note that matting algorithm [20], which we use, is capable of producing high-quality image mattes from very few, sparse strokes. Thus it will be enough to have a few temporal connections, provided that they are reliable ones and non-contradicting with each other.

Our approach is inspired by [25]. They state that for constraint dropping a deterministic approach may be preferable to a probabilistic one. The obvious criterion for which flow vectors should be dropped is a large difference in alpha (or color). Two-frame flow methods usually compare this difference with a fixed threshold [26] or use a histogram [27]. As this may lead to inconsistencies between frames, we would like to exploit temporal connectivity of temporal constraints, assuming that occlusions are sparse in the temporal domain.

[5] More temporal connections may occur from incoming vectors of previous/next frames if they happen to go into the same pixel.

Since our temporal constraints are of the form

$$(\alpha(x,\, y,\, t) - \alpha(x + V_x,\, y + V_y,\, t + 1))^2, \tag{4}$$

pruning them with a binary occlusion mask ρ, gives the following set of constraints

$$(1 - \rho)(\alpha(x,\, y,\, t) - \alpha(x + V_x,\, y + V_y,\, t + 1))^2 \tag{5}$$

with an additional sparsity constraint for ρ. Alpha constancy constraints can be viewed as part of a Gibbs free energy (they are log-likelihoods). Hence, we can view ρ as a parameter of this distribution, thus we are interested in minimizing its entropy. We now describe an MDL-based formulation for the temporal sparsity on ρ.

We want to segment the video volume into temporal chains (superpixels) which are short trajectories constructed from consecutive flow vectors. We treat forward and backward vectors as undirected edges between pixels of adjacent frames. We specifically want these trajectories to start and end at occlusions.

We explain our algorithm in terms of color, which is used on the first iteration of our algorithm. On consecutive iterations we use alpha. It is assumed that changes in color(alpha), on a non-occluded motion path, are explained by Gaussian distributions.

For each temporal chain we store the mean color μ, starting frame t_1 and length τ. For each pixel we store the color relative to the mean color of chain, spatial coordinates of the pixel $(x(t),\, y(t))$. See technical report [28] for details.

Finding the exact optimum is computationally hard, so we use an approximate solution. We perform a greedy optimization that at each step joins a pair of superpixels along a motion vector that gives maximum decrease in the total cost. The gain (cost decrease) is computed for each flow vector and updated for vectors adjacent to superpixel end-pixels, when merging is performed. Vectors are stored in a priority queue ordered by the gain value.

The average trajectory length for a given video-sequence is usually 2-4 frames. Many trajectories consist of just one pixel, but this is compensated by rather long reliable trajectories, some of which run through the whole sequence.

Typically, long superpixels are constructed in areas where $\alpha \in \{0, 1\}$, while areas with $0 < \alpha < 1$ consist mostly of individual pixels. Note that trajectories which are only one pixel long do not create any constraints on alpha. Thus partitioning of the video-volume into superpixels can be viewed as a relaxed trimap constraint (see trajectory lengths plotted in figure 5).

After segmenting the video volume into many temporal chains, we remove constraints (motion vectors) that connect different chains. Finally we observe that the remaining constraints are usually

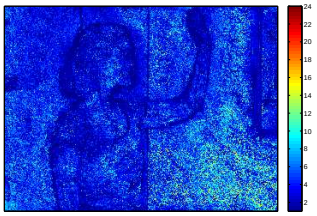


Fig. 5. Trajectory lengths in frame 15 of the *amira* sequence. Reliable trajectories inside/outside the object are long, trajectories near boundary are very short. This can be thought of as a relaxed trimap constraint with variable confidence.

satisfied almost exactly, due to the usually high weight w_α in (2). This allows us to save memory and gain computation speed by replacing them with hard constraints:

$$\alpha(x, y, t) = \alpha(x + V_x, y + V_y, t + 1) \tag{6}$$

in pixels where $\rho(x, y, t) = 0$ (no occlusion).

Considering the segmentation of the video volume into temporal superpixels, we introduce the vector z for alpha values of superpixels: $\alpha = Mz$, where M is a sparse matrix of 0's and 1's that maps superpixels to pixels. We minimize the objective function (8) for z, thus enforcing the equation (6) along motion trajectories (superpixels). The number of variables in z is usually 2-4 times less than the number of pixels since on average the trajectory length is around 2-4 frames. The spatio-temporal Laplacian matrix for superpixels is then expressed as follows:

$$\hat{L} = M^T\{L_t\}M, \qquad J(z) = z^T\hat{L}z, \tag{7}$$

where $\{L_t\}$ denotes a block-diagonal matrix of all 2D Laplacians for $t = 1...T$ (assuming that linear indices of components of the vector α are ordered by t).

Boundary constraints (keyframes) are applied to superpixels. In some cases long trajectories may receive two different labels from two keyframes. This is resolved by breaking the superpixel into two, at the point of maximum color (alpha) difference along it.

Our temporal superpixels differ from other representations. They are dense, as opposed to sparse and semi-dense point tracks ([15]), which often do not cover important parts a of video frame. Unlike 2D and 3D superpixels ([29, 6]), our trajectories cover exactly one pixel in each frame, so they do not join pixels spatially, which may pose problems when foreground and background colors are similar. Thus we defer the decision of spatial affinity to the matting stage.

2.3 Matting

This section addresses the task of computing alpha given alpha-flow. We use the closed-form matting algorithm [20]. The objective function for alpha is

$$J(\alpha) = \sum_{t=1}^{T} \alpha_t^T L_t \alpha_t, \tag{8}$$

where L_t is Levin's Laplacian matrix for frame t, α is a vector of alpha values for the whole video volume and α_t is a vector of alpha values for frame t. This provides spatial connectivity in each frame. For temporal connectivity we use the computed flow V. So the full energy optimized in this step becomes:

$$E(\alpha) = \sum_{x,y,t} w_\alpha(\alpha(x,\, y,\, t) - \alpha(x + V_x,\, y + V_y,\, t + 1))^2 + J(\alpha). \tag{9}$$

Unlike 3D volume methods, we do not violate the color-line model of the matting Laplacian. We apply the color-line model only in 2D, in each frame, while temporal connections are based on the alpha-flow principle.

2.4 Sparsity in alpha

Often superpixels receive conflicting labeling information from their neighbors (due to heavy occlusions or illumination and appearance changes). The quadratic nature of the matting Laplacian leads to intermediate alpha values (≈ 0.5) being assigned to large image regions. The standard approach to prevent this is to use a robust norm (e.g. L_1-norm) or an additional energy term that pushes alpha to zero or one values and thus increase the sparsity of the solution. It was observed that a similar result can be achieved with a very fast technique, which simply filters the image with the so-called guided filter [5].

We apply the guided filter to the alpha values of superpixels using the whole 3D Laplacian $\hat{L}$ (7), i.e. using the equation (12) in [5], which shows how to compute the guided filter operator from the Laplacian matrix. As a result, large semi-transparent artifacts are filtered out.

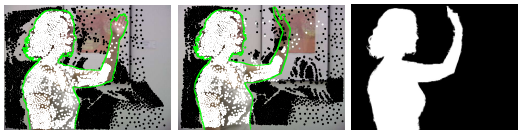
We perform guided filtering at each iteration of our algorithm. Note, if want to extract truly semi-transparent objects (e.g. smoke) then the guided filter, step 4, can be omitted in the last iteration of our method.

3 Results and comparison

We have selected several state-of-the-art algorithms for comparison with our approach. The first two methods are video-volume based with hard [7] or soft segmentation [8, 9], both of them softly constraining opacity between frames. The third technique is Particle Video [15], which is an example of a technique that can be used to propagate a trimap between frames via optical flow. The forth competitor is SnapCut [2] that uses optical flow to propagate a set of boundary classifiers. The last one is [18], which is most similar to our approach. In contrast to our work they utilize a hard segmentation for the tracked object. More importantly, they can only process a few frames at a time, due to computational complexity, whereas we process the full video volume.

Video-volume based techniques suffers from frequent "flickering" or "spilling" artefacts as shown on the the *amira-queen* sequence in Fig.3. In contrast, our result in Fig. 8 (b) is considerably superior.

The main problematic areas for Particle Video method are textureless regions, as illustrated in Fig. 6. The result is imperfect after 4 frames, and this becomes worse as we move further away from the keyframe. For this reason we do not want to rely on sparse set of long point tracks as [14], but rather try to construct dense set of short tracks (temporal superpixels) for which we are more certain that they



(a) (b) (c)

Fig. 6. Result of Particle Video [15]. (a) In the keyframe 30 we assign the true alpha values to the particle trajectories. The green contour shows the ground truth object boundary. **(b)** The same particles tracked backward to frame 26. Foreground particles (white) from the hand region are wrongly spilled into the background wall, as well as background particles (black) propagate wrongly into the arm. **(c)** Result of our algorithm.

(a) (b) (c) (d) (e) (f)

Fig. 7. Comparison. (a) Frame 21 of *Penguin* sequence from SegTrack database (top) and result by Tsai et al. [18] (bottom). **(b)** Our result (top) and ground truth from [18] (bottom). **(c)** Frames 13, 14 of *girl* sequence (first frame 1, last frame 21). **(d)** Result of [18] – a sudden change occurs between these frames. **(e)** Video SnapCut [2]. **(f)** Our result.

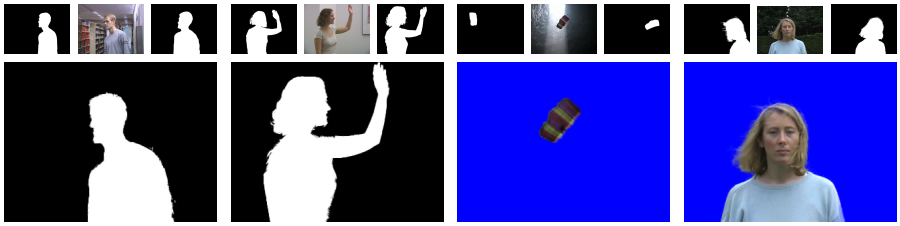
have the same label, i.e. alpha value. By recomputing the alpha-flow for several iterations we achieve stable motion in such textureless regions, driven by the nearby textured regions and user strokes.

The main disadvantage of Video SnapCut [2][6] is that it processes frames sequentially and often produces artifacts at occlusion boundaries, see Fig. 7. The only possibility to reliably correct the artifacts is to find the frame in which it started to occur. In our method corrections are possible by adding strokes at arbitrary frames, because we process the whole video volume.

The sequential processing of frames in [18] can also lead to sudden changes in opacity between consecutive frames for both slow-moving and fast-moving objects, examples are shown in Fig. 7. In comparison, our approach is more stable for slow-moving objects, even if they are not in good contrast with the background and there is very little motion. For fast-moving objects our algorithm also provides good refinement capabilities, preventing the errors from spreading to other frames.

We also addressed the interesting question of whether it is sufficient to first perform binary video segmentation and then apply matting only on the segmentation boundary. We believe that our approach has two main advantages. The key advantage is that dealing with fractional alpha values makes the alpha flow computation more stable. To illustrate this, we run our algorithm but change the guided filter settings in step 4 such that the result is a binary segmentation mask. Fig. 4 (d) shows that the result is worse, compared to (e). If we add Gaussian blur to the binary matte in each iteration the result gets better, but still not as good as solving for fractional alpha with the use of Laplacian, see supp. material. Secondly, if matting is applied as a postprocessing step then it is not clear how to choose the width of the video trimap. Furthermore, frame-by-frame matting may produce temporally inconsistent results, which, to some degree, has been addressed in recent work [30].

[6] Implemented in Adobe After Effects CS5 as RotoBrush tool.



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

Fig. 8. Results of our approach. Top row, for each test sequence: first keyframe, frame in the middle of the sequence, last keyframe. Bottom row: matte or composite corresponding to the middle frame. Besides the shown keyframes only a few additional strokes were used (see sup. material). Sequences: **(a)** *adam-lib-walk* (30 frames, from [17]), **(b)** *amira-queen* (31 frames), from [17], **(c)** *parachute* (51 frames, from [18]), **(d)** *maud* (61 frames, a part from [13]).

Several results for our approach are given in Fig. 8 (see also supplementary material). In our experiments we used two full keyframes for each sequence (first and last frame) and optional strokes to fix obvious problems (we tried to keep the number of strokes to a minimum). We show all the user interactions for each test sequence in the supplementary video.

4 Conclusion

We have proposed a new video matting algorithm that introduces ideas from rotoscoping into an optical flow based approach. Unlike previous techniques, our algorithm replaces optical flow in RGB by alpha-flow, which is the flow of the latent opacity channel. We then solve jointly for continuous alpha and alpha-flow. This reduces flickering artifacts, which is a common problem of most other techniques. By utilizing the first and last frame as keyframes, we are able to prevent the opacity mask from drifting. This is supported by the bi-directional flow estimation that handles occlusions reliably.

References

1. Rav-Acha, A., Kohli, P., Rother, C., Fitzgibbon, A.: Unwrap mosaics: A new representation for video editing. In: SIGGRAPH (2008) 17:1–17:11
2. Bai, X., Wang, J., Simons, D., Sapiro, G.: Video SnapCut: robust video object cutout using localized classifiers. In: SIGGRAPH. (2009) 1–11
3. Wang, J., Cohen, M.F.: Optimized color sampling for robust matting. Computer Vision and Pattern Recognition (2007) 1–8
4. Rother, C., Kolmogorov, V., Blake, A.: "GrabCut": interactive foreground extraction using iterated graph cuts. ACM Trans. Graph. **23** (2004) 309–314
5. He, K., Sun, J., Tang, X.: Guided image filtering. In: ECCV. (2010) 1–14
6. Wang, J., Bhat, P., Colburn, R.A., Agrawala, M., Cohen, M.F.: Interactive video cutout. In: SIGGRAPH. (2005) 585–594
7. Li, Y., Sun, J., Shum, H.Y.: Video object cut and paste. In: SIGGRAPH. (2005) 595–600
8. Bai, X., Sapiro, G.: Geodesic matting: A framework for fast interactive image and video segmentation and matting. IJVC **82** (2009) 113–132

9. Ding, Z., Chen, H., Guan, Y., Chen, W., Peng, Q.: GPU accelerated interactive space-time video matting. In: Computer Graphics International. (2010)
10. Tang, Z., Miao, Z., Wan, Y., Zhang, D.: Video matting via opacity propagation. The Visual Computer (2011) 1–15
11. Lee, S.Y., Yoon, J.C., Lee, I.K.: Temporally coherent video matting. Graphical Models **72** (2010) 25–33
12. Chuang, Y.Y., Agarwala, A., Curless, B., Salesin, D.H., Szeliski, R.: Video matting of complex scenes. SIGGRAPH **21** (2002) 243–248
13. Apostoloff, N.E., Fitzgibbon, A.W.: Automatic video segmentation using spatiotemporal T-junctions. In: BMVC. (2006)
14. Lezama, J., Alahari, K., Sivic, J., Laptev, I.: Track to the future: Spatio-temporal video segmentation with long-range motion cues. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2011)
15. Sand, P., Teller, S.J.: Particle video: Long-range motion estimation using point trajectories. International Journal of Computer Vision **80** (2008) 72–91
16. Chen, J., Paris, S., Wang, J., Cohen, M., Cohen, M., Durand, F.: The video mesh: A data structure for image-based video editing. Artificial Intelligence (2009)
17. Agarwala, A., Hertzmann, A., Salesin, D.H., Seitz, S.M.: Keyframe-based tracking for rotoscoping and animation. In: SIGGRAPH. (2004) 584–591
18. Tsai, D., Flagg, M., M.Rehg, J.: Motion coherent tracking with multi-label MRF optimization. BMVC (2010)
19. Sun, D., Roth, S., Black, M.J.: Secrets of optical flow estimation and their principles. In: CVPR. (2010) 2432–2439
20. Levin, A., Lischinski, D., Weiss, Y.: A closed form solution to natural image matting. In: IEEE CVPR. Volume 1. (2006) 61–68
21. Rhemann, C., Rother, C., Kohli, P., Gelautz, M.: A Spatially Varying PSF-based Prior for Alpha Matting. In: IEEE Conference on Computer Vision and Pattern Recognition. (2010)
22. Liu, C.: Beyond pixels: Exploring new representations and applications for motion analysis. Doctoral Thesis. Massachusetts Institute of Technology (2009)
23. Steinbrucker, F., Pock, T., Cremers, D.: Large displacement optical flow computation without warping. In: ICCV. (2009) 1609–1614
24. Barnes, C., Shechtman, E., Finkelstein, A., Goldman, D.B.: PatchMatch: A randomized correspondence algorithm for structural image editing. In: SIGGRAPH. (2009)
25. Agrawal, A., Raskar, R., Chellappa, R.: What is the range of surface reconstructions from a gradient field? In: Proceedings of the 9th European conference on Computer Vision - Volume Part I. ECCV'06 (2006) 578–591
26. Xiao, J., Cheng, H., Sawhney, H., Rao, C., Isnardi, M., Corporation, S.: Bilateral filtering-based optical flow estimation with occlusion detection. In: ECCV, volume I. (2006) 211–224
27. Strecha, C., Fransens, R., Gool, L.V.: A probabilistic approach to large displacement. In: In proc of ECCV Workshop SMVP. (2004) 71–82
28. Sindeev, M., Konushin, A., Rother, C.: Alpha-flow for video matting. Technical Report (2012)
29. Grundmann, M., Kwatra, V., Han, M., Essa, I.: Efficient hierarchical graph based video segmentation. IEEE CVPR (2010)
30. Bai, X., Wang, J., Simons, D.: Towards temporally-coherent video matting. In: Proceedings of the 5th international conference on Computer vision/computer graphics collaboration techniques. MIRAGE'11, Springer-Verlag (2011) 63–74