Video Object Segmentation Using Graphs

Abstract. This paper presents an approach for video object segmentation. The main idea of our approach is to generate a planar, triangulated, and labeled graph that describes the scene, foreground objects and background. With the help of the KLT Tracker, corner points are tracked within a video sequence. Then the movement of the points adaptively generates a planar triangulation. The triangles are labeled as *rigid*, *articulated*, and *separating* depending on the variation of the length of their edges.

Key words: Video object segmentation, adaptive triangulation, articulated objects

1 Introduction

Video object segmentation (VOS) is an important task in computer vision to separate foreground from background and initialize tracking systems. VOS methods can be divided up like in [1] into (1) two-frame motion/object segmentation and (2) multi-frame spatio-temporal segmentation/tracking.

Former methods are [2–5]. Alatan et al. present in [2] the activities of the COST 211^{ter} group dedicated toward image and video sequence analysis and segmentation, which is an important technological aspect for the success of emerging object-based MPEG-4 and MPEG-7 multimedia applications. In [3], Altunbasak et al. describe a combination of pixel-based and region-based methods to obtain the best possible segmentation results on a variety of image sequences. Castagno et al. present in [4] a scheme for interactive video segmentation. A key feature of the system is the distinction between two levels of segmentation, namely, regions and object segmentation. Chen et al. describe in [5] a method to segment highly articulated video objects with weak-prior random forests. The random forests are used to derive the prior probabilities of the objects's configuration for an input frame. The prior is used to guide the grouping of over-segmented regions.

Latter methods are [1, 6–8]. Celasun et al. write in [6] and [1] about 2-D mesh-based VOS. In [8], Tekalp et al. also present a 2-D mesh-based approach. They describe 2-D mesh-based modeling of video objects as a compact representation of motion and shape for interactive, synthetic/natural video manipulation, compression, and indexing. The affine motion model is often used in VOS, because it is simple and locally a good approximation of smooth motion. With only six parameters it is able to describe complex motions like for example rotation, scaling and shearing. Li et al. present in [7] an approach where they use the affine motion model to estimate the motion of homogeneous regions.

Artner et al. present in [9] a kernel-based tracking method using a spatial structure. They showed with their experiments that structure (graph-based representation of the target object) can enhance the results of tracking. In [9], the

target objects were rigid and the initialization for the tracking process was done manually by selecting the target object.

This paper presents a new approach of VOS, where the result is a triangulated, labeled graph of the scene, which describes foreground objects and background. Each triangle of the graph is labeled either as *rigid*, *articulated* or *separating*, depending on its behavior during a video sequence. We plan to combine the approach in this paper with the method described in [9] to automatically initialize the tracking process and allow to track articulated objects.

The paper is organized as follows: Section 2 recalls the approach in [9]. Section 3 describes our approach. Section 4 shows results with different video sequences. In Section 5 conclusions and future work are given.

2 Tracking Using Spatial Structure

Artner et al. propose in [9] an initial concept for combining deterministic tracking of object parts with graph representation encoding structural dependencies between the parts. In general, image graphs can be used to represent structure and topology. The Maximally Stable Extremal Regions (MSER) detector [10] is used in [9] to generate regions which represent the vertices of the graph. The MSER computation is used only once to initialize the graph structure (Delaunay triangulation) and the Mean Shift trackers at each vertex. On the vertices, color histograms are computed to obtain an attributed graph (AG). The edges between the vertices define the region adjacencies.

The objective of Artner et al. is to link the processes of (1) structural energy minimization of the graph and (2) color histogram similarity maximization at the vertices by Mean Shift tracking. The algorithmic combination of Mean Shift and graph relaxation represents a joint iterative mode seeking process on the color similarity and on the structural energy surfaces. As the tracked objects in [9] are rigid, the objective of the relaxation is to maintain the tracked structure as similar as possible to the initial structure.

3 Extracting Structure

Our approach can be divided into 3 steps: track interest points in video with any tracker, build a planar, triangulated graph and analyse the movement of the points (vertices) of the graph over time and label its triangles.

The Kanade-Lucas-Tomasi tracker is used to track corner points. We use the implementation of [11]. At the beginning the algorithm selects good features to track and then keeps track of this features. The main idea of the KLT tracker is that feature extraction should not be separated from tracking. If a feature is lost during the tracking process, it is not considered in the following steps of our approach, because it is not dependable.

3.1 Adaptive Triangulation

Let $l(\mathbf{p_1}, \mathbf{p_2})$ denote the *likelihood* that two image points $\mathbf{p_1}$, $\mathbf{p_2}$ belong together i.e. are part of the same (rigid) structure. $l(\mathbf{p_1}, \mathbf{p_2}) = l(\mathbf{p_2}, \mathbf{p_1})$, and $l(\mathbf{p_1}, \mathbf{p_2}) < l(\mathbf{p_2}, \mathbf{p_3})$ means that $\mathbf{p_1}$, $\mathbf{p_2}$ are more likely to be part of the same rigid structure than $\mathbf{p_2}$, $\mathbf{p_3}$. l is understood as a nonnegative continuous measure. Note that in the ideal case l should also depend on the type of object considered.

Pairs of points and their properties can be used to decide whether they belong to the same rigid structure or not, but cannot be applied directly to find articulation points. More global information or local information of higher dimensional cells (e.g. triangles) has to be considered. The latter one is used.

A *triangulation* of a set of points $\mathbf{P} \in \mathbb{R}^2$ is a subdivision of the convex hull of the points into triangles. A frequently used triangulation [12] is the *Delaunay triangulation* [13]. The Delaunay triangulation $DT(\mathbf{P})$ of a set of points $\mathbf{P}$ ensures that no point $\mathbf{p} \in \mathbf{P}$ lies within the circumcircle of any triangle. It maximizes the minimum angle of all the angles of the triangles in the triangulation; it tends to avoid "sliver" triangles. $DT(\mathbf{P})$ is not always unique (e.g. for 4 points on the same circle) and can be computed in $O(n \log n)$ in the number of points. The disadvantage of using $DT(\mathbf{P})$ is that the presence of edges is decided solely on the position of the points in $\mathbb{R}^2$, and additional hints like the likelihood l are hard to integrate. One could imagine finding a new set of points for which the euclidean distance is proportional to l and then computing the triangulation, but such a set does not always exist. The *constrained Delaunay triangulation* requires a priory knowledge of the edges that are necessarily part of it.

In the ideal case, the used triangulation mainly contains edges with high l, increasing the chance that vertices of the same rigid part are connected, and articulation points are correctly detected. The following formulation is used: given a fully connected graph $G = (\mathbf{V}, \mathbf{E})$, where all $\mathbf{v} \in \mathbf{V}$ correspond to a unique point $\mathbf{p} \in \mathbf{P}$ (bijection), and all $\mathbf{e} = (\mathbf{v_1}, \mathbf{v_2}) \in \mathbf{E}$ are weighted with $l(\mathbf{p_1}, \mathbf{p_2})$, where $\mathbf{p_1}$, $\mathbf{p_2}$ are the points associated to $\mathbf{v_1}$, respectively $\mathbf{v_2}$, find the connected subgraph $T = (\mathbf{V}, \mathbf{E^T})$ s.t. T is a triangulation, trying to keep edges with higher likelihoods. Algorithm 1 computes T, for a given fully connected graph G with $|\mathbf{V}| > 2$. Note that Algorithm 1 converges after maximum $|E|$ steps, as in each iteration at least one edge is removed from $\mathbf{S}$ (Line 7).

Property 1. If G is a fully connected graph, Algorithm 1 returns a triangulation.

Proof. Assume T contains one face F with $k > 3$ vertices. As G is a fully connected graph, all k vertices of the face F are pairwise connected. Any "internal" edge i is removed only if it intersects a "surviving" edge e (Line 6 in Algorithm 1). For F, the surviving edge e could connect:

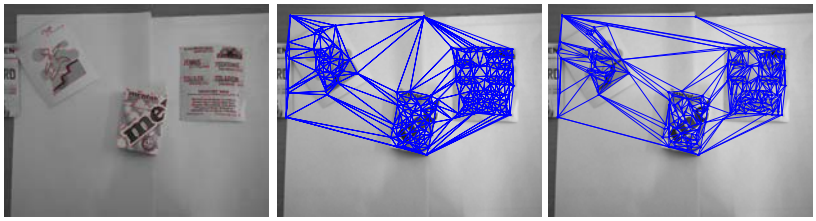
- two vertices not part of F: would not produce F as a face, as it would remove also at least 2 other edges of F and disconnect it;
- one vertex of F with one vertex not part of F: would not produce F as a face, as it would remove also at least 1 other edge of F and disconnect it;

Algorithm 1 *ComputeTriangulation*(G)

Input: Weighted Graph $G = (\mathbf{V}, \mathbf{E})$

1: $\mathbf{E^T} \leftarrow \mathbf{E}$
2: $\mathbf{S} \leftarrow$ sort edges $\mathbf{E}$ in decreasing order of their likelihood
3: **while** $\mathbf{S} \neq \emptyset$ **do**
4: $\quad \mathbf{e} \leftarrow$ first edge in $\mathbf{S}$
5: $\quad \mathbf{I} \leftarrow$ edges in $\mathbf{E}$ "intersecting" edge $\mathbf{e}$ /**common end points not included*/
6: $\quad \mathbf{E^T} \leftarrow \mathbf{E^T} - \mathbf{I}$ /**remove cutting edges from result*/
7: $\quad \mathbf{S} \leftarrow \mathbf{S} - (\{\mathbf{e}\} \cup \mathbf{I})$ /**processed edges will not be considered again*/
8: **end while**

Output: Triangulation $T = (\mathbf{V}, \mathbf{E^T})$.



(a) KLT corners (b) Delaunay (c) Adaptive

Fig. 1. Triangulation of corner points selected by the KLT Tracker.

– two vertices of F, not defining a boundary edge: edge e would divide F in two parts being either triangles or fully connected subgraphs. □

Property 2. If $G(\mathbf{V}, \mathbf{E})$ is a supergraph of a triangulation s.t. $\forall e \in \mathbf{E}$, with $\mathbf{I} \subset \mathbf{E}$ all edges intersecting e, $G'(\mathbf{V}, \mathbf{E} - \mathbf{I})$ is triangulation or a supergraph of a triangulation with the same property, Algorithm 1 returns a triangulation. (proof similar to Property 1)

Starting with a fully connected graph is computationally expensive ($|E|^2 = |V|^4$ intersection), thus a faster approximation solution is proposed. We propose G to be a "non optimal" triangulation T_i to which all end vertices of paths of length two are added. Algorithm 1 is applied to select the triangulation T considering l. Note that Property 2 is not satisfied for any triangulation T_i, and to make sure, additional conditions are necessary (e.g. adding only edges connecting the non common vertices of two adjacent triangles).

We have set G to be $DT(\mathbf{P})$ to which edges connecting all end vertices of paths of length two were added. $l(\mathbf{p_1}, \mathbf{p_2})$ was set inversely proportional to the standard deviation of the distance between $\mathbf{p_1}, \mathbf{p_2}$. For $DT(\mathbf{P})$, the points that have been tracked successfully over the whole sequence and their position in the first frame was used.

For the previous, Algorithm 1 always produced a triangulation. Figure 1 (c) shows an example of the produced triangulation.

An alternative approach to the one in Algorithm 1 would have been to weight the edges with the inverese of the likelihood l, build the *Minimum Spanning Tree*

of G and add edges with high l to produce a triangulation. Planarity would still have to be ensured, requiring the edge intersections to be done.

3.2 Motion Analysis and Labeling

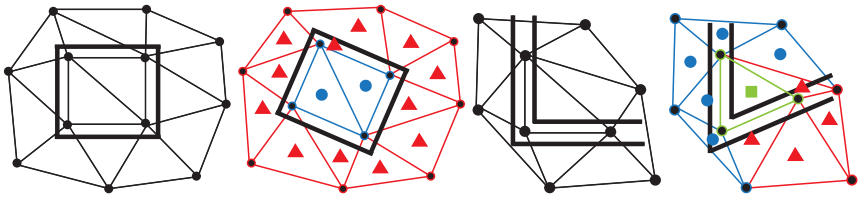
The labeling of the triangles depends on the movement of their vertices $\mathbf{v}$ and so on the variation of the length of their edges $\mathbf{e}$. Every edge $\mathbf{e}$ is weighted depending on its variation with $w(\mathbf{e}) = max(e_t) - min(e_t)$, where $max(e_t)$ and $min(e_t)$ are the maximum and the minimum length of edge e during the whole sequence, respectively. In our experiments, $w(e)$ proved to be robust against repeated small errors in the tracking. The standard deviation was also considered, but constant edge length for a longer time reduces the effect of length variation on the weight.

If $w(e) > \epsilon$, the edge is treated as *eventful* during the analysis. The threshold ϵ is used to compensate small variations in the edge length due to discretization, noise or tracking errors. The triangles are labeled with the following rules:

- ● *rigid*: The weights of all three edges of a triangle lie over the threshold ϵ.
- ■ *articulated*: The weight of one edge of a triangle is under ϵ.
- ▲ *separating*: At least two edges are eventful, which indicates that the triangle is a connection between foreground and background or another object.

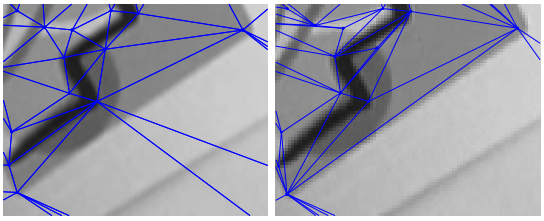
In Figure 2, two simple examples of our labeling mechanism are shown. The sequence in the examples consists only of two frames, because of simplicity. In our experiments (see Section 4) we used a full sequence of frames. Figure 2 (a) and (b) show the two frames $E_a(1)$ and $E_a(2)$ of example E_a. Due to the movement of the vertices (points) from frame (a) to (b), the square is detected as a rigid foreground object, because its vertices did not move and so its triangles are labeled with ● rigid (blue). The triangles that connect the square to the background are labeled with ▲ separating (red), which means this edges should be cut to separate the foreground object from the background.

E_b in Figure 2 (c) and (d) is a labeling example with all three kinds of labels: ● rigid (blue), ■ articulated (green), and ▲ separating (red). One triangle in E_b is labeled as articulated, because on edge of the triangle changed its length during the initialization phase over the threshold ϵ.



(a) $E_a(1)$ (b) $E_a(2)$ (c) $E_b(1)$ (d) $E_b(2)$

Fig. 2. Labeling triangles: ● rigid (blue), ■ articulated (green), ▲ separating (red).



(Delaunay) (Adaptive triangulation)

Fig. 3. Comparison of triangulations. Detailed view.

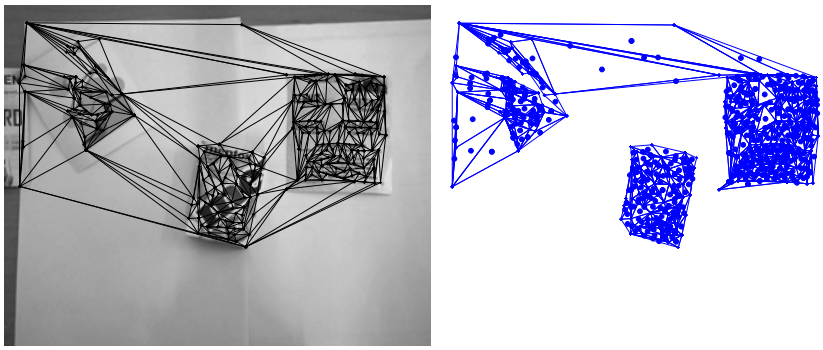
4 Experiments and discussion

The KLT Tracker [14] was used to track corner points and the triangulation is build as described in Algorithm 1. Then the movement of the tracked points, grouped in triangles, is analysed and the graph is labeled (see Section 3.2).

Adaptive triangulation vs Delaunay triangulation: For this experiment we used video sequence 1, which shows a box of candy (rigid object) moving, while the rest of the objects are static (see Figure 1). The proposed adaptive triangulation gives higher priority to connections between points keeping the same relative position (e.g. long horizontal edge in the top-center of Figure 1 (c), and long diagonal edge in Figure 3 (b)). Methods like [9] benefit from edges inside the same rigid object.

Labeling rigid triangles: In sequence 1 the background and the static objects (left and right) are detected as a rigid structure (Fig. 4). The moving candy box (center) is also identified as a rigid object, and the triangles between the candy box and the background are correctly labeled as separating (are not drawn). No articulation points are detected, which is correct too.

Labeling articulated triangles: Video sequence 2 (Fig. 5) shows a human moving his head and arms. Articulation points located in shoulder, elbows and

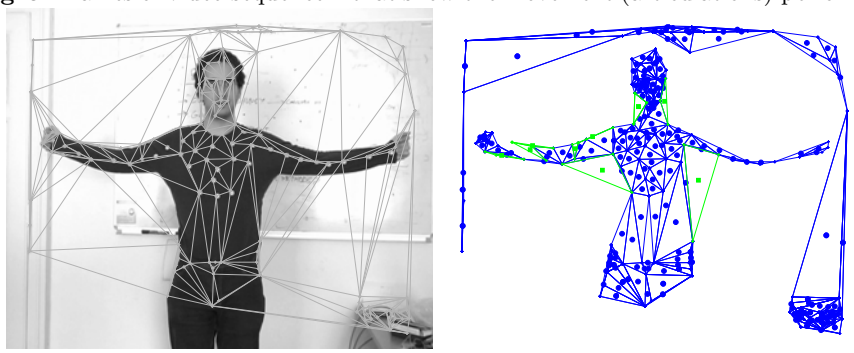


(a) Adaptive triangulation (b) Labeling without "separating"

Fig. 4. Results for video sequence 1. Triangles labeled as separating are not shown.

(Frame 1) (Frame 422) (Frame 455) (Frame 663)

Fig. 5. Frames of video sequence 2 that show the movement (articulations) performed.



(a) Adaptive triangulation (b) Labeling without "separating"

Fig. 6. Results for video sequence 2. Triangles labeled as separating are not shown.

neck can be detected. Round stickers were used to provide sufficient corner points for the KLT tracker in structurally interesting positions.

Problems during the tracking process produced some inconsistent results (i.e. some separating and articulation triangles inside the right arm). Nevertheless, the labeling is satisfying as most of the articulation points are located (only the left elbow is not detected) and the rigid and separating triangles are correctly labeled (Fig. 6).

5 Conclusion and Future Work

The adaptive triangulation presented in this paper allows a better description of the structure of a scene than the Delaunay triangulation, because it takes into account the likelihood l of the points belonging together. The spatio-temporal analysis of the movement of the points during the video sequence classifies triangles into rigid, articulated and separating. Our labeling allows to separate foreground and background objects and identifies articulation points.

In future we plan to combine this approach with kernel-based tracking [9] to track articulated objects. The structural constraints of the graph relaxation can be deduced from the labeled graph. No structural constraints are applied on triangles which are labeled as separating, while for rigid parts the objective is to maintain the tracking consistent with the structure of the object. In the

articulated parts edge variations up to a defined degree of freedom are allowed to model articulated motion (e.g. human motion).

References

1. Celasun, I., Tekalp, A.M., Gokcetekin, M.H., Harmanci, D.M.: 2-d mesh-based video object segmentation and tracking with occlusion resolution. Signal Processing: Image Communication **16**(10) (August 2001) 949–962
2. Alatan, A.A., Onural, L., Wollborn, M., Mech, R., Tuncel, E., Sikora, T.: Image sequence analysis for emerging interactive multimedia services. Circuits and Systems for Video Technology, IEEE Transactions on **8**(7) (Nov 1998) 802–813
3. Altunbasak, Y., Eren, P.E., Tekalp, A.M.: Region-based parametric motion segmentation using color information. Graphical Models and Image Processing **60**(1) (January 1998) 13–23
4. Castagno, R., Ebrahimi, T., Kunt, M.: Video segmentation based on multiple features for interactive multimedia applications. Circuits and Systems for Video Technology, IEEE Transactions on **8**(5) (Sep 1998) 562–571
5. Chen, H.T., Liu, T.L., Fuh, C.S.: Segmenting highly articulated video objects with weak-prior random forests. In: ECCV, Graz, Austria, Springer (2006) 373–385
6. Celasun, I., Tekalp, A.: Optimal 2-d hierarchical content-based mesh design and update for object-based video. Circuits and Systems for Video Technology, IEEE Transactions on **10**(7) (Oct 2000) 1135–1153
7. Li, H., Lin, W., Tye, B., Ong, E., Ko, C.: Object segmentation with affine motion similarity measure. Multimedia and Expo, 2001. ICME 2001. IEEE International Conference on (22-25 Aug. 2001) 841–844
8. Tekalp, A., Van Beek, P., Toklu, C., Gunsel, B.: Two-dimensional mesh-based visual-object representation for interactive synthetic/natural digital video. Proceedings of the IEEE **86**(6) (Jun 1998) 1029–1051
9. Artner, N., López Mámol, S.B., Beleznai, C., Kropatsch, W.G.: Kernel-based tracking using spatial structure (accepted). In: 32nd annual workshop of the Austrian Association for Pattern Recognition. (2008)
10. Matas, J., Chum, O., Urban, M., Pajdla, T.: Robust wide baseline stereo from maximally stable extremal regions. Image and Vision Computing **22**(10) (September 2004) 761–767
11. Birchfeld, S.: Klt: An implementation of the kanade-lucas-tomasi feature tracker. `http://www.ces.clemson.edu/~stb/klt/` (March 2008)
12. Moss, S., Wilson, R.C., Hancock, E.R.: A mixture model for pose clustering. Pattern Recognition Letters **20**(11–13) (November 1999) 1093–1101
13. Klette, R., Rosenfeld, A.: Digital Geometry. Morgan Kaufmann (2004)
14. Shi, J., Tomasi, C.: Good features to track. In: Proceedings of the Conference on Computer Vision and Pattern Recognition, Los Alamitos, CA, USA, IEEE Computer Society Press (June 1994) 593–600