Spatiotemporal Closure

Alex Levinshtein[1], Cristian Sminchisescu[2], and Sven Dickinson[1]

[1] University of Toronto
{babalex,sven}@cs.toronto.edu
[2] University of Bonn
cristian.sminchisescu@ins.uni-bonn.de

Abstract. Spatiotemporal segmentation is an essential task for video analysis. The strong interconnection between finding an object's spatial support and finding its motion characteristics makes the problem particularly challenging. Motivated by closure detection techniques in 2D images, this paper introduces the concept of *spatiotemporal closure*. Treating the spatiotemporal volume as a single entity, we extract contiguous "tubes" whose overall surface is supported by strong appearance and motion discontinuities. Formulating our closure cost over a graph of spatiotemporal superpixels, we show how it can be globally minimized using the parametric maxflow framework in an efficient manner. The resulting approach automatically recovers coherent spatiotemporal components, corresponding to objects, object parts, and object unions, providing a good set of multiscale spatiotemporal hypotheses for high-level video analysis.

1 Introduction

Spatiotemporal segmentation refers to the task of partitioning a video sequence into coherently moving objects. While such partitioning does not correspond to a full video interpretation, it can prove to be an essential component for higher-level tasks, including tracking, object recognition, video retrieval, or activity recognition. What makes spatiotemporal segmentation challenging is the strong coupling that exists between the estimation of an object's spatial support and the estimation of its motion parameters. On one hand, local motion estimates may be unreliable, especially in untextured regions, and larger spatial support is needed for accurate motion estimation. On the other hand, appearance alone may not be enough to recover the object's spatial support in cases of heterogeneous object appearance or low contrast with the background, and we may need to rely on motion to define the correct spatial support for objects. This chicken and egg problem forces most spatiotemporal segmentation techniques to resort to restrictive modeling assumptions or suboptimal solutions to the problem.

This paper introduces a novel spatiotemporal grouping approach with minimal modeling assumptions and a globally optimal algorithm for segmentation. Similar to prior methods, we represent the whole video stack using a graph with node affinities encoding appearance and motion similarity. In this manner, our

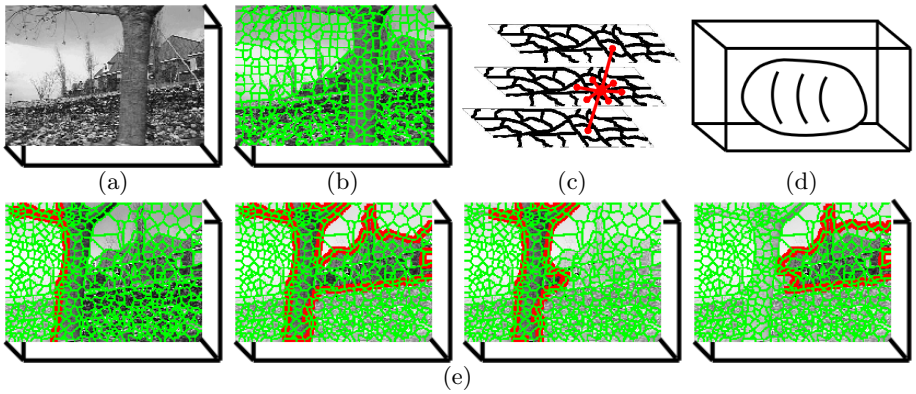


Fig. 1: Overview of our approach illustrated on the flower garden sequence. (a) Spatiotemporal volume; (b) Spatiotemporal superpixels; (c) Superpixel graph with edges encoding appearance and motion affinity; (d) Optimizing our spatiotemporal closure corresponds to finding a closed surface cutting low affinity graph edges; (e) Our optimization framework results in multiple multiscale hypotheses, corresponding to objects, object unions, and object parts.

segmentation approach encodes global information without making overly restrictive modeling assumptions. A number of methods approach the problem in a similar manner. However, they commonly employ greedy clustering algorithms [1–5], resort to approximate methods for optimizing global NP-hard costs [6–8], assume a known number of objects [6, 7, 9, 8], or work with pixels whose small spatial support results in unreliable motion or appearance features [9, 5].

We propose to solve the video segmentation problem by extending the concept of 2D image closure into the spatiotemporal domain, where the perception of closure would correspond to 3D "tubes" whose overall boundary is strongly supported by appearance and motion discontinuities. Fig. 1 illustrates the main steps of our approach. Building on our 2D closure detection framework [10], we formulate spatiotemporal closure detection inside a spatiotemporal volume (Fig. 1a) as selecting a subset of spatiotemporal superpixels whose collective boundary falls on such discontinuities (Fig. 1b). Our spatiotemporal superpixels, based on the framework of [11], provide good spatiotemporal support regions for the extraction of appearance and motion features, while limiting the undersegmentation effects that plague other superpixel extraction techniques due to their lack of compactness and temporal stability. We proceed by forming a superpixel graph whose edges encode appearance and motion similarity of adjacent superpixels (Fig. 1c). Closure detection is posed as the optimization of a global, unbalanced normalized cuts (Ncuts) cost over the superpixel graph (Fig. 1d). Similar to [12], we optimize our unbalanced Ncuts cost with the parametric maxflow approach [13] that is not only able to efficiently find a globally optimal closure solution, but returns multiple closure hypotheses (Fig. 1e). This not only eliminates the need for estimating the number of objects in a video sequence, as all objects exhibiting sufficient closure are extracted, but can result in hypotheses

that oversegment objects into parts or merge adjacent objects. The use of such multiscale hypotheses was shown to facilitate state-of-the-art object recognition in images [14]. Similarly, multiple spatiotemporal segmentation hypotheses can serve tasks such as action recognition, video synopsis and indexing [15].

In the following sections, we begin by reviewing related work on spatiotemporal segmentation (Section 2). Next, in Section 3, we introduce our problem formulation. It is here that our cost function is described. Section 4 details all the steps of our algorithm. In Section 5, we evaluate our framework, comparing different superpixel affinities and evaluating against an alternative optimization framework. Finally, in Section 6, we draw conclusions and outline our plans for future work.

2 Related Work

A full interpretation of a dynamic scene is a great challenge in computer vision. Tracking methods often adopt a high-level probabilistic scene representation, where objects are modeled with low-dimensional state vectors whose probability at any given instance is a function of the observed data and the temporal dynamics. Inferring object states in real world motion sequences is a difficult task in the face of occlusion, camera motion, and variability in object appearance, dynamics, and shape. As a result, tracking techniques are forced to restrict their models of observed data likelihood and motion [16, 17], or resort to approximate techniques to infer object states [18, 19]. In contrast, our focus in this paper is on spatiotemporal segmentation. Unlike tracking, where objects are represented at a high level, spatiotemporal segmentation is a low-level task that aims to automatically extract precise object boundaries given generic perceptual grouping regularities, such as similarity, proximity and common fate.

Spatiotemporal segmentation methods can be divided into two categories, layer-based approaches and generic segmentation techniques (a good review is provided in Megret and Dementhon [20]). In the first category, a scene is represented using overlapping layers, with each layer capturing a coherently moving object or part [21–25]. Most such approaches are limited by either assuming a fixed number of layers, assuming a restricted motion model per layer, or resorting to suboptimal techniques that iteratively estimate the spatial extent and the motion of each layer. Nevertheless, this strong global model of a scene enables layer-based methods to successfully segment objects in video sequences in the presence of occlusion, appearance changes, and other effects. In this work, however, we will focus on more generic, less restrictive models for spatiotemporal segmentation.

The second category of approaches does not enforce strong models and attempts to segment a video based on generic spatiotemporal information. Methods mainly differ in their segmentation algorithms and their treatment of the spatiotemporal volume, with some methods analyzing the volume in a framewise manner and others treating it as a single 3D entity. One set of techniques models moving objects with active contours. In Bascle and Deriche [26], motion is mod-

eled with a global warp which is found by correlating internal region appearance in successive frames. After the warp, however, only appearance information is used to update the region's contour. Paragios and Deriche [27] propose a more elegant geodesic active contour formulation. Unlike [26], both motion and appearance information are used in active contour evolution and their level-set framework enables them to easily handle the splitting and merging of contours. However, they assume a static background model to facilitate automatic contour initialization and better tracking. A similar method is proposed by Chung et al. [28], who employ the EM framework to iterate between region motion estimation and segmentation using active contours, but unlike [27] do not rely on a static background. Cremers and Soatto [29] propose a more holistic approach and treat the spatiotemporal volume as a single entity instead of working with pairs of frames. However, their approach provides no automatic initialization and does not estimate the number of objects in a scene.

A different set of techniques opts for a more bottom-up approach, and finds spatially and temporally coherent clusters. Similar to methods based on active contours, some of these approaches handle the spatiotemporal volume in a frame-wise fashion [2, 1, 4, 3, 30]. While such techniques are more applicable to realtime segmentation, some opt to treat the video stack as a single entity facilitating more global constraints. Dementhon [5] and Greenspan et al. [9] are examples of two techniques that represent videos with distributions in a low-dimensional feature space (7D in [5] and 6D in [9]). While it enables them to efficiently segment videos by employing non-parametric (a mean-shift-based technique in [5]) or parametric (GMM in [9]) clustering, such low-dimensional models may prove too restrictive for many motion sequences. Instead of explicitly modeling video sequences in some Euclidean space, segmentation can be formulated as an optimization of a global cost that is based on pairwise similarities between neighboring points in the spatiotemporal stack. For example, in [6, 7], video segmentation is formulated as a normalized cuts problem, further extended by Huang et al. [8] to handle more global interactions. Our approach falls under this category and is closely related to Ncuts as it also defines a global cost function. However, unlike Ncuts-based techniques that are forced to resort to approximate solutions, we are able to find an exact global optimum of our cost. Moreover, the number of clusters does not have to be specified a priori, as we automatically detect a multiscale set of spatiotemporal clusters.

3 Problem formulation

We formulate the detection of spatiotemporal segments as a superpixel selection problem. To that end, we define our closure cost to be the unbalanced normalized cuts cost over a superpixel graph. Out of an exponential number of superpixel subsets we will efficiently select subsets corresponding to coherent spatiotemporal segments.

Given a superpixel segmentation of every frame in a video, we start by building a superpixel graph with spatial and temporal connections. Let $\boldsymbol{X}$ be an

indicator vector for all the superpixels across all frames, with each element being in the set $\{0, 1\}$. We connect each superpixel to its spatial and temporal neighbors and define an affinity W_{ij} for each pair of neighboring superpixels i and j, encoding the similarity of the two superpixels. Setting $D_i = \sum_j W_{ij}$, we optimize the following closure cost:

$$C(\boldsymbol{X}) = \frac{cut(\boldsymbol{X})}{volume(\boldsymbol{X})} = \frac{\sum_{ij} X_i(1 - X_j)W_{ij}}{\sum_i D_i X_i} = \frac{\sum_i D_i X_i - 2\sum_{i<j} X_i X_j W_{ij}}{\sum_i D_i X_i} \tag{1}$$

where $cut(\boldsymbol{X})$ is the sum of the affinities of all the edges between selected and unselected superpixels, and $volume(\boldsymbol{X})$ is the sum of all the affinities originating from the selected superpixels. Minimizing the ratio $C(\boldsymbol{X})$ is equivalent to minimizing the numerator $cut(\boldsymbol{X})$ while maximizing the denominator $volume(\boldsymbol{X})$. The cut between selected and unselected superpixels is small when selected superpixels are strongly separated from the rest in terms of their appearance and motion. Normalization by volume pushes the solution towards large and compact subsets of superpixels that are homogeneous in terms of appearance and motion.

The above is called the unbalanced normalized cuts cost. It is similar to our 2D closure cost in [10], with the exception that the numerator measures the cut instead of the gap and is normalized by affinity volume instead of area. That said, we will show that the affinities W_{ij} can also include the length of the boundary between superpixels or their area to give larger superpixels a greater influence.

Unlike the standard normalized cuts cost, which is NP-hard to optimize, our closure cost can be minimized efficiently using parametric maxflow [13]. In parametric maxflow, the problem of ratio minimization is converted to minimizing a parametrized difference of the numerator and the denominator. For the cost in Eqn. 1, the parametric maxflow cost is:

$$\begin{aligned} C(\boldsymbol{X}, \lambda) &= cut(\boldsymbol{X}) - \lambda \cdot volume(\boldsymbol{X}) \\ &= \sum_i D_i X_i - 2\sum_{i<j} X_i X_j W_{ij} - \lambda \sum_i D_i X_i \end{aligned} \tag{2}$$

Different λ's correspond to different weights of the cut against the affinity volume. Parametric maxflow can optimize the above parametrized cost, efficiently finding all the different *breakpoints* (interval boundaries) of λ between which the optimal solution X is fixed, resulting in an increasing sequence of breakpoints $\lambda_0, \lambda_1, \lambda_2, \ldots$. Kolmogorov et al. [13] show that while the solution $\boldsymbol{X}^*$ in range $0 \leq \lambda \leq \lambda_0$ corresponds to the global minimum of $C(\boldsymbol{X})$, consecutively larger breakpoints $\lambda_1, \lambda_2, \ldots$ are also related to ratio optimization. In fact, the optimal solution $\boldsymbol{X}^i$ of $C(\boldsymbol{X}, \lambda)$ in the interval $[\lambda_i, \lambda_{i+1}]$, is also an optimal solution of $\min_{volume(\boldsymbol{X}) \geq T} C(\boldsymbol{X})$, where $T = volume(\boldsymbol{X}^i)$. Therefore, employing parametric maxflow results in several solutions where optimal cuts are found with increasing affinity volume constraints. We refer the reader to [13] for more details on the parametric maxflow method.

4 Algorithm details

Our algorithm consists of several stages. We start by extracting the superpixels for each frame of the video. Subsequently, we construct a superpixel graph where each superpixel is connected to its spatial and temporal neighbors. Each superpixel edge is assigned an affinity that measures the degree of superpixel similarity. Once the graph is built, optimal cuts are found using parametric maxflow. Finally, we post-process the solutions to detect connected components, remove similar or spurious results, and generate other potentially good solutions. The following subsections describe each of these stages[3].

4.1 Superpixel Extraction

We begin by extracting superpixels from every frame using the TurboPixels approach of Levinshtein et al. [11]. Instead of using the algorithm in its raw form, we modify it to obtain more temporally coherent superpixels. We start by extracting superpixels in the first frame using the original form of the superpixel algorithm in [11]. Instead of reseeding the superpixels in the next frame on a regular grid, we use the current frame's superpixel to drive the seeding procedure. To that end, we first compute the optical flow using the Lucas-Kanade (LK) algorithm. The LK algorithm returns the flow for every pixel in every frame, together with a measure of reliability for each pixel flow. For every superpixel, we compute a weighted average of the flow over all the reliable pixels, where pixels that are closer to the superpixel centroid have larger weights. Superpixels with an insufficient number of reliably flowing pixels get a flow of $(0, 0)$. The result is a *superpixel flow*, with motion flow vector $\boldsymbol{V_i}$ for every superpixel i (Fig. 2).

Taking the superpixel flow for every superpixel, we project the center of each superpixel to the next frame according to the computed flow. These projected centers serve as the initial seeds for the superpixel evolution in the next frame. We repeat this process for all the frames in the video, giving us a much more temporally stable superpixel segmentation. In addition, we also modify the superpixel algorithm to use a Pb-based [31] affinity rather than the original grayscale gradient-based affinity proposed in [11][4].

4.2 Superpixel Affinity

Once the superpixels are extracted, we form spatial and temporal edges in the superpixel graph. Every edge is assigned an affinity W_{ij} that measures the similarity of the two superpixels (Fig. 1c). To form spatial connections, we find the

[3] See the Approach Overview section at `http://www.cs.toronto.edu/~babalex/SpatiotemporalClosure/supplementary_material.html` for a graphical overview of the method.

[4] See the Superpixel Extraction section at `http://www.cs.toronto.edu/~babalex/SpatiotemporalClosure/supplementary_material.html` for a better visualization of superpixel extraction.

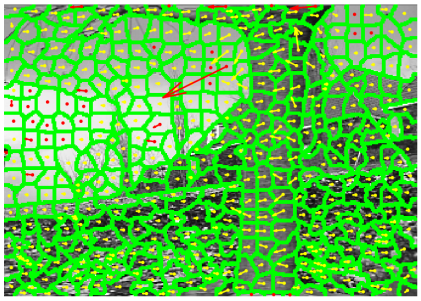


Fig. 2: Superpixel flow. The arrow within each superpixel indicates the motion flow vector of this superpixel. Yellow arrows indicate reliable flows, while red arrows correspond to unreliable flows.

immediate spatial neighbors of each superpixel in each frame. Spatial neighbors of superpixel i are defined as superpixels in the same frame that share some boundary with superpixel i. The formation of temporal connections follows the same approach as was used in the superpixel extraction technique. Each superpixel in frame f (except the superpixels in the last frame) is connected to one superpixel in frame $f+1$. The correspondence is determined based on the superpixel flow vectors. The center of superpixel i from frame f is projected to frame $f+1$ according to the superpixel flow $\boldsymbol{V_i}$. We form an edge between superpixels i and j, where superpixel j is the superpixel in frame $f+1$ that contains the projected center of superpixel i.

Motivated by [8], our superpixel affinity W_{ij} for a spatial edge (i,j) is defined as the combination of appearance (W^a_{ij}) and motion (W^m_{ij}) affinities. Appearance affinity is obtained by computing the histogram intersection of the grayscale (or color, if available) histograms of the two superpixel regions (we use 30 bin histograms for grayscale and $4 \times 4 \times 4$ histograms for RGB). Motion affinity is computed by comparing the flow vectors of the two superpixels, $\boldsymbol{V_i}$ and $\boldsymbol{V_j}$, and is equal to $W^m_{ij} = 1 - \frac{\|\boldsymbol{V_i} - \boldsymbol{V_j}\|}{\max\{\|\boldsymbol{V_i}\|, \|\boldsymbol{V_j}\|\}}$ capped to the range $(0,1)$. Since our superpixel graph construction incorporates superpixel flow already, we include the motion affinity only for spatial edges. Finally, to give larger superpixels more influence, we augment the affinity by weighting it with the product of areas of the two superpixels (A_i and A_j). Combining that with the goal of not grouping two superpixels if either their appearance or motion is dissimilar results in the following superpixel affinity:

$$W_{ij} = \begin{cases} A_i A_j \min\left(W^a_{ij}, W^m_{ij}\right), & (i,j) \text{ are in the same frame} \\ A_i A_j W^a_{ij}, & (i,j) \text{ are in different frames} \end{cases} \tag{3}$$

Since our graph has edges for only a small spatial neighborhood of superpixels with edge affinities encoding both appearance and motion, we will refer to it as S-AM. In Section 5 we will compare this graph construction to other graphs with modified spatial connectivity and different superpixel affinities.

4.3 Optimal Cuts for Each Shot

At this point, we have a superpixel graph and thus can apply the parametric maxflow framework to optimize the cost in Eqn. 1. However, prior to running the optimization framework, we first detect the shot boundaries in the video with the goal of independently finding closures for each shot.

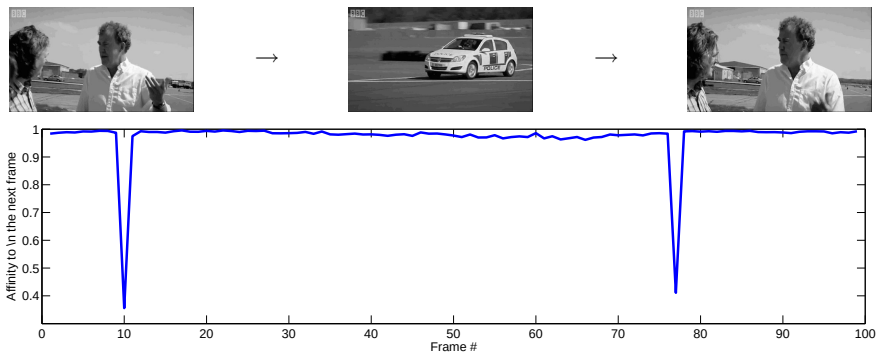


Fig. 3: Shot detection by finding minima in consecutive frame affinities. The top row shows a video containing 3 shots. The shot changes from people to car at frame 11 and back to people at frame 78. The bottom row shows a corresponding drop in consecutive frame affinity for these frames. These minima are detected in order to find the shot boundaries.

Temporal superpixel edges across shot boundaries are unreliable. Thus if a video is composed of multiple shots, running the optimization on the whole video results in undesirable solutions. Since this is not the focus of this work, we take a very simplistic approach to shot boundary detection. Similar to the appearance affinity between superpixels, we compute an appearance affinity between consecutive frames by comparing the grayscale histograms of whole frames using the histogram intersection kernel. This results in a $F-1$ dimensional vector of consecutive frame affinities (where F is the number of frames). The shot boundaries correspond to the detected minima in this vector (Fig. 3). Given the detected shots, we build a subgraph for every shot by selecting the superpixels and the edges that are contained in the shot. We optimize the cost in Eqn. 1 for all the subgraphs and concatenate the results.

Note that optimizing the cost in Eqn. 1 directly results in a trivial solution where all the superpixels are selected for which $cut(\boldsymbol{X}) = 0$ and $volume(\boldsymbol{X}) > 0$, resulting in $C(\boldsymbol{X}) = 0$. Moreover, we want to be able to weaken affinities in order to handle the cases of potential bleeding between foreground and background due to appearance or motion similarity. We solve the first problem by introducing infinite penalties for a subset of superpixels in the graph, preventing the trivial solution. Specifically, we run the optimization six times for each shot. In the first four runs, all the superpixels on the left, right, top, and bottom frame boundary

respectively, are assigned an infinite penalty. In the two additional runs, we assign infinite penalties first to all top and bottom superpixels, and then to all left and right superpixels. To handle the second issue, we augment the closure affinity in Eqn. 3 to :

$$W'_{ij} = \begin{cases} A_i A_j \left(\min\left(W^a_{ij}, W^m_{ij}\right)\right)^{\alpha}, & (i,j) \text{ are in the same frame} \\ A_i A_j \left(W^a_{ij}\right)^{\alpha}, & (i,j) \text{ are in different frames} \end{cases} \quad (4)$$

The exponent α controls the contribution of weak affinities. Increasing the exponent effectively lowers all the affinities towards 0, thereby preventing bleeding, but also increases the relative difference between weak and strong affinities. In the results section, we will analyze the effect of changing α on performance and suggest an optimal value for α.

4.4 Post-processing

Running parametric maxflow on the spatiotemporal superpixel graph results in hundreds and sometimes thousands of breakpoints. Some of the solutions differ by a very minor increase in area, while others contain multiple connected components. Furthermore, some desirable solutions are missed. Since our goal is to yield a small number of spatiotemporal hypotheses that capture coherently moving objects in the scene, we post-process the results to narrow down the number of solutions to a more manageable number and in the process generate additional good solutions. While such post-processing no longer guarantees the optimality of the solutions according to Eqn. 1, the resulting solutions still have a low closure cost and empirically yield a better set of hypotheses than the original solutions from parametric maxflow. Post-processing consists of the following 3 stages:

1. **Filtering solutions and generating new ones by analyzing the area change:** As previously stated, parametric maxflow results in solutions that minimize the cut with increasing area constraints. Some solutions corresponding to consecutive breakpoints $(\lambda_i, \lambda_{i+1})$ are almost equivalent in their superpixel selections and differ by a very small increase in area. We filter out the solutions where such an increase is insignificant (less then 1% of relative area increase). Conversely, for all other solutions we detect consecutive solution pairs where the relative area increase is above a threshold (more than 5%) and generate a new solution subtracting one superpixel subset from another.
2. **Selecting connected components and removing small solutions:** Some solutions up to this point contain only a few superpixels or select superpixels in a very small number of frames. We filter out these solutions by keeping only the solutions with at least 2 superpixels, with total area that is at least 1% of the frame area, and that participate in at least 5 frames. We run a connected component analysis for all the remaining solutions. Each solution that contains multiple connected components in space-time is split, generating one solution for each connected component.

3. **Removing duplicate solutions:** The above post-processing steps can result in the generation of duplicate solutions. In this final step we remove duplicate solutions.

For our test videos, this post-processing step reduces the number of solutions of a single run of parametric maxflow from several hundreds to an average of $20 - 80$ solutions.

5 Evaluation

We first perform a qualitative analysis of our approach on several short video sequences. Some sequences (such as the flower garden sequence) are grayscale, while others contain color. In the case of color sequences, we make use of this additional information, comparing color histograms instead of grayscale when computing superpixel affinities. The frame size for each video is on the order of 300×300 pixels, with the length of a video ranging from around 10 frames to 250 frames (hippo sequence). Based on quantitative evaluation (described in later paragraphs), we set $\alpha = 6$ for our qualitative experiments. We also perform a quantitative evaluation on a test dataset [32], comparing different graph constructions and affinity variations, as well as evaluating our approach against standard normalized cuts on the same graphs. The computational bottlenecks of the approach are the preprocessing steps: Pb edge detection, superpixel extraction, and optical flow computation, each taking several seconds per frame. Once a superpixel graph is built, each run of the optimization using parametric maxflow finishes in less than 5 seconds on the whole video, followed by all the postprocessing steps taking approximately 1 second.

Fig. 4 shows our qualitative results. For each sequence we show a frame from the original video and visualize several interesting solutions[5]. In the car sequence, several objects of interest were successfully recovered, such as the car and the heads of the people. Moreover, a part of the car (windshield) is also recovered in one of the solutions, indicating that our method can be used for part-based object recognition in videos or for action recognition that requires the tracking of parts. In the galloping horse sequence, the horse was correctly recovered in the middle of the sequence. A fence is also discovered as one of the solutions. However, in the beginning of the sequence it is partially merged with the horse due to poor superpixel boundaries and affinities between the horse and the background, which is also the reason for the incomplete solution in the Pepsi sequence. The horse example also illustrates that our framework works best when with large objects, as small objects usually have higher closure cost and tend to be undersegmented by superpixels. The table sequence illustrates that our framework can detect most objects in the scene. Finally, the hippo sequence

[5] See the Results at `http://www.cs.toronto.edu/~babalex/SpatiotemporalClosure/supplementary_material.html` for a video visualization of the results.

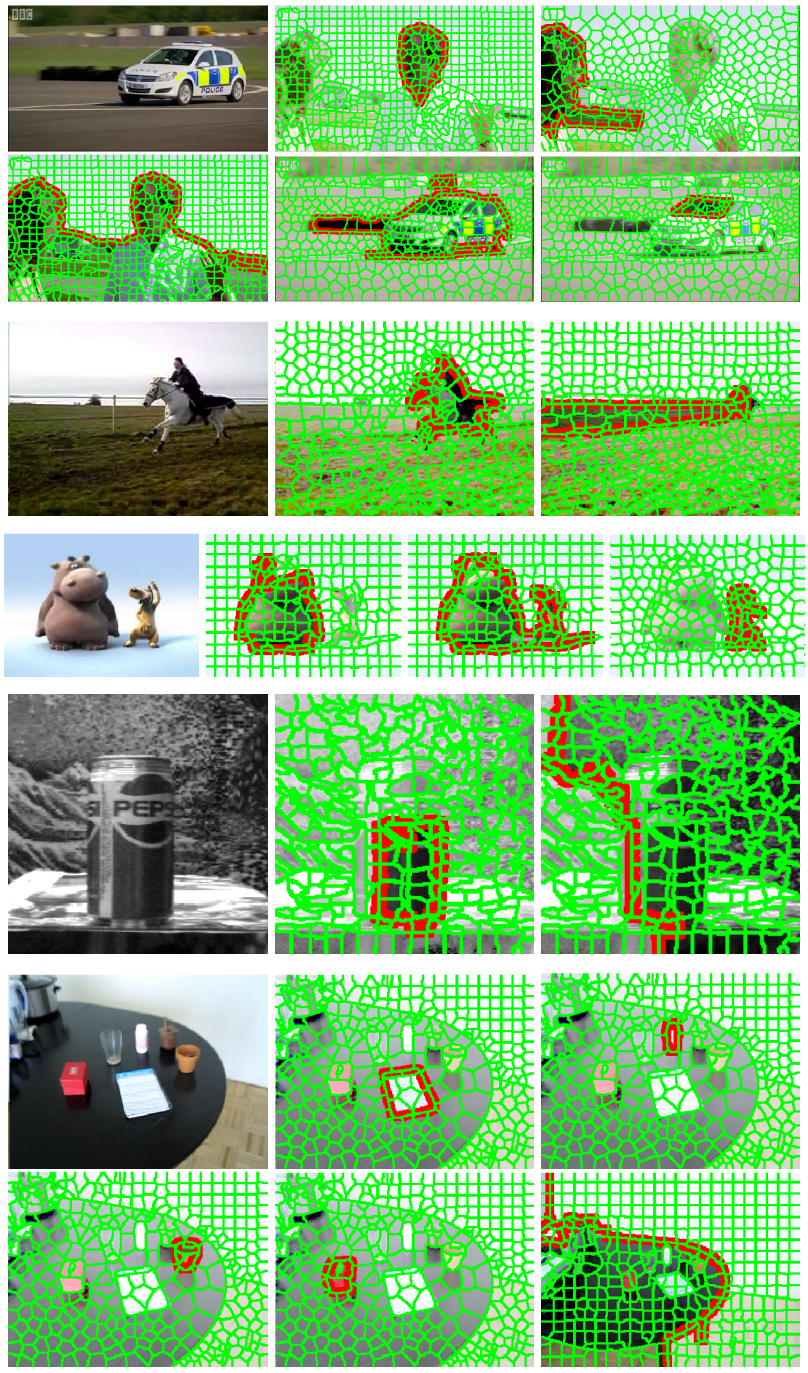


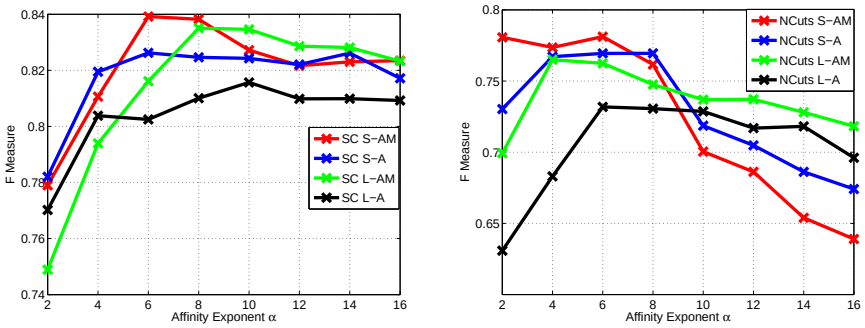
Fig. 4: Qualitative video figure/ground segmentation results. We display one sample frame from a sequence, followed by several interesting solutions.

illustrates how an additional solution (dog) can be generated by subtracting one solution (hippo) from another (hippo and dog).

For quantitative evaluation of our method we use 27 sequences from the dataset of Stein et al. [32]. Each sequence has a ground truth video segmentation mask, marking one foreground object. Given a set of detected spatiotemporal figures for a sequence, we choose the solution with the maximal F measure ($\frac{2 \cdot Precision \cdot Recall}{Precision + Recall}$) relative to the ground truth. We report the average F measure across all sequences.

We compare different variations of our algorithm, as well as replace our parametric maxflow minimization of the unbalanced normalized cuts cost with standard normalized cuts. Unlike our method, normalized cuts requires a user specified number of clusters. Therefore, to compare with our approach we run normalized cuts with 5, 10, 15, 20, and 25 clusters and concatenate all the results. Recall that our previously described graph construction (S-AM) includes only the immediate spatial neighbors and adds the motion affinity W_{ij}^m for spatial edges. We define additional variations over this construction:

- S-A - Same graph as S-AM, but with affinity only including **appearance** $W_{ij} = A_i A_j \left(W_{ij}^a\right)^\alpha$
- L-AM - Same as S-AM but with **larger** spatial connectivity. In addition to the edges in S-AM we add edges between all superpixels in the same frame whose centroids are less than R apart, where R is five times the radius of an average superpixel.
- L-A - Same as L-AM, but with affinity only including appearance $W_{ij} = A_i A_j \left(W_{ij}^a\right)^\alpha$



SC NCuts

Fig. 5: Quantitative evaluation of spatiotemporal closure detection. We compare the performance of each method (SC on the left and NCuts on the right) on four different graph constructions.

We compare our method (SC) to normalized cuts (NCuts) for all the above graph constructions. While we are able to solve the unbalanced normalized cuts

problem in a globally optimal fashion, normalized cuts cost is NP-hard to optimize and therefore only an approximation is provided. Despite that, the cut balancing in NCuts further constrains the solutions to be balanced and compact and helps to avoid bleeding, while our closure cost pushes the solutions to contain more superpixels which may result in undersegmentation. Fig. 5 illustrates the performance as we vary α. We also observe that our method achieves comparable results using S-AM and L-AM, indicating that our increase of spatial connectivity has only a marginal effect on the results. Note that the video sequences in the test dataset mostly contain large objects. Thus undersegmentation as a result of incorrect superpixels or our unbalanced normalized cuts closure cost is less of a concern, resulting in SC outperfoming the standard NCuts.

6 Conclusions

We began by motivating the problem of bottom-up spatiotemporal segmentation. We proceeded by extending work in bottom-up 2D closure detection to spatiotemporal closure detection in videos. Defining our closure cost over spatiotemporal superpixels was shown to facilitate better affinity computation and lead to more stable solutions. Finally, we employ parametric maxflow not only to efficiently find a global optimum of our spatiotemporal closure cost, but recover several multiscale segmentations giving a full hierarchical description of a dynamic scene. The limitations of our framework are particularly apparent when small, low-contrast objects are present, occasionaly leading to object undersegmentation. Therefore, in future work we will improve our spatiotemporal superpixel approach to recover larger, more meaningful superpixels, without sacrificing speed or accuracy. In addition, we will also explore other graph constructions and will design a better superpixel affinity by learning the best composition of motion and appearance cues in a supervised manner.

References

1. Wang, D.: Unsupervised video segmentation based on watersheds and temporal tracking. CirSysVideo **8** (1998) 539–546
2. Moscheni, F., Bhattacharjee, S., Kunt, M.: Spatiotemporal segmentation based on region merging. PAMI **20** (1998) 897–915
3. Gelgon, M., Bouthemy, P.: A region-level motion-based graph representation and labeling for tracking a spatial image partition. Pattern Recognition **33** (2000) 725 – 740
4. Deng, Y., Manjunath, B.: Unsupervised segmentation of color-texture regions in images and video. PAMI **23** (2001) 800–810
5. DeMenthon, D.: Spatio-temporal segmentation of video by hierarchical mean shift analysis. In: SMVP. (2002)
6. Shi, J., Malik, J.: Motion segmentation and tracking using normalized cuts. In: ICCV. (1998) 1154
7. Fowlkes, C., Belongie, S., Malik, J.: Efficient spatiotemporal grouping using the nyström method. In: CVPR. (2001) 231–238

8. Huang, Y., Liu, Q., Metaxas, D.: Video object segmentation by hypergraph cut. CVPR (2009) 1738–1745
9. Greenspan, H., Goldberger, J., Mayer, A.: Probabilistic space-time video modeling via piecewise gmm. PAMI **26** (2004) 384–396
10. Levinshtein, A., Sminchisescu, C., Dickinson, S.: Optimal Contour Closure by Superpixel Grouping. In: ECCV. (2010)
11. Levinshtein, A., Stere, A., Kutulakos, K.N., Fleet, D.J., Dickinson, S.J., Siddiqi, K.: Turbopixels: Fast superpixels using geometric flows. PAMI **31** (2009) 2290–2297
12. Carreira, J., Sminchisescu, C.: Constrained parametric min-cuts for automatic object segmentation. In: CVPR. (2010)
13. Kolmogorov, V., Boykov, Y., Rother, C.: Applications of parametric maxflow in computer vision. In: ICCV. (2007)
14. Li, F., Carreira, J., Sminchisescu, C.: Object Recognition as Ranking Holistic Figure-Ground Hypotheses. In: CVPR. (2010)
15. Pritch, Y., Rav-Acha, A., Peleg, S.: Nonchronological video synopsis and indexing. PAMI **30** (2008) 1971–1984
16. Welch, G., Bishop, G.: An introduction to the kalman filter. Technical report (1995)
17. Black, M., Jepson, A.: Eigentracking: Robust matching and tracking of articulated objects using a view-based representation. IJCV **26** (1998) 63–84
18. Isard, M., Blake, A.: Condensation - conditional density propagation for visual tracking. IJCV **29** (1998) 5–28
19. Comaniciu, D., Ramesh, V., Meer, P.: Kernel-based object tracking. PAMI **25** (2003) 564–577
20. Megret, R., DeMenthon, D.: A survey of spatio-temporal grouping techniques. Technical report, University of Maryland, College Park (2002)
21. Wang, J., Adelson, E.: Representing moving images with layers. TIP **3** (1994) 625–638
22. Weiss, Y., Adelson, E.H.: A unified mixture framework for motion segmentation: Incorporating spatial coherence and estimating the number of models. In: CVPR. (1996) 321
23. Weiss, Y.: Smoothness in layers: Motion segmentation using nonparametric mixture estimation. In: CVPR. (1997) 520
24. Jojic, N., Frey, B.J.: Learning flexible sprites in video layers. CVPR **1** (2001) 199
25. Jepson, A.D., Fleet, D.J., Black, M.J.: A layered motion representation with occlusion and compact spatial support. In: ECCV. (2002) 692–706
26. Bascle, B., Deriche, R.: Region tracking through image sequences. ICCV **0** (1995) 302
27. Paragios, N., Deriche, R.: Geodesic active contours and level sets for the detection and tracking of moving objects. PAMI **22** (2000) 266–280
28. Chung, D., MacLean, W., Dickinson, S.: Integrating region and boundary information for spatially coherent object tracking. IVC **24** (2006) 680–692
29. Cremers, D., Soatto, S.: Motion competition: A variational approach to piecewise parametric motion segmentation. IJCV **62** (2005) 249–265
30. Patras, I., Lagendijk, R.L., Hendriks, E.A.: Video segmentation by map labeling of watershed segments. PAMI **23** (2001) 326–332
31. Martin, D.R., Fowlkes, C.C., Malik, J.: Learning to detect natural image boundaries using local brightness, color, and texture cues. PAMI **26** (2004) 530–549
32. Stein, A., Hoiem, D., Hebert, M.: Learning to find object boundaries using motion cues. In: ICCV. (2007)