Fast Non-Rigid Object Boundary Tracking

Michael Donoser and Horst Bischof
Institute of Computer Graphics and Vision
Graz University of Technology, Austria
{donoser,bischof}@icg.tugraz.at

Abstract

This paper introduces a method which provides robust tracking results and accurately segmented object boundaries in short computation time. The first step of the algorithm is to apply a novel edge detector on efficiently calculated color probability maps in an object-specific Fisher color space. The proposed edge detector exploits context information by finding the maximally stable boundaries of connected regions in threshold results outperforming purely local edge detectors. Finally, based on the estimated edge maps a probabilistic particle filtering framework hypothesizes rigid transformations for initializing an active contour model to provide accurate object segmentations in each frame. Experimental evaluations show that robust tracking results with accurate segmentations are obtained on challenging data sets.

1 Introduction

Visual tracking is a topic of high interest in computer vision and has versatile applications, as e. g. in video surveillance, motion analysis or gesture recognition. Although tracking algorithms have to cope with major complications as e. g. clutter, occlusions, background distraction, illumination changes or changes in appearance and shape of the object, many robust tracking frameworks have been proposed.

In general, current tracking algorithms can be divided into deterministic and probabilistic approaches. Deterministic approaches as e. g. the mean shift tracker [4] or the covariance tracker [15] are calculated in an efficient manner, but they are sensitive to clutter and occlusions. Recently, probabilistic approaches based on various extensions of the classical particle filtering framework [9] were quite popular. Particle filtering maintains multiple hypotheses of the current object state and therefore provides impressively robust tracking results.

Most of the currently available tracking frameworks only provide a simple bounding box as tracking result assuming that the object-to-be-tracked has approximately a rectangular shape. Recently, similar as in the field of object detection, different authors, as e. g. Ren and Malik [17], Angelopoulous et al. [1], Kohli and Torr [11] or Moreno-Noguer et al. [13] focused on providing accurate object segmentations instead of simple bounding boxes for tracking.

The main issue of most of these methods is the computational complexity required to be able to perform subsequent figure-ground segmentations. For example, Moreno-Noguer et al. [13] integrated multiple cues into a particle filtering framework and showed

BMVC 2008 doi:10.5244/C.22.1

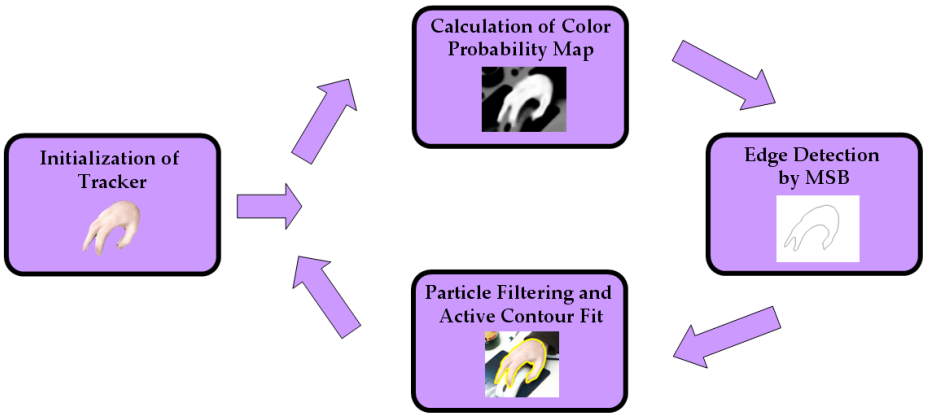


Figure 1: Illustration of tracking framework. After initialization three iteratively repeated steps enable robust tracking results and accurately segmented region boundaries.

impressive results on challenging data sets. But their algorithm requires approximately a minute per frame for tracking.

In this paper we propose a novel tracking framework, which enables robust tracking results and additionally provides accurate object segmentations per frame. Furthermore, all required steps have a low computational complexity allowing to use the tracker in real-time applications. The main idea of our tracking framework is similar to interactive segmentation algorithms, as e. g. in the seminal work of Boykov and Jolly [2], where a user has to initialize the segmentation method by drawing free-form-lines into the image. Instead of using an user provided initialization we use the segmentation of the object in the previous frame `t-1` to perform the figure-ground segmentation in frame `t`. If the figure-ground segmentation has low computational complexity such an approach can be used for real-time tracking applications. In such a way we do not need to model any dynamics of the scene and no a-priori knowledge of the object is required.

The main contributions of this paper are: we first propose to use a modified version of an efficient color segmentation algorithm for tracking purposes. Second, we show how an object-specific color space is elegantly integrated into segmentation. Third, we introduce a novel edge detector exploiting context information which outperforms purely local detectors. Finally, we demonstrate that robust tracking results are obtained providing accurate object segmentations per frame.

The outline of the paper is as follows. Section 2 introduces the method which is based on fitting active contours to efficiently calculated color probability edge maps. Section 3 shows an experimental evaluation on challenging data sets which demonstrate that robust tracking results with accurately segmented object boundaries are obtained. Finally Section 4 draws some conclusions and gives an outlook to future work.

2 Tracking framework

Our method requires a segmented object-of-interest as initialization. Tracking then mainly consists of three subsequently repeated steps, as it is illustrated in Figure 1. First, for a new frame of the input sequence a color probability map is calculated, which measures the similarity of the local neighborhood of every pixel to the current object-of-interest color distribution in an object-specific Fisher color space (Section 2.1). Second, a novel edge detector is applied to the color similarity maps which exploits context information and outperforms purely local detectors (Section 2.2). Third, a probabilistic particle filtering framework is used to hypothesize rigid transformations based on the edge maps. Because the particle filtering step only considers rigid transformations, an active contour model is finally applied to be able to adapt to non-rigid deformations (Section 2.3).

2.1 Calculation of color probability map

The first step of the tracking framework is to calculate a color probability map $p(\mathrm{O}|x)$ for the currently analyzed frame t, which measures the similarity between the color appearance of the object-to-be-tracked O and the local neighborhood of every pixel x. The resulting map $p(\mathrm{O}|x)$ can be visualized as shown in Figure 2(c), where bright values indicate areas with high color similarity.

The values $p(\mathrm{O}|x)$ are calculated by a sliding window approach, where subsequently a model of the color appearance of the local neighborhood – defined by a window of size $L \times L$ – is compared to the appearance of the object O. Since we focus on a real-time capable method the calculation of these color probability maps has to be as efficient as possible. We propose to exploit the properties of a modified version of a color segmentation algorithm introduced by Donoser and Bischof [7] for tracking purposes. They demonstrated that state-of-the-art unsupervised color segmentation results can be achieved within short computation time by combining differently focused sub-segmentations. In their sub-segmentation step an efficient algorithm for calculating local color probability values is introduced. We adapt their method by converting pixel values into an object-specific color space especially suited for tracking applications and by using the Kullback-Leibler divergence as a better suited distance measure.

We start by modeling the color appearance of a segmented object O by a Gaussian Mixture Model Θ_{rgb} in the RGB color space, defined by

$$p(x, \Theta_{rgb}) = \sum_{c=1}^{C} \omega_c \, \mathrm{N}(\mu_c, \Sigma_c), \tag{1}$$

which consists of a weighted linear combination of C Gaussian distributions $\mathrm{N}(\mu_c, \Sigma_c)$, where μ_c is the 3×1 mean vector and Σ_c is the 3×3 covariance matrix of the c-th component. The weights ω_c describe the proportion of the c-th component and fulfill $\sum_{c=1}^{C} \omega_c = 1$. The parameters of the model are estimated as maximum likelihood parameters using the Expectation–Maximization (EM) algorithm [5] initialized by a Mean Shift process in the manner of Cho et al. [3], which automatically chooses the number of components.

In order to better discriminate the color distribution of the object from its background we transform each component of the GMM to a component-specific color space. The idea of transforming into an object-dependent color space for tracking applications was

proposed by Moreno-Noguer et al. [13], who showed that the best feature space is the one that maximizes the distance between the object and its surrounding background color values. They used a single transformation to a 2D Fisher space, whereas we extend this idea by introducing an optimal space for every component of the GMM.

After fitting the GMM Θ_{rgb} to the object O we are able to assign every pixel of the object to one of the estimated components, providing C point sets P_c. Furthermore, for every point set P_c we can also define a set N_c of neighborhood pixels n_c^i fulfilling

$$\forall n_c^i \in N_c \rightarrow d\left(n_c^i, P_c\right) \leq \delta \wedge n_c^i \notin P_c, \tag{2}$$

where $d\left(n_c^i, P_c\right)$ denotes the Euclidean distance of the pixel n_c^i to the nearest pixel in the set P_c and δ is a maximum neighborhood distance parameter.

Based on these point sets we calculate an optimal color space for each component that discriminates the color values of the pixels in P_c from its corresponding non-object neighbors N_c by non-parametric discriminant analysis (NDA) [8], which is an extension of the classical linear discriminant analysis (LDA). In general, LDA finds the hyperplane in the feature space, which best separates the two sets P_c and N_c by maximizing the between class scatter matrix S_b and minimizing the within class scatter matrix S_w. In the classical LDA setup, the rank of the between class scatter matrix S_b is only one for our two class problem and the projected data points would lie on a line. To overcome this limitation we use, similar to Moreno-Noguer et al. [13], the non-parametric version of LDA denoted as NDA [8], where the calculation of the between class scatter matrix S_b is extended by including local information based on a K-nearest neighbor rule which makes the matrix a full-rank one. Thus, NDA provides a projection of a pixel p^{rgb} in the three-dimensional RGB space to a sample p^{fis} in the two-dimensional Fisher space by a 2×3 matrix W by

$$p^{fis} = W\, p^{rgb}. \tag{3}$$

In our case, NDA provides a component-dependent 2×3 mapping W_c for each of the C components of the GMM Θ_{rgb}. Each component in the three-dimensional RGB space is then transformed into its corresponding Fisher space by multiplying the means and the covariances by

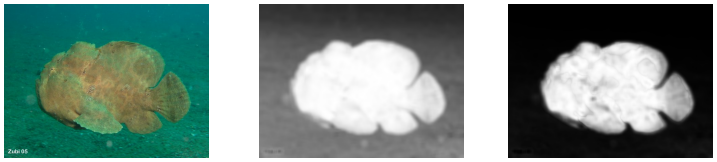
$$\mu_c^{fis} = W_c\, \mu_c^{rgb} \quad \text{and} \quad \Sigma_c^{fis} = W_c\, \Sigma_c^{rgb}\, {W_c}^T, \tag{4}$$

and the GMM Θ_{fis} in the Fisher space is defined by

$$p(x, \Theta_{fis}) = \sum_{c=1}^{C} \omega_c \,\mathrm{N}\left(W_c\, \mu_c^{rgb}, W_c\, \Sigma_c^{rgb}\, {W_c}^T\right) = \sum_{c=1}^{C} \omega_c \,\mathrm{N}\left(\mu_c^{fis}, \Sigma_c^{fis}\right). \tag{5}$$

This GMM Θ_{fis} is then compared to the color appearance of the local neighborhoods of each pixel by a sliding window approach for calculating the required color probability map $p(\mathrm{O}|x)$. First, a single Gaussian $\mathrm{N}(\mu_w^{rgb}, \Sigma_w^{rgb})$ is fitted to the color values of a window of size $L \times L$ located on the currently investigated pixel, assuming locally uniform distribution. Then this Gaussian is transformed to the estimated C Fisher spaces of the GMM Θ_{fis} by applying the corresponding transformation matrices W_c based on Equation 4 resulting in C Gaussians $\mathrm{N}(\mu_{wc}^{fis}, \Sigma_{wc}^{fis})$. Finally, in such a way a color likelihood value $p(\mathrm{O}|x)$ is calculated between the GMM Θ_{fis} and the single Gaussian by

$$p(\mathrm{O}|x) = exp\left(-\sum_{c=1}^{C} \omega_c\, \beta\left(\mathrm{N}(\mu_{wc}^{fis}, \Sigma_{wc}^{fis}), \mathrm{N}(\mu_c^{fis}, \Sigma_c^{fis})\right)\right) \tag{6}$$



(a) Object to track (b) RGB modeling (c) Fisher modeling

Figure 2: Illustration of differences in color probability maps $p(\mathrm{O}|x)$ of the object O (a) by modeling in RGB color space (b) and in Fisher color space (c).

where $\beta(\mathrm{N}_1, \mathrm{N}_2)$ denotes the Kullback-Leibler (KL) divergence between two single Gaussians N_1 and N_2 which has an analytical solution defined by

$$\beta(\mathrm{N}_1, \mathrm{N}_2) = \frac{1}{2}\left(\log\frac{|\Sigma_2|}{|\Sigma_1|}\right) + tr\left(\Sigma_2^{-1}\Sigma_1\right) - d + (\mu_1 - \mu_2)^T \Sigma_2^{-1}(\mu_1 - \mu_2), \quad (7)$$

where d is the dimensionality of the feature space used, i. e. two in our case and $tr(\dots)$ denotes the trace of the matrix. Please note, that this is just an approximation since the KL divergence is in general not linearly separable. By sliding the window of size $L \times L$ all over the image, the KL divergence is calculated for every pixel describing the similarity between the color appearance of the local neighborhood and the object model. Figure 2 shows examples of probability maps projected to a gray scale range. Please note, how the usage of the Fisher space improves the discrimination of the object from its background.

To be able to calculate the mean and covariance matrix of the RGB color values of a local neighborhood in the most efficient way, we use the integral image concept, which e. g. has been applied for fast calculation of histograms by Porikli [16], for covariance matrix computation by Tuzel et al. [18] or for Bhattacharyya distance calculation by Donoser and Bischof [7]. As shown in [7] nine different integral images are required for efficient calculation of Gaussian distributions. Please note, that the necessary transformation into the corresponding Fisher spaces can be elegantly integrated just by multiplication of the means and covariances with the corresponding NDA matrices W_c.

2.2 Maximally Stable Boundaries

The second step of our framework consists of applying an edge detector to the color similarity map $p(\mathrm{O}|x)$ estimated by the algorithm presented in Section 2.1. Standard edge detection algorithms as the Canny detector just look for local discontinuities in image brightness, while recent work focused on integrating context, mid- and high-level information to improve edge detection results as done by Dollar et al. [6] or Zheng et al. [19]. These frameworks provide excellent results but their main drawback is the required computation time in the range of 12 seconds [6] and 90 seconds [19] which makes them not feasible for fast tracking applications.

To be able to use an edge detection result in a real-time capable tracking framework its calculation has to be fast, but nevertheless at least contextual information should be integrated to be able to outperform purely local detectors. Therefore, we introduce a novel edge detection framework which outperforms local edge detectors as e. g. the Canny detector and furthermore is calculated in a very efficient manner.

The underlying idea of the method is to detect edges by finding boundary chains of connected regions within binary threshold results, which are stable over a sequence of thresholds. This idea is quite similar to the detection of Maximally Stable Extremal Regions (MSERs) as proposed by Matas et al. [12]. We detect edges by analyzing a data structure named the component tree. The component tree is a rooted, connected tree and can be built for any image I_{in} with pixel values coming from a totally ordered set. Each node of the tree represents a single extremal region, i. e. a connected region extracted from a binary threshold image $I_{bin}^{g} = I_{in} \geq g$, where g denotes a threshold value. The edges of the tree are defined by the inclusion relationship between the extremal regions found in results for subsequently increased threshold values g. By moving in the component tree upwards the size of the extremal regions increases and the root of the tree contains a region which includes all pixels of the input image I_{in}.

Then for every region of the tree a stability criterion is analyzed which measures if parts of the region boundary remain more or less the same over several levels of the component tree. We define the stability value $\Psi(R_i^g)$ of a region R_i^g by comparing its neighboring regions $R_j^{g-\Delta}$ and $R_k^{g+\Delta}$, where $R_j^{g-\Delta}$ and $R_k^{g+\Delta}$ are the extremal regions that are obtained by moving upwards respectively downwards in the component tree from region R_i^g until a region with gray value $g-\Delta$ respectively $g+\Delta$ is found. Let $\mathscr{C}_j$ and $\mathscr{C}_k$ be the corresponding region boundaries. As a first step we calculate the distance transformation DT_j for the binary map containing the boundary pixels of $\mathscr{C}_j$. This distance transformation DT_j allows to find partial matches between the boundaries $\mathscr{C}_j$ and $\mathscr{C}_k$ by finding connected chains $\overline{\mathscr{C}}_k \subset \mathscr{C}_k$ fulfilling

$$\overline{\mathscr{C}}_k \subset \mathscr{C}_k \rightarrow DT_j\left(\overline{\mathscr{C}}_k\right) \leq \Phi, \tag{8}$$

where Φ is a maximum boundary distance parameter. Then, the stability value $\Psi(R_i^g)$ is set to the average chamfer distance D_{cham} of the matched boundary pixels $\overline{\mathscr{C}}_k$ by

$$\Psi(R_i^g) = \frac{1}{N}\sum_{n=1}^{N} DT_j\left(\overline{\mathscr{C}}_k\right), \tag{9}$$

where N is the number of matched pixel. In such a way we get a similarity measure $\Psi(R_i^g)$ and matched connected boundary chains for every comparison.

After calculation of the stability values for the entire component tree, we detect the most stable ones within the tree and return the corresponding boundary chains as detection result. Therefore, we denote the resulting binary edge map as Maximally Stable Boundary (MSB) detection result.

Please note, that MSB results are quite different from simply returning the boundaries of an MSER detection result. First, since we find connected sub-chains that are similar, the returned MSBs need not to be closed. Further, since we really analyze if local parts of the boundary are stable and not only the region sizes, the selected most stable regions can differ significantly. Nevertheless, MSBs possess the same properties as MSERs. They are detected at every scale and are invariant to affine intensity changes. Furthermore, algorithms have been proposed which allow to detect the component tree in linear time [14]. In our framework, the proposed edge detection algorithm is only applied on the color probability maps $p(\mathrm{O}|x)$, estimated with the algorithm described in Section 2.1, for tracking purposes. This application enables to further decrease the computation time by focusing on the upper threshold levels, i. e. on regions with similar color appearance.

(a) Input Image (b) Canny edge result (c) MSB result

Figure 3: Edge detection results of post-processed Canny edge detector and Maximally Stable Boundary (MSB) detector.

Figure 3 shows the difference between a post-processed canny edge result and our MSB result applied to the color probability map shown in Figure 2(c). Please note, that due to the previously mentioned constraints the MSB detection exclusively finds the boundary chain of interest.

2.3 Particle filtering and active contour fit

The final step of our tracking algorithm is to obtain the current object segmentation based on the calculated color edges. For that purpose we first estimate a transformation of the object O between the previous and the current frame. We use a probabilistic particle filtering framework [9] for hypothesizing rigid transformations of the object between subsequent frames. At each frame a set of P particles s_i^t with corresponding weights π_i^t and $i = 1 \dots P$ approximates the posterior probability function $p(x_t|z_{1:t})$, where x is the state vector and z are the observations during tracking.

We use a five-dimensional state space representing similarity transformations with translation, scaling and rotation. Resampling and particle propagation follow the standard particle filter procedure [9]. The weight update, i. e. the analysis of the current observation, is based on analyzing the Chamfer distance of the estimated transformed shape to the edge map calculated by the algorithm described in Section 2.2. Finally, the particle set $\{s_i^t, \pi_i^t\}$ approximates the posteriori probability function $p(x_t|z_{1:t})$, estimating the most likely transformation between the frames.

Because particle filtering only considers rigid transformations we finally apply an active contour model [10] to the edge map. Due to the reason that fitting an active contour model on edge maps strongly depends on the initialization, we use the estimated transformed shape returned by particle filtering as initialization. Because the initialization is on average close to the desired output only few iterations are required to get a parametric model as the final result.

3 Experiments

The main focus of the experimental evaluation lies on demonstrating the high accuracy of our method on a simple hand tracking sequence and its robustness on a challenging sports video sequence. In both experiments we perform a direct comparison to a state-of-

Figure 4: Tracking hands through a video sequence by the proposed method. Accurate segmentations of the hand are provided in each frame.

the-art tracker: the covariance tracker proposed by Porikli et al. [15] which has proven to provide impressively robust tracking results. In the original paper of Porikli et al. [15] the current tracking location is set to the location of maximum similarity. To further increase its robustness we embedded the tracker in a particle filtering framework [9], which also allows to dynamically adjust the scale of the object to be tracked. We demonstrate that our framework robustly tracks complex objects without any model of dynamics or a-priori knowledge of the object outperforming the covariance tracker in terms of tracking accuracy and robustness for fast changing object shapes.

The presented results were achieved with a non-optimized Matlab implementation, enabling a tracking speed of one frame per second for the 640×480 sequences. Therefore, an efficient implementation in `C/C++` will allow to use our framework for real-time tracking applications. In all experiments the parameters were fixed to $\delta = 10$, $L = 9$, $\Delta = 10$, $\Phi = 5$ and $P = 100$.

In a first experiment we applied the proposed method for tracking hands, as e. g. required in applications like gesture recognition or human-computer interaction (HCI). Figure 4 shows selected frames of the tracking sequence where the hand segmentation results are highlighted by a yellow boundary. As can be seen our tracker provides accurate segmentations of the object in every frame. For comparison, the covariance tracker output is also able to track the hand robustly, but lacks concerning the accuracy of the provided bounding boxes.

As second tracking example Figure 5 shows the tracking and segmentation results of a challenging ski sequence, which includes lightning changes, appearance changes and occlusions. Despite these difficulties our framework provides robust results, with accurate boundary segmentations. In comparison as can be seen in Figure 6 the covariance tracker fails to track the fast changing object in this sequence. The severe changes of the object appearance and shape are correctly handled by our tracker because of the repeated figure/ground segmentation in every frame which allows to adapt to new shapes.

(a) (b)

Figure 5: Tracking sports events – (a) shows cropped segmentation bounding boxes and (b) selected frames including segmentation results.

Figure 6: Result of particle filter embedded covariance tracker. The provided bounding boxes are not accurate and the track is lost after 100 frames.

4 Conclusion and outlook

This paper introduced a novel real-time capable tracking framework, based on the idea of repeatedly performing figure/ground segmentations on the subsequent frames by fitting an active contour on efficiently calculated color similarity edge maps. We presented the Maximally Stable Boundary detector which outperforms purely local edge detectors by integrating context information. Experimental evaluations proved that robust tracking results in addition to accurately segmented boundaries are achieved. Future work will focus on the integration of a fast partial shape matching concept to find the object in the edge maps, subsequently performing accurate object boundary detection instead of applying active contours.

References

[1] A. Angelopoulous, A. Psarrou, and G. Gupta. Robust modelling and tracking of nonrigid objects using active-gng. In *International Conf. on Computer Vision*, 2007.

[2] Y. Y. Boykov and M. P. Jolly. Interactive graph cuts for optimal boundary & region segmentation of objects in N-D images. In *International Conf. on Computer Vision*, volume 1, pages 105–112, 2001.

[3] W. Cho, J. Park, M. Lee, and S. Park. Unsupervised color image segmentation using mean shift and deterministic annealing EM. In *ICCSA*, pages 867–876, 2005.

[4] V. Comaniciu, D. Ramesh and P. Meer. Kernel-based object tracking. *Transactions on Pattern Analysis and Machine Intelligence*, 25:564–577, 2003.

[5] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society*, 39(1):1–38, 1977.

[6] P. Dollar, Z. Tu, and S. Belongie. Supervised learning of edges and object boundaries. In *Conf. on Computer Vision and Pattern Recognition*, pages 1964–1971, 2006.

[7] M. Donoser and H. Bischof. ROI-SEG: Unsupervised color segmentation by combining differently focused sub results. In *Conf. on Computer Vision and Pattern Recognition*, 2007.

[8] K. Fukunaga and J. Mantock. Nonparametric discriminant analysis. *Trans. on Pattern Anal. and Mach. Intelligence*, 5:671–678, 1983.

[9] M. Isard and A. Blake. Condensation – conditional density propagation for visual tracking. *International Journal of Computer Vision*, 29(1):5–28, 1998.

[10] M. Kass, A. Witkin, and D. Terzopoulos. Snakes - active contour models. *International Journal of Computer Vision*, 1:321–331, 1987.

[11] P. Kohli and P. Torr. Efficiently solving dynamic markov random fields using graph cuts. In *International Conf. on Computer Vision*, volume 2, pages 922–929, 2005.

[12] J. Matas, O. Chum, M. Urban, and T. Pajdla. Robust wide baseline stereo from maximally stable extremal regions. In *British Machine Vision Conf.*, 2002.

[13] F. Moreno-Noguer, A. Sanfeliu, and D. Samaras. Dependent multiple cue integration for robust tracking. *Trans. on Pattern Anal. and Mach. Intelligence*, 2008.

[14] David Nister and Henrik Stewenius. Linear time maximally stable extremal regions. In *European Conf. on Computer Vision*, 2008.

[15] F. Porikli, O. Tuzel, and P. Meer. Covariance tracking using model update based on lie algebra. In *Conf. on Computer Vision and Pattern Recognition*, volume 1, pages 728–735, 2006.

[16] F. M. Porikli. Integral histogram: A fast way to extract histograms in cartesian spaces. In *Conf. on Computer Vision and Pattern Recognition*, pages 829–836, 2005.

[17] X. Ren and J. Malik. Tracking as repeated figure/ground segmentation. In *Conf. on Computer Vision and Pattern Recognition*, 2007.

[18] O. Tuzel, F. Porikli, and P. Meer. Region covariance: A fast descriptor for detection and classification. In *European Conf. on Computer Vision*, pages 589–600, 2006.

[19] S. Zheng, Z. Tu, and A.L. Yuille. Detecting object boundaries using low-, mid-, and high-level information. In *Conf. on Computer Vision and Pattern Recognition*, 2007.