JumpCut: Non-Successive Mask Transfer and Interpolation for Video Cutout

Qingnan Fan[1] Fan Zhong[1] Dani Lischinski[2] Daniel Cohen-Or[3] Baoquan Chen[1]
[1]*Shandong University* [2]*The Hebrew University of Jerusalem* [3]*Tel Aviv University*

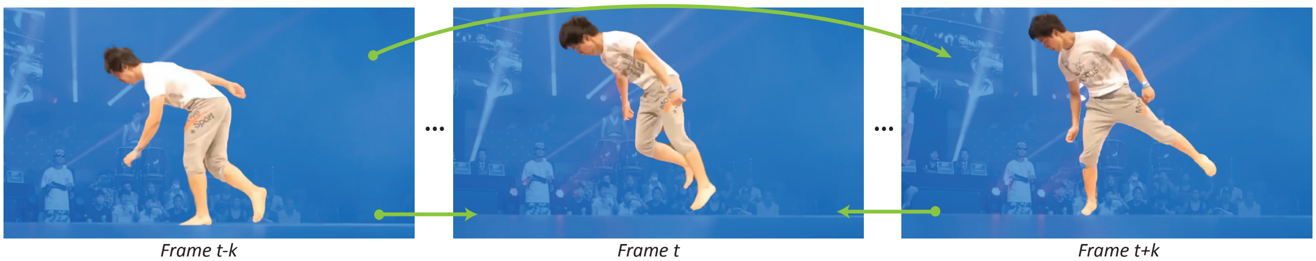


Figure 1: *Given a video frame* $(t-k)$ *with a* pre-segmented *foreground subject, our new mask transfer approach is able to predict the mask at non-successive unsegmented frames (e.g.,* $t+k$*), in a more accurate manner than by sequential frame-to-frame propagation. We are then able to accurately estimate the mask at intermediate frames, such as t, using bi-directional mask transfer, referred to as mask interpolation. Note that no manual corrections were applied to the segmentations shown in frames* $t+k$ *and* t*. Please view the companion video!*

Abstract

We introduce JumpCut, a new mask transfer and interpolation method for interactive video cutout. Given a source frame for which a foreground mask is already available, we compute an estimate of the foreground mask at another, typically non-successive, target frame. Observing that the background and foreground regions typically exhibit different motions, we leverage these differences by computing two separate nearest-neighbor fields (split-NNF) from the target to the source frame. These NNFs are then used to jointly predict a coherent labeling of the pixels in the target frame. The same split-NNF is also used to aid a novel edge classifier in detecting silhouette edges (S-edges) that separate the foreground from the background. A modified level set method is then applied to produce a clean mask, based on the pixel labels and the S-edges computed by the previous two steps. The resulting mask transfer method may also be used for coherently interpolating the foreground masks between two distant source frames. Our results demonstrate that the proposed method is significantly more accurate than the existing state-of-the-art on a wide variety of video sequences. Thus, it reduces the required amount of user effort, and provides a basis for an effective interactive video object cutout tool.

CR Categories: I.4.6 [Image Processing and Computer Vision]: Segmentation—pixel classification;

Keywords: video segmentation, foreground extraction, object cutout

Project webpage: http://irc.cs.sdu.edu.cn/JumpCut/

1 Introduction

Video foreground extraction is routinely used in TV and movie production for compositing visual elements from a variety of different sources onto a new video background. The challenges involved in accurately separating complex dynamic objects from the background in natural videos, as well as the need for creative control, render automatic methods impractical for the task. Thus, over the past decade, considerable research efforts focused on development of effective interactive tools. However, despite impressive progress in interactive foreground extraction [Bai et al. 2009; Zhong et al. 2012], the task still requires considerable user effort and expertise.

Specifically, state-of-the-art methods are still challenged by color and/or texture ambiguities between foreground and background regions, by low contrast edges separating the foreground elements from the background, and by fast foreground motion, which typically involves non-rigid deformations and drastic changes in the foreground region's topology.

In this work, we describe a new tool for interactive foreground extraction that specifically targets large and non-rigid foreground motions. We address the problem of interactive *video cutout*, where the task is to extract a binary mask labeling the pixels of each frame as foreground or background. A soft-edged alpha matte can be obtained, if needed, by using our mask as a basis for a trimap [Chuang et al. 2002].

The basic computational element that we focus on in this work is *mask transfer*: given a correctly segmented source frame, the goal is to compute the foreground mask in another, non-successive target frame (Figure 1). In practice, the mask transfer distance is typically between 4 and 32 frames, depending on the speed of motion and the amount of foreground deformation present in the video. As we demonstrate in our results and companion video, the ability to perform such mask transfer accurately provides a basis for an effective interactive video cutout tool.

Current state-of-the-art interactive methods, e.g., [Bai et al. 2009; Zhong et al. 2012], rely on combinations of local and global classifiers. Bai et al. use local classifiers in square windows along the foreground boundary, for which distant propagation is challenging, while Zhong et al. use long directional windows, whose appropriate sizes and directions are difficult to determine. Recent results indicate that PatchMatch [Barnes et al. 2009], a fast method for computing the nearest neighbor field (NNF) between two images, is effective for handling large displacement optical flow [Bao et al. 2014]. PatchMatch finds correspondences based on differences between patches, simultaneously taking into account color and texture. It computes NNFs that are piecewise spatially coherent, but may contain distant correspondences due to its use of random search. However, the PatchMatch method sometimes gets stuck in local minima, especially in the presence of color and texture ambiguities.

Our basic insight in this work, is that the displacements in the background (BG) and the foreground (FG) regions typically exhibit different and uncorrelated behavior. Specifically, the BG motion is typically induced by camera motion with respect to the scene, while FG motion is independent of the camera, and often features larger displacements and non-rigid deformations. Thus, our approach attempts to explicitly account for such differences, by tracking the BG and FG regions using two separate NNFs (split-NNF). These separate NNFs are better localized, and are thus less susceptible to local minima.

A second key component of our approach is a new edge classification process that leverages our split-NNF. Specifically, each salient edge in the unsegmented target frame is classified into one of three classes: edges inside the background region (B-edges), inside the foreground (F-edges), and, most importantly, silhouette edges that separate the two regions (S-edges). The classification is accomplished by a non-parametric, supervised classifier, which uses the edges and the mask in the source frame as its training data.

To obtain the final, clean, mask for the target frame we apply a method based on Level Sets [Osher and Fedkiw 2003]. This final step is facilitated by the results of the previous two stages: the level set method is initialized with the pixelwise mask predicted by our split-NNF, and relies on the results of our edge classifier to snap to nearby silhouette edges, while smoothing the target mask contour.

The entire process outlined above results in a new mask transfer method that succeeds in propagating masks farther and more accurately than existing state-of-the-art methods. Armed with this improved non-consecutive mask transfer ability, we are also able to implement a new mask interpolation mechanism, where given two non-adjacent segmented frames we reconstruct the foreground mask in the intermediate frames, in a coherent and accurate manner.

2 Related Work

Video foreground extraction is routinely used in TV and movie production for combining visual elements from different sources into a single video stream. In a studio, foreground extraction may be accomplished using a constant color screen [Smith and Blinn 1996]. Extracting a dynamic foreground element from a natural video is a much more challenging problem, which has attracted significant research attention [Chuang et al. 2002; Agarwala et al. 2004; Li et al. 2005; Wang et al. 2005; Bai et al. 2009; Price et al. 2009; Tong et al. 2011; Zhong et al. 2012].

In their pioneering work, Chuang et al. [2002] combine bidirectional optical flow with background estimation to interpolate a trimap across a video volume. Bayesian matting [Chuang et al. 2001] is then applied to compute a foreground matte at each frame.

Agarwala et al. [2004] describe a keyframe-based system for *rotoscoping*: tracking the curve-based representation of a foreground contour in a video sequence. Through the use of spacetime optimization they propagate user constraints forward and backward in the sequence.

Li et al. [2005] formulate the foreground extraction as a 3D graphcut problem over the spatio-temporal video volume, further refining the results using 2D graphcuts inside tracked local windows. Wang et al. [2005] also operate on the 3D video volume, and provide a user interface for painting constraints directly in this volume. Tong et al. [2011] describe a more intuitive and efficient user interface for painting the constraints, based on local 3D graphcuts.

The *LIVEcut* system of Price et al. [2009] uses multiple cues with 2D graphcut optimization in order to propagate a selection forward, frame by frame. The *Video SnapCut* system of Bai et al. [2009] obtains the segmentation at each frame via a collaboration of a set of overlapping localized classifiers, each of which integrates multiple local features. This approach has been incorporated into Adobe AfterEffects, as the *Rotobrush* tool, and was later improved by a better integration of motion and color models [Bai et al. 2010]. Because these methods are based on statistics collected by a set of square local windows centered along the foreground object's boundary, they have difficulty handling the temporal discontinuities that arise due to large motions, or videos with inseparable statistics.

More recently, Zhong et al. [2012] advance the state-of-the-art further by introducing unbiased directional classifiers, designed to cope with larger foreground motions, while maintaining the advantages of local statistics. They also propose a new way of combining together multiple classifiers. However, for distant transfer it can be difficult to determine the appropriate window sizes, and sampling only four directions (0, 45, 90, 135 degrees) is not always enough.

Thus, despite impressive progress in interactive foreground extraction, it is still easy to find video sequences that challenge the existing state-of-the-art methods: videos with color and texture ambiguities between foreground and background regions, where the motion of the camera and/or the dynamic foreground object is significant.

In contrast to these previous methods, we base our mask propagation on nearest neighbor fields (NNFs) between frames, and use PatchMatch [Barnes et al. 2009] to compute them. PatchMatch is based on differences between patches, thus seeking similarity in both color and texture. Another advantage of PatchMatch, compared to parametric classifiers, is that it explicitly encourages spatially coherent NNFs. Nevertheless, it is still capable of capturing large displacements and non-rigid motion due to its use of random search. The effectiveness of PatchMatch for computing large displacement optical flow has recently been demonstrated [Chen et al. 2013; Bao et al. 2014]. Differently from these methods, our method explicitly accounts for the typically different motion exhibited by the background and the foreground regions. Specifically, we track these two regions using two separately computed NNFs, each of which is better localized, thereby reducing the susceptibility of PatchMatch to local minima.

In the computer vision community there has been significant research on segmentation tracking in video sequences (e.g., [Tsai et al. 2010; Faktor and Irani 2014; Ramakanth and Babu 2014]). Faktor and Irani [2014] also use the different motions of background and foreground to initialize their FG/BG classifier. Their method appears to represent the state-of-the-art in automatic (unsupervised) foreground extraction. However, the results of this and other automatic methods are not yet robust or precise enough to replace interactive cutout methods. This is shown in Section 4, where we compare our method with SeamSeg [Ramakanth and Babu 2014], currently the top performer on the SegTrack benchmark [Tsai et al. 2010].

The Level Set method has been extensively used in the context of automatic and interactive image segmentation. We refer the reader to Cremers et al. [2007] for an extensive review of the related literature. Below, we only highlight a few particularly related works.

Wang et al. [2014] describe an interactive system for image and video segmentation, geared towards mobile devices with a touch interface. Their approach is based on a level set framework, with an appearance model sampled in the vicinity of the touched point. Since this method relies on locally trained parametric models, it is limited in practice to images with non-overlapping statistics of foreground and background regions, and to videos with small displacements.

Liu and Yu [2012] describe an interactive image segmentation method, which employs a level set method that uses edge suppression to remove non-salient edges. Their level set has an edge field term, which is defined using the distance transform with respect to the salient edges. Our approach uses a level set with a similar term, however, our edge classifier is discriminatively trained using the segmented source frame from which we propagate the mask.

3 Method

3.1 Mask Transfer

Given a segmented source frame I_s with a binary mask M_s that indicates the foreground pixels in I_s, and a target frame I_t, our goal is to determine the target foreground mask M_t.

One approach to mask transfer is to classify each pixel in I_t as either background (BG) or foreground (FG) using various classifiers trained on the pair (I_s, M_s). Another alternative is to compute a dense correspondence φ matching each pixel $x \in I_t$ with $\varphi(x) \in I_s$, and setting $M_t(x) = M_s(\varphi(x))$. In this work, we choose the latter approach, since we believe it to be better suited for transfers where the source and target frames are far from each other in the video sequence, and both the BG and the FG regions may be strongly displaced. Large displacements can also occur between successive frames in video sequences with fast camera and/or FG object motion.

As mentioned earlier, we use a method based on PatchMatch [Barnes et al. 2009] in order to compute the nearest neighbor field (NNF) that serves as φ. We observe that the BG and FG regions in a sequence are typically displaced in a different manner. The BG displacement is often induced by the motion of the camera with respect to the (mostly static) scene, while the FG displacement is typically larger and less rigid. Thus, our idea is to leverage the two different displacement modes by computing a separate NNF for each of the two regions. This approach is illustrated in Figure 2.

Specifically, we start by aligning I_s and I_t with respect to the motion of foreground and background, respectively. The FG alignment is computed by template matching, where the masked foreground of I_s is the template, with the sum of squared differences inside the mask serving as the matching metric. Having found the optimal translation for the foreground region, we apply it to I_s to obtain the FG-aligned source I_s^f. To align the background, we perform feature matching between I_s and I_t using SURF [Bay et al. 2008], with outlier rejection via RANSAC. Next, we warp I_s using the rigid moving least squares (MLS) method [Schaefer et al. 2006] under the constraints of the matched feature points, resulting in a BG-aligned source I_s^b.

Given the two aligned sources $I_s^*, * \in \{f, b\}$, we then compute a pair of *localized* NNFs from I_t to I_s^*. Our method is based on PatchMatch [Barnes et al. 2009], but restricted to search only locally in order to achieve better accuracy and coherency. PatchMatch computes an NNF by interleaving coherent propagation and random search. Our localized PatchMatch differs from the original method in three aspects: First, since I_t and I_s^* are roughly aligned, it is not necessary to randomly initialize the NNF, and an initial NNF with a zero offset at each pixel is used instead. Second, in each iteration of the random search, it is not necessary to sample the entire image; instead, we randomly search only inside a local region of size $W \times W$ around each pixel, where W is set to one third of the image's diagonal. Although it might seem as a large window, one must remember that the random search in PatchMatch is not an exhaustive one: only a few random samples are drawn from that window. Third, to further encourage local matches, we use a weighted combination of the patch difference metric with the Euclidean distance. The corresponding weights are 10 and 3 for transfers of up to 4 frames, and 10 and 1.5 for more distant transfers.

The computation of two separately aligned NNFs, as described above, results in fewer errors in the NNF due to the use of local search and the modified distance term. Having computed the two NNFs, we now fuse them to a single NNF by selecting, for each pixel in I_t, the NNF offset that yields the smaller patch difference. The mask M_t is then predicted as described earlier in this section $(M_t(x) = M_s(\varphi(x)))$. The entire process is illustrated in Figure 2, which also shows the transferred mask after a final refinement with the level set method, as described in Section 3.4.

3.2 Mask Interpolation

The above technique can be extended to the case of mask interpolation, or bi-directional mask transfer, where we wish to predict the foreground mask at a frame I_t, given two already segmented frames, e.g., I_{t-k} and I_{t+k}. By leveraging two segmented source frames, bi-directional transfer is usually able to predict the target mask more accurately. Another advantage of this process is that given the foreground contour on both sides, we can use shape interpolation as a coherent shape prior. Our system performs interpolation hierarchically using binary subdivision; that is, given the segmented source frames I_{t-k} and I_{t+k}, we first interpolate to the midway frame I_t, and then recurse on each half until $k = 1$.

Our approach is to apply the technique described in the previous section twice, once to transfer the mask from I_{t-k} to I_t and once from I_{t+k} to I_t. Thus, we in fact make use of four separate NNFs. But, here, instead of relying on template matching to roughly align the segmented foreground with the target frame, we perform shape matching on the two segmented masks, and use shape interpolation to predict the location and shape of the intermediate frame mask. The predicted shape is then used to more accurately align each of the two segmented foreground regions with the target frame, resulting in more accurate foreground NNFs.

Shape matching is a well studied area, and we have initially considered using the state-of-the-art method of Ling and Jacobs [2007], which uses inner-distances, and was shown to match well articulated shapes with overall similarity. However, a drawback of this approach is that it uses dynamic programming for globally optimal correspondences, rendering it sensitive to occlusion and topology changes, which frequently occur in our context. Furthermore, in addition to the shape geometry, we would also like to leverage color and texture features, which allow us to find reliable correspondences without order constraints. Thus, we use the method described below.

To perform shape matching, we start by extracting and simplifying the two source mask contours[1]. Next, we compute correspondences between pairs of points on the two contours. The similarity of two contour points p_i and p_j is measured as a sum of three terms: shape difference d_{ij}^{shape}, color difference d_{ij}^{color}, and spatial distance d_{ij}^{space}. We use the mask patch (of size 31×31) centered

[1] We use the standard OpenCV functions findContours and approxPolyDP.

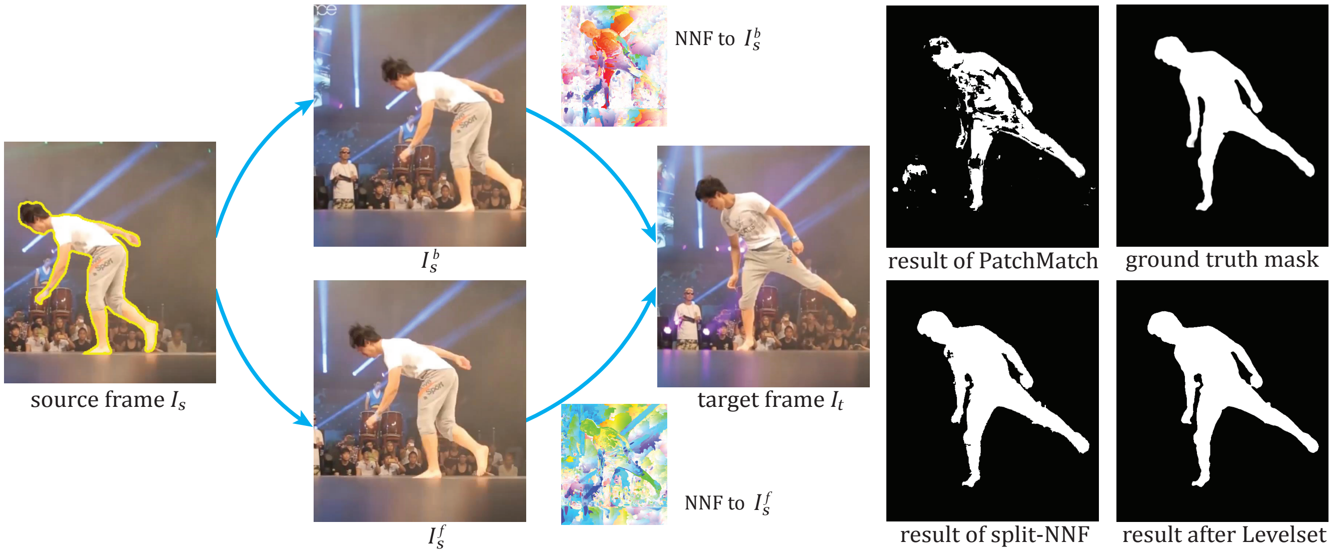


Figure 2: *Mask transfer using split localized NNFs. Two warped versions I_s^b and I_s^f are generated from the source frame, and a separate localized NNF is computed between the target frame and each of the two. Using the split-NNF results in a more accurate mask transfer than using a single NNF. The mask is further improved by level set refinement (Section 3.4).*

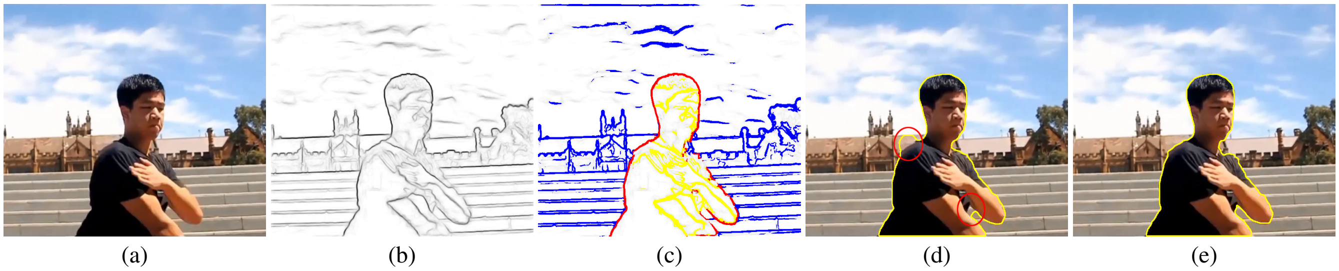


Figure 3: *Edge classification and level set refinement. (a) input target image; (b) edge map produced by the method of Dollár and Zitnick [2013]; (c) result of edge classification (S-edges are drawn in red, F-edges in yellow, B-edges in blue); (d) level set result without using edge classification, note the errors inside the red circles; (e) level set result using edge classification.*

at each contour point to represent its local shape, and compute the shape difference between two contour points as the SSD between their corresponding patches (divided by the number of pixels in the mask patch). The color associated with each point is computed as the mean foreground color in a 15 × 15 image patch centered at it, and d^{color} is the L_2 distance between these colors. Note that only the colors inside the foreground are used, since the position of the foreground shape over the background may change significantly between different frames. The spatial distance d_{ij}^{space} is defined as $\|p_i + m - p_j\|^2$, where m is the difference between the centers (mean position of points) of the contours in I_{t-k} and I_{t+k}. The final distance d_{ij} is computed as $d_{ij} = d_{ij}^{shape} + d_{ij}^{color} + d_{ij}^{space}$.

The pairwise correspondences computed using the point similarity described above may contain many outliers, which are removed by two additional steps. For each contour point i in I_{t-k}, we find the best matching contour point $\mathcal{M}_i$ in I_{t+k}, and do the same in the other direction. A correspondence is discarded unless the matches in both directions agree. Furthermore, we insist that the ordering of the matched points is monotonic; in other words, $i > j$ implies $\mathcal{M}_i > \mathcal{M}_j$. Matches that violate monotonicity are also discarded as outliers.

Given the resulting set of matched pairs between the contours of I_{t-k} and I_{t+k}, we predict the locations of the corresponding points on the target frame I_t by linear interpolation. Next, each of I_{t-k} and I_{t+k} is warped towards I_t using rigid MLS [Schaefer et al. 2006], resulting in two warped images I_{t-k}^f and I_{t+k}^f whose foreground regions roughly predict the position and shape of the foreground region in I_t.

Next, we also align the background regions of I_{t-k} and I_{t+k} with the target I_t, exactly as described in Section 3.1, yielding another pair of warped images I_{t-k}^b and I_{t+k}^b. Finally, we compute four separate NNFs from the target frame I_t to each of the four warped images above, and transfer the mask by using the NNF that yields the best match at each pixel (again, as was done in Section 3.1).

3.3 Edge Classification

It has long been recognized that edges play a crucial role in figure-ground separation in our visual perception [Rubin 2001], and many image segmentation algorithms incorporate edges into the process, reasoning that region boundaries should coincide with salient edges. However, the mask transfer techniques described earlier are based on NNFs and do not explicitly take edges into account. Furthermore, the transferred masks are eventually cleaned up using the level set method, as described in the next section. Thus, to incorporate edges into the process, and to prevent the mask from aligning itself with irrelevant salient edges that might exist in its vicinity, we attempt to classify all the salient edges in the target frame.

Specifically, we attempt to classify the salient edges in the image into one of three classes: B-edges (entirely contained in the background), F-edges (entirely contained in the foreground), and S-edges (silhouette edges separating the foreground from the background). It should be noted, however, that in the remainder of our pipeline, we currently only make use of the S-edges. The extra classification of B-edges and F-edges comes for free without any extra computational effort, since each edge is classified by examining its two sides as explained later. We use a simple non-parametric supervised three-way classifier, whose training data is given by the source frame and its mask. The edge classifier is in principle similar to a pixel classifier, since it is applied at individual edge points, but it incorporates the orientation of the edge, and makes use of the two colors on its two sides.

We begin by extracting an edge map from both source and target frames using the fast state-of-the-art method of Dollár and Zitnick [2013]. For each point whose edge response exceeds a certain threshold we associate two pixels, one on each side of the edge. For each such pixel we extract a 7-dimensional feature vector, consisting of $(r, g, b, x, y, \sin\theta, \cos\theta)$, where r, g, b are the colors, x, y are the spatial coordinates, and θ is the orientation of the gradient. Each dimension is normalized to the range of $[0, 1]$, and then the color, space coordinates, and orientation are scaled by a factor 10.0, 2.0 and 5.0, respectively. We then classify the two pixels separately. This separation of the two sides is important, since we are primarily interested in correctly classifying the S-edges, and since the foreground is moving across the background, points on S-edges rarely retain both of their colors.

Let p_i and p_j be the two pixels associated with an edge point, as described above. Each of these points is classified as foreground or background using a k-NN classifier. It should be noted that the classifier adds the offsets provided by the NNFs to the spatial (x, y) coordinates of the feature vectors. The edge point is then identified as an S-edge point, if the labels of p_i and p_j differ. Our current implementation uses FLANN [Muja and Lowe 2009] to accelerate the nearest neighbor search, and $k = 15$ in all our results. Note that having classified an edge point as silhouette, we also know on which side of the edge lies the foreground. Figure 3 demonstrates the results of our edge classifier, and their effect on the subsequent level set contour.

3.4 Level Set Mask Refinement

The results of mask transfer and mask interpolation usually contain some noise and do not always correctly follow the silhouette edges of the foreground object. They may also contain some small isolated incorrectly classified islands. We use an approach based on the level set method to clean up the mask boundary, by smoothing it and snapping it to nearby S-edges. Our method is an extension of the popular Geodesic Active Contour method [Caselles et al. 1997]. Let $\phi : \Omega \rightarrow R$ denote the level set function, where Ω is the image domain. We design ϕ such that its zero level yields the desired foreground contour C:

$$C_t = \{x \in \Omega \,|\, \phi_t(x) = 0\}, \tag{1}$$

where t is introduced for the evolution process. C_0 is initialized with the result of our NNF-based mask transfer (Section 3.1), and ϕ_0 is set to the signed distance transform applied to C_0, with a positive sign in the interior (foreground) region, and negative elsewhere.

Having initialized ϕ_0, the contour C is evolved by iteratively updating ϕ: $\phi_t = \phi_{t-1} + d\phi$, with $d\phi$ defined as:

$$d\phi = \alpha E^g + \beta E^e + \gamma E^m \tag{2}$$

with the balancing weights set to $\alpha = 1, \beta = \gamma = 0.5$ in all our results (we did not observe particular sensitivity to the values of these parameters).

The first term E^g is the geodesic active contour functional:

$$E^g = \|\nabla\phi\| \text{div}\left(\frac{1}{1+\tau|g(I)|^2} \frac{\nabla\phi}{\|\nabla\phi\|}\right) \tag{3}$$

which combines both image edges and a smoothness prior. $g(I)$ is the edge response produced by Dollár and Zitnick's method, and τ is set to 100.

For each classified S-edge e, we compute the L_2 distance transform ψ_e in its vicinity. The edge term is then defined as $E^e = s\,\psi_e$, where $s \in \{-1, +1\}$ is positive on the interior side of the edge and negative on the exterior side. The information of which side is interior and which is exterior comes from the edge classifier. The purpose of the edge term E^e is thus to attract the zero level set contour towards nearby silhouette edges. A similar edge field term was previously used by Liu and Yu [2012], but without the advantage of having classified edges.

Finally, the mask prior E^m is designed to penalize large deviations from the results of the initial NNF-based mask, since our goal is only to remove small errors and smooth the resulting contour. It is defined as $E^m = \omega\, s\,\psi_c$, where $s\,\psi_c$ is the signed L_2 distance transform from the initial mask contour C_0, and ω is a weight field computed from the residual error of the NNF: $\omega = \exp(-u\,\Delta)$. Here Δ is the color difference to the nearest neighbor and u is a scale factor (0.01 for all our results).

Figure 3 demonstrates the effect of the mask refinement step, and in particular, shows the effect of the edge term E^e.

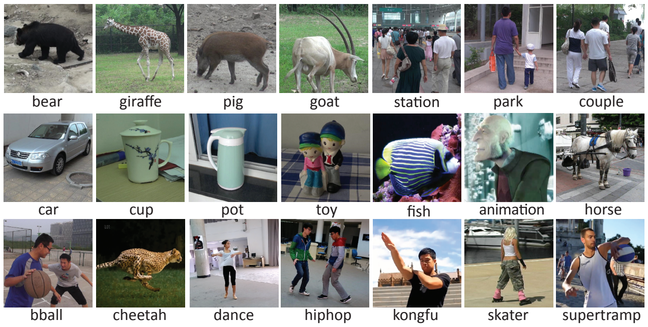


Figure 4: *Dataset used in our experiments. An additional example,* tricking*, is shown in Figure 1.*

4 Results

In this section we present some of the results that we were able to obtain using our method, and evaluate its performance, including comparisons to several existing state-of-the-art methods.

We implemented our method in C++, with a GPU-accelerated implementation of the PatchMatch algorithm. Our current implementation typically requires under 1.4 seconds to transfer a mask between two frames of size 960 × 540, on an Intel Core i7-2700K 3.50GHz CPU with 32G RAM and a GeForce 680 GPU with 4GB of RAM. Currently, about 45 percent of the time is spent on PatchMatch, 30 percent on edge classification, and 20 percent on level

Table 1: *Error rates for automatic mask transfer across different frame distances for several different methods: RB – RotoBrush (based on [Bai et al. 2009]), Z12 – [Zhong et al. 2012], SS – SeamSeg [Ramakanth and Babu 2014], JC – our JumpCut method (split localized NNFs). The numbers in the top row indicate the mask transfer distance. The JumpCut method yields the smallest error in the vast majority of cases.*

		1				4				8				16				32			
		RB	Z12	SS	JC	RB	Z12	SS	JC	RB	Z12	SS	JC	RB	Z12	SS	JC	RB	Z12	SS	JC
ANIMAL	bear	1.82	**1.07**	1.84	1.36	6.74	**1.83**	2.93	2.91	5.58	**2.72**	3.44	3.18	4.58	4.48	4.21	**4.00**	**4.10**	7.19	7.37	5.05
	giraffe	8.49	6.99	4.77	**3.83**	12.7	8.08	8.82	**6.11**	14.4	9.27	11.6	**6.69**	22.0	11.2	17.4	**7.40**	24.9	17.3	25.1	**7.95**
	pig	3.86	**2.08**	3.39	2.97	7.02	3.74	5.84	**3.27**	11.3	7.08	9.07	**3.58**	9.22	9.85	10.3	**3.43**	10.5	9.43	14.0	**4.17**
	goat	3.68	2.57	3.30	**2.00**	7.00	4.74	5.45	**3.35**	10.6	8.61	6.34	**3.41**	13.1	13.3	8.22	**4.14**	24.8	21.1	12.3	**5.39**
HUMAN	station	2.53	2.01	2.37	**1.53**	4.24	6.66	8.58	**3.82**	**6.66**	14.0	16.0	6.80	**8.85**	20.9	21.3	9.01	9.81	24.5	24.7	**9.68**
	couple	4.09	3.54	3.78	**2.27**	10.0	5.90	12.7	**4.35**	18.1	11.0	17.2	**4.81**	17.5	16.0	23.4	**5.13**	7.36	26.6	31.7	**6.22**
	park	3.95	3.49	3.33	**2.93**	6.28	**4.14**	4.47	5.06	8.97	**4.60**	5.07	5.19	11.8	6.54	6.91	**5.39**	20.5	9.18	8.10	**5.85**
STATIC	car	1.35	1.38	0.73	**0.54**	1.43	1.42	1.45	**0.87**	1.52	1.79	2.18	**0.99**	**1.76**	5.93	5.08	2.26	**2.90**	17.5	12.4	8.10
	cup	3.72	1.34	1.19	**1.16**	4.14	1.87	**1.61**	1.73	5.16	4.65	3.02	**1.93**	5.45	12.9	9.31	**2.15**	8.21	28.2	25.3	**4.72**
	pot	0.94	1.49	0.80	**0.70**	1.58	1.56	**1.05**	1.28	1.71	2.11	**1.41**	1.49	**2.43**	5.03	2.98	2.95	**3.96**	12.3	6.90	5.41
	toy	1.02	1.32	0.70	**0.58**	**1.15**	1.45	1.29	1.18	1.25	2.05	1.44	**1.23**	**1.28**	3.19	2.16	1.30	**1.34**	7.66	2.95	1.38
SNAPCUT	animation	1.98	**1.26**	1.83	1.59	5.18	**3.43**	3.52	4.46	12.9	5.59	7.45	**4.46**	11.9	6.38	6.78	**4.55**	22.0	10.1	11.4	**6.90**
	fish	2.80	1.97	2.54	**1.80**	7.68	5.87	7.32	**5.66**	13.8	9.25	10.8	**7.61**	51.8	21.7	25.7	**17.5**	105	41.8	48.2	**39.6**
	horse	3.99	4.14	3.00	**2.62**	5.18	11.3	12.6	**3.93**	7.43	28.4	25.9	**5.48**	8.39	45.1	37.8	**6.80**	10.9	68.1	48.0	**8.77**
FAST	bball	**1.55**	1.71	1.90	1.61	6.86	3.24	2.97	**2.16**	10.0	4.70	4.95	**2.75**	18.4	8.47	8.89	**3.90**	20.0	10.8	10.8	**4.37**
	cheetah	7.17	**3.99**	5.07	4.41	13.4	6.82	6.06	**4.87**	21.1	10.6	7.47	**5.97**	31.5	16.6	**7.68**	8.16	59.4	26.6	**8.21**	9.59
	dance	6.65	9.19	7.55	**6.62**	29.8	17.7	30.9	**12.4**	40.2	37.9	53.4	**18.3**	56.1	50.8	43.0	**18.7**	109	64.4	48.6	**34.0**
	hiphop	8.02	4.62	6.94	**3.37**	36.4	19.3	27.8	**8.48**	73.4	32.0	39.1	**11.0**	67.5	51.1	33.7	**14.2**	111	73.6	44.8	**21.0**
	kongfu	5.42	3.71	5.78	**3.28**	26.6	18.4	12.3	**5.30**	25.6	24.8	18.8	**6.59**	40.2	40.8	17.9	**8.00**	21.6	32.9	16.5	**7.95**
	skater	6.33	5.33	5.09	**4.89**	11.5	8.93	8.84	**6.78**	25.4	21.2	11.7	**8.02**	38.7	40.8	29.6	**22.8**	48.5	72.0	85.2	**46.8**
	supertramp	14.7	8.99	17.4	**6.17**	52.7	32.2	35.4	**22.6**	76.8	42.2	41.7	**31.3**	129	60.5	57.4	**42.9**	159	91.9	55.2	**50.8**
	tricking	42.2	9.71	11.9	**5.02**	31.3	21.4	29.0	**8.31**	48.4	41.6	46.6	**17.9**	79.4	70.9	35.8	**21.3**	93.9	94.3	**47.3**	61.6

set refinement. While our implementation could be optimized further, the current running times already enable interactive response times. This is particularly true for a multi-threaded implementation, which can propagate some masks while the user is interacting with another frame. A real-time capture of an interactive session is included in the supplementary video.

To thoroughly evaluate our method, we collected five sets of video clips, with representative frames shown in Figure 4. The SNAPCUT set contains three examples from Bai et al. [2009]. The ANIMAL, HUMAN, and STATIC sets are from the training data set of Zhong et al. [2012]. The FAST set is a new set we collected, featuring very fast motion and significant foreground deformations. There are 22 video clips in total, and for each example, we obtained the ground truth foreground masks from the original datasets (for ANIMAL, HUMAN, STATIC) or by careful manual segmentation.

4.1 Comparison and Evaluation

We compare our method's mask transfer accuracy with several alternatives, including the state-of-the-art discontinuity-aware method of Zhong et al. [2012] (Z12), and the Rotobrush tool in Adobe AfterEffects (RB), which is based on the Video SnapCut method of [Bai et al. 2009]. In order to compare with the state-of-the-art segmentation tracking results in computer vision, we also include in this comparison the recent SeamSeg method [Ramakanth and Babu 2014], which has proved itself as a top performer on the SegTrack benchmark [Tsai et al. 2010][2].

The error rates of the different methods are reported in Table 1. Errors are computed as the ratio between the wrongly classified areas of the transferred mask and the foreground area of the ground truth mask. For each test sequence we use 128 frames, and compute the *automatic* transfer of a ground truth mask from frame i to $i + d$, for $i = 0, 16, \ldots, 96$, for several different transfer distances $d \in [1, 4, 8, 16, 32]$. Each error reported in the table is the average error for all of the mask transfers for a particular distance d computed for the sequence. In addition to the quantitative comparison, Figure 5 shows a visual comparison of some of the resulting masks produced by the different methods.

As may be seen from Table 1, our method outperforms all of the other methods on the vast majority of examples across the five datasets. The Rotobrush tool exhibits the best performance on distant mask transfer in most of the STATIC sequences. However, it uses only local classifiers for mask propagation, so it is sensitive to fast motion and to any topological changes, as evidenced by the high errors on examples like "giraffe", "goat", and the clips in the FAST set. Z12 handles temporal discontinuities better than Rotobrush, so its error rates are lower on most sequences (except the STATIC set). However, for faster motions and longer transfer distances it still generates significant errors, due to the limited and fixed sampling range of its classifiers. The SeamSeg method performs better than Z12 on distant transfer in nearly all of the FAST sequences, and is the best performer on two of these, for a stride of 32. Our method performs better than each of these methods overall, achieving the lowest error in the vast majority of cases.

A visual examination of the resulting masks shows that our method produces more coherent results than Z12, which sometimes tends to introduce false background holes in the foreground region (see the "couple" example in Figure 5). This is because Z12 does not enforce any coherency constraint in each directional window. In

[2] We also evaluated our method on the SegTrack benchmark, which is not well suited for video cutout performance evaluation (low resolution and low quality video sequences, without a sufficiently accurate ground truth). The results of this evaluation are included in the supplementary materials.

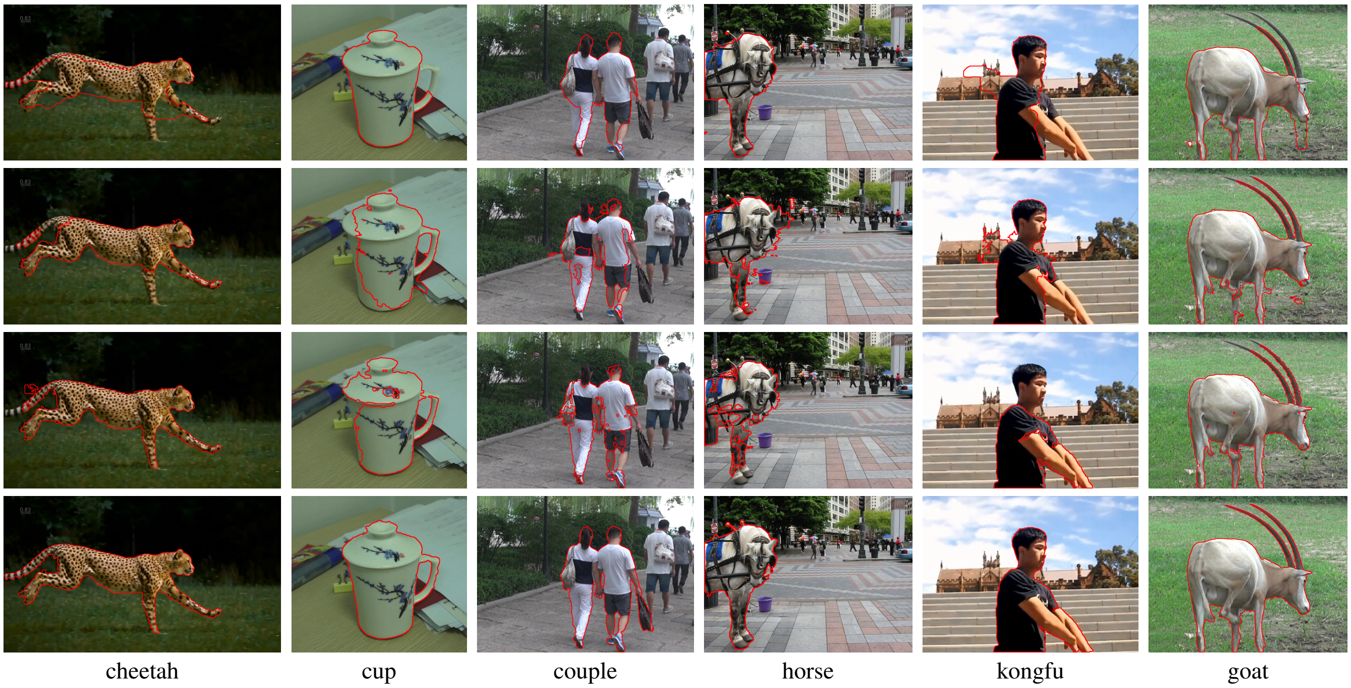


Figure 5: *From top to bottom row we show results of Rotobrush, Zhong et al. [2012], SeamSEG, and our method. Each row shows the result of automatic mask transfer across a distance of 16 frames, for six different examples. Please watch the companion video!*

addition, due to the lack of coherency and the use of single pixel color for classification, Z12 is more sensitive to similar colors between foreground and background than our patch-based classification method (see "horse" in Figure 5). Our results are also noticeably more coherent than those produced by SeamSeg.

In Table 2 we examine the error rates for mask transfer by our method when performing the transfer from frame i to frame $i{+}8$ using different strides. A stride of 1 means that we start with a ground truth mask in frame i, transfer it to frame $i+1$, from frame $i+1$ to $i{+}2$, and so forth, until we obtain the mask at frame $i{+}8$. The entire process is automatic, without manual correction of errors along the way, and the reported error is computed with respect to the ground truth mask at frame $i+8$. This table shows that the accuracy of our mask transfer generally increases when using a larger stride. While this may appear counter-intuitive at first glance, the reason for this behavior is that when using smaller strides there is an accumulation of errors from the multiple steps. Figure 6 shows such an example. Small errors may be difficult for the user to notice. And if these small errors are not fixed at every frame, they quickly accumulate and add up to a larger error after a few propagation steps. However, it may also be seen that on some of the fast moving sequences, our method performs better with a smaller stride of 4 frames.

The same table also reports the error rates for interpolating an intermediate frame i from two source frames (with ground truth masks) either 4 frames (i4) or 8 frames away (i8). It may be seen that interpolation (bi-directional transfer) results in a smaller error rate than one-sided transfer for the same stride in all cases but one (a particularly challenging sequence). These results demonstrate that our distant mask propagation is well suited for a workflow that combines distant mask transfer with interpolation. Note that we did not measure the i4/i8 errors with ground-truth on one side and propagated mask on the other side, since in our proposed interactive workflow the user is expected to fix errors in the i+4/i+8 frame before interpolation is applied to generate the intermediate frame. An example of such a workflow is shown in the real-time captured interactive sessions in the companion video and the supplementary materials.

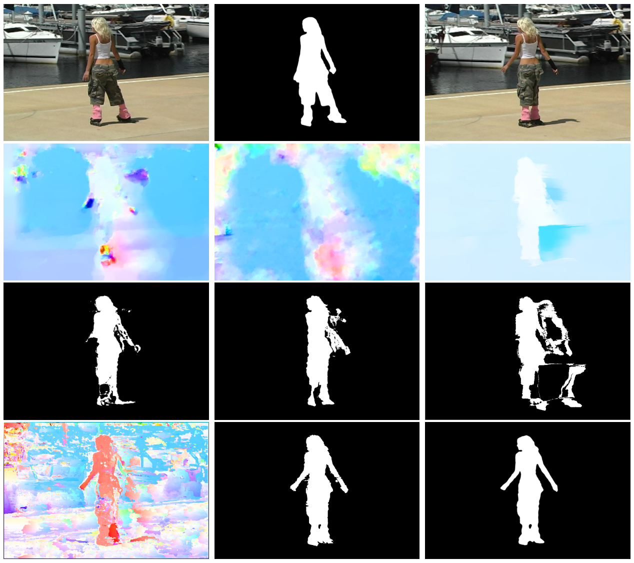
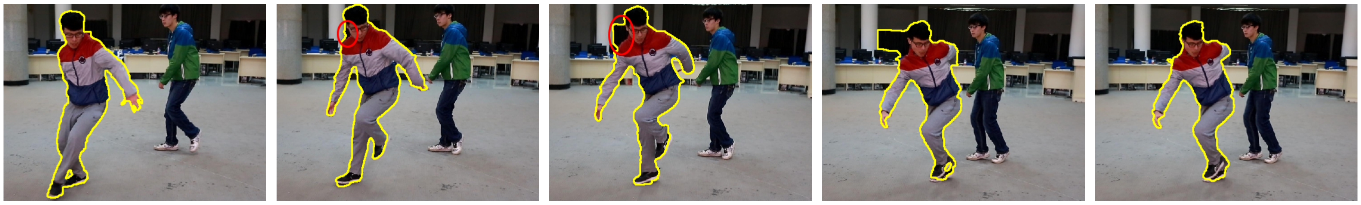


Figure 7: *Comparison with large displacement optical flow. Top row (left to right): source frame, source mask, and target frame. 2nd row: optical flow fields of Brox et al., DeepFlow, and Bao et al. 3rd row: corresponding resulting masks. 4th row: our combined NNF, resulting mask, and ground truth mask.*

In principle, one could consider performing mask transfer using optical flow. Figure 7 compares our results to ones that we were able to obtain using several optical flow methods, including the classical method by Brox et al. [2004], DeepFlow [Weinzaepfel et al. 2013], and Bao et al. [2014]. Note that the last two methods specifically



(a) F80 (b) F83 (c) F85 (d) F88 (e) F88

Figure 6: *Comparison of distant transfer and sequential transfer. (a)-(d) are the results of sequential propagation from frame 80 to frame 88; note that the small error in frame 83 (circled in red) is difficult to notice, yet it accumulates to a significant error in frame 85 and onward. (e) The distant transfer result by our method.*

Table 2: *Error rates for automatic mask transfer by our method from frame i to $i + 8$ using four different strides. Also shown (in parentheses) the error rates for interpolation: i4 means interpolating frame i from $i - 4$ and $i + 4$.*

		1	2	4 (i4)	8 (i8)
ANIMAL	bear	2.94	2.85	**2.79** (1.91)	2.82 (2.10)
	giraffe	10.3	8.28	7.35 (4.51)	**6.27** (4.79)
	pig	5.22	4.00	3.49 (2.44)	**3.24** (2.69)
	goat	4.53	3.77	3.11 (2.39)	**2.98** (2.57)
HUMAN	station	6.14	6.03	**5.72** (2.62)	6.55 (4.38)
	couple	5.95	5.21	**4.64** (3.52)	4.71 (4.03)
	park	5.89	5.33	**4.71** (2.24)	4.89 (2.70)
STATIC	car	1.52	1.30	1.09 (0.82)	**0.98** (0.90)
	cup	2.79	2.46	**2.13** (1.28)	2.20 (1.36)
	pot	1.47	1.41	1.35 (0.93)	**1.29** (0.93)
	toy	1.51	1.44	1.22 (0.78)	**1.11** (0.78)
SNAPCUT	animation	3.64	3.66	3.26 (1.59)	**3.08** (2.07)
	fish	3.35	2.95	**2.91** (2.25)	4.05 (2.43)
	horse	5.81	4.99	4.61 (3.46)	**4.55** (3.95)
FAST	bball	2.57	2.42	**2.10** (1.68)	2.12 (1.67)
	cheetah	6.10	5.94	**5.98** (5.03)	6.11 (5.35)
	dance	12.8	10.4	**13.7** (10.6)	15.9 (10.4)
	hiphop	8.66	7.95	**7.57** (3.48)	8.05 (4.85)
	kongfu	6.47	7.26	**5.94** (3.82)	6.94 (6.93)
	skater	15.1	11.9	8.28 (5.56)	**7.57** (5.98)
	supertramp	19.3	16.4	**15.5** (10.2)	17.0 (19.1)
	tricking	15.6	16.1	**14.9** (7.47)	18.6 (10.1)

target large displacement optical flow. As can be seen in Figure 7, these methods are less successful in propagating the foreground mask, even between two frames with relatively mild displacements. A quantitative comparison with the latter two methods, as well as the unmodified PatchMatch algorithm [Barnes et al. 2009] is reported in Table 3.

Figure 8 shows an example of our method for automatic 8-frame interpolation, where the foreground undergoes a drastic change in both shape and appearance (turning around). Our automatically interpolated results are accurate except at some blurry edges, and note that topology changes are also handled well. For comparison, we show the results of Agarwala et al. [2004], which fails on this example, due to the large change in the foreground shape. Note that the method of [Agarwala et al. 2004] requires correspondences between the two ends frames to be specified by the user, which is difficult for the shown case, even when using the interactive tool provided in the author's software.

User study. We conducted a small scale informal user study to assess the degree to which our more accurate mask transfer and mask interpolation mechanism can reduce the user effort involved in extracting a cutout from a video sequence. Nine users took part in our user study. None of the users had prior experience with any interactive video segmentation tools, but four of them were familiar with object selection in Adobe Photoshop. The users were first trained in the use of the different video cutout tools, using several training video clips. A user passed his/her training, only once he/she could achieve an error rate below a certain threshold, within an allotted amount of time. Each trained user was then asked to segment several test video clips (different from the training ones), as accurately and as quickly as possible. Specifically, each user was assigned the task of performing a video cutout of five short (32 frames) video sequences from our test set, with each of the three systems (RotoBrush, Z12 and JumpCut), resulting in a total of 15 video cutout tasks. The tasks were assigned to each user in a random order.

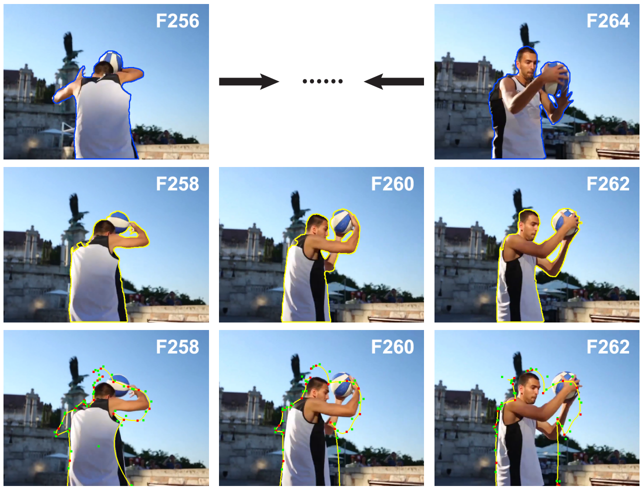


Figure 8: *Automatic interpolation from frames 256 and 264 (top row) to intermediate frames. The middle and bottom rows are the results of our method and [Agarwala et al. 2004], shown for frames 258, 260 and 262.*

We recorded the number of strokes that each user had to place for each sequence, as well as the combined length of these strokes. The sum of stroke lengths is also reported, because some users may use a few long strokes to same effect as many short strokes. The total time that it took each user to complete each task was also recorded. Finally, we computed the average mask error using the available ground truth masks, to verify that the extracted masks were all comparable in terms of their accuracy. The results of the study are summarized in Table 4, averaged over all users for each task. It may be seen that in almost all cases, our method required consider-

Table 3: *Error rates for mask transfer using optical flow methods, as well as unmodified PatchMatch to compute the NNF. Masks are propagated from frame i to $i + 8$ with stride = 1. B14 – [Bao et al. 2014], DF – [Weinzaepfel et al. 2013], PM – [Barnes et al. 2009], JC – our split localized NNFs.*

		B14	DF	PM	JC
ANIMAL	bear	28.6	50.0	3.19	**2.86**
	giraffe	42.0	38.8	10.6	**9.34**
	pig	34.2	50.1	5.09	**4.35**
	goat	16.3	39.5	5.92	**5.26**
HUMAN	station	16.0	30.3	18.9	**7.64**
	couple	17.6	35.3	22.7	**5.77**
	park	19.6	24.0	7.21	**5.69**
STATIC	car	5.08	17.7	5.35	**1.05**
	cup	7.92	35.8	5.24	**2.17**
	pot	7.24	17.9	1.92	**1.47**
	toy	6.56	6.47	4.05	**1.24**
SNAPCUT	animation	15.8	17.4	6.69	**4.25**
	fish	7.59	18.6	14.0	**3.09**
	horse	12.2	58.7	19.4	**4.50**
FAST	bball	31.1	45.3	6.87	**5.03**
	cheetah	32.5	47.4	9.97	**7.06**
	dance	55.3	123	50.3	**13.4**
	hiphop	59.2	94.3	46.0	**9.11**
	kongfu	58.5	83.9	20.4	**13.2**
	skater	56.9	61.4	22.4	**11.6**
	supertramp	77.0	73.4	43.2	**19.1**
	tricking	103	122	44.7	**15.0**

ably fewer strokes to be placed, with a correspondingly smaller total stroke length. As a result, the total task completion time is lower in most cases. It should be noted that the task completion times is greatly affected by the UI, whose design is outside the scope of this paper, as well as by other factors, which our informal study does not attempt to account for.

5 Conclusions and Future Work

We have presented a video cutout technique designed specifically to excel in fast motion sequences. The challenge in transferring a mask to a frame exhibiting a large motion is that techniques based on local statistics are likely to be less effective, as the changes are non-local. Our approach is based on the efficiency of advanced methods for computing the nearest neighbor field (NNF) between two images. The key is that these methods are non-local, but at the same time they generate a coherent mapping. Often, in fast motion sequences, it is the foreground object that changes fast and significantly, while the background is more stable, reflecting the camera motion only. Thus, we have separated the NNF of the foreground and background, avoiding the often harmful effort to keep the two coherent. The same split-NNF is also used in our edge classifier, which in turn further improves the performance of level-set based cutout.

Our method is still challenged by ambiguities in color and texture between the foreground and the background. Such ambiguities introduce errors into the NNFs that we compute and use, as well as into our edge classifier. Furthermore, the edge classifier heavily relies on edge detection (currently provided by Dollár and Zitnick's method [2013]), thus carrying over the limitations of the edge detector, especially when blurry edges are present. This is demonstrated in Figure 9. To overcome this weakness, a stronger shape prior is required. Another challenge is presented by disocclusions, as demonstrated in Figure 10.

Table 4: *User study of three different real-time interactive video cutout methods. RB – RotoBrush (based on [Bai et al. 2009]), Z12 – [Zhong et al. 2012], JC – our JumpCut method.*

		RB	Z12	JC
couple	number of strokes	95.44	151.22	**75.22**
	sum of stroke lengths	3285.11	4091.33	**2499.22**
	run time	547.33	501.44	**396.67**
	mask error	4.36	3.60	**3.15**
cup	number of strokes	163.00	60.33	**19.22**
	sum of stroke lengths	12166.56	1339.11	**679.11**
	run time	576.44	196.89	**175.44**
	mask error	2.01	1.63	**1.38**
fish	number of strokes	51.11	132.89	**34.33**
	sum of stroke lengths	3214.00	8166.67	**2024.33**
	run time	266.67	313.44	**198.22**
	mask error	2.74	3.10	**2.08**
pig	number of strokes	85.78	74.00	**29.67**
	sum of stroke lengths	5689.00	2509.00	**1172.78**
	run time	354.11	214.11	**201.22**
	mask error	3.63	**2.01**	2.03
tricking	number of strokes	123.33	250.33	**119.44**
	sum of stroke lengths	6482.56	13096.56	**5677.89**
	run time	**406.33**	443.11	534.22
	mask error	4.07	4.31	**3.29**

Being able to handle large changes in the foreground has two immediate applications. First, it allows dealing with fast motion sequences where consecutive frames exhibit large changes with large magnitude optical flow. Second, it allows leaping between distant frames and then interpolating the in-between frames, using decreasing strides. This interpolating scheme is somewhat counterintuitive, since it seems that incremental steps between more similar frames should yield fewer errors. However, as we have shown, the incremental approach accumulates more errors, than leaping forward and then interpolating.

One avenue for future work is to associate a confidence value to a mask transfer, allowing the system to automatically employ an adaptive stride size for mask transfer and interpolation. Another possible improvement is to take advantage of cyclic motions, where similar foreground poses may be found many frames apart, and extremely long strides can be used effectively. An intriguing direction, which we also consider, is to extend this idea further, and treat

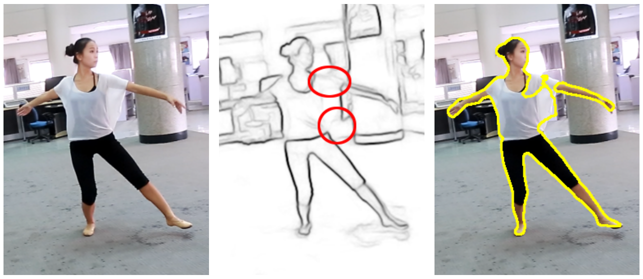


Figure 9: *Error due to failure to detect blurry edges. Left: target image. Middle: edge map, note that no strong edges are detected in the red circles. Right: mask transfer result.*

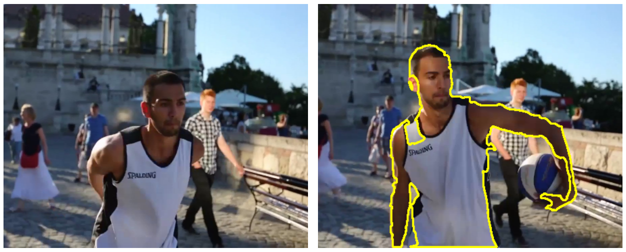


Figure 10: *Failure case of our method due to complex color and dis-occlusion: the ball is hidden in the source frame (left). Right: the target frame and the resulting contour.*

the video sequence as an unordered set of frames. Similar frames can then be clustered and processed together, without relying on the natural time coherency along the video.

Acknowledgements

We thank the anonymous reviewers for their valuable comments. This paper is supported by 973 program of China (No. 2015CB352501), NSF of China (No. 61232011,61572290), Young Scholars Program of Shandong University, and by the Israel Science Foundation (ISF).

References

Agarwala, A., Hertzmann, A., Salesin, D. H., and Seitz, S. M. 2004. Keyframe-based tracking for rotoscoping and animation. *ACM Trans. Graph. 23*, 3 (Aug.), 584–591.

Bai, X., Wang, J., Simons, D., and Sapiro, G. 2009. Video SnapCut: Robust video object cutout using localized classifiers. *ACM Trans. Graph. 28*, 3 (July), 70:1–11.

Bai, X., Wang, J., and Sapiro, G. 2010. Dynamic color flow: A motion-adaptive color model for object segmentation in video. In *Proc. ECCV*, Springer-Verlag, vol. V, 617–630.

Bao, L., Yang, Q., and Jin, H. 2014. Fast edge-preserving patchmatch for large displacement optical flow. *IEEE Trans. Image Processing 23*, 12 (Dec.), 4996–5006.

Barnes, C., Shechtman, E., Finkelstein, A., and Goldman, D. 2009. PatchMatch: A randomized correspondence algorithm for structural image editing. *ACM Trans. Graph 28*, 3, 24.

Bay, H., Ess, A., Tuytelaars, T., and Van Gool, L. 2008. Speeded-up robust features (SURF). *Computer Vision and Image Understanding 110*, 3, 346–359.

Brox, T., Bruhn, A., Papenberg, N., and Weickert, J. 2004. High accuracy optical flow estimation based on a theory for warping. In *Proc. ECCV'04*. Springer, 25–36.

Caselles, V., Kimmel, R., and Sapiro, G. 1997. Geodesic active contours. *Intl. J. Comp. Vision 22*, 1, 61–79.

Chen, Z., Jin, H., Lin, Z., Cohen, S., and Wu, Y. 2013. Large displacement optical flow from nearest neighbor fields. In *Proc. CVPR'13*, 2443–2450.

Chuang, Y.-Y., Curless, B., Salesin, D. H., and Szeliski, R. 2001. A bayesian approach to digital matting. In *Proc. CVPR 2001*, IEEE Computer Society, vol. 2, 264–271.

Chuang, Y.-Y., Agarwala, A., Curless, B., Salesin, D. H., and Szeliski, R. 2002. Video matting of complex scenes. *ACM Trans. Graph. 21*, 3 (July), 243–248.

Cremers, D., Rousson, M., and Deriche, R. 2007. A review of statistical approaches to level set segmentation: Integrating color, texture, motion and shape. *Intl. J. Comp. Vision 72*, 2, 195–215.

Dollár, P., and Zitnick, C. L. 2013. Structured forests for fast edge detection. In *Proc. ICCV*, IEEE, 1841–1848.

Faktor, A., and Irani, M. 2014. Video segmentation by non-local consensus voting. In *Proc. BMVC 2014*.

Li, Y., Sun, J., and Shum, H.-Y. 2005. Video object cut and paste. *ACM Trans. Graph. 24*, 3 (July), 595–600.

Ling, H., and Jacobs, D. W. 2007. Shape classification using the inner-distance. *IEEE Trans. PAMI 29*, 2, 286–299.

Liu, Y., and Yu, Y. 2012. Interactive image segmentation based on level sets of probabilities. *IEEE Trans. Vis. Comp. Graphics 18*, 2 (Feb), 202–213.

Muja, M., and Lowe, D. G. 2009. Fast approximate nearest neighbors with automatic algorithm configuration. In *International Conference on Computer Vision Theory and Applications (VISAPP'09)*.

Osher, S. J., and Fedkiw, R. P. 2003. *Level Set Methods and Dynamic Implicit Surfaces*, 1st ed. Springer-Verlag.

Price, B., Morse, B., and Cohen, S. 2009. LIVEcut: Learning-based interactive video segmentation by evaluation of multiple propagated cues. In *Proc. ICCV*, 779–786.

Ramakanth, S. A., and Babu, R. V. 2014. SeamSeg: Video object segmentation using patch seams. In *Proc. CVPR'14*.

Rubin, E. 2001. Figure and ground. In *Visual Perception*, S. Yantis, Ed. Psychology Press, Philadelphia, 225–229.

Schaefer, S., McPhail, T., and Warren, J. 2006. Image deformation using moving least squares. *ACM Trans. Graph. 25*, 3, 533–540.

Smith, A. R., and Blinn, J. F. 1996. Blue screen matting. In *Proc. SIGGRAPH '96*, ACM, 259–268.

Tong, R.-F., Zhang, Y., and Ding, M. 2011. Video brush: A novel interface for efficient video cutout. *Computer Graphics Forum 30*, 7, 2049–2057.

Tsai, D., Flagg, M., and Rehg, J. M. 2010. Motion coherent tracking with multi-label MRF optimization. In *Proc. BMVC*.

Wang, J., Bhat, P., Colburn, R. A., Agrawala, M., and Cohen, M. F. 2005. Interactive video cutout. *ACM Trans. Graph. 24*, 3 (July), 585–594.

Wang, T., Han, B., and Collomosse, J. 2014. TouchCut: Fast image and video segmentation using single-touch interaction. *Comp. Vis. Im. Understanding 120*, 14 – 30.

Weinzaepfel, P., Revaud, J., Harchaoui, Z., and Schmid, C. 2013. DeepFlow: Large displacement optical flow with deep matching. In *Proc. ICCV*, IEEE, 1385–1392.

Zhong, F., Qin, X., Peng, Q., and Meng, X. 2012. Discontinuity-aware video object cutout. *ACM Trans. Graph. (SIGGRAPH Asia 2012) 31*, 6 (November), 175:1–10.