Video Segmentation by Non-Local Consensus Voting

Alon Faktor
http://www.wisdom.weizmann.ac.il/~alonf/
Michal Irani
http://www.wisdom.weizmann.ac.il/~irani/

Dept. of Computer Science and
Applied Math
The Weizmann Institute of Science
ISRAEL

Abstract

We address the problem of Foreground/Background segmentation of "unconstrained" video. By "unconstrained" we mean that the moving objects and the background scene may be highly non-rigid (e.g., waves in the sea); the camera may undergo a complex motion with 3D parallax; moving objects may suffer from motion blur, large scale and illumination changes, etc. Most existing segmentation methods fail on such unconstrained videos, especially in the presence of highly non-rigid motion and low resolution. We propose a computationally efficient algorithm which is able to produce accurate results on a large variety of unconstrained videos. This is obtained by casting the video segmentation problem as a voting scheme on the graph of similar ('re-occurring') regions in the video sequence. We start from crude saliency votes at each pixel, and iteratively correct those votes by 'consensus voting' of re-occurring regions across the video sequence. The power of our consensus voting comes from the *non-locality* of the region re-occurrence, both in space and in time – enabling propagation of diverse and rich information across the entire video sequence. Qualitative and quantitative experiments indicate that our approach outperforms current state-of-the-art methods.

1 Introduction

The video segmentation problem considered here is that of consistently separating between the foreground moving objects and the background in complex unconstrained video sequences. Obtaining such a segmentation can be beneficial for many computer vision applications such as action recognition, video summarization, video retrieval and video editing.

Foreground/Background (fg/bg) video segmentation has been widely addressed in the vision community for more than two decades. Older video segmentation methods were mostly *geometry-based*, constrained to specific families of background-induced motions. These were assumed to be either: (i) stationary backgrounds [5, 9, 12, 33] or backgrounds undergoing 2D parametric motion [16, 18, 30] (thus image registration can be used for background stabilization); or (ii) backgrounds undergoing 3D motions with 3D parallax [17, 34, 37]. However, all the geometry-based methods are sensitive to model selection (2D or 3D), and cannot handle non-rigid backgrounds.

Later, *trajectory-based* methods have been proposed, some factorization-based [8, 13, 29], others based on spectral-clustering [4, 14, 25, 26, 31]. These methods, however, heavily rely on the accuracy of optical flow estimation and tracking, and thus encounter difficulties

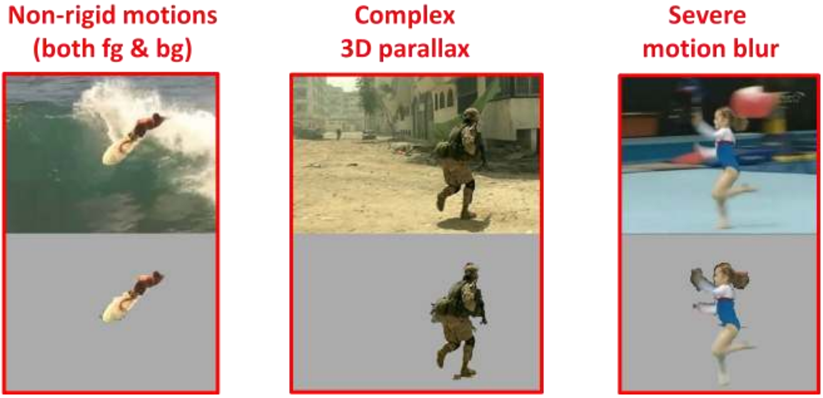

Figure 1: **A unified approach to fg/bg video segmentation in *unconstrained* videos.** Our algorithm can handle in a single framework video sequences which contain highly non-rigid foreground and background motions, complex 3D parallax and simple 2D motions, and severe motion blur.

in the presence of highly non-rigid motions. They further have their own "model selection" problems, in the form of rank selection (in factorization), or the number of clusters.

Advancement in handling non-rigid motions has been made by *supervoxel-based* methods [3, 10, 15, 21, 36]. These methods segment the video into consistent space-time supervoxels by merging image regions using *local* space-time similarity. However, the local nature of supervoxel methods results in severe over-segmentation of the video, both in space and in time. Finally, in *semi-supervised methods* [1, 6, 7, 21, 28, 35] the user manually annotates several video frames, and segmentation is then propagated to all other frames.

Thus, fully unsupervised fg/bg segmentation of unconstrained videos still remains a very challenging task. The camera motion can be very large, the moving object/s can deform rapidly and non-rigidly (yielding high errors in optical flow estimation), the camera motion may induce 3D parallax (thus background stabilization is impossible), the background may further undergo complex non-rigid deformations (e.g., water in the sea), etc. Moreover, the moving objects may suffer from motion blur, may have similar appearance to some of the background, and may undergo large scale and illumination changes. Fig. 1 shows several such examples (see full videos in our **project website** ***www.wisdom.weizmann.ac.il/~vision/NonLocalVideoSegmentation.html***). Unconstrained video has thus become the focus of most recent video segmentation methods [20, 23, 27, 38]. In Sec. 2 we highlight the differences between our approach and these methods.

In this work, we suggest a simple yet general algorithm for performing fg/bg video segmentation, which can handle complex unconstrained videos.We cast the fg/bg segmentation problem as a voting scheme on the graph of re-occurring regions across the video sequence (see Fig. 2). By "re-occurring regions" we mean spatial regions that have similar appearance features (e.g., color, local structure). We start from crude *local* motion saliency votes at each pixel, and iteratively correct these votes by reaching *consensus* of *non-local* re-occurring regions across the video sequence. We allow for re-occurring regions to be quite far both in space and in time, but constrain them to be close in the appearance feature space. This allows fast propagation of information from faraway parts of the video, which enables the correction of large errors in the initial votes.

In contrast to trajectory-based methods, we do not try to explicitly estimate long-term correspondences via flow estimation or tracking, but rather obtain long-term "probabilistic"

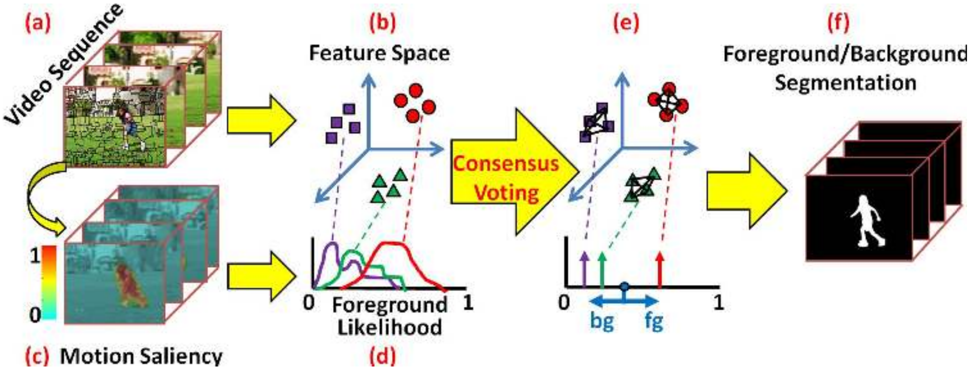

Figure 2: **Scheme of the Algorithm.** (a) Each video frame is split into many regions (superpixels in our current implementation), each reprsented by an appearance descriptor. (b) Neighboring descriptors in feature space represent similar ('re-occurring') regions, which are possibly far in space and time in the video. (c) Each region is associated with a fg likelihood vote, initialized by a motion saliency measure. (d) These votes are very noisy, thus votes of neighboring descriptors in feature space may have large entropy. (e) Consensus voting of neighboring descriptors (NNs) reveals the true fg likelihood of each region. (f) Thresholding the consensus values leads to a fg/bg segmentation of the video.

correspondences using re-occurring regions across distant frames. This avoids the inherent uncertainties of explicit optical flow estimation, whose errors tend to accumulate over time.

We show significant improvement over current state-of-the-art methods on the SegTrack dataset [35] – the current benchmark for unconstrained video segmentation. Moreover, we are able to reduce computation time by an order of magnitude relative to most existing methods. To further demonstrate the power of our approach, we test it on several new and extremely challenging video sequences, with very fast non-rigid foreground and background motions and severe motion blur, showing significant improvement over current state-of-the-art methods. Results can be found in Figs. 4 and 5 and in our **project website**.

The rest of this paper is organized as follows: Sec. 2 places our contributions in the context of other related graph-based video segmentation methods. Sec. 3 describes our non-local iterative voting scheme, which is initialized by our local saliency measure described in Sec. 4. Experimental results are reported in Sec. 5.

2 Related Graph-Based Video Segmentation Methods

Most current methods for unconstrained fg/bg video segmentation are graph-based [20, 23, 27, 38]. In this section we highlight the cardinal differences between our and other graph-based methods, which gives us the leap in improvement.

In methods [20, 23, 27, 38], the video is represented using an Markov Random Field (MRF) graphical model which consists of a fg/bg data term for each pixel and pairwise terms between neighboring pixels. The data term is iteratively refined by learning color models of the foreground and background. For computational reasons, the pairwise terms are typically considered only between adjacent pixels in the same frame and corresponding pixels in adjacent frames (using optical flow). In practice, however, such a *local* graph structure which uses temporal links based on optical-flow, limits how far information can propagate. This is because optical-flow errors rapidly accumulate, thus inducing weak (often

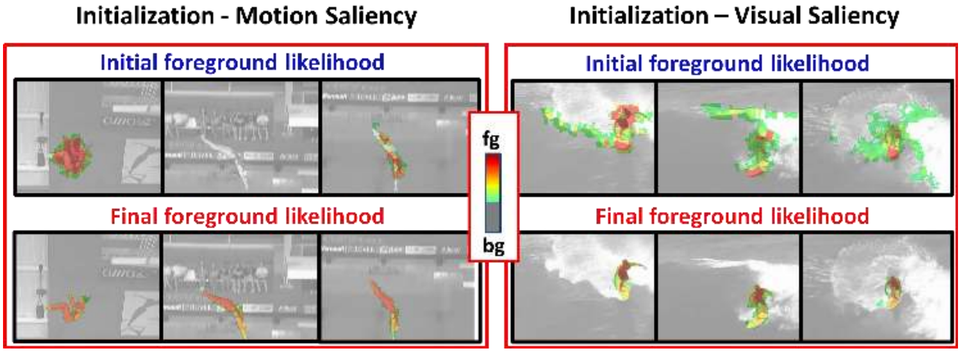
Initialization - Motion Saliency
Initialization – Visual Saliency
Initial foreground likelihood
Initial foreground likelihood
fg
bg
Final foreground likelihood
Final foreground likelihood

Figure 3: **Foreground Likelihood Votes.** The fg-likelihood votes are initialized by crude motion or visual saliency measures (top row). Red represents high fg-likelihood; grey represents very low fg-likelihood (i.e, background pixels). This results in an initial 'noisy' and incomplete fg-likelihood maps (in some frames the fg-likelihood is uniformly set to zero – see text). Our non-local consensus voting 'cleans' the errors in the votes and converges to quite accurate fg-likelihood maps (bottom row).

zero) weights between related video parts in faraway frames. Thus, the segmentation performance of video-MRF methods strongly depends on the quality of the initial fg/bg data term. However, fg/bg initializations tend to be quite noisy, whether based on mining moving object proposals [20, 23, 38], or based on motion saliency maps [27], especially in unconstrained low-quality videos. Therefore, current video segmentation methods encounter difficulties in such challenging videos.

In contrast, we consider an entirely different graph structure which is *non-local*, and allows to propagate diverse information from distant parts of the video. This allows us to start from very 'noisy' foreground votes for all video pixels, and clean them according to 'consensus voting' of re-occurring regions across the video sequence.

The use of non-local neighborhoods was proposed also by several interactive video matting methods [7, 21]. However, their neighborhoods are non-local only in space and not in time, and segmentation is propagated in a causal manner between consecutive frames, starting from a user input in the first frame. We, on the other hand, use non-local neighborhoods both in space and in time with cross talk between frames from all times.

3 Segmentation as a Non-Local Voting Scheme

To perform video segmentation, we consider both *short-term cues* (motion saliency), and *long-term cues* (visual similarity across large time-laps). Neither of these cues alone does not suffice to provide good segmentation results. Our approach is to fuse these two cues in a simple, yet effective, way.

Short-term motion saliency is not a strong enough cue on its own in video sequences with diverse and complex motions. For example, it does not suffice if only some of the foreground moves at part of the video while the rest stays static, or if the entire moving foreground stops moving for several frames. As can be seen in the top row of Fig. 3, the motion saliency produces a noisy map with respect to the true moving foreground (foreground pixels may get low values; background pixels may get high values). Similarly, detecting visual similarity ("re-occurring" regions) across distant frames will generate long-term connections, but will not suffice on its own. This is because in distant frames the moving objects tend to undergo significant non-rigid deformations, changes in illumination, changes in scale, etc. Such distant matches will thus tend to be very noisy and fragmented.

We suggest to cast the video segmentation problem as a consensus voting scheme which combines both short and long term cues. The short-term cues will be used as initial foreground likelihood votes, which will be diffused on the graph of re-occurring regions by the long-term cues. These re-occurring regions are close to each other only in feature space, but can be very distant from each other (can form a *non-local neighborhood*) both in space and in time. One should note that trying to use small spatio-temporal neighborhoods, as commonly used in video-MRFs, will not 'denoise' the votes well since information will probably not propagate too far (see Sec. 2). On the other hand, averaging votes of 're-occurring structures' across distant parts of the video will better 'denoise' the votes, by propagation and 'consensus voting' of the foreground likelihood across distant video parts. A schematic visualization of these ideas is shown in Fig. 2. Examples of votes before and after the long-term vote propagation ("consensus voting") can be found in Fig. 3.

3.1 The Algorithm

We first provide a high-level sketch of our algorithm, and then explain it in more detail.

1. Extract many non-overlapping regions $\{R\}$ from each frame in the video sequence.
2. Represent each region R by a high-dimensional descriptor $d(R)$.
3. For each region R find its M Nearest Neighbors (NNs) in the feature space of $\{d(R)\}$.
4. Compute the similarity between R and each of its NNs $\{NN_m(R)\}_{m=1}^{M}$ by:
$$w(R,NN_m(R)) = exp(-\|d(R)-d(NN_m(R))\|^2/\sigma^2) \tag{1}$$
5. Construct a graph from all the video regions $\{R\}$ and a random-walk transition matrix P over the graph:
$$W(i,j) = \begin{cases} w(R_i,R_j) & R_j \in NN(R_i) \\ 1 & i=j \\ 0 & otherwise \end{cases} \tag{2}$$
and normalizing $P = D^{-1}W$, where $D = diag\{W\mathbf{1}\}$.
6. Initialize fg-likelihood votes: Compute a crude saliency map (Sec. 4) to assign an initial fg likelihood vote $v^{(0)}(R)$ for each region R (by averaging the saliency values of all pixels in the region). The votes of all regions are gathered into one vector $\vec{v}^{(0)}$.
7. for $t = 1 : T$ do:
   - Diffuse: $\vec{v}^{(t)} = P\vec{v}^{(t-1)}$ (this is equivalent to updating the vote of each region R with the weighted average of the votes of its M Nearest Neighbors).
   - Normalize the votes in each frame so they will occupy the entire range $[0,1]$.
8. Threshold the votes to obtain the final binary segmentation of the video.

The above algorithm is repeated in *two phases*, which increases the probability of obtaining reliable matches, while seeking distant matches in space and time. Note that unless *the majority* of the NNs are indeed inliers (i.e., fg regions are mostly matched to fg regions; bg is mostly matched to bg), our voting scheme will not suppress the outliers and will blur the fg/bg separation. Similarly, unless NNs come from far enough frames, we may not be able to fix local temporal errors.

***Phase I* (Constrained space-time search space)**: We restrict the initial NN search space, both for computational efficiency, as well as to ensure that initial *reliable* re-occurring regions are found. Restricted space-time neighborhoods reduce the chance of confusing fg and bg regions. Motions are limited in small time-laps, and when visual similarity search is further restricted to the local spatial vicinity, it increases the chance of matching fg with fg, and bg with bg. We thus initially restrict each region R to find NNs only within a *temporal* radius of $F = 15$ frames, and penalize NNs that are *spatially* distant (more details below).

Phase II **(Relaxed space-time search space)**: We remove the spatial and temporal restriction, and find reliable re-occurring regions across the entire video. This is done as follows: after obtaining a crude segmentation in *Phase I*, we open a slightly larger spatial window around each foreground segment (a bounding box around the segment, with an additional margin of 10%). These "foreground windows" include the moving objects, along with some surrounding background. For all regions R within these "foreground windows", we allow for a new NN search, this time across the entire video (no temporal restriction), but only within the collection of "foreground windows". This allows for distant connections, while also refining the boundaries of the moving objects. In *Phase II*, the votes $v^{(0)}(R)$ in Step 6 are initialized using the *final* votes of *Phase I* (and not the initial saliency-based votes).

3.2 Detailed Description of the Algorithm

We next describe how we extract regions, generate their descriptors, and provide various other implementation details and parameters.

Region Extraction: We choose to use superpixels as image regions. We generate the superpixels by applying the watershed transform to each frame after applying the efficient image boundary detector of [11]. We used a low threshold for the boundary detection, resulting in quite a large number of superpixels (∼1000-2000 per frame). This number is a good tradeoff between keeping a compact frame representation in the graph, while retaining high accuracy of object boundaries. This over-fragmentation allows us to extract meaningful boundaries even in the presence of high motion blur or low resolution.

Region Descriptor: We represent each superpixel by a concatenation of several types of descriptors: RGB and LAB color histograms (6 channels, each of 20 bins), HOG descriptor (9 cells with 6 orientation bins) computed over a 15×15 patch around the center of the superpixel, and "relative spatial coordinates"(normalized between 0 and 1). Incorporating the relative spatial coordinates of the superpixel into the descriptors allows to implicitly penalize spatially distant NNs in the NN-search. In *Phase I*, the "relative spatial coordinates" of each superpixel is with respect to the center of the image. In *Phase II*, the "relative spatial coordinates" of each superpixel is with respect to center of the "foreground window" in which it resides (extracted in *Phase I*). This allows to move from the static frame coordinate system in *Phase I*, to the dynamic moving object coordinate system in *Phase II*. To further obtain a scale invariant representation in *Phase II* (since NNs are now searched in distant frames), we re-scale all the "foreground windows" to a constant size (150^2 pixels per window). The HOG is then recomputed for each superpixel after rescaling.

Nearest Neighbor (NNs) Search: We find approximate NNs using an efficient KD-tree search. In *Phase I*, each superpixel searches for NNs within a temporal radius of $F = 15$ frames, including its own (i.e., $2F + 1$ frames). Each superpixel can have several good matches in a frame (and not only one), since fg superpixels tend to be similar to nearby fg superpixels, and the same holds for bg superpixels. Thus, we set the number of NNs per superpixel to be $M = L(2F + 1)$, where $L = 4$. In *Phase II* we still keep the number of NNs per superpixel to the same number M, but allow to search them in a much larger temporal window (typically 200 frames). This however is done only for superpixels within

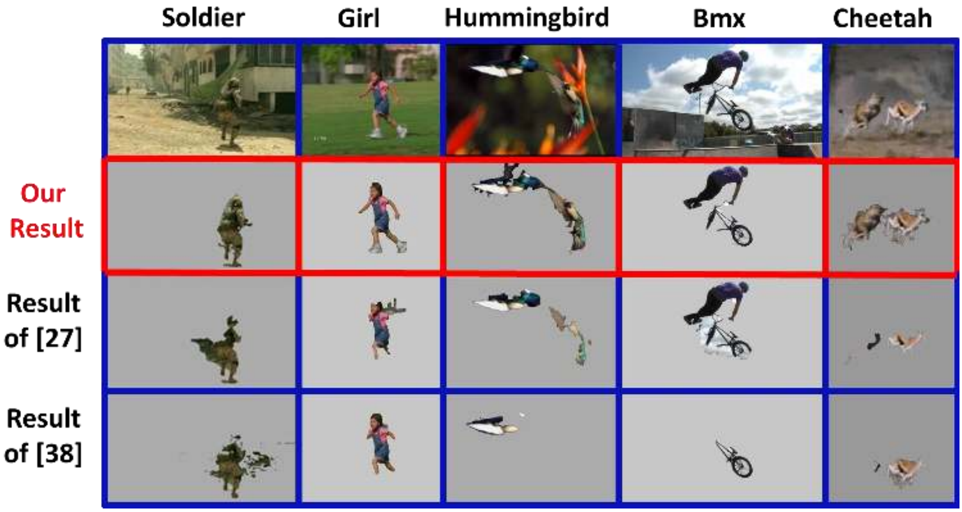

Figure 4: **The SegTrack Benchmark – visual comparison.** Visual comparisons to [27, 38] using their publicly available code. For 'Bmx', we show results of [38] using object selection *without* Grab-Cut (whereas for all other sequences with Grab-Cut), since these settings gave best results for [38]. See full videos on `www.wisdom.weizmann.ac.il/~vision/NonLocalVideoSegmentation.html`.

the "foreground windows". This maintains both computational and memory efficiency (a larger temporal extent, but a smaller spatial portion of each frame).

Runtime: Our algorithm is computationally efficient relative to other methods for fg/bg segmentation of unconstrained video. Extraction of superpixels and their descriptors takes a few seconds per frame, as well as computation of the approximate Nearest Neighbors. Each voting iteration is very efficient since it is just a multiplication of a very sparse matrix with the vector of votes. The overall runtime (including saliency based initialization) takes around 12 seconds per frame on a regular PC, which is comparable to the runtime of [27] on a regular PC, and is much faster than [20, 23, 38], which take several minutes per frame. To handle long videos, we use a sliding temporal window of 200 frames.

4 Initializing the Voting Scheme

Motion Saliency Cues: We seek frames with a "dominant" motion direction, which can with high likelihood, be associated with bg camera motion. By "dominant" motion direction, we mean that either the camera is close to static or that it is translating. Such frames allow more reliable initial (crude) separation into fg/bg pixels than other frames. Then we compute for each of these frames a crude motion saliency map, which is used to initialize the fg/bg votes. For the rest of the frames, we just set the initial fg/bg votes to zero. This frame selection mechanism makes sense, assuming that most videos contain enough frames with simple camera motion (almost static camera or translating), especially if long enough. However, when this is not the case, we resort to visual saliency cues (see below). Another advantage of this frame selection is that it filters out frames with unreliable optical flow (which may happen if the camera moves very fast).

More specifically, we compute optical flow between frames which were down-sampled

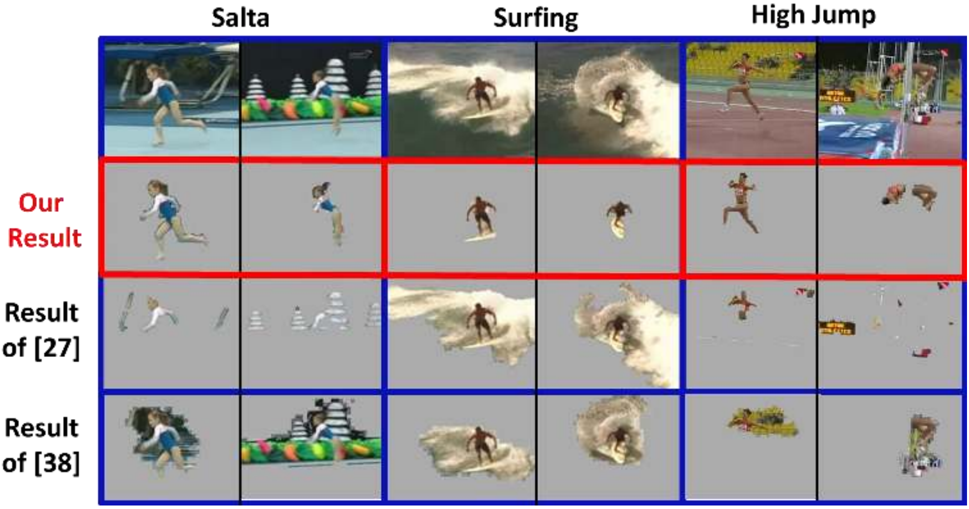

Figure 5: **Extremely difficult videos – visual comparison.** Results on new challenging sequences (with non-rigid background and severe motion blur). For 'Salta' and 'High Jump', we show results of [38] using object selection *without* Grab-Cut (whereas for 'Surfing' with Grab-Cut), since these settings gave best results for [38]. See full videos on `www.wisdom.weizmann.ac.il/~vision/NonLocalVideoSegmentation.html`.

to a low resolution (size 100X100) using the efficient code of [22] (runtime is less than a second per frame). We first look for frames with dominant motion close to zero (i.e., static camera) – we check if the median of optical flow *magnitude* is below a certain threshold (we use 1 pixel). Then we proceed to look for frames in which there is camera translation in some dominant direction. This is done by computing a global histogram of the optical flow *orientations* in each frame (with bins weighted according to the flow magnitude) and check if the maximal bin has a weight above some threshold (we use 0.75). Since only the orientation of the flow is considered and not its magnitude, this applies both to 2D motions and to complex 3D motions with parallax.

We then check if there exist enough (more than 50%) selected frames. If so, we compute for each of those frames a motion saliency map by the following process. For each pixel, we take the flow vectors in its surrounding 5×5 patch, and compute their deviations from the estimated "dominant" motion. In frames with a 'static dominant motion', we compute the deviations of the flow *magnitudes* from zero; in frames with a 'dominant translation', we compute the deviations of the flow *orientations* from the dominant direction. These deviations provide the saliency score of the pixel.

In practice, for improved robustness, both the dominant motion estimation and the saliency map are estimated with respect to frames which are within a temporal radius of 3. In other words, for each frame we compute optical flow with respect to 6 different frames (3 before and 3 after). For those that have a "dominant motion", we estimate the motion-saliency map, and average all these intermediate saliency maps to obtain the final saliency map of the center frame. Examples of motion-saliency maps are shown in the top-left part of Fig. 3.

Visual Saliency Cues: When a "dominant" motion direction is not found in enough frames, this often implies the existence of non-rigid background motion (especially when no dominant motion in many successive frames). However, non-rigid backgrounds (e.g. water in the sea, flickering fire, etc.) often tend to have quite repetitive structures. Thus, in those cases,

Average Segmentation Precision	**Ours**	**[27]**	**[38] (Object Selection + MRF)**	**[38] (Object Selection only)**
1. Birdfall	0.74	0.59	0.71	0.63
2. Girl	0.91	0.73	0.82	0.65
3. Monkeydog	0.78	0.79	0.75	0.74
4. Parachute	0.94	0.91	0.94	0.87
5. Monkey	0.71	0.65	0.62	0.65
6. Soldier	0.83	0.69	0.6	0.35
7. Frog	0.83	0.77	0.74	0.59
8. Worm	0.81	0.74	0.6	0.75
9. Bmx	0.79	0.67	0.17	0.42
10. Drift	0.86	0.75	0.14	0.54
11. Cheetah	0.69	0.28	0.4	0.33
12. HummingBird	0.75	0.52	0.37	0.19
Average Precision (1-12)	**0.8**	**0.67**	**0.57**	**0.56**
Runtime per frame	**12 sec**	**15 sec**	**3.5 min**	**3 min**

Table 1: **Performance evaluation on the full SegTrack Dataset**

we resort to visual saliency, namely, detecting deviations from repetitive spatial structures. This yields an initial rough estimate of the location of moving (salient) foreground objects. These initial votes will obviously be very noisy (see example in the top-right part of Fig. 3), but provide a good enough initialization for the consensus voting. We use the efficient visual saliency implementation of [24] for every frame separately.

5 Experimental Results

We tested our algorithm on various datasets, ranging from benchmark evaluation datasets (SegTrack [21, 35]), on which we empirically compared results to others, to more difficult video sequences (taken from human action datasets like UCF101 [32] and ASLAN [19]), on which current state-of-the-art methods fail to produce reasonable segmentations. For all the experiments, we used the same parameters and the same threshold on the votes in order to obtain the final binary segmentation.

Experiments on the SegTrack Benchmark Dataset: We use the SegTrack Dataset [21, 35] and its ground truth segmentations in order to empirically compare results against [2, 23, 27, 38]. This dataset is considered challenging for most video segmentation methods since it contains diverse scenarios which break classical assumptions.

We first compared results on the entire dataset against two state-of-the-art methods [27, 38] using their publicly available code. A full visual comparison can be found in our **project website** ***www.wisdom.weizmann.ac.il/~vision/NonLocalVideoSegmentation.html***, and some examples are shown in Fig. 4. Table 1 further provides a quantitative comparison. For [38], we report the final results both before and after the Grab-cut refinement phase (sometimes their Grab-cut phase hurts their results). The segmentation quality is measured using average segmentation *precision*, namely: In each frame we measure the intersection divided by the union of the segmentation result and the ground truth segmentation, and average those scores over all frames in the sequence. The average precision measure is insensitive to the object size, thus allowing to compare performance on different sequences. As can be seen in Table 1, we obtain a relative improvement of 20% compared to [27], and 40% com-

Average Segmentation Error (in pixels)	**Ours**	**[2]**	**[38]**	**[23]**	**[27]**
1. Birdfall	146	166	155	189	239
2. Girl	797	1214	1488	1698	2404
3. Monkeydog	361	322	394	472	306
4. Parachute	219	218	220	221	347

Table 2: **Performance evaluation on a subset of the SegTrack Dataset**

pared to [38]. However, [38] is not originally designed to handle *multiple* moving objects (sequences 10-12). When comparing to [38] only on sequences with a single moving object (1-9), we obtain a relative improvement of 23%.

We further compared results to [2, 23]. Since their code is not available, we could compare only on the subset of 4 sequences from the SegTrack dataset reported in their papers. We used their average segmentation error measure (the average number of mislabeled pixels), which is much less informative than the average precision measure. For example, if we set a video frame with a small moving object to be uniformity bg (no fg), the average segmentation error will be small (which seems good), but the precision will be zero! Furthermore, averaging this measure across sequences is meaningless. Results on all methods [2, 23, 27, 38] are reported in Table 2. As can be seen, we obtain comparable or better results than other methods, even with this measure.

Experiments on extremely difficult videos: Finally, to demonstrate the power of our approach, we apply our algorithm to several additional challenging video sequences with very fast and non-rigid foreground motion, which may be of low quality and have severe motion blur, as well as to sequences with highly non-rigid background motion (see Figs. 1,3 and 5). The full sequences and segmentation results appear in our **project website**. For these kinds of video sequences, state-of-the art methods like [27, 38] typically obtain quite poor performance, yielding segmentations which are inconsistent, fractured, or contain significant parts from the background. Our algorithm, on the other hand, is still able to consistently produce meaningful and quite accurate segmentations.

Limitations: In our current implementation, the dominant motion estimation is restricted to detecting dominant translation. Hence, we are currently unable to obtain reliable initializations when there is only camera zoom and/or rotation throughout the entire sequence. Extending the dominant motion estimation to capture also dominant rotation and/or dominant zoom will relax this limitation.

6 Conclusion

We suggest a new approach to Foreground/Background video segmentation based on consensus-voting on a graph of similar ('re-occurring') regions in the video sequence. The power of our approach comes from the non-locality of the consensus voting (of distant 're-occurring' regions), both in space and in time – enabling propagation of diverse and rich information across the entire video sequence. Our approach is fast, accurate, and deals well within a single framework with a wide variety of 'unconstrained' video sequences (e.g., highly non-rigid foreground motion, severe motion blur, non-rigid background motion), leading to state-of-the-art results.

Acknowledgments. The authors would like to thank Shahar Kovalski, Yuval Bahat and Tomer Michaeli. This work was funded in part by the Israel Ministry of Science.

References

[1] X. Bai, J. Wang, D. Simons, and G. Sapiro. Video snapcut: robust video object cutout using localized classifiers. In *SIGGRAPH*, 2009.

[2] D. Banica, A. Agape, A. Ion, and C. Sminchisescu. Video object segmentation by salient segment chain composition. In *ICCV*, 2013.

[3] W. Brendel and S. Todorovic. Video object segmentation by tracking regions. In *ICCV*, 2009.

[4] T. Brox and J. Malik. Object segmentation by long term analysis of point trajectories. In *ECCV*, 2010.

[5] S. Brutzer, B. Hoeferlin, and G. Heidemann. Evaluation of background subtraction techniques for video surveillance. In *CVPR*, 2011.

[6] I. Budvytis, V. Badrinarayanan, and R. Cipolla. Mot - mixture of trees probabilistic graphical model for video segmentation. In *BMVC*, 2012.

[7] I. Choi, M. Lee, and Y. W. Tai. Video matting using multi-frame nonlocal matting laplacian. In *ECCV*, 2012.

[8] J. Costeira and T. Kanande. A multi-body factorization method for motion analysis. In *ICCV*, 1995.

[9] A. Criminisi, G. Cross, A. Blake, and V. Kolmogorov. Bilayer segmentation of live video. In *CVPR*, 2006.

[10] M. Van den Bergh, G. Roig, X. Boix, S. Manen, and L. Van Gool. Online video seeds for temporal window objectness. In *ICCV*, 2013.

[11] P. Dollár and C. Zitnick. Structured forests for fast edge detection. In *ICCV*, 2013.

[12] A. Elgammal, R. Duraiswami, D. Harwood, and L. Davis. Background and foreground modeling using nonparametric kernel density estimation for visual surveillance. In *Proceedings of the IEEE*, pages 1151–1163, 2002.

[13] E. Elhamifar and R. Vidal. Sparse subspace clustering. In *CVPR*, 2009.

[14] K. Fragkiadaki and J. Shi. Video segmentation by tracing discontinuities in a trajectory embedding. In *CVPR*, 2012.

[15] M. Grundmann, V. Kwatra, M. Han, and I. Essa. Efficient hierarchical graph-based video segmentation. In *CVPR*, 2010.

[16] E. Hayman and J. O. Eklundh. Statistical background subtraction for a mobile observer. In *ICCV*, 2003.

[17] M. Irani and P. Anandan. A unified approach to moving object detection in 2d and 3d scenes. *IEEE transactions on Pattern Analysis and Machine Intelligence*, 20, 1998.

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

[19] O. Kliper-Gross, T. Hassner, and L. Wolf. The action similarity labeling challenge. *IEEE transactions on Pattern Analysis and Machine Intelligence*, 34(3):615–621, 2012.

[20] Y. J. Lee, J. Kim, and K. Grauman. Key-segments for video object segmentation. In *ICCV*, 2011.

[21] F. Li, T. Kim, A. Humayun, D. Tsai, and J. Rehg. Video segmentation by tracking many figure-ground segments. In *ICCV*, 2013.

[22] C. Liu. Beyond pixels: Exploring new representations and applications for motion analysis. In *Doctoral Thesis. Massachusetts Institute of Technology*, 2009.

[23] T. Ma and L. J. Latecki. Maximum weight cliques with mutex constraints for video object segmentation. In *CVPR*, 2012.

[24] R. Margolin, A. Tal, and L. Zelnik-Manor. What makes a patch distinct? In *CVPR*, 2013.

[25] P. Ochs and T. Brox. Object segmentation in video: A hierarchical variational approach for turning point trajectories into dense regions. In *ICCV*, 2011.

[26] P. Ochs and T. Brox. Higher order motion models and spectral clustering. In *CVPR*, 2012.

[27] A. Papazoglou and V. Ferrari. Fast object segmentation in unconstrained video. In *ICCV*, 2013.

[28] B. L. Price, B. S. Morse, and S. Cohen. Livecut: Learningbased interactive video segmentation by evaluation of multiple propagated cues. In *ICCV*, 2009.

[29] S. R. Rao, R. Tron, R.Vidal, and Y. Ma. Motion segmentation via robust subspace separation in the presence of outlying, incomplete, or corrupted trajectories. In *CVPR*, 2008.

[30] Y. Ren, C. S. Chua, and Y. K. Ho. Statistical background modeling for non-stationary camera. *Pattern Recognition Letters*, 24:183–196, 2003.

[31] J. Shi and J.Malik. Motion segmentation and tracking using normalized cuts. In *ICCV*, 1998.

[32] K. Soomro, A. R. Zamir, and M. Shah. Ucf 101: A dataset of 101 human actions classes from videos in the wild. *arXiv preprint arXiv: 1212.0402*, 2012.

[33] C. Stauffer and E. Grimson. Learning patterns of activity using realtime tracking. *IEEE transactions on Pattern Analysis and Machine Intelligence*, 22:747–757, 2001.

[34] P. H. S. Torr and A. Zisserman. Concerning bayesian motion segmentation, model averaging, matching and the trifocal tensor. In *ECCV*, 1998.

[35] D. Tsai, M. Flagg, and J. Rehg. Motion coherent tracking with multi-label mrf optimization. In *BMVC*, 2010.

[36] A. Vazques-Reina, S. Avidan, H. Pfister, and E. Miller. Multiple hypothesis video segmentation from superpixel flows. In *ECCV*, 2010.

[37] C. Yuan, G. Medioni, J. Kang, and I. Cohen. Detecting motion regions in the presence of a strong parallax from a moving camera by multiview geometric constraints. *IEEE transactions on Pattern Analysis and Machine Intelligence*, 29:1627–1641, 2007.

[38] Dong Zhang, Omar Javed, and Mubarak Shah. Video object segmentation through spatially accurate and temporally dense extraction of primary object regions. In *CVPR*, 2013.