Pacific Graphics (2016)
E. Grinspun, B. Bickel, and Y. Dobashi (Editors) *Short Paper*

Interactive Multicut Video Segmentation

Evgeny Levinkov[1], James Tompkin[2], Nicolas Bonneel[3], Steffen Kirchhoff[2], Bjoern Andres[1], and Hanspeter Pfister[2]

[1]MPI für Informatik, [2]Harvard Paulson SEAS, [3]LIRIS CNRS

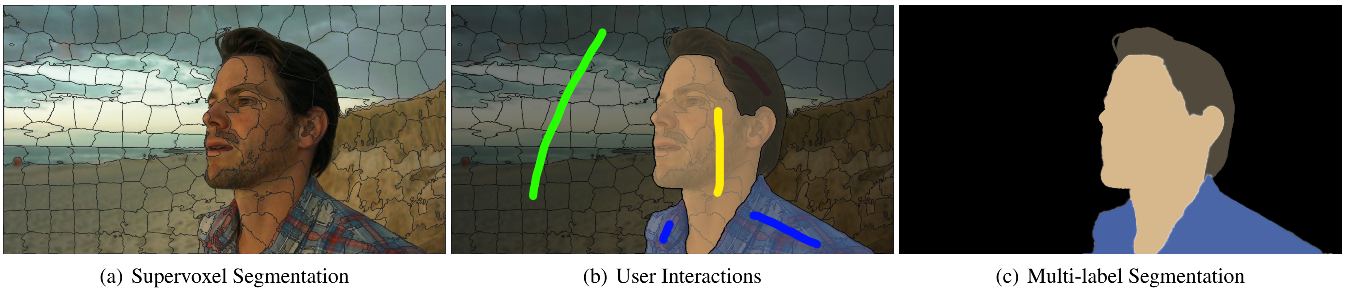

Figure 1: *Starting with a supervoxel segmentation of the input* (a)*, the user scribbles different regions to segment with multi-colored brushes* (b)*. These interactive constraints quickly cut the video into any number of segments, providing a fast and simple way to accomplish multi-part segmentation tasks* (c).

Abstract
Video segmentation requires separating foreground from background, but the general problem extends to more complicated scene segmentations of different objects and their multiple parts. We develop a new approach to interactive multi-label video segmentation where many objects are segmented simultaneously with consistent spatio-temporal boundaries, based on intuitive multi-colored brush scribbles. From these scribbles, we derive constraints to define a combinatorial problem known as the multicut*—a problem notoriously difficult and slow to solve. We describe a solution using efficient heuristics to make multi-label video segmentation interactive. As our solution generalizes typical binary segmentation tasks, while also improving efficiency in multi-label tasks, our work shows the promise of multicuts for interactive video segmentation.*

Categories and Subject Descriptors (according to ACM CCS): I.4.6 [Computer Graphics]: Image Processing and Computer Vision—SegmentationRelaxation G.2.2 [Computer Graphics]: Discrete Mathematics—Graph TheoryGraph labeling

1. Introduction

Video segmentation is a common task. In media production, it is key to composite or remove elements in a scene, or to apply effects to different objects independently. In computer vision, video segmentation is key to creating 'ground truth' sequences from which to train models for object classification. Professional visual effects (VFX) tools like Silhouette [Sil14] and Mocha [Imagineer Systems Ltd.14] allow artists to segment a foreground from uncontrolled backgrounds; however, these often require hours of work for a few seconds of footage. Acceptable results are achievable in simpler tools, but these can still take time (e.g., Adobe After Effects Rotobrush [BWSS09]) or be too inflexible (e.g., Power Matte [Dig]).

We aim to make a fast video segmentation tool which requires just a few user interactions, but which is still able to produce acceptable results for applications such as consumer VFX work or 'ground truth' labeling of video databases. One avenue for potential gains is to consider scenes where multiple objects or regions need to be segmented. Instead of many binary segmentations, it is possible to take an approach so that user interactions inform the segmentation of all parts simultaneously—producing a so-called *multicut*.

To this end, we introduce an approach to quickly solve the video multicut problem with user constraints, and so create a fast interactive video segmentation system. We begin with a supervoxel over-segmentation of the video. This over-segmentation forms a non-planar graph in spacetime, which must be cut into meaningful segments. The user interactively scribbles onto the video with different colored brushes, one color per desired segment. From these scribbles, we resolve split and merge constraints to inform the solution to the multicut problem. If the constraints conflict with the underlying supervoxel graph, i.e., the over-segmentation is too coarse, then we automatically refine the graph to resolve the constraints. In principle, many different multicuts may exist which meet these constraints. To pick a good multicut, we assign costs to cutting graph edges by dynamically clustering video features based on user scribbles. With this approach, each added user stroke takes

DOI: 10.2312/pg.20161332

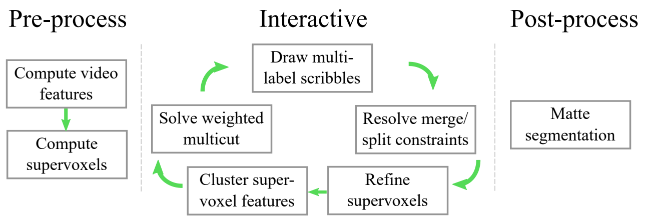

Figure 2: *After an initial pre-process to compute superpixels and aggregated features, the user interactively refines the segmentation with multiple scribbles. Finally, the solution is matted.*

≈1 second to resolve, and the solution automatically generalizes to segmenting any number of regions.

We perform a comparison of time and quality on 4 sequences against two expert users who took 10–12 hours with professional tools, and against novices who used RotoBrush. Our method was 50x faster than the experts, though with less output quality. Against Rotobrush, our tool was 3x faster with comparable quality. From these results, we believe our approach hold promise as an interactive video segmentation tool for consumers, and as a quick tool to label 'ground truth' datasets for computer vision.

We state our contributions explicitly:

1. A formulation of the multicut problem that is appropriate for *interactive* video segmentation, with an efficient way to provide fast feedback to users.
2. A system for interactive video segmentation which demonstrates the speed and quality achievable with a supervoxel-based multicut solution.

2. Related Work

Interactive segmentation of videos into spatially and temporally connected components (Fig. 1(b)) is a prerequisite for many video post-production applications, such as matting [CAC*02], color transfer [BSPP13], or white balancing [HMP*08]. It is often performed as a binary segmentation task where a foreground object is extracted from the background [WBC*05, BS07, ZQPM12, NSB15, FZL*15, LBSW16, LVS*16]. Prominent examples include the consumer Rotobrush tool in Adobe After Effects, which is based on the work of Bai et al. [BWSS09], and professional commercial tools such as Silhouette, Mocha, and Power Matte. These tools do not guaranteed to create consistent segment boundaries between different objects.

In the computer vision literature, consistent boundaries can be guaranteed by multi-label segmentations [GKHE10, OB11, GNC*13]. However, existing work in this area has focused on fully automatic segmentation. While automatic approaches will work some of the time, it is currently very difficult to reliably generate semantic-level segmentations. To complete the task of disambiguating regions and creating meaningful labels, a user is required, and so interactive segmentation tools are needed.

Our approach builds on the formalization of segmentation as a multicut problem. The multicut problem has recently become an active topic of computer vision research [KLB*15, AG12, AKB*11, AKB*12, AYM*13, KSA*11, KNKY11, VB10, YIF12]. A one-to-one relation exists between segmentations and multicuts; thus, the study of segmentations *is* the study of multicuts. The constraints of the multicut problem make explicit the statement that contours of segments are closed. Many models of segmentation have been proposed which do not enforce this constraint, for instance, in segmentation based on sketched silhouettes [AHSS04]. These models benefit from assumptions such as the piecewise smoothness of contours which sometimes yield closed contours in practice. However, since the closedness of contours is not enforced, the result is not guaranteed to define a segmentation. In this paper, we focus on the formalization of video segmentation as a multicut problem and refer the reader to [LM01, Zha06] for a review of different approaches.

The multicut problem is an integer linear program [CR93], well-known for its application in correlation clustering [DEFI06]. Solving instances of the multicut problem for the purpose of video segmentation is challenging. Typically, the video is represented as a graph, with pixels (or superpixels) as nodes joined by edges, where a multicut is a set of edges which separate the nodes into isolated subgraphs—the desired segments. This combinatorial optimization problem is NP-hard, even for planar graphs [BKKZ13]. Multicut approaches consider a set of linear constraints, where a segmentation is any solution of a minimization problem under these constraints. Exact solvers have been used for video segmentation [AKB*11, AKB*12], but they remain impractically slow, requiring more than two minutes for a 0.17 mega-pixel video, which precludes user interaction. Beier et al. [BKK*14] showed that using heuristics can achieve solutions which are very close to the global optimal, but take orders of magnitude less computation time. In this work, we employed the heuristic KLj [KLB*15], which shows a state-of-the-art runtime vs. solution quality trade off.

3. Our Approach

A block diagram for our system is shown in Fig. 2. To begin, we preprocess the video to compute both supervoxels and an associated feature vector per supervoxel which describes its appearance. Then, interactively, the user employs a 'paint by numbers' approach with multiple colored brushes to scribble coarsely and sparsely over the video. These multiple labels are automatically resolved into a set of constraints which specify which supervoxels belong to the same segment and which belong to different segments. Should any constraints conflict with the underlying supervoxels, we automatically refine the supervoxel over-segmentation, e.g., to capture fine details.

Then, we solve the constrained multicut problem and find a good segmentation. To provide graph edge weights with which to score potential multicut solutions, we cluster all supervoxels by training on the provided user labels and appearance features. Once a multicut solution is optimized, the presented segmentation respects both the user constraints and the video appearance. Finally, after the user is satisfied, we matte the result.

To make this model applicable to video, there are two major challenges to overcome. The first challenge is making it fast while maintaining segmentation accuracy. Multicuts are typically computed on the graph of a supervoxel over-segmentation. Many supervoxels are usually used through a fine over-segmentation, but this implies a large graph which is slow to solve. In employing a very fast heuristic—KLj (Sec. 3.3.1)—we can start with many supervoxels

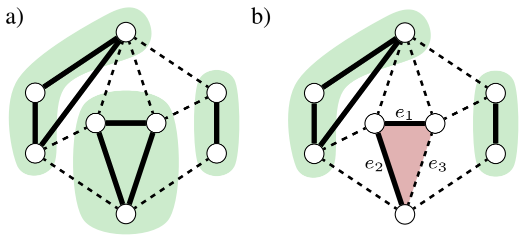

Figure 3: ***a**. Every segmentation of a graph is defined uniquely by the set of edges (dashed lines) that straddle different segments (green). Not every subset of edges defines a segmentation. A negative example is depicted in **b**. If precisely the edges depicted as dashed lines straddled different segments, the three nodes connected by the edges e_1 and e_2 would belong to the same segment, in contradiction to e_3 which straddles different segments.*

(5,000) to increase output accuracy and still maintain interactivity with ≈1 second response.

The second challenge is implementing constraints to split different brush strokes. Whenever the user wants two pixels to belong to different segments, there must exist at least one segment boundary on *any* path between these two pixels. This is a large space, and many constraints must be added to the optimization problem to constrain the set of feasible segmentations. As this is expensive, we propose a solution to this problem in Sec. 3.3.

3.1. Multicuts of Supervoxel Graphs

A segmentation of a graph is a partitioning of its node set into connected subsets, where the partitioning is defined by a subset of graph edges. A multicut is a binary labeling $x \in \{0,1\}^E$ of edges in the graph, with edges labeled 0 and 1 defining, correspondingly, connectivity within and between segments (solid and dashed lines in Fig. 3).

Chopra and Rao [CR93] showed a necessary and sufficient condition for a binary labeling to be a valid multicut: for edge set $\{e \in E | x_e = 1\}$, along each graph cycle, either no edge or more than one edge is labeled 1. We write this as a set of linear inequalities:

$$\forall T \in \text{cycles}(V,E)\ \forall e \in c \quad x_e \leq \sum_{e' \in T \setminus \{e\}} x_{e'} \tag{1}$$

Here, cycles(V,E) denotes the set of all cycles in the graph (V,E).

3.2. User Interactions

Users provides evidence about the correct segmentation via scribbling over each intended segment with a different colored brush. From these scribbles, we automatically extract a set of constraints:

- A set $E_0 \subseteq E$ of edges known to connect adjacent supervoxels of the same segment, i.e., supervoxels along a single brush stroke.
- A set $E_1 \subseteq E$ of edges known to connect adjacent supervoxels of different segments.
- A set $F_1 \subseteq (V \times V) \setminus E$ of pairs of non-adjacent supervoxels known to be in different segments, i.e., supervoxels along different colored brush strokes.

This evidence constrains the set of feasible segmentations.

We can write these constraints formally as linear inequalities:

$$\forall e \in E_0 \quad x_e = 0 \tag{2}$$
$$\forall e \in E_1 \quad x_e = 1 \tag{3}$$
$$\forall \{v,w\} \in F_1 \quad \forall \pi \in \text{path}(v,w) \quad 1 \leq \sum_{e \in \pi} x_e \tag{4}$$

where path(v,w) denotes the set of all paths in the graph (V,E) from node v to node w. The inequalities in (4) state that no path exists in the graph that connects supervoxels v and w. Constraining non-adjacent nodes to the same segment is possible [NL10], but is a substantial complication that is beyond the scope of this work. This means that, in our solutions, disconnected but similarly-brush-colored segments are assigned different segment labels in the result. In practice, this is not a problem as we simply relabel the result segments based on their intersecting input brush stroke colors.

3.3. Optimization Problem

From the set of all possible segmentations which satisfy these constraints, we wish to favor segmentations in which similar appearing supervoxels belong to the same segments, and dissimilar supervoxels belong to different segments. To this end, we model the similarity between supervoxels by the edge features described in Sec. 3.3.2. Then, the segmentation problem is posed as a minimization over probabilities that adjacent supervoxels belong to different segments.

The probability p_e that an edge between adjacent supervoxels $e \in E$ belongs to a segment boundary with $x_e = 1$ can be modeled by any probabilistic model for classification (Sec. 3.3.2). We convert the vector of probabilities $p \in [0,1]^{|E|}$ into a cost vector $c \in \mathbb{R}^{|E|}$ by $c_e = \log((1-p_e)/p_e)$. The problem of inferring the most probable segmentation x given the user-defined constraints (2)–(4) is an integer linear program (ILP):

$$\min_{x \in \{0,1\}^E} \quad \sum_{e \in E} c_e x_e \tag{5}$$
$$\text{subject to} \qquad (1), (2)\text{-}(4) \tag{6}$$

If an edge e is likely to be a segment boundary, i.e. if $p_e > 0.5$, its cost c_e is negative. Thus, the value of the objective function is minimized if the edges that are likely to be segment boundaries are labeled 1. Constraint (1) enforces the edge labeling to be a valid segmentation with closed contours, and constraints (2)-(4) enforce the segmentation to respect the user input.

3.3.1. Kernighan-Lin-type Heuristic Algorithm (KLj)

Solving the resulting optimization problem is NP-hard, therefore exact solvers are not suitable for our interactive scenario. However, primal feasible heuristics are known to work well for the multicut problem [BKK*14, KLB*15], delivering good solutions in little time. In these cases, it is impossible to impose hard constraints (2)-(4). However, we circumvent this problem by assigning very large positive or negative weights to edges which connect supervoxels that, through resolving constraints derived from the multi-colored brush strokes, were marked by a user as belonging to the same or different segments. This restricts the search space of the heuristics to the solutions that respect the user input constraints, because the solutions that violate them will have a much higher objective values. One benefit of this approach is that if a user specifies contradicting constraints, we can still arrive at a solution that is feasible w.r.t. the multicut constraints (1), i.e., a valid segmentation. This makes our system more stable under arbitrary user interactions.

KLj extends the classic Kernighan-Lin algorithm for the set partitioning problem [KL70], which is equivalent to solving the multicut problem defined w.r.t. a complete graph. KLj starts with an initial feasible solution (all nodes V are in the same component) and then greedily improves the boundary between a pair of neighboring segments by moving nodes locally from one set into the other and assessing the objective cost of the move. KLj can also introduce new segments by the same procedure carried out against an empty set. While KLj does not provide any approximation or runtime guarantees, in practice it is much faster than exact ILP solvers and still produces usable results. For a more detailed description of the algorithm, we refer the reader to [KLB*15].

3.3.2. Edge Features and Classification

The cost vector c in (5) describes the similarity between neighboring supervoxels. To describe this similarity, we compute the mean and variance of per pixel grayscale intensity, sum of the squared horizontal, vertical and temporal gradient magnitudes, Lab color, HSV color, and bi-directional optical flow [Liu09]. This results in a 24-d feature vector of means and variances per supervoxel.

These costs c depend on the probability that adjacent supervoxels belong to different segments. To predict these probabilities, we use the user labels to train a random forest classifier with 64 trees with features for every edge between adjacent supervoxels, where the element-wise distance between the feature vectors represents the overall distance. In principle, it is possible to pre-train an edge classifier from a database of videos. This could be useful for domain-specific videos, such as for sports videos.However, in general, retraining the classifier for every video is the most flexible approach to cope with unseen videos of widely-varying appearance.

3.4. Supervoxels

As we solve for a solution on a graph of supervoxels, we rely on these supervoxels for temporal consistency. Given that, an initial over-segmentation should not be too coarse so as to avoid missing important contours, and should not be too fine so as to remain computationally tractable. We favored fast feedback, and refine the supervoxel segmentation whenever user constraints deem it necessary. We compute SLIC supervoxels [ASS*12] which we re-label into connected components. Moreover, SLIC sometimes produces small cluttered regions. We remove regions smaller than 500 pixels per frame and relabel their pixels with that of their nearest neighbors. This provides a good trade-off between speed and quality among a set of tested alternatives [SGS*06, FH04, RM00].

Refining Supervoxels Our fast solver allows us to use many supervoxels, but this still may not be enough to capture fine details.This manifests as conflicting constraints in the system of equations, e.g., two different scribble colors in the same superpixel. We detect such conflicts and automatically refine the supervoxel segmentation to

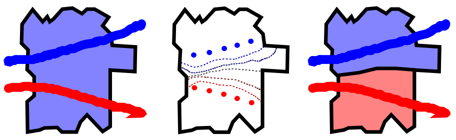

Figure 4: Left: *A blue superpixel is in conflict with a new red scribble.* Middle: *To automatically resolve the conflict, we split the supervoxel in a 3D region-growing process (dotted lines converging on center line), where samples along the brush strokes act as seeds.* Right: *The supervoxel is split and the constraint resolved.*

match the constraints (Fig. 4). Coordinates along the conflicting scribble lines are used as seeds to a 3D region growing algorithm, which uses the sum of squared gradient magnitudes in horizontal, vertical, and temporal directions as features. This refinement takes less than a second. This allows the user to quickly make coarse segmentations using the discovered supervoxel edges, but also to specify new segmentation edges in arbitrary locations by drawing conflicting brush strokes very close together.

3.5. Matting

After segmentation, we compute a matting solution to refine our supervoxel edges and to cope with transparency. We use a spatio-temporal cross-bilateral filter [TM98] with fast bilateral grids [CPD07]. This filter takes two video volumes as input: a binary mask, and per-pixel weights to modulate the Gaussian kernel used in filtering. We use a grayscale version of the video as our weights. Each label is filtered separately, then the resulting mattes are normalized so that they sum to 1. This solution can be parallelized across labels. Other state-of-the-art matting solutions would improve our results further [BWS11, SPCR14].

3.6. Web Implementation and Collaborative Editing

Our system is built as a web application: All core algorithmic components are implemented in C++, then wrapped through Cython and served from Tornado Web Server. The client is written in JavaScript, with WebGL rendering. First, the video and supervoxel segmentation are sent from the server and cached on the client. During interaction, only the constraints are sent to the server, with the segmentation returned to the client. Updates to the supervoxels are sent to the client as incremental changes. This approach allows for new applications such as collaborative segmentation on difficult or long sequences, or for segmenting large video databases as the task can be crowdsourced and completed on light-weight terminals.

For interaction, the user selects a brush, scribbles, and the segmentation is returned. The user iterates until they are satisfied. There are two optional brushes: the first turns off retraining the classifier, for use when the clustering is well-defined and should not be swayed by newly-marked regions; the second is a pure non-constraint-forming scribble, to paint the segmentation labels of supervoxels directly. In practice, the novice user need not use these expert features.

4. Results

For a 1280×720×150 video, on an Intel Xeon E5-2609 CPU, our system spends 12 min in unoptimized pre-processing: 10 min to

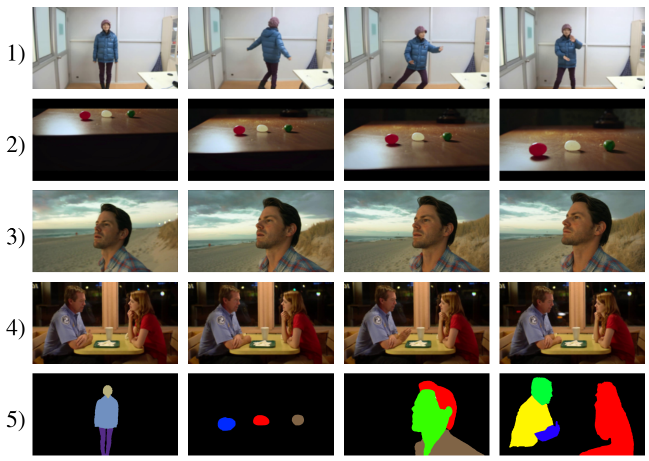

Figure 5: *The four different videos in the comparison. 1) 'Kung Fu', 960×540×270. 2) 'The Secret Number', 1280×720×210. 3) 'Sunset', 1280×960×60. 4) 'Time Expired', 1280×960×150. 5) The required segments for each sequence.*

compute edge features (Sec. 3.3.2), then 2 min to compute supervoxels and aggregate features. During interaction, with 5k supervoxels and 25 user constraints, training and classifying takes 0.3 s, with the multicut problem solved in a further 0.25 s. After network transfer, the total response time to interaction is under one second. As an extreme case, with 25k supervoxels, training and classifying takes 1.4 s, with the multicut problem solved in 5 s.

To assess the accuracy and efficiency of our approach, we compare our tool with After Effects Rotobrush which is used in a repeated binary segmentation fashion by a novice user. For comparison, we also hired two professional VFX artists to segment our videos with professional tools of their choice (namely Silhouette and mocha Pro). In total, four videos were segmented with different levels of difficulty (Fig. 5). Completion times are shown in Table 1. We can see from the expert timings that high-quality multi-label segmentation is a laborious task. Ideally, one would re-use the shared edges between different segments. While this is possible in existing foreground/background segmentation tools, our approach solves edges jointly and guarantees consistent non-overlapping boundaries. Our novice user was able to perform multi-label segmentation tasks more quickly with our tool than with After Effects Rotobrush, and to a similar segmentation quality (Fig. 6). However, our automatic pre-process takes longer. While we only tested four videos, these results show some promise for the generality of the tool.

5. Discussion

Pro-level tools can achieve high edge accuracy given enough time, and this is where our tool cannot compete yet. In principle, the system of arbitrary supervoxel refinement (Sec. 3.4) could be augmented to take input not from scribbles but from parametric curves, similar to Silhouette. This would provide the control necessary to describe arbitrary edges with more control, though it requires future work. That said, our current results are suitable for many consumer-level editing tasks, and the speed and multi-label nature of the tool makes it suitable for labeling large databases of videos to generate 'ground truth' for training computer vision models.

One limitation with re-learning a classifier at every new stroke is that the resulting segmentation can change in unexpected ways, e.g., an edge in the video which has similar content in appearance on either side may change segment as new strokes are added. This typically manifests itself in the very early stages of segmenting a video, when the classification boundaries have very few user labels to inform them. One solution is to simply educate the user to trust the process as it is a part of the learning system; another is to use a static pre-trained classifier, though this may be inflexible to new 'unseen' videos which do not match the training database.

Table 1: *Completion time in minutes. Participants completed the multi-label segmentation tasks faster with our tool than with Rotobrush (RB).*

	Ours Novice	RotoBrush Novice	Silhouette Expert 1	mocha Pro Expert 2
Video 1	11.29	27.61	405.75	174.25
Video 2	5.48	35.88	33.10	35.36
Video 3	8.99	15.31	103.50	147.34
Video 4	13.04	31.68	180.21	219.89
Mean	9.70	27.63	180.64	144.21

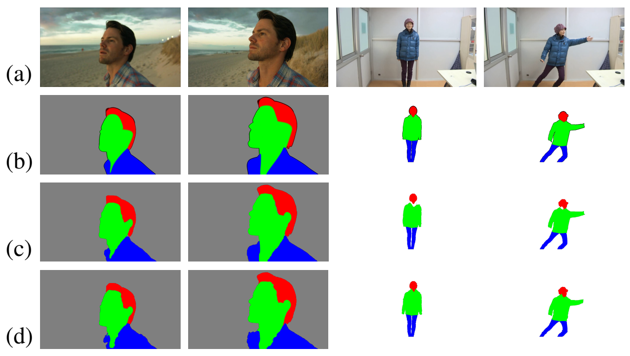

Figure 6: *(a) 'Sunset' (video 3) and 'Kung Fu' (video 1). (b) Segmentation from professional VFX artist using Silhouette (SI). (c) Novice Rotobrush (RB) result. (d) Novice result with our tool.*

6. Conclusion

We propose an interactive multicut video segmentation system. We solve two challenges with creating an interactive video multicut tool: making it fast through a state-of-the-art multicut solver, and integrating user constraints into the problem. Our system produces similar quality to current prosumer tools such as After Effects Rotobrush, while being faster to use. Overall, these results show the promise of interactive multicut video segmentation, easing the way for more advanced video editing and database labeling.

7. Acknowledgements

Thank you to Daniel Haehn for help with the user study, and to Jeff Desom and Pawel Galazka for expert segmentations.

References

[AG12] ALUSH A., GOLDBERGER J.: Ensemble segmentation using efficient integer linear programming. *TPAMI 34*, 10 (2012). 2

[AHSS04] AGARWALA A., HERTZMANN A., SALESIN D. H., SEITZ S. M.: Keyframe-based tracking for rotoscoping and animation. *ACM Trans. Graph. (SIGGRAPH) 23*, 3 (2004). 2

[AKB*11] ANDRES B., KAPPES J. H., BEIER T., KÖTHE U., HAMPRECHT F. A.: Probabilistic image segmentation with closedness constraints. In *IEEE Int. Conf. Comp. Vision (ICCV)* (2011). 2

[AKB*12] ANDRES B., KRÖGER T., BRIGGMAN K. L., DENK W., KOROGOD N., KNOTT G., KÖTHE U., HAMPRECHT F. A.: Globally optimal closed-surface segmentation for connectomics. In *ECCV* (2012). 2

[ASS*12] ACHANTA R., SHAJI A., SMITH K., LUCCHI A., FUA P., SÜSSTRUNK S.: SLIC superpixels compared to state-of-the-art superpixel methods. *IEEE Trans. Patt. Anal. and Machine Intell. 34*, 11 (2012). 4

[AYM*13] ANDRES B., YARKONY J., MANJUNATH B. S., KIRCHHOFF S., TURETKEN E., FOWLKES C., PFISTER H.: Segmenting planar superpixel adjacency graphs w.r.t. non-planar superpixel affinity graphs. In *EMMCVPR* (2013). 2

[BKK*14] BEIER T., KROEGER T., KAPPES J. H., KOETHE U., HAMPRECHT F.: Cut, glue cut: A fast, approximate solver for multicut partitioning. In *IEEE Conf. Comp. Vision and Patt. Rec.* (2014). 2, 3

[BKKZ13] BACHRACH Y., KOHLI P., KOLMOGOROV V., ZADIMOGHADDAM M.: Optimal coalition structures in graph games, 2013. 2

[BS07] BAI X., SAPIRO G.: A geodesic framework for fast interactive image and video segmentation and matting. In *IEEE Int. Conf. Comp. Vision (ICCV)* (2007). 2

[BSPP13] BONNEEL N., SUNKAVALLI K., PARIS S., PFISTER H.: Example-based video color grading. *ACM Trans. Graph. (SIGGRAPH) 32*, 4 (2013). 2

[BWS11] BAI X., WANG J., SIMONS D.: Towards temporally-coherent video matting. In *Computer Vision/Computer Graphics Collaboration Techniques*, vol. 6930. 2011. 4

[BWSS09] BAI X., WANG J., SIMONS D., SAPIRO G.: Video snapcut: Robust video object cutout using localized classifiers. In *ACM Trans. Graph. (SIGGRAPH)* (2009). 1, 2

[CAC*02] CHUANG Y.-Y., AGARWALA A., CURLESS B., SALESIN D. H., SZELISKI R.: Video matting of complex scenes. *ACM Trans. Graph. (SIGGRAPH) 21*, 3 (2002). 2

[CPD07] CHEN J., PARIS S., DURAND F.: Real-time edge-aware image processing with the bilateral grid. *ACM Trans. Graph. (SIGGRAPH) 26*, 3 (2007). 4

[CR93] CHOPRA S., RAO M. R.: The partition problem. *Math. Program. 59* (1993). 2, 3

[DEFI06] DEMAINE E. D., EMANUEL D., FIAT A., IMMORLICA N.: Correlation clustering in general weighted graphs. *Theoretical Computer Science 361*, 2–3 (2006). 2

[Dig] DIGITAL FILM TOOLS, LLC.: Power Matte v2.0.1.3 software. . 1

[FH04] FELZENSZWALB P. F., HUTTENLOCHER D. P.: Efficient graph-based image segmentation. *Int. J. Comput. Vision 59*, 2 (2004). 4

[FZL*15] FAN Q., ZHONG F., LISCHINSKI D., COHEN-OR D., CHEN B.: Jumpcut: Non-successive mask transfer and interpolation for video cutout. *ACM Trans. Graph. 34*, 6 (Oct. 2015), 195:1–195:10. 2

[GKHE10] GRUNDMANN M., KWATRA V., HAN M., ESSA I.: Efficient hierarchical graph based video segmentation. *IEEE CVPR* (2010). 2

[GNC*13] GALASSO F., NAGARAJA N. S., CARDENAS T. J., BROX T., SCHIELE B.: A unified video segmentation benchmark: Annotation, metrics and analysis. In *IEEE Int. Conf. Comp. Vision (ICCV)* (2013). 2

[HMP*08] HSU E., MERTENS T., PARIS S., AVIDAN S., DURAND F.: Light mixture estimation for spatially varying white balance. *ACM Trans. Graph. (SIGGRAPH) 27*, 3 (2008). 2

[Imagineer Systems Ltd.14] IMAGINEER SYSTEMS LTD.: mocha Pro v3.1 software. , 2014. 1

[KL70] KERNIGHAN B. W., LIN S.: An efficient heuristic procedure for partitioning graphs. *Bell Systems Tech. Journal 49* (1970). 4

[KLB*15] KEUPER M., LEVINKOV E., BONNEEL N., LAVOUÉ G., BROX T., ANDRES B.: Efficient Decomposition of Image and Mesh Graphs by Lifted Multicuts. In *IEEE Int. Conf. Comp. Vision (ICCV)* (2015). 2, 3, 4

[KNKY11] KIM S., NOWOZIN S., KOHLI P., YOO C. D.: Higher-order correlation clustering for image segmentation. In *NIPS* (2011). 2

[KSA*11] KAPPES J. H., SPETH M., ANDRES B., REINELT G., SCHNÖRR C.: Globally optimal image partitioning by multicuts. In *EMMCVPR* (2011). 2

[LBSW16] LU Y., BAI X., SHAPIRO L., WANG J.: Coherent parametric contours for interactive video object segmentation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (June 2016). 2

[Liu09] LIU C.: *Beyond Pixels: Exploring New Representations and Applications for Motion Analysis*. PhD thesis, Massachusetts Institute of Technology, May 2009. 4

[LM01] LUCCHESE L., MITRA S. K.: Color image segmentation: A state-of-the-art survey. In *Proc. Indian Nat. Science Academy* (2001). 2

[LVS*16] LI W., VIOLA F., STARCK J., BROSTOW G. J., CAMPBELL N. D. F.: Roto++: Accelerating professional rotoscoping using shape manifolds. *ACM Trans. Graph. 35*, 4 (July 2016), 62:1–62:15. 2

[NL10] NOWOZIN S., LAMPERT C. H.: Global interactions in random field models: A potential function ensuring connectedness. *SIAM J. Img. Sci. 3*, 4 (2010). 3

[NSB15] NAGARAJA N., SCHMIDT F., BROX T.: Video segmentation with just a few strokes. In *IEEE Int. Conf. Comp. Vision (ICCV)* (2015). 2

[OB11] OCHS P., BROX T.: Object segmentation in video: A hierarchical variational approach for turning point trajectories into dense regions. In *IEEE Int. Conf. Comp. Vision (ICCV)* (2011). 2

[RM00] ROERDINK J. B. T. M., MEIJSTER A.: The watershed transform: definitions, algorithms and parallelization strategies. *Fundam. Inf. 41*, 1-2 (2000). 4

[SGS*06] SHARON E., GALUN M., SHARON D., BASRI R., BRANDT A.: Hierarchy and adaptivity in segmenting visual scenes. *Nature 442*, 7104 (2006). 4

[Sil14] SILHOUETTEFX, LLC.: Silhouette v5 software. , 2004–2014. 1

[SPCR14] SHAHRIAN E., PRICE B., COHEN S., RAJAN D.: Temporally coherent and spatially accurate video matting. *Computer Graphics Forum 33* (2014). 4

[TM98] TOMASI C., MANDUCHI R.: Bilateral filtering for gray and color images. In *IEEE Int. Conf. Comp. Vision (ICCV)* (1998). 4

[VB10] VITALADEVUNI S. N. P., BASRI R.: Co-clustering of image segments using convex optimization applied to EM neuronal reconstruction. In *CVPR* (2010). 2

[WBC*05] WANG J., BHAT P., COLBURN R. A., AGRAWALA M., COHEN M. F.: Interactive video cutout. In *ACM Trans. Graph. (SIGGRAPH)* (2005). 2

[YIF12] YARKONY J., IHLER A., FOWLKES C.: Fast planar correlation clustering for image segmentation. In *ECCV* (2012). 2

[Zha06] ZHANG Y.: *Advances in image and video segmentation*. IGI Global research collection. 2006. 2

[ZQPM12] ZHONG F., QIN X., PENG Q., MENG X.: Discontinuity-aware video object cutout. *ACM Trans. Graph. (SIGGRAPH Asia) 31*, 6 (2012). 2