Semi-Automatic Video Object Segmentation by Advanced Manipulation of Segmentation Hierarchies

Jordi Pont-Tuset Miquel A. Farré Aljoscha Smolic
Disney Research Zurich

***Abstract*—For applications that require very accurate video object segmentations, semi-automatic algorithms are typically used, which help operators to minimize the annotation time, as off-the-shelf automatic segmentation techniques are still far from being precise enough in this context. This paper presents a novel interface based on a click-and-drag interaction that allows to rapidly select regions from state-of-the-art segmentation hierarchies. The interface is very responsive, allows to obtain very accurate segmentations, and is designed to minimize the human interaction. To evaluate the results, we provide a new set of object video ground truth data.**

(a) (b) (c) (d) (e)

Fig. 1. Overview of our approach: (a) mouse down on a region, (b) drag to make the selection grow, (c) mouse up when we cannot grow more inside the object; (d) iterate until all the object is marked, and (e) track the mask to following frames. Images from [8].

I. Introduction

Image and video segmentation is one of the most fundamental yet challenging problems in computer vision. Dividing the image into *meaningful* regions implies a high level interpretation of the image that cannot be satisfactorily solved by looking, for instance, for the *homogeneous* areas in the image, as many of the classical approaches do. In Figure 8.a, for instance, the car is very diverse in color, but humans recognize it as object because they *understand* that it is a car.

In the era of *big data* and vast computing power, one approach to model this high level interpretation of images is to make use of powerful machine learning tools [1] on huge annotated databases [2], [3]. While significant advances have been made in recent years, automatic image segmentation is still far from providing accurate results in a generic scenario.

The approach followed in this paper to achieve this high level interpretation is to incorporate a human in the loop. This type of techniques can be referred to as *semi-automatic image segmentation*, which aims at minimizing the interaction that the human has to do to obtain an accurate result. To achieve this goal, we will make use of a representation of the image known as segmentation hierarchy [4], [5], [6]. It transforms an image from a matrix of pixels to a tree-like graph whose nodes represent regions, from fine superpixels at the leafs to the root representing the whole image.

Hierarchies have demonstrated state-of-the-art performance [4] in image segmentation, object proposals, and contour detection. Therefore, we use them here as the base for our novel semi-automatic video object segmentation algorithm. Intuitively, we cast object segmentation as selecting a set of regions from the hierarchy, and then object tracking as finding the set of regions in the following hierarchies.

We present an interface that allows the user to intuitively and easily select a set of regions from the hierarchy via click-and-drag interaction, which is suitable for both mouse and touchscreen operation. We provide some algorithms that minimize the number of time that the user has to spend annotating, by keeping track not only of the regions that the user has selected, but also those that have been skipped.

When the user has finished annotating the mask of the object in a certain frame of a sequence, we propagate it using optical flow [7], and then we look for regions in the hierarchy that match the propagated pixels to refine the mask and correct potential errors. At each step, the user can easily further refine small errors, and thus the propagation can be done fast.

To validate our approach, we provide new object-based annotations on the Berkeley Segmentation Video Dataset (BVSD) [8], that allow us to estimate the maximum expectable performance, to validate the use of segmentation hierarchies, and to assess our proposal.

The remainder of the paper is organized as follows. Section II gives an overview of related work, and Section III provides the needed background on segmentation hierarchies. The proposed algorithm is described in Section IV, which explains how we make use of hierarchies to help in the annotation of a certain frame, and in Section V, which describes how we propagate the mask to following frames. Section VI describes the experimental validation, and finally we draw the conclusions in Section VII.

II. State of the Art

Video segmentation techniques [9], [10], [11], [12] take a video as input and output a partition of the frames into a set of regions aiming at being as temporally consistent as possible, and at regions between partitions being as connected as possible.

Video object tracking [13], [14] provide trajectories of objects represented by bounding boxes, thus providing only an approximate location of the object. While this type of tracking

978-1-4673-6870-4/15/$31.00

is useful for some applications like pedestrian tracking, others require more precise shape, such as object inpainting.

Video object segmentation and tracking algorithms [15], [16], [17] take a binary mask representing the shape of the object at a given frame, and output the propagated mask to the subsequent frames. Our application fits within this type of algorithms. Other techniques working and optimizing on segmentation hierarchies may be found in [9], [10], but they output full partitions instead of objects, by means of cuts of the tree.

The closely related work to our paper may be found in [18], which presents the Graphical Annotator Tool (GAT). As in our approach, the authors present a tool to annotate objects using segmentation hierarchies. However, in contrast to GAT, our method presents a more advanced interaction approach, is based on an improved segmentation hierarchy, and is much more responsive thanks to how we code the hierarchies. Further, we also cover video object segmentation by tracking and refining the masks over time.

III. Segmentation Hierarchies

This section is devoted to give an overview of segmentation hierarchies, reviewing the state of the art, and providing the necessary background for the following sections.

Current state-of-the-art segmentation techniques are based on a first step of contour detection, represented as a probability map. Najman and Schmitt [19] refer to these probability maps as contour saliency maps and they were the first to show that there is a bijection between contour saliency maps and hierarchies of regions. This idea was later popularized by Arbeláez et al. [5] as Ultrametric Contour Maps (UCM), which have been the state-of-the-art technique for image segmentation since then. Working directly on regions of hierarchies, without starting from a contour map, has also been explored in the so-called Binary Partition Trees [6].

In this work, we will use a recent improvement over UCM called Multiscale Combinatorial Grouping (MCG) [4], which runs significantly faster while obtaining better results than UCM. Let us delve into how these hierarchies are represented. As introduced before, current state-of-the-art segmentation hierarchies are based on contour detectors [20], [21] (MCG is based on [20]), whose output is the probability of each pixel boundary segment of being a contour. Thresholding this probability, we would have a binary contour map, which classifies pixel boundaries into contour/no-contour. The contour probability map is then transformed into a contour saliency map, or UCM, which has the interesting behavior that for any threshold t, the output binary contour map produces closed boundaries; and thus a segmentation of the image whose contours are the ones obtained by the UCM.

This way, each piece of contour in the UCM can be seen as the boundary between two (or more) regions in the image, and thus augmenting the contour strength threshold can be interpreted as merging the neighboring regions. If we represent regions as nodes of a graph, and the merged region as the parent of the original regions, we can represent a UCM as a tree (dendrogram to be more precise) of regions, which we will refer to as segmentation hierarchy.

Figure 2 shows a graphical representation of this duality. On the lower left corner, we have a simple ultrametric contour map. By thresholding it at different contour strengths λ_i we obtain a sequence of closed boundaries, and so partitions, that we refer to as merging-sequence partitions, as depicted in the left-most column. On the right of the plot, each region in the partitions is represented by a node of the graph, and the merging process forms the region tree, or segmentation hierarchy.

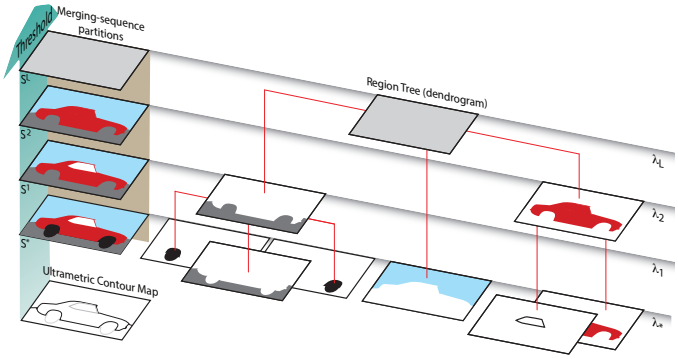


Fig. 2. Graphical representation of a hierarchy as a region dendrogram.

As we will see in next sections, the intuition behind our algorithm is that we process the image directly in the dendrogram of regions, using tailored data structures that allow us to compute the optimized proposals and their features very efficiently.

Figure 3 illustrates how object segmentation is casted to selecting regions in a segmentation hierarchy: (a) shows a simplified version of the hierarchy introduced in Figure 2 and (b) shows the single region in the hierarchy that better represents the car. As we can see, there is no single region that covers the whole car, so (c) shows the best object representation from regions in the hierarchy, in this case, 3 of them.

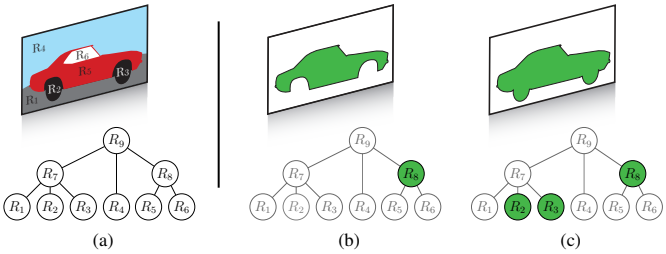


Fig. 3. Examples of objects (b), (c), formed by selecting regions from a hierarchy (a).

This example also illustrates the benefits from working on hierarchies. In a single structure, we can represent objects at different levels of detail, with less regions than on a flat partition. In this particular case, we would need 4 regions (R_2, R_3, R_5, R_6) to represent the same object.

IV. Semi-Automatic Object Segmentation on Hierarchies

This section explains the interface we propose to allow a user to annotate an object via the selection of regions in a hierarchy. First, we describe how our hierarchy representation

allows us to preview the selected regions in real time, very responsively. Then, the core strategy of the algorithm is a click-and-drag interaction that allows to rapidly explore the whole hierarchy. We improve the interaction by keeping track not only of the regions that the user has selected but also those that he has skipped. Finally, we introduce a modification of the hierarchy in order not to have holes when we perform a selection. The following sections expound on these four points. Please also refer to the supplementary video and Figure 1.

A. Real-time selected region preview

One of the most relevant aspects of a tool that involves a human in the loop is its responsiveness, which means that the delay between an order and the response should be as short as possible. In the case of our tool, we want to highlight the region where the cursor is located in real time.

To achieve this goal, let us analyze how the hierarchy can be stored. A first possibility would be to store a matrix of labels, one label per pixel. To highlight region R_i using this representation, we would need to scan all the matrix of labels looking for those pixels with label R_i, at every mouse move, which would make interaction prohibitively slow.

Instead of storing the hierarchy as a matrix of labels, we propose a contour-based representation, in which we store the coordinates of the pieces of contour between neighboring regions; and we index all neighbors of each region. To highlight region R_i, therefore, we just have to scan the reduced set of neighbors, say $\{R_j, R_k, R_l\}$, and paint the contour coordinates between $\{(R_i, R_j), (R_i, R_k), (R_i, R_l)\}$. This way, we reduce the number of operations needed at every mouse move, as well as the memory footprint, and we obtain a smooth and responsive interaction.

B. Click-and-Drag: General region selection

The main idea of our interaction strategy is that the user clicks on a certain area of the image, and then the leaf of the hierarchy at that position (previously highlighted) is selected. While still with the mouse/touchscreen pressed, dragging the pointer makes the selected region grow, by selecting the parent of the region selected at each time instant.

As expounded in the introduction, [18] is based on a similar principle but with the following significant differences. Instead of click and drag, they use the mouse wheel, which prevents the algorithm to be used on touch interfaces. In order to make a click-and-drag approach feasible, the contour representation of the hierarchy plays a key role, which makes it responsive enough to be usable. Apart from this, we use state-of-the-art segmentation techniques and we optimize their parameters, tailored to the semi-automatic task, apart from taking advantage of the non-selected regions to minimize the number of clicks (see following section).

Figure 4 illustrates how the click-and-drag algorithm works. First, the user clicks on a certain leaf (a), and the region is highlighted. He starts dragging, and the leaf is substituted by its parent (b). If the user keeps dragging, it will get to a point where the selected region is too big (c), and so he will have to drag back to the last acceptable region.

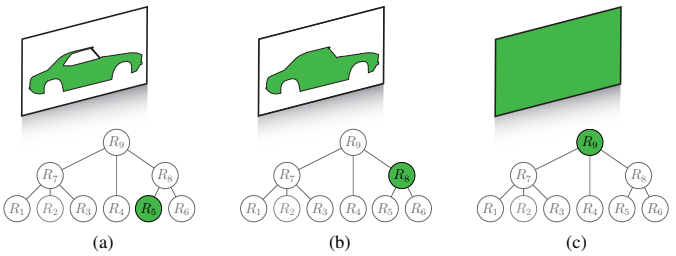


Fig. 4. General region selection on hierarchies via click and drag. We start by clicking on a leaf and growing the region through its ancestors by dragging the pointer.

C. Non-selected regions avoidance

As depicted in the previous example (Figure 4), given that the hierarchies are not perfect, the object of interest won't usually be represented by a single region. This way, the common behavior from a user is to keep growing the object until the selected region expands out of the object. Figure 5 shows the typical behavior in these cases: (a) the object is not complete yet, so the user keeps making the mask grow, until it grows too much (b). He then needs to take a step back (c).

Apart from getting the selected part of the object (c), we can also deduce that the regions added in the last step (b), as a whole, are not part of the object. Observing the hierarchy in (c), for instance, we realize that regions R_7 and R_4 are not both part of the object, and thus are marked as *non-selected regions* (in gray in the figure). We keep track of these sets of regions and we avoid them in future interactions. Figure 6 shows an example of interaction with an avoided region (red stripes in (d)).

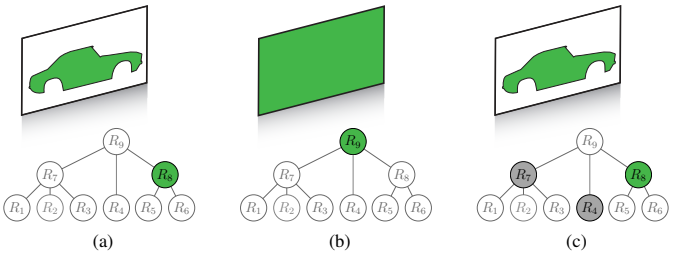


Fig. 5. Non-selected regions avoidance: We keep track of the regions that the user did not select to avoid them in future interactions.

D. Hierarchy hole filling

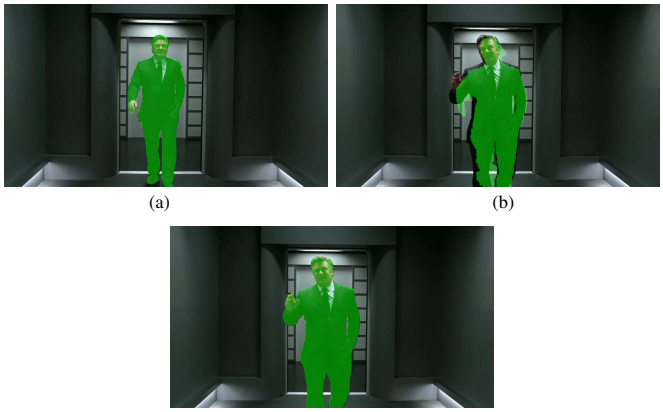
When performing user tests with the tool, we realized that a significant part of the interaction time was devoted to fill the holes that the hierarchy would create. Sometimes they were objects that contained holes, but the common case was that small artificial holes appeared due to textured areas. To solve this issue, we considered filling the holes automatically while the user was interacting.

This approach, however, would entail adding computation to the front-end of the tool. Since we envision our tool to be used on mobile devices, we wanted to minimize the computation at the front end.

We therefore decided to modify the hierarchies in a way that they would contain no hole. We analyzed all regions there were merged at each step of the creation of the hierarchy.

(a) (b) (c) (d)

Fig. 6. Non-selected regions avoidance example: (a) the user clicks on a certain region, (b) drags the mouse until the selection is in the object, (c) one step more and the selection gets background regions, (d) when taking a step back and releasing the mouse, the non-selected region is selected as to avoid (red stripes). Images from [22].



(a) (b) (c)

Fig. 7. Object tracking overview: (a) Marked objects by the user at frame 1, (b) propagated object with optical flow after 40 frames, and (c) mask propagated with optical flow and adapted to the hierarchy. Images from [8].

In case the merging created a hole, we modified the tree so that the hole region was merged together in the same merging. We created an algorithm to detect the holes directly in the hierarchical graph not to have to rely on morphological operations that would have been prohibitive to do so many times.

As a result, we have a very efficient algorithm that converts the hierarchy into another one that has no holes when performing a click-and-drag. This way, the user does not have to fill the holes, but now one cannot creates tags with holes. To solve it, we added the *negative* mode, in which the tool removes selected regions. In this mode, the user can create holes. We observed that the interaction was much faster using this mode.

V. Object Segmentation Tracking

Once the objects are annotated in one frame, we propagate the masks to the following frames to minimize the user interaction. To do so, we pre-compute the optical flow [7], which links the pixels from one frame to the position where they have moved in the following frame. Using this information, we propagate the marked pixels to the next frame, which gives us an estimate of the mask of the objects in that frame.

In order to refine this mask and also to allow the user to further correct potential errors, we look for the set of regions that best match the propagated mask. To do so, one naive approach would be to select those regions whose overlap with the propagated mask is higher that 0.5. As we will show below, this would not achieve the optimum representation in all cases. Let us therefore analyze the problem formally.

Given the propagated mask and a set of non-overlapping regions (leafs of the hierarchy), and assuming we have some regions selected R, let us evaluate whether adding another region R' to it would increase the Jaccard index between the selected regions and the mask we want to fit (O). Let us define: $tp_R = \left|R \cap O\right|$ and $fp_R = \left|R \cap \overline{O}\right|$, the true and false positives in R, respectively. The Jaccard index of this solution would then be:

$$J(R) = \frac{\left|R \cap O\right|}{\left|R \cup O\right|} = \frac{tp_R}{|O| + fp_R}$$

.

We would like to evaluate whether adding a non-overlapping region R' improves the result, that is, we would like to check when:

$$J(R) < J(R \cup R') \;\Leftrightarrow\; \frac{tp_R}{|O| + fp_R} < \frac{tp_R + tp_{R'}}{|O| + fp_R + fp_{R'}}$$

where $fp_{R \cup R'} = fp_R + fp_{R'}$ and $tp_{R \cup R'} = tp_R + tp_{R'}$ hold because R and R' do not overlap. Manipulating the expression we derive that the inequality holds if, and only if

$$J(R) < \frac{tp_{R'}}{fp_{R'}}$$

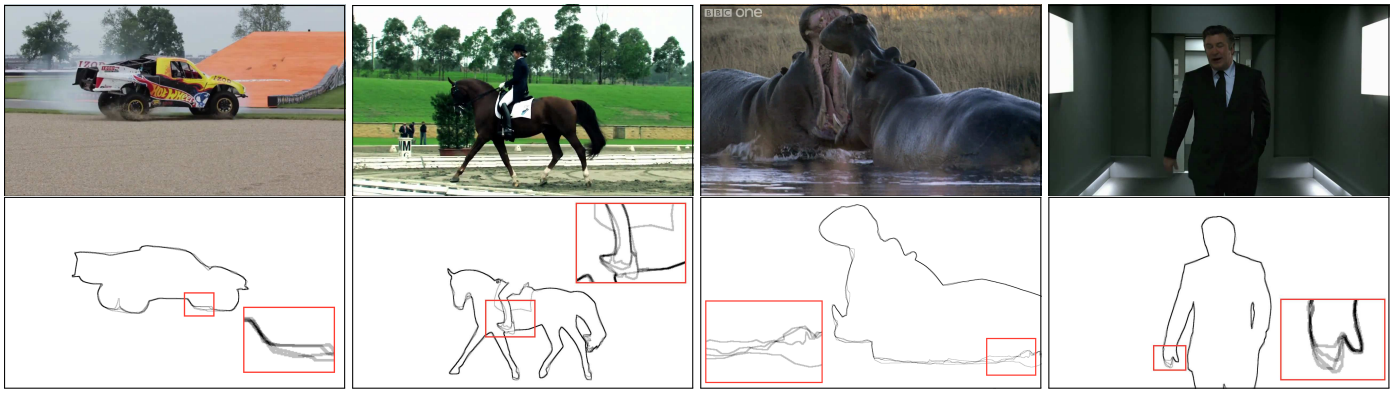
In other words, there is no need to calculate $J(R \cup R')$ to know whether adding R' to R would improve the result. Since we can calculate $\frac{tp_{R'}}{fp_{R'}}$ beforehand for all regions, we can use the inequality to speed the algorithm up.

We therefore sort the regions with respect to $\frac{tp_{R'}}{fp_{R'}}$ and we keep adding regions while the inequality holds, guaranteeing that we reach the optimum representation. Figure 7 shows an example to illustrate the tracking algorithm. It shows that propagating the selected mask using optical flow introduces errors in the long term, especially in the boundaries and in high-motion parts. This errors can be minimized by adapting the propagated masks to the hierarchy.

VI. Experimental Validation

New object annotations:: We base our experiments on the annotated Berkeley Segmentation Video Dataset (BVSD) [8], consisting of 100 high definition sequences. The original ground-truth partitions were only provided for the central frame of each sequence. The database was later annotated by different subjects as reported in [23], for every 20th frame in each sequence, also providing full partitions instead of foreground-background masks.

In this work, we generate foreground-background masks from these multiple partitions and we make them publicly



(a) $J=0.94$ (b) $J=0.84$ (c) $J=0.92$ (d) $J=0.97$

Fig. 8. Example annotations from the database. Even human annotations have a certain degree of variability due to motion blur, low-contrast boundaries, or different levels of semantic interpretation. Images and annotations from [8].

available as High-Definition Video Object Annotations (HD-VOA)[24]. Having multiple foreground masks for each object will allow us to assess the human variance within the annotations and thus normalize the results with respect to the acceptable variability of human masks. Figure 8 shows some ground-truth masks from the dataset, in which the boundaries of the different annotations are all overlaid.

The resulting database consists of 88 sequences (39 training, 49 testing), containing a total of 9 585 frames. In them, 200 objects are annotated every 20 frames, making a total set of 1 270 masks (each of them annotated four times).

Evaluation of human performance:: We start by evaluating the maximum expected quality in the database by evaluating one human against the others; that is, we will evaluate the ground truth created by one annotator as if it were a machine-generated result, by comparing it to the rest of annotations created by the other subjects.

We use the Jaccard index [25] (Intersection over Union) to measure similarity with ground truth, being J=1 a perfect result and J=0 the worst result. In the case of humans, we obtain a mean Jaccard value of $J = 0.89 \pm 0.08$, which corroborates that the dataset is very challenging, because even human results are not perfectly coherent. In other words, the exact position of the object boundaries is inherently ambiguous. This value is in practice a quality reference for what we can expect from any machine-generated result.

Evaluation of performance:: As our system is interactive, the user can improve results until he is satisfied up to a maximum achievable quality, which is a property of our algorithm. It is given as best possible approximation of the ground-truth by selection of elements from the hierarchy, and can be computed to evaluate the system performance for a certain configuration (number of nodes, see below). For that, we use the Single-scale Combinatorial Grouping (SCG) hierarchy [4], given its state of the art quality and computational efficiency.

In a region-based representation of images as ours, we have the situation that the larger the number of regions (nodes in the graph) is, the better the maximum achievable quality is, with per-pixel resolution as theoretical limit. On the other hand, the bigger the graph is, the slower and more memory consuming the processing is. Therefore, we have a classical trade off between quality and computational efficiency, with the number of nodes as crucial parameter for system design.

To assess this trade-off and to choose an optimum number of regions, we computed the maximum achievable quality as function of the number of regions, as plotted in Figure 9. It also includes human performance as reference.

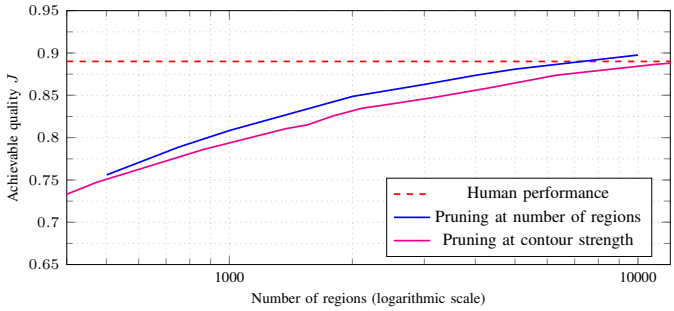


Fig. 9. Achievable quality of the hierarchies with respect to the number of regions in the representation for two different strategies of reducing the number of regions.

The plot shows that, as expected, the more regions in the representation, the more quality we can achieve by selecting regions from the hierarchy, at the expense of being slower. Comparing pruning strategies, pruning with respect to the number of regions gives better performance, at all ranges, so we choose the hierarchy obtained by pruning at 5000 regions for the following evaluations, as it is very close to human performance, although still with an affordable number of regions.

To further analyze the comparison with human performance, Figure 10 shows, for the 102 annotated objects in training, the maximum achievable quality (for 5000 regions) in the hierarchy with respect to the human annotation quality of the same object.

We observe that the quality obtained from hierarchies is very correlated with that of humans, which validates our approach. We find some outliers in which hierarchies do not perform well, which correspond to objects with very low contrasted contours (no low-level contour), in which humans mark

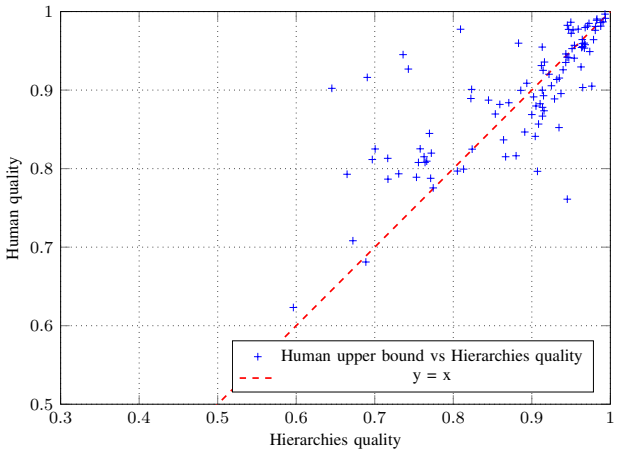


Fig. 10. Human upper bound versus the achievable quality in the hierarchies, for the 102 annotated objects of BVSD.

it by context and semantics. In a semi-automatic environment, the annotator will correct these mistakes.

VII. Conclusions

This paper presents a novel semi-automatic video object segmentation based on a click-and-drag interface to select regions from segmentation hierarchies. The main contributions are the following:
(i) We optimize the state-of-the-art hierarchical segmentation algorithm to find a good point in the quality-speed trade-off.
(ii) To make the interface very responsive, we propose a contour-based representation of the hierarchies.
(iii) To minimize the needed interactions, we keep track both the selected regions and the ones avoided, and we take advantage of it. We also propose a mask tracking algorithm.
(iv) We release a new ground-truth data set [24] of object annotations for the evaluation of video object segmentation algorithms.
In consequence, our approach provides very accurate results, it is very responsive, and the human interaction is minimized.

References

[1] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[2] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A Large-Scale Hierarchical Image Database," in *CVPR*, 2009.

[3] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollr, and C. Zitnick, "Microsoft COCO: Common Objects in Context," in *ECCV*, 2014.

[4] P. Arbeláez, J. Pont-Tuset, J. Barron, F. Marques, and J. Malik, "Multiscale combinatorial grouping," in *CVPR*, 2014.

[5] P. Arbeláez, M. Maire, C. C. Fowlkes, and J. Malik, "Contour detection and hierarchical image segmentation," *IEEE TPAMI*, vol. 33, no. 5, pp. 898–916, 2011.

[6] P. Salembier and L. Garrido, "Binary partition tree as an efficient representation for image processing, segmentation, and information retrieval," *IEEE Transactions on Image Processing*, vol. 9, no. 4, pp. 561–576, 2000.

[7] P. Weinzaepfel, J. Revaud, Z. Harchaoui, and C. Schmid, "DeepFlow: Large displacement optical flow with deep matching," in *ICCV*, 2013.

[8] P. Sundberg, T. Brox, M. Maire, P. Arbeláez, and J. Malik, "Occlusion boundary detection and figure/ground assignment from optical flow," in *CVPR*, 2011.

[9] G. Palou and P. Salembier, "Hierarchical video representation with trajectory binary partition tree," in *CVPR*, 2013.

[10] C. Xu, S. Whitt, and J. J. Corso, "Flattening supervoxel hierarchies by the uniform entropy slice," in *ICCV*, 2013.

[11] F. Galasso, R. Cipolla, and B. Schiele, "Video segmentation with superpixels," in *ACCV*, 2013.

[12] C. Couprie, C. Farabet, and Y. LeCun, "Causal graph-based video segmentation," *arXiv preprint arXiv:1301.1671*, 2013.

[13] L. Leal-Taixé, M. Fenzi, A. Kuznetsova, B. Rosenhahn, and S. Savarese, "Learning an image-based motion context for multiple people tracking," in *CVPR*, 2014.

[14] Y. Wu, J. Lim, and M.-H. Yang, "Online object tracking: A benchmark," in *CVPR*, 2013.

[15] D. Varas and F. Marques, "Region-based particle filter for video object segmentation," in *CVPR*, 2014.

[16] D. Banica, A. Agape, A. Ion, and C. Sminchisescu, "Video object segmentation by salient segment chain composition," in *ICCVW*, 2013.

[17] D. Zhang, O. Javed, and M. Shah, "Video object segmentation through spatially accurate and temporally dense extraction of primary object regions," in *CVPR*, 2013, pp. 628–635.

[18] X. Giro-i Nieto, N. Camps, and F. Marques, "GAT: a Graphical Annotation Tool for semantic regions," *Multimedia Tools and Applications*, vol. 46, no. 2-3, pp. 155–174, 2010. [Online]. Available: http://dx.doi.org/10.1007/s11042-009-0389-2

[19] L. Najman and M. Schmitt, "Geodesic saliency of watershed contours and hierarchical segmentation," *TPAMI*, vol. 18, no. 12, pp. 1163–1173, 1996.

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

[21] X. Ren and L. Bo, "Discriminatively trained sparse code gradients for contour detection," in *NIPS*, 2012.

[22] (CC) Blender Foundation. Tears of steel. mango.blender.org.

[23] F. Galasso, N. S. Nagaraja, T. J. Cárdenas, T. Brox, and B. Schiele, "A unified video segmentation benchmark: Annotation, metrics and analysis," in *ICCV*. IEEE, 2013, pp. 3527–3534.

[24] "HD-VOA: High-Definition Video Object Annotations," http://www.vision.ee.ethz.ch/~jpont/hdvoa/.

[25] P. Jaccard, "Étude comparative de la distribution florale dans une portion des alpes et des jura," *Bulletin del la Société Vaudoise des Sciences Naturelles*, vol. 37, pp. 547–579, 1901.