Efficient Hierarchical Graph-Based Video Segmentation

Matthias Grundmann[1,2] Vivek Kwatra[2] Mei Han[2] Irfan Essa[1]
grundman@cc.gatech.edu kwatra@google.com meihan@google.com irfan@cc.gatech.edu
[1]Georgia Institute of Technology, Atlanta, GA, USA [2]Google Research, Mountain View, CA, USA
http://www.cc.gatech.edu/cpl/projects/videosegmentation

Abstract

We present an efficient and scalable technique for spatio-temporal segmentation of long video sequences using a hierarchical graph-based algorithm. We begin by over-segmenting a volumetric video graph into space-time regions grouped by appearance. We then construct a "region graph" over the obtained segmentation and iteratively repeat this process over multiple levels to create a tree of spatio-temporal segmentations. This hierarchical approach generates high quality segmentations, which are temporally coherent with stable region boundaries, and allows subsequent applications to choose from varying levels of granularity. We further improve segmentation quality by using dense optical flow to guide temporal connections in the initial graph. We also propose two novel approaches to improve the scalability of our technique: (a) a parallel out-of-core algorithm that can process volumes much larger than an in-core algorithm, and (b) a clip-based processing algorithm that divides the video into overlapping clips in time, and segments them successively while enforcing consistency. We demonstrate hierarchical segmentations on video shots as long as 40 seconds, and even support a streaming mode for arbitrarily long videos, albeit without the ability to process them hierarchically.

1. Introduction

Image segmentation aims to group perceptually similar pixels into regions and is a fundamental problem in computer vision. Video segmentation generalizes this concept to the grouping of pixels into *spatio-temporal* regions that exhibit coherence in both appearance and motion. Such segmentation is useful for several higher-level vision tasks such as activity recognition, object tracking, content-based retrieval, and visual enhancement. To illustrate the complexity of video segmentation, we identify three major challenges.

Temporal coherence: Image segmentation approaches applied to each frame independently produce unstable segmentation results, owing to the fact that even small frame-to-frame changes cannot be expressed as a continuous function in general. Consequently, posing video segmentation as spatial region matching problem cannot always enforce consistency of region boundaries over time in the same way as volumetric approaches can. For volumetric techniques, *short-term* coherence (~ 5 frames) can be obtained by generalizing image segmentation methods to a 3-D domain. However, we demonstrate that for *long-term* coherence, it is imperative to go beyond pure pixel-level approaches to a hierarchical approach.

Automatic processing: Segmenting perceptually homogeneous regions in dynamic scenes is related to tracking regions over time. In contrast to tracking, however, it is not known *a priori*, which regions to track, what frames contain those regions, or the time-direction for tracking (forward or backward). We develop a fully automatic approach to segmentation, while leaving selection and tracking of specific regions as a post-process that may involve a user.

Scalability: Given the large amount of pixels or features in a video, video segmentation approaches tend to be slow and have a large memory footprint. Consequently, previous advances concentrate on short video sequences (usually less than a second) or reduce complexity, which can adversely affect long-term temporal coherence. We achieve scalability by employing a graph-based approach with linear time complexity and develop memory-efficient algorithms that enable reliable segmentation of long videos.

Our novel video segmentation algorithm addresses all of the above challenges. We build a 3-D graph from the video volume and generalize Felzenszwalb and Huttenlocher's [7] graph-based image segmentation to obtain an initial over-segmentation of the video volume into relatively small space-time regions. Instead of employing a regular grid graph, we use dense optical flow to modify the graph structure along the temporal dimension, accounting for the distortion of the spatio-temporal volume caused by sweeping motions. We propose a hierarchical segmentation scheme that constructs a region graph from the previous level of segmentation and iteratively applies the same segmentation algorithm. By combining a volumetric over-segmentation with a hierarchical re-segmentation, we obtain regions that exhibit long-term temporal coherence in their identities *and* boundaries. The use of optical flow as a region descriptor for graph nodes further improves coherence. We use a tree-structure to represent the segmentation hierarchy, effectively enabling subsequent systems to choose the desired granularity *post*-segmentation, as opposed to re-running the algorithm with different parameters. Granularity could also be specified as a desired minimum or average region size, which may be application dependent.

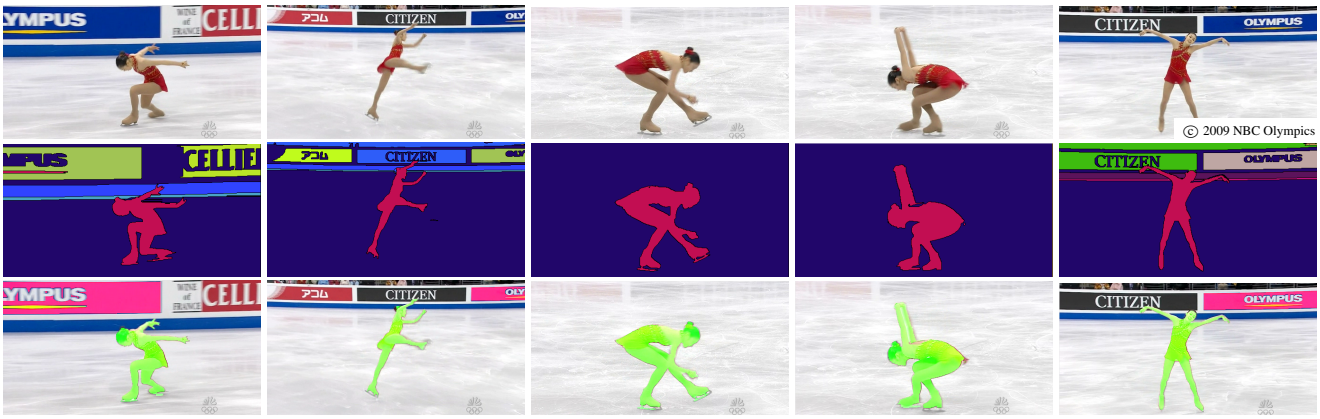


Figure 1: Top: Ice-skater Yu-Na Kim, 2009 World Championships, ©2009 NBC Olympics. Middle: Segmentation result computed in 20 min. Our algorithm is able to segment video of non-trivial length into perceptually distinct spatio-temporal regions. We maintain region identity and clear boundaries over all frames, despite significant motion, camera movement and zoom. Bottom: User-selected regions, Ice-skater (green) selected by a *single* mouse click in *one* frame, Olympus sign (magenta) selected by two clicks. Please see video.

To overcome memory and runtime limitations, we propose two methods (used during over-segmentation): a novel parallel out-of-core algorithm and a clip-based processing approach. These techniques allow us to process long video shots[1] (>40 seconds) fairly efficiently (in ∼20 minutes ≡ 1 *fps*). The clip-based processing can be used for segmenting streaming videos of *arbitrary* length, in case a single (initial) level of the segmentation hierarchy is sufficient.

We demonstrate several applications of our algorithm, including efficient user-selection and tracking of important objects and video tooning. See Fig. 1 for an example.

2. Related work

An obvious step towards video segmentation is to apply image segmentation techniques to video frames without considering temporal coherence [4, 22]. These methods are inherently scalable and may generate segmentation results in real time. However, lack of temporal information from neighboring frames may cause jitter across frames. Freedman and Kisilev [8] applied a sampling-based fast mean-shift approach to a cluster of 10 frames as a larger set of image features to generate smoother results without taking into account temporal information.

Spatio-temporal video segmentation techniques can be distinguished by whether the information from future frames is used in addition to past frames. Causal methods apply Kalman filtering to aggregate data over time, which only consider past data [11, 15]. Paris et al. [14] derived the equivalent tool of mean-shift image segmentation [5] for video streams based on the ubiquitous use of the Gaussian kernel. They achieved real-time performance without considering future frames in the video.

[1]It is reasonable to segment videos only within *shot boundaries*, *i.e.* time instances where the camera cuts to a different scene.

Another class of spatio-temporal techniques take advantage of both past and future data in a video. They treat the video as a 3D space-time volume [12], and typically use a variant of the mean shift algorithm [5] for segmentation [6, 18]. Dementhon [6] applied mean shift on a 3D lattice and used a hierarchical strategy to cluster the space-time video stack for computational efficiency. Wang et al. [19] used anisotropic kernel mean shift segmentation [18] for video tooning. Wang and Adelson [20] used motion heuristics to iteratively segment video frames into motion consistent layers. Tracking-based video segmentation methods generally define segments at frame-level and use motion, color and spatial cues to force temporal coherence [10, 23]. Following the same line of work, Brendel and Todorovic [3] used contour cues to allow splitting and merging of segments to boost the tracking performance.

Interactive object segmentation has recently shown significant progress [1, 2, 9, 13, 16]. These systems produce high quality segmentations driven by user input. We exhibit a similar interactive framework driven by our segmentation.

Our video segmentation method builds on Felzenszwalb and Huttenlocher's [7] graph-based image segmentation technique. Their algorithm is efficient, being nearly linear in the number of edges in the graph, which makes it suitable for extension to spatio-temporal segmentation. We extend the technique to video making use of both past and future frames, and improve the performance and efficiency using a hierarchical framework.

3. Graph-based Algorithm Review

Our spatio-temporal video segmentation builds upon Felzenszwalb and Huttenlocher's [7] graph-based algorithm for image segmentation. We start with a brief overview of their approach. Their objective is to group pixels that ex-

hibit similar appearance, where similarity is based on color difference but also takes the color variation within a region into account. For example, homogeneous regions should not be merged with pixels of different color, but the merging process should be more tolerant to textured regions. Consequently, the notion of *internal variation* of a region is introduced, whereby regions are merged only if their color difference is less than each region's internal variation.

Specifically, for image segmentation, a graph is defined with the pixels as nodes, connected by edges based on 8-neighborhood. Edge weights are derived from the per-pixel normalized color difference. The internal variation $\mathrm{Int}(R)$ of a region R is defined as the maximum edge weight $e_{\max}$ of its Minimum Spanning Tree (MST):

$$\mathrm{Int}\, R := \max_{e \in \mathrm{MST}(R)} w(e)$$

with $w(e)$ being the edge weight of e. The motivating argument is that since the MST spans a region through a set of edges of minimal cost, any other connected set of same cardinality will have at least one edge with weight $\geq e_{\max}$. Therefore $e_{\max}$ defines a lower bound on the maximal internal color variation of the region (see [7] for more details).

We quickly review the original segmentation algorithm. Initially, a graph is constructed over the entire image, with each pixel p being its own unique region $\{p\}$. Subsequently, regions are merged by traversing the edges in a sorted order by increasing weight and evaluating whether the edge weight is smaller than the internal variation of both regions incident to the edge. If true, the regions are merged and the internal variation of the compound region is updated. Since the internal variation of a single node is zero (its MST has no edges), only edges of zero weight can cause an initial merge. To alleviate this behavior the internal variation is substituted with the relaxed internal variation $\mathrm{RInt}(R)$:

$$\mathrm{RInt}(R) := \mathrm{Int}(R) + \delta(R), \quad \text{with } \delta(R) := \frac{\tau}{|R|} \qquad (1)$$

where $|R|$ is the size of region R in pixels, and τ is a constant parameter. This allows regions to be merged even if the weight of the edge connecting them is larger than their internal variations. As the regions grow, $\mathrm{RInt}(R)$ approaches $\mathrm{Int}(R)$ in the limit and is therefore compatible with the original definition. The parameter τ indirectly influences the granularity of the final segmentation, with a larger τ usually leading to larger regions but also with a higher likelihood of missed segmentation boundaries.

4. Hierarchical Spatio-Temporal Segmentation

The above algorithm can be extended to video by constructing a graph over the spatio-temporal video volume with edges based on a 26-neighborhood in 3D space-time. Following this, the same segmentation algorithm can be applied to obtain volumetric regions. This simple approach

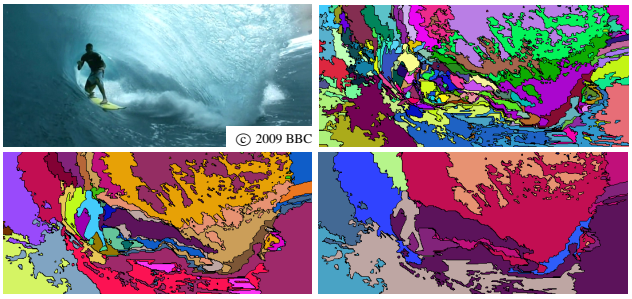


Figure 2: Multiple levels of segmentation hierarchy. Pixel-level over-segmentation on top-right. Larger region granularity in bottom row, with bottom-right having largest regions. Original frame from South Pacific, ©2009 BBC.

generally leads to somewhat underwhelming results due to the following drawbacks.

Firstly, τ does not control the desired region size very well. Increasing τ leads to larger regions but with inconsistent and unstable boundaries, while a small τ leads to a consistent over-segmentation, but the average region size is too small for many applications. Our hierarchical approach solves this problem by eliminating the need to control τ altogether. Instead the desired region granularity can be chosen *post*-segmentation. Secondly, the internal variation $\mathrm{Int}(R)$ reliably discriminates homogeneous from textured regions. However, being simply the maximal color variation, it becomes increasingly unreliable for discriminating between regions of the same type as their sizes grow. We use a *region*-based measure instead of a *pixel*-based measure to overcome this limitation. Thirdly, the segmentation graph of the video has to fit in memory. For ordinary 640×480 resolution, memory issues already occur after one second of video. We address this issue in later sections.

Our hierarchical algorithm begins with a pixel-level segmentation of the graph with a small $\tau \sim 0.02$, to obtain an over-segmentation as shown in Fig. 2. This choice of τ ensures that all important edges in a frame are coincident with region borders. We also enforce a minimum region size by iteratively merging low-cost edges until all regions contain at least 100 voxels. Next, we compute a descriptor for each region in form of its *Lab* histogram[2] with 20 bins along each dimension. These descriptors offer a much richer description of appearance than local per-pixel measurements (even if gathered in a multi-scale manner as in [17]). For instance, textured regions will have flat scattered histograms while homogeneous regions exhibit peaks.

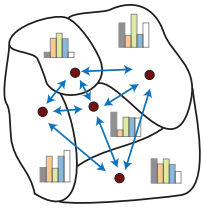


Figure 3: Region Graph.

[2]We experimented with supplementing this descriptor by a spatial gradient histogram but it did not improve results

We use the regions obtained from the over-segmentation to form a graph as indicated in Fig. 3. Each region forms a node and is connected to its incident regions by an edge with a weight based on the difference of their local descriptors. We choose the χ^2 distance between the two histograms to be the edge weight. In contrast to the pixel-level graph, we refer to the so constructed graph as the *region graph*.

The region graph is used to segment the initial set of over-segmented regions into super-regions, *i.e.* regions composed from smaller regions. The super-regions in-turn form a region-graph that can be segmented again. Successively applied, the algorithm computes a hierarchy or a bottom-up tree of regions. At each step of the hierarchy, we scale the minimal region size as well as τ by a factor s. In our implementation $s = 1.1$.

We save the region hierarchy in a tree-structure, which allows selection of the desired segmentation at any desired granularity level. This is much better than manipulating τ directly to control region size. Moreover, the region hierarchy is less prone to segmentation errors and preserves the important region borders. Fig. 2 shows different levels of the hierarchy for an example video frame.

5. Parallel Out-of-Core Segmentation

The algorithm described so far is successful in generating coherent segmentations for a variety of videos. However, as it defines a graph over the entire video volume, there is a restriction on the size of the video that it can process, especially for the pixel-level over-segmentation stage[3]. To overcome this issue, we designed a multi-grid-inspired out-of-core algorithm that operates on a subset of the video volume. Performing multiple passes over windows of increasing size, it still generates a segmentation identical to the in-memory algorithm. Besides segmenting large videos, this algorithm takes advantage of modern multi-core processors, and segments several parts of the same video in parallel.

Consider a connected axis-aligned subset of the nodes of the segmentation graph, *i.e.* a set of nodes that corresponds to a cube of pixels within the video volume, referred to as a *window*. Recall that the original algorithm traverses the edges in a sorted order and evaluates, for each edge, whether the incident regions should be merged or not. If we limit ourselves to process only regions inside a window, there are three types of edges we may encounter:

Boundary edge: If only one region (of the two regions incident upon the edge) is contained in the window, we cannot decide whether or not to merge the incident regions without looking *outside* the window. So we delay our decision and also flag the region inside the window as ambiguous.

[3]For example, the graph of a one second long 25 *fps*, 640×480 video has 7.6 million nodes with 198 million edges (based on 26-neighborhood) and consumes at least 2.2 GB (12 bytes per edges).

Figure 4: Clip-based processing, optical flow edges and region features for segmenting long video clips (28 s) from Goodfellas, ©1990 Warner Bros. Region identity of the actors is preserved under partial occlusions, motion blur and complex background.

Interior edge: If both incident regions are contained inside the window, and not flagged as ambiguous by the step above, we can resolve this edge without making any error (Resolving an edge involves determining whether or not to merge its incident regions. Once an edge is resolved, it is not considered again.)

Exterior edge: If an edge is not incident to any region inside the window, we simply skip it, since edges outside the window have no effect on the regions within it. This implies that for a specific window we only need to sort *its* boundary and interior edges.

We derive our algorithm from these key observations.The last two observations allow us to process windows independently in parallel, while the first observation ensures equivalence to the in-core algorithm. In a single processing thread, we only consider a window of nodes at a time, sorting and processing only the interior and boundary edges of that window. To resolve delayed decisions, we grow the windows by merging them together to obtain a larger window whose edges consists of the unresolved edges from both windows and their common boundary. We iterate this process until all edges are resolved and we obtain our final segmentation.

We implemented both in-core and out-of-core versions of our video segmentation algorithm, which produce the same segmentations, but the in-core algorithm is limited to videos around 30 frames in length. The out-of-core implementation is more scalable, and we have successfully used it to segment videos on the order of 10 seconds in 23 minutes on a laptop. We can achieve further speed-up by employing clip-based processing as described in the next section.

6. Clip-based Processing and Streaming Mode

Our goal is to segment videos beyond the 10 seconds we can achieve by using our out-of-core approach. To that end, we propose a novel clip-based segmentation method that scales well while maintaining temporal coherence, without processing the entire volume at once.

We start by partitioning the video into equally sized clips of n frames ($n = 25$ in our experiments). To preserve tem-

poral coherence, we add a fraction (*one-third*) of the last frames from the previous clip to the current one.

By using a graph representation and observing that zero weight edges always cause a merge, we are able to *constrain* the solution of clip $i+1$ to be coherent in the overlap region with clip i. After constructing the 3D graph for clip $i+1$, we scale the edge weights $w(e_{p,q})$ in the overlap region by:

$$S(e_{p,q}) = \begin{cases} \alpha & \text{if } R_i(p) = R_i(q), \\ 100 \times (1-\alpha) & \text{otherwise,} \end{cases} \quad (2)$$

with $\alpha \in [0,1]$, $\alpha = 0$ for the first frame in the overlap, $\alpha = 1$ for the last frame in the overlap, and linear in between. The function $R_i(p)$ assigns the region id from the segmentation of the previous clip i to each voxel p in the overlap. As a result all edges within the same region have zero weight and all edges along different regions will have a high weight. This forces the segmentation of the clip $i+1$ to agree with the segmentation of clip i on the first frame of the overlap, while being able to diverge more and more from it in later frames. Therefore, each clip can be segmented quasi-independently while constraining the segmentation over all clips to be temporally consistent. Note that the parallel out-of-core algorithm described earlier can still be used for each clip. Fig. 4 shows frames from a single 28 second shot segmented using clip-based processing.

If we decide against additionally performing hierarchical segmentation, *i.e.* if the initial pixel-level segmentation is sufficient, then the clip-based processing allows us to segment videos of arbitrary length in a streaming fashion, one clip at a time. We only need to save the last segmentation in the overlap region, as well as a lookup table to associate regions in the overlap region with the previous region identities. An example of our streaming mode as well as a comparison to hierarchical segmentation is shown in Fig. 7. In order to avoid over-segmenting in streaming mode, we force a minimum size for all spatio-temporal regions (8000 *voxels* in our experiments). We establish this by repeatedly iterating over edges in sorted order and merging regions until their size is larger than the minimum size.

7. External Optical Flow

While our video segmentation algorithm described so far is self-contained, the supplemental use of *dense* optical flow can considerably improve segmentation results. We make use of optical flow in two ways. Firstly, instead of connecting a voxel (i,j,t) to its immediate 9 neighbors $(i+\mu, j+\nu, t-1), \mu,\nu \in \{-1,0,1\}$ in the previous frame, we connect it to its 9 neighbors along the backward flow vector (u,v), *i.e.* $(i+u(i,j)+\mu, j+v(i,j)+\nu, t-1)$. This is a generalization of prior grid-based volumetric approaches, something we are only able to achieve using a graph representation. Secondly, we use optical flow as a

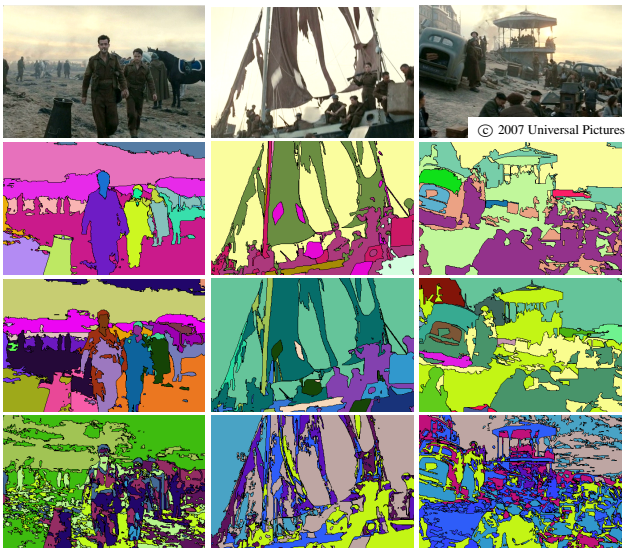


Figure 5: Comparison of segmentation with and without optical flow edges and features for a complex scene. (from Atonement, ©2007 Universal Pictures) From top to bottom: original frame, with flow features and edges, with flow features only, without flow.

feature for each region during hierarchical segmentation. As optical flow is only consistent within a frame, we use a per-frame flow-histogram discretized w.r.t. to the angle, similar to SIFT. We use 16 orientation bins and accumulate each bin by the magnitudes of flow-vectors within that bin. Matching the flow-descriptors of two regions then involves averaging the χ^2 distance of their normalized per-frame flow-histograms over time.

We combine the χ^2 distance of the normalized color histograms $d_c \in [0,1]$ with the χ^2 distance of the normalized flow histograms $d_f \in [0,1]$ by

$$(d_c, d_f) \rightarrow \big(1-(1-d_c)(1-d_f)\big)^2. \quad (3)$$

This function is close to zero if both distances are close to zero, and close to one if either one is close to one. In our paper we use the GPU-based dense optical flow of Werlberger et al. [21] as it runs close to real-time and produces very good results on the Middlebury Optical Flow Dataset. Fig. 5 shows a comparison between segmentation results with and without optical flow for a complex dynamic scene.

8. Results

We apply our segmentation algorithm to a wide range of videos, from classic examples to long dynamic movie shots, studying the contribution of each part of our algorithm. Fig. 8 demonstrates segmentation results on four video clips. Each region is given a unique color to evaluate long-term coherence and boundary consistency. Our segmentation exhibits consistent region identity and stable boundaries under conditions such as significant motion (water-skier rotating around own axis, first row) and dynamic

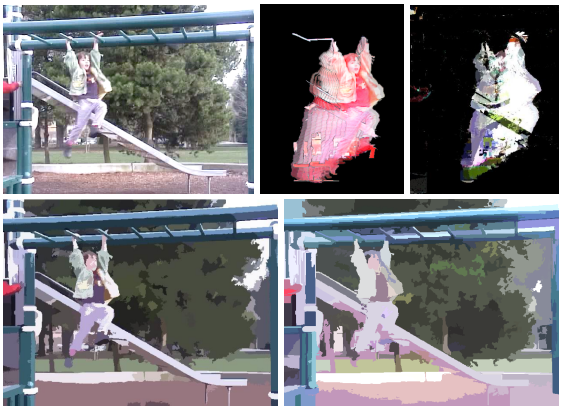


Figure 6: Top-Left: "Lena on Monkey Bars", courtesy of Michael Cohen. Bottom-Left: Result of Wang et al. [18, 19]. Bottom-Right: Our tooned segmentation result is similar but features better region boundaries, indicated by evaluating the boundary of the girl over 10 frames (top middle). Wang et al.'s feature based mean-shift approach can lead to spatially disconnected regions (top right) while our regions are temporally *and* spatially connected.

surfaces like water, partial occlusions (numbers on football players, second row) camera motion (panning down, third row) and illumination changes (explosion in the background, fourth row).

We compare our results against others on the classic flower garden sequence in Fig. 9a. Our segmentation (2^{nd} from left) successfully separates the motion layers while providing more details than other approaches (compare outlines of the houses to Wang et al. [20]), 3^{rd} from left). We track region identity consistently in contrast to other approaches; compare to identity change of the houses on the right in Khan and Shah's [10] result (4^{th} from left) and identity of the sky, houses and the flower field in Brendel and Todorovic's [3] result (5^{th} from left). A finer segmentation of the flower garden sequence is shown in Fig 9b on the left, with tooned version to the right of it obtained by averaging the color over spatio-temporal regions. This is an example of selecting the desired granularity *post*-segmentation.

Fig. 6 illustrates an important difference of our approach to techniques that segment in feature space, such as Wang et al.'s [18]. While our tooned segmentation result looks similar, a detailed analysis of the outline of the segmented girl shows that our approach ensures temporal *and* spatial connectedness of regions. We believe that region connectedness is a crucial property for several video analysis algorithms and conforms to human perception.

We study the effect using optical flow as a region feature in Fig. 5 on a 40 second movie sequence. Mostly set in a grayish tone, it is a hard sequence to segment using color alone (4^{th} row). Adding optical flow as a region feature differentiates between perceptually similar, but independently moving segments (3^{rd} row). The additional use of optical

Figure 7: Comparison to Paris [14]. Column 1: 3 frames from a grayscale sequence of about 100 frames. Column 2: Paris result [14]: note that regions such as the windscreen and body of the truck, and the jumpsuit of the woman change identity over time. Column 3: Our hierarchical segmentation: has greater temporal coherence and reliably segments fine details as well as homogeneous regions. Column 4: Our streaming mode result: also temporally coherent, but lacks the perceptual boundaries that our hierarchical segmentation is able to achieve.

flow edges in the graph allows tracking region boundaries more consistently and leads to our final result (2^{nd} row).

A validation of our streaming mode as well as a comparison to Paris's [14] streaming mean-shift approach is illustrated in Fig. 7. Our algorithm achieves better temporal coherence in both streaming mode as well as hierarchical segmentation. However, Paris [14] algorithm runs in real-time on gray-scale video while our streaming algorithm achieves 1 *fps* on color video (on a dual-core 2.4GHz laptop with 4GB RAM) with clip-based processing. Fig. 7 also evaluates the effectiveness of hierarchical segmentation compared to pixel-level segmentation. Perceptual boundaries are better maintained, fine details preserved and similar pixels are successfully grouped into homogeneous regions in the former. The effectiveness of our clip based processing to segment long video sequences coherently is displayed in Fig. 1 (30 s), Fig. 4 (28 s) and Fig. 5 (40 s).

Our technique can be used as a pre-processing step for various video based algorithms that could benefit from temporally coherent segmentation, such as video editing or selective filtering such as video tooning. We demonstrate some of these applications in Fig. 1, Fig. 6, Fig. 9b and Fig. 10. We believe that the automatic computation of spatio-temporal regions combined with user-guided selection of the granularity post-segmentation is a valuable tool for video editing as well as content analysis.

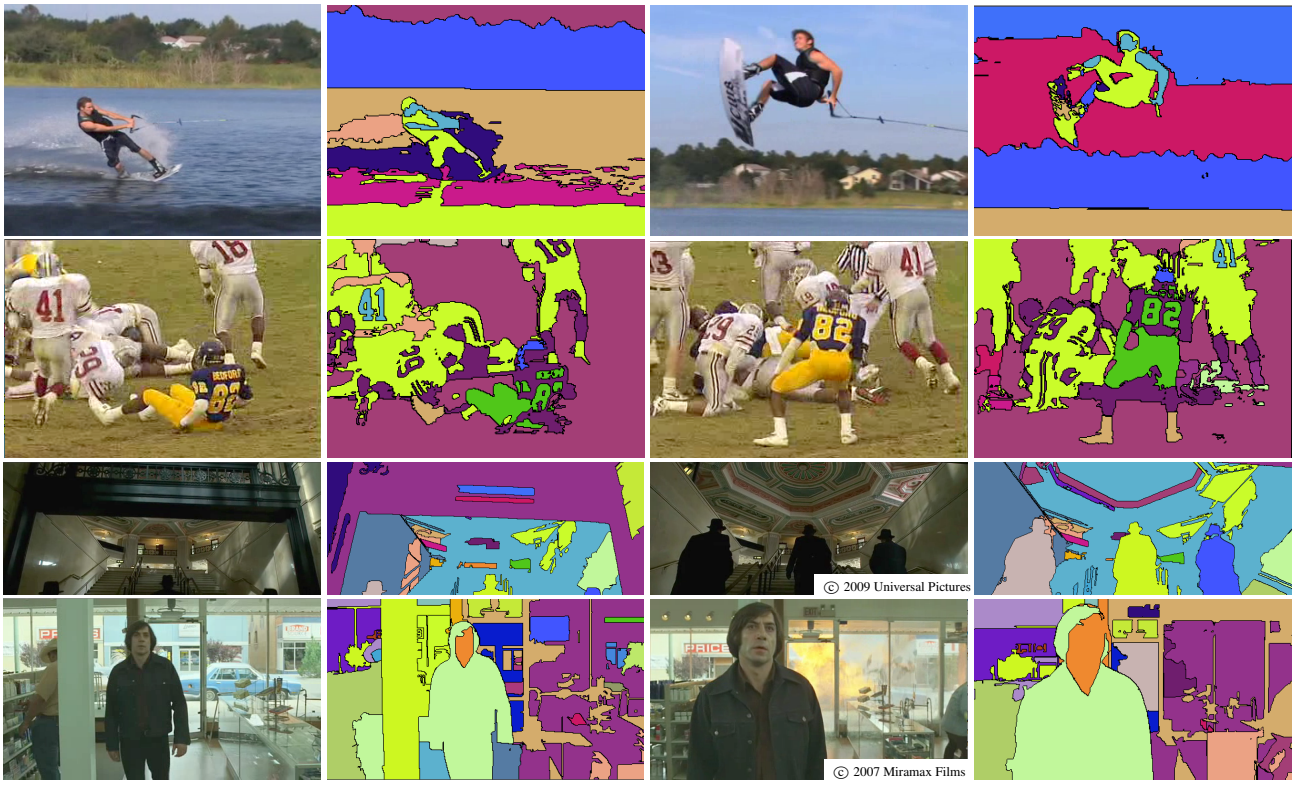
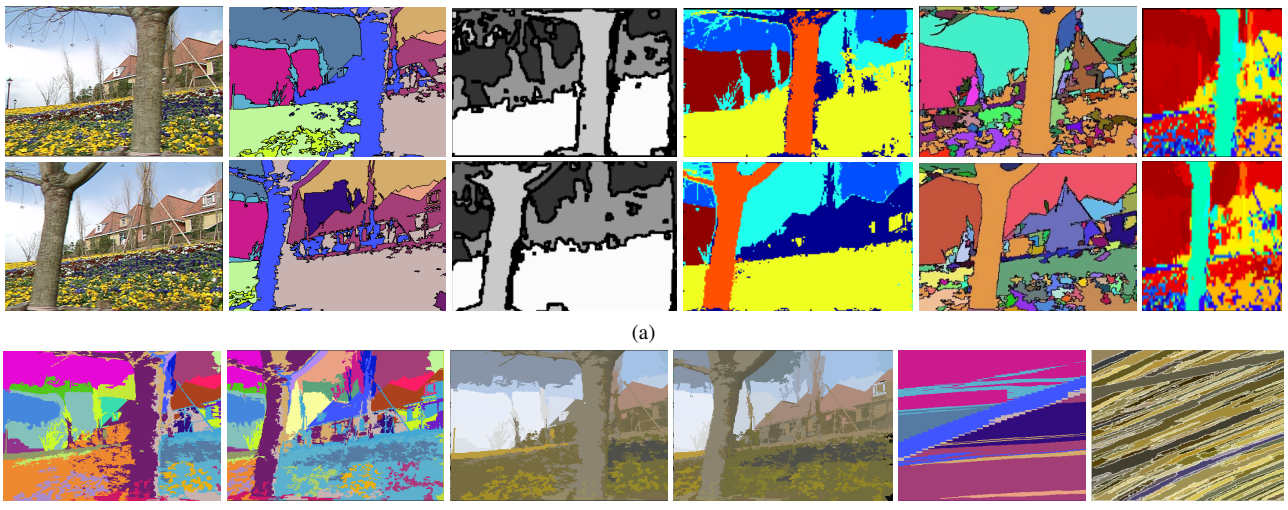


Figure 8: Spatio-temporal segmentation. Shown are two frames each from video sequences with their corresponding segmentations. Same color denotes the same spatio-temporal region. Region boundaries and identity are tracked reliably (note body and skin of the water-skier, football player numbers and persons in bottom videos. 3rd row: from Public Enemies, ©2009 Universal Pictures, 4th row: from No country for old men, ©2007 Miramax Films.



(a)

(b)

Figure 9: Flower garden sequence ($\sim$ 30 frames apart). (a) From left to right: Original sequence, our segmentation, Wang et al.'s [20] result, Khan and Shah's [10] result, Brendel and Todorovic's [3] results, Dementhons' [6] result. Our segmentation result is coherent over all 30 frames. Brendel and Todorovic's [3] result (5th from left) changes region identity noticeably (sky, houses and flower field) while Khan and Shah's [10] result (4th from left) is inconsistent on the right hand side (houses identity changes). Our segmentation retains important details like the houses in the background while Wang et al.'s [20] (3rd from left) as well as Dementhons' [6] result (right-most) do not show the same clear-cut boundaries (*e.g.* the roof of the houses). Dementhons' [6] result (right-most) also exemplifies a typical property when segmenting in feature space: Regions are not spatially connected and exhibit significant holes making them hard to use for later analysis stages. (b) A finer granularity of our segmentation (left 2 frames), the consistent tooned result by averaging the color over the spatio-temporal regions (middle two frames), and a time-slice from our segmentation (5th from left) compared to the time-slice of Wang et al. [18] (last frame). Our time-slice is less fragmented indicating better temporal coherence.

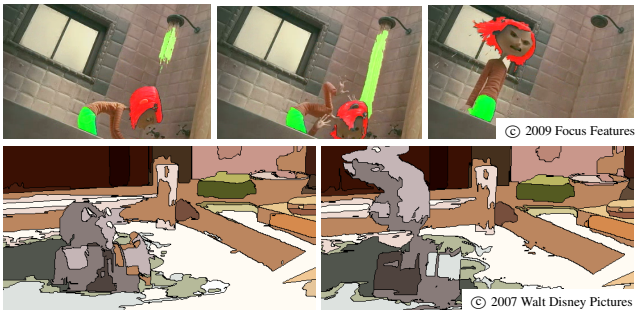


Figure 10: Applications of our algorithm. Top: Spatio-temporal tracking of user-selected regions (shown in green and red) over multiple frames. Note temporal coherence even in presence of fast motions (girl) and dynamic shapes (water). Original frame from: Coraline, ©2009 Focus Features. Bottom: Tooning result by color averaging over spatio-temporal regions. Original frame from: Ratatouille, ©2007 Walt Disney Pictures.

Figure 11: Failure case. Encoding artifacts cause fragmentation (wall on the right). The algorithm can be sensitive to smooth illumination changes (background) and hard shadows (on the face). Original frame from Public Enemies, ©2009 Universal Pictures.

9. Conclusion, Limitations and Future Work

We propose a novel approach to segment dynamic scenes in video, achieving a high-quality, hierarchical segmentation that allows users and applications to select the desired granularity after segmentation. Our algorithm is computationally and memory efficient, reposes the original segmentation algorithm to allow parallel and out-of-core processing and scales well to processing videos of non-trivial length.

We have tested our approach on a wide variety of challenging videos, studied the individual components of our algorithm, and explored interesting applications that build upon segmentation. We believe our algorithm provides an effective solution to an important low-level vision problem.

Currently, our algorithm can be sensitive to MPEG encoding artifacts and smooth illumination changes as displayed in Fig. 11. In the future, we plan to enforce shape consistency over time, to deal with occlusions or partial scene changes. Our current average processing times are around 20 min for a 40 s video, and we hope to move our parallel out-of-core algorithm to GPUs.

References

[1] X. Bai, J. Wang, D. Simons, and G. Sapiro. Video snapcut: robust video object cutout using localized classifiers. *ACM SIGGRAPH*, 28, 2009. 2

[2] Y. Boykov and G. Funka-Lea. Graph cuts and efficient n-d image segmentation. *Int. J. Comput. Vision*, 70(2), 2006. 2

[3] W. Brendel and S. Todorovic. Video object segmentation by tracking regions. In *ICCV*, 2009. 2, 6, 7

[4] J. Chen, S. Paris, and F. Durand. Real-time edge-aware image processing with the bilateral grid. *SIGGRAPH*, 07. 2

[5] D. Comaniciu and P. Meer. Mean shift: a robust approach toward feature space analysis. *IEEE PAMI*, 24(5), 2002. 2

[6] D. DeMenthon. Spatio-temporal segmentation of video by hierarchical mean shift analysis. In *Statistical Methods in Video Processing Workshop (SMVP)*, 2002. 2, 7

[7] P. Felzenszwalb and D. Huttenlocher. Efficient graph-based image segmentation. *IJCV*, 59(2), 2004. 1, 2, 3

[8] D. Freedman and P. Kisilev. Fast mean shift by compact density representation. In *CVPR*, 2009. 2

[9] Y. Huang, Q. Liu, and D. Metaxas. Video object segmentation by hypergraph cut. In *CVPR*, 2009. 2

[10] S. Khan and M. Shah. Object based segmentation of video using color, motion and spatial information. In *CVPR*, 2001. 2, 6, 7

[11] J. Kim and J. Woods. Spatiotemporal adaptive 3-d kalman filter for video. *IEEE Trans. on Image Proc.*, 6, 1997. 2

[12] A. Klein, P. Sloan, A. Finkelstein, and M. Cohen. Stylized video cubes. In *Symp. on Computer Animation*, 2002. 2

[13] Y. Li, J. Sun, and H.-Y. Shum. Video object cut and paste. In *SIGGRAPH '05: ACM SIGGRAPH 2005 Papers*, pages 595–600, New York, NY, USA, 2005. ACM Press. 2

[14] S. Paris. Edge-preserving smoothing and mean-shift segmentation of video streams. In *ECCV*, 2008. 2, 6

[15] A. Patti, A. Tekalp, and M. Sezan. A new motion-compensated reduced-order model kalman filter for space-varying restoration of progressive and interlaced video. *IEEE Transactions on Image Processing*, 7, 1998. 2

[16] B. Price, B. Morse, and S. Cohen. Livecut: Learning-based interactive video segmentation by evaluation of multiple propagated cues. In *ICCV*, 2009. 2

[17] E. Sharon, A. Brandt, and R. Basri. Fast multiscale image segmentation. In *CVPR*, 2000. 3

[18] J. Wang, B. Thiesson, Y. Xu, and M. Cohen. Image and video segmentation by anisotropic kernel mean shift. In *ECCV*, 2004. 2, 6, 7

[19] J. Wang, Y. Xu, H.-Y. Shum, and M. F. Cohen. Video tooning. In *SIGGRAPH*, 2004. 2, 6

[20] J. Y. A. Wang and E. H. Adelson. Representing moving images with layers. *IEEE Trans. on Image Proc.*, 3(5):625–638, 1994. 2, 6, 7

[21] M. Werlberger, W. Trobin, T. Pock, A. Wedel, D. Cremers, and H. Bischof. Anisotropic Huber-L1 optical flow. In *BMVC*, London, UK, 2009. 5

[22] H. Winnemöller, S. C. Olsen, and B. Gooch. Real-time video abstraction. *ACM SIGGRAPH*, 25(3):1221–1226, 2006. 2

[23] C. L. Zitnick, N. Jojic, and S. B. Kang. Consistent segmentation for optical flow estimation. In *ICCV*, 2005. 2