Object Tracking with Occlusion Handling: Improvements and Performance Evaluation

Pablo Ballesty[1], Matías Colotto[1], Juliana Gambini[1,2] and Ezequiel Scaruli[1]
[1] Departamento de Ingeniería en Informática, Instituto Tecnológico de Buenos Aires
Av. Madero 399 (C1106ACD) - Buenos Aires - Argentina
[2] Departamento de Inegeniería en Computación- UNTref- Pcia. de Buenos Aires- Argentina
Email: paulballesty@gmail.com, mcolotto@gmail.com, juliana.gambini@gmail.com, ezequiel.scaruli@gmail.com

***Abstract*—In this work, an object tracking method based on contour detection is analyzed, improved and assessed. The original tracking algorithm is very efficient, even in applications that require real time processing but it fails in violent illumination changes and occlusion presence. Further developments on the original algorithm are presented in order to solve these problems. It consists in using the HSV color representation to decide if a pixel belongs to the tracking region. As regards to occlusion handling, alternative ways are introduced to support obstruction with different feature objects. The performance evaluation of the resulting algorithm is carried out with two kinds of tests. The first one consists in using a specifically designed metric to test its precision. The latter is a measurement of the method speed.**

I. Introduction

Object tracking techniques is essential in many computer vision applications including image-based medical diagnosis [1], surveillance [2], [3] and robotics [4]. There exist several methods in the literature for the purpose of object tracking in image sequences. The level set approach is a powerful technique and various models have been proposed [5], [6], [7]. All these methods are based on solving partial differential equation, they are robust and they allow topology changes, but their high computational cost is a limitation. Another type of trackers are those that use two sets of markers to segment the object of interest and graph-cut [8], [9]. In [10] the authors introduce a method that can handle partially occluded objects using a set of markers. They compare the performance of the algorithm with other published algorithms that deal with the same problem as [11], [12], [13], [14]. In [15] a method which estimates the likelihood of the matching residual between the object representation model and the new candidate image based on previous matching errors, is presented and it is robust to occlusion. In [16] Mean-Shift and Kalman filter are combined for object tracking with occlusion handling, but all these methods do not work in real time. On the other hand, there exist several approaches for real time object tracking, as those based on the analytical derivation of the Jacobian [17], or learning based methods [18], [19] which have been applied successfully. There are other, fast tracking algorithms that are also suitable for real-time requirements, based on the level set approach but without solving differential partial equations, as the one proposed by Maška-Matula-Daněk-Kosuvek in [20]. In the article [21], another level set formulation is proposed, where the level set is modeled as a continuous parametric function expressed in a B-spline basis. In [22], a novel approach of video tracking is proposed using a fast level set implementation. With this approach, a real time video tracking based on level set was achieved. However, all of these methods present convergence problems in presence of violent illumination changes and occlusion. Shi and Karl's procedure is based on the level set approach but it does not solve partial differential equations. It represents each object with its contour curve and performs the tracking by adapting the curve in each frame by switching elements between two lists of neighboring pixels. One of the main challenges related with object tracking is occlusion, this problem appears in situations in which the tracked object moves behind another one, disappearing from the video for an interval of time. There are two kinds of occlusion. The first one, where the tracked object is occluded by another with different features, so the algorithm loses its position; and occlusion by an object with similar features, where the tracked object is obstructed with another of similar features, so the algorithm considers both of them as the object of interest. Occlusion handling requires detecting when it occurs. In [23], the authors use two metrics: the distance between the occluded and the occluding objects, and the ratio between the area of the occluded object in the current frame and the mean area along the all frames. This approach is effective, but has the disadvantage of requiring the occluding object to be also tracked by the algorithm. In [24], the contour sizes of the occluded object in adjacent frames are compared. If the ratio between them is less than a threshold, occlusion is detected and the contour of the occluded object is expanded until it includes the occluding object as well. This method works well however it fails when both objects, the occluded and the occluding, are being tracked.

The goal of this paper is to improve the object tracking method proposed by Shi and Karl in [22] and to improve the occlusion handling method presented in [24]. In addition, a performance evaluation of the resulting algorithm is carried out with two kinds of tests. The first one consists in using a specifically designed metric to test its precision. The latter is a measurement of the method speed.

This work is composed as follows: in section II The original algorithm based on switching pixels is described in detail. In section III-A the proposed improvements in object representation are presented. In section III-B, a solution to the convergence problem in occlusion presence, are explained. In section IV the metrics to measure the performance evaluation, are shown. In section V, we present the results of the performance evaluation of the implemented algorithm. Finally, in section VI, conclusions and future works are presented.

II. Contour-based tracking algorithm

The contour of an object is a closed curve that divides an image into two sets of pixels, where the inner set represents the object of interest. Contour-based tracking algorithms find the edges through diverse characteristics (e.g. color, texture, etc.). The user initializes the method in the first frame in a supervised way, then the object is tracked in each frame by considering the solution for the previous frame as an initial curve. In this section, the theory of the contour based tracking algorithms is briefly explained. For more details see [7], [6], [22], [25].

The contour of an object through a whole video can be represented as a parametric closed curve

$$\begin{aligned} C(s,t) &= (x(s), y(s), t) \text{ with } 0 \leq s \leq S, \\ C(0,t) &= C(s,t). \end{aligned}$$

where t is time, and $(x(s), y(s)) \in \Omega$ where Ω is the domain of the frame. Then, Ω is divided into two disjoint regions: Ω^-(internal) y Ω^+(external), separated by the curve $C(s,t)$. In the fronts evolution approach, the curve evolves in the normal direction with a velocity $F(s)$. Then, the curve evolution equation from an initial curve is given by:

$$\begin{aligned} C_t(s,t) &= \overrightarrow{N}_{C(s,t)} F(s) \\ C(s,0) &= C_0(s) \end{aligned}$$

where $\overrightarrow{N}_{C(s,t)}$ is the unitary vector normal of $C(s,t)$ and $C_0(s)$ is the initial curve. From the space-scale theory ([26], [27]) we know that when the speed is the curvature, $F(s) = -\kappa$, where $\kappa = \nabla \cdot \left(\frac{\nabla C}{|\nabla C|}\right)$, the curve evolution is equivalent to the applying of a Gaussian filter.

Another option is to represent the curve implicitly as the zero level set of a function of a greater dimension, known as level set function. Therefore, $C(s,t) = \{(x(s), y(s) \in \Omega \mid \phi(x(s), y(s), t) = 0\}$ where $\phi(x,y,t)$ is the level set function,

$$\begin{aligned} \phi(x,y,t) &< 0 \quad \text{if } (x,y) \in \Omega^- \\ \phi(x,y,t) &> 0 \quad \text{if } (x,y) \in \Omega^+ \\ \phi(x,y,t) &= 0 \quad \text{if } (x,y) \in C(s,t). \end{aligned}$$

It can be proven that the evolution of the curve $C(s,t)$ with velocity F in the normal direction is governed by the equation:

$$\begin{aligned} \phi_t + F|\nabla\phi| &= 0 \\ \phi(x,y,0) &= C(s,0) \end{aligned} \tag{1}$$

The definition of the velocity function F depends on the specific application. Due to the high computational cost of calculating a solution of Eq. 1, a different proposal has appeared in the recent years, which formulates fast algorithms with another strategy.

The method developed in [22] is based on the exchange of neighboring pixels between two lists that represent the contour of the object of interest. The set of M regions to be tracked are defined: $\{\Omega_1, \Omega_2, \ldots, \Omega_M\}$, with $\Omega_i \cap \Omega_j = \emptyset$ if $i \neq j$, those regions are initially chosen by the observer. The background, $\{\Omega_1 \cup \Omega_2 \ldots, \Omega_M\}^C$ called Ω_0 and the set of contours of the M regions of interest: $\{C_1, C_2, \ldots, C_M\}$. Each region has an associated a vector $\Theta^i = (\theta_1^i, \theta_2^i, \ldots, \theta_n^i), i \in \{1, 2, \ldots, M\}$, which contains representative information about the region of interest. Therefore, it is referred to as the *feature vector*. An RGB tuple representing the color of the region is a possible example of a feature vector. The curve evolution formula is given by: $C_t(s,t) = (F_d + F_s)\overrightarrow{N}_{C(s,t)}$, where F_d is the speed of evolution and F_s makes the curve smooth. In this case, F_d is given by $F_d(\mathbf{x}) = \log(p(\Theta^i(\mathbf{x}) \mid \Omega_i / p(\Theta^j(\mathbf{x}) \mid \Omega_j))$, where Θ^i and Θ^j are the feature vectors of regions Ω_i and Ω_j, respectively, $p(\Theta^i(\mathbf{x}) \mid \Omega_i)$ is the probability of pixel $\mathbf{x}$ belonging to the i-th region. F_s is given by $F_s(\mathbf{x}) = -2\lambda\kappa(\mathbf{x})$, where κ is the curvature. The algorithm used is based on the level set theory. Function ϕ is defined in the following way:

$$\phi(\mathbf{x}) = \begin{cases} 3 & \text{if } \mathbf{x} \in \Omega_0 \text{ and } \mathbf{x} \notin L_{out} \\ 1 & \text{if } \mathbf{x} \in L_{out} \\ -1 & \text{if } \mathbf{x} \in L_{in} \\ -3 & \text{if } \mathbf{x} \in \Omega_1 \text{ and } \mathbf{x} \notin L_{in} \end{cases}$$

where L_{in} and L_{out} are pixel sets used to represent the edge of the region. They correspond to its internal and external edges respectively. They are defined as: $L_{in} = \{\mathbf{x} \mid \phi(\mathbf{x}) < 0 \wedge \exists \mathbf{y} \in N_4(\mathbf{x}) \mid \phi(\mathbf{y}) > 0\}$ and $L_{out} = \{\mathbf{x} \mid \phi(\mathbf{x}) > 0 \wedge \exists \mathbf{y} \in N_4(\mathbf{x}) \mid \phi(\mathbf{y}) < 0\}$, where $N_4(\mathbf{x})$ represents the four neighbors of $\mathbf{x}$ $N_4(\mathbf{x}) = \{\mathbf{y} \mid |\mathbf{x} - \mathbf{y}| = 1\}$. An example of these sets can be seen in Fig. 1.

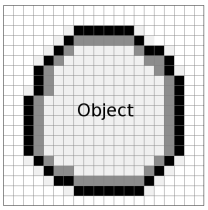


Fig. 1. Inner and outer edges of the boundary of a region.

The curve expands or contracts according to the sign of F_d. At the beginning, ϕ and the two lists L_{in} and L_{out} are initialized according to a supervised selected region, then two cycles are performed in each frame. The first cycle adapts the sets L_{in} and L_{out} to the edge of the object. The second cycle has the objective of obtaining the softening of the curve.

III. Improvements in Object Representation and Occlusion Handling

A. The HSV color representation

The feature vector $\Theta^i,\ i = 1, \ldots, M$ must be defined. We represent the object of interest by its color, and we analyze two different models. An alternative is the use of the RGB components with the euclidean norm as similarity measure, which was developed in the original article (see [22]). However, this representation model is not robust when there are sudden illumination changes in the video. Therefore, our proposal is to use the HSV color model which has the component V that corresponds to the color brightness. If this component is not considered in the analysis, two colors which differ only in brightness are considered the same. This idea solves the

problems arising from sudden illumination changes. Using this model, a pixel p is denoted by $p = (h_p, s_p) \in [0, 1) \times [0, 1]$ where h_p and s_p are the hue and saturation components of HSV color system, respectively. For simplicity, the analysis is done for a single object tracking. In the HSV representation, the feature vector of Ω is $\Theta_{HSV} = (\mu_h, \mu_s)$ where μ_h, μ_s are the sample means of the hue and saturation components, respectively, calculated using the initial region. In this case, we need to use the following function to compute the difference between two hue values:

$$hdiff(h1, h2) = \begin{cases} |h1 - h2| & if \quad |h1 - h2| \leq 0.5 \\ 1 - |h1 - h2| & if \quad |h1 - h2| > 0.5 \end{cases}$$

Therefore, $p = (h_p, s_p) \in \Omega$ if $||(hdiff(\mu_h, h_p), \mu_s - s_p)||_{1,2} < T_{HSV}$ where T_{HSV} is a tolerance given by the user.

The result of applying the algorithm to a real video using both the RGB and HSV representations can be seen in Fig. 2 and Fig. 3, respectively. In Fig. 2, it can be observed that the object of interest produces a shadow in the background and the method fails in fitting the edge of the leaf. Fig. 3 shows that HSV color model solves this problem.

Fig. 2. Results of applying the algorithm using RGB Color Model.

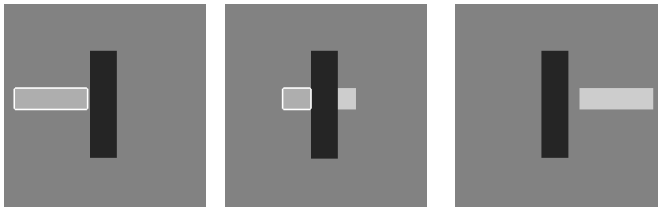
Fig. 3. Results of applying the algorithm using HSV Color Model.

B. Occlusion Handling

The occlusion with an object that has different features is detected by analyzing the contour of the tracked object. If its size decreases sharply, it means that it is being occluded. This situation is shown in Fig. 4.

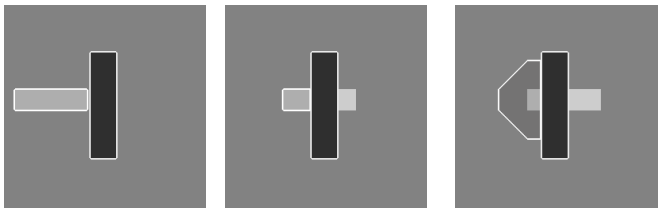
When the tracked object is occluded, one approach to reach it again is to expand its contour curve until it gets over the obstacle. This technique is presented in [24] with relatively good results. The main disadvantage of this way of occlusion handling is that it does not work well when both objects, the occluded and the occluding, are being tracked by the algorithm. This situation is shown in Fig. 5.

To deal with this issue, we propose another technique that consists in moving the contour of the occluded object to a new position, which depends on the direction of the contour movement along the previous frames. It is necessary



(a) Frame 1 (b) Frame 10 (c) Frame 24

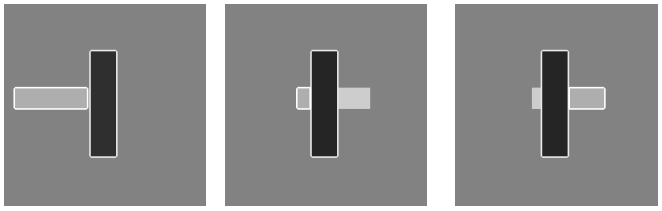
Fig. 4. Result of applying the original algorithm in a synthetic video where occlusion with an object of different features occurs and the original method fails.



(a) Frame 1 (b) Frame 11 (c) Frame 14

Fig. 5. Problem that occurs when both objects are tracked by the algorithm and the curve expansion approach is applied (see [24]). The method fails.

to calculate the average displacement $\mathbf{\Delta}_i$ of the tracked object in frame i. If $\mathbf{c}_k$ is its center of mass in frame k, then $\mathbf{\Delta}_i$ can be defined as $\mathbf{\Delta}_i = \frac{1}{i} \sum_{k=1}^{i} (\mathbf{c}_k - \mathbf{c}_{k-1})$. Once occlusion is detected in frame m, the curve is moved in the direction of $\mathbf{\Delta}_m$, assuming that the tracked object is likely to move in that direction after it is occluded. So the new position p_{m+1} of each pixel of the contour is $p_{m+1} = p_m + \lambda \Delta_m$, where p_m is the previous position and $\lambda \in \Re$ is a factor that indicates how much the curve is moved in the direction of Δ_m. However, it is necessary that the contour in the new position contains the object previously occluded, or at least a part of it. To verify that the contour in the new position contains a part of the object of interest, a sample of pixels near $\mathbf{c}_m + \Delta_m$ is taken. If most pixels belong to the tracked object, the contour is moved. In another case the movement is not performed, because the occluded object has moved in another direction. In Fig. 6, the moment in which the curve movement takes place is illustrated.



(a) Frame 1 (b) Frame 14 (c) Frame 15

Fig. 6. Result of applying the new technique to the video in Fig. 5

IV. Metrics for Performance Evaluation

In order to evaluate the performance of our methods, we use the metrics proposed in [28]. They are based on the comparison with the *ground truth* which shows the expected results of the object tracking obtained manually.

The first metric is (*Label Tracking Detection Rate*) or $LTDR$. It is aimed to determine if the algorithm is capable of correctly matching the position of the objects of interest. It is defined as follows:

$$LTDR = \frac{1}{L}\sum_{i=0}^{L-1}\frac{TPM_i}{OAF_i}$$

where L is the number of tracked objects; TPM_i is the number of frames in which the position of the i-th object obtained by the algorithm matches the position of the ground truth for the i-th tracked object. OAF_i is the number of frames in which the object is present in the video according to the ground truth. The position of an object is represented by the centroid of the region. The value of $LTDR$ belongs to the interval $[0, 1]$. A value of 0 means that the algorithm is not able to track the position of the objects of interest in any frame, while a value of 1 indicates that the position of all the objects in every frame is successfully matched with the ground truth.

The second metric is $ASDR$ (*Average Size Detection Rate*) which provides information about the ability of an algorithm to correctly obtain the size of the objects of interest. It is defined as follows:

$$ASDR = \frac{1}{L}\sum_{i=0}^{L-1}\frac{SDR_i}{TPM_i}$$

where SDR_i is the number of frames in which the size of the i-th region is correctly calculated according to the ground truth and TPM_i is the number of frames in which the i-th region is in the correct position, according to the ground truth. The size is calculated using the area of the smallest rectangle containing the region of interest. The value of $ASDR$ belongs to the interval $[0, 1]$. A value of 0 means that the algorithm is not able to obtain the object size correctly in any frame, instead a value of 1 means that the obtained object size is well calculated during the whole video.

In order to determine whether the algorithm runs in real time or not, we propose calculating the number of frames that it is able to process in a second. We use the following magnitude:

$$MPT = \frac{1}{N}\sum_{i=1}^{N} T_i$$

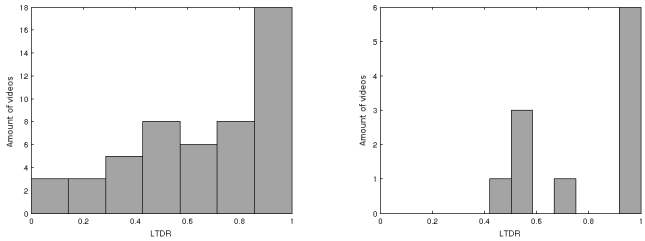
where T_i is the time spent processing the i-th frame and N is the number of frames. We also use

$$\overline{FPS} = \frac{1}{MPT}$$

MPT is the *Mean Processing Time*, the mean time that the algorithm takes to perform the tracking in one frame. $\overline{FPS}$ represents the mean of the frames per second that the algorithm is able to track in the video. If $\overline{FPS}$ is greater than 30, we consider that the algorithm runs in real time.

V. Experimental Results

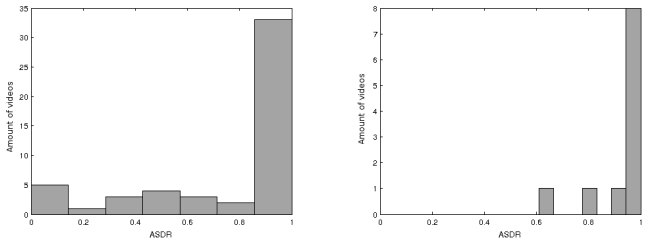
In order to test the proposed algorithm it was applied to several real and synthetic videos. In addition, the method was tested using the same videos with Gaussian noise added in an artificial way. For each video, several combinations of features and criteria are tested. Videos *Akiyo*, *Carphone*, *Bus*, *Ice* and *Claire* are available for download at *Xiph.org Video Test Media [derf's collection]* (http://media.xiph.org/video/derf/). The rest can be found at http://goo.gl/iFH1d. In 7 out of 11 videos, 64% of the total number, the value of $LDTR$ is greater than 0.7, so an accurate tracking of the object position is achieved under certain parameters of the algorithm. In Fig. 7(a) a comparative chart of $LTDR$ values obtained from these experiments, is illustrated. Axis y shows the number of videos and axis x shows the $LTDR$ output value. For example, there are eigtheen videos where the $LTDR$ output value is between 0.9 and 1. Fig. 7(b) shows a chart of the best values of $LTDR$ outputs for each video. This chart shows that the new algorithm achieves a tracking, at least partial, in all the tested videos.



(a) All Experiments (b) Best Experiments

Fig. 7. Comparative chart of $LTDR$ values in each experiment.

With regard to size metrics, the results indicate that the 70% of the experiments show an $ASDR$ value greater than 0.7. This suggests that the method is capable of successfully detecting the size of the objects in most frames. In Fig. 8(a) a chart of $ASDR$ values obtained in each experiment is illustrated. Fig. 8(b) shows a chart of the best values of $ASDR$ outputs for each video. In both cases, there is a great amount of values near to 1 which suggests that, using all its alternatives, the algorithm is able to match the original size in a consistent way.

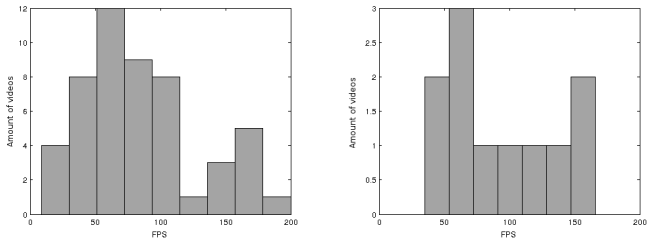


(a) All experiments (b) The best $ASDR$ values for each video.

Fig. 8. Comparative chart of $ASDR$ values in each experiment.

In addition, it can be observed that the algorithm works in real time in more than a 90% of tested videos. Moreover, it works in real time in videos in which more than one object is tracked, and have occlusion handling enabled. Fig. 9(a) shows a comparative chart of the $\overline{FPS}$ values obtained from the experiment. The y-axis represents the number of videos and the x-axis the $\overline{FPS}$ values. $\overline{FPS}$ values greater than 200 are excluded from this graphic, since they correspond to experiments with non acceptable values of $LTDR$. Fig. 9(b)

shows the best $\overline{FPS}$ values for each video. In both charts, it can be seen that most values suggest a real time tracking.



(a) All experiments

(b) Best $\overline{FPS}$ values obtained for each video.

Fig. 9. Comparative chart of $\overline{FPS}$ value in each experiment.

Even for videos that have Gaussian noise applied, the method is able to successfully track the object with occlusion handling, however the tracking is slower.

VI. Conclusions and future work

This paper deals with a different approach to overcome the difficulties of the object tracking algorithm. The first proposal in our approach consists in using the HSV color representation to decide if a pixel belongs to the tracking region. The second is a proposal to support obstruction with different feature objects. In order to evaluate the performance of the algorithm, three kinds of tests are performed with the resulting algorithm. We achieve improvements in the results. Several experiments with test videos validate our approach. The results of the performance evaluation of the method show that in 70% of the videos the algorithm fits the object of interest in each frame and satisfies the real time requirements even in occlusion presence.

As some aspects of future work that arise from this paper, we have to take into account: an automatic selection of initial regions, the usage of a less amount of parameters and an increment of the robustness against the noise present at initial frames.

References

[1] T. Behrens, K. Rohr, and H. Stiehl, "Robust segmentation of tubular structures in 3-d medical images by parametric object detection and tracking," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 33, pp. 554–561, 2003.

[2] W. Hu, T. Tan, L. Wang, and S. Maybank, "A survey on visual surveillance of object motion and behaviors," *IEEE Transactions on Systems, Man and Cybernetics, Part C (Applications and Reviews)*, vol. 34, pp. 334–352, 2004.

[3] B. Coifman, D. Beymer, P. McLauchlan, and J. Malik, "A real-time computer vision system for vehicle tracking and traffic surveillance," *Transportation Research Part C: Emerging Technologies*, vol. 6, pp. 271–288, 1998.

[4] V. Baier and F. Sahin, "Detection and tracking of high motion objects in arm robotics," in *Fifth International Conference on Soft Computing, Computing with Words and Perceptions in System Analysis, Decision and Control, ICSCCW*, 2009.

[5] S. Osher and J. Sethian, "Fronts propagating with curvature-dependent speed: algorithms based on hamilton-jacobi formulations." *Journal of Computational Phisics*, vol. 79, pp. 12–49, 1988.

[6] J. Sethian, "Fast marching methods and level set methods for propagating interfaces von karman insitute lecture series," 1998.

[7] S. Zhu and A. Yuille, "Region competition: Unifying snakes, region growing and bayes/MDL for multiband image segmentation," *IEEE Transaction on Pattern Anal. and Machine Intelligence*, vol. 18, no. 9, pp. 884–900, 1996.

[8] A. Protiere and G. Sapiro, "Interactive image segmentation via adaptative weighted distances," *IEEE Transactions on Image Processing*, vol. 16, pp. 1046–1057, 2007.

[9] S. Vicente, V. Kolmogorov, and C. Rother, "Graph cut based image segmentation with connectivity priors," in *Computing Vision and Pattern Recognition*, 2008.

[10] R. Minetto, T. V. Spina, A. X. Falao, N. J. Leite, J. Papa, and J. Stolfi, "Iftrace: Video segmentation of deformable objects using the image foresting transform," *Computer Vision and Image Understanding*, vol. 116, pp. 274–291, 2012.

[11] D. Zhong and S. Chang, "Long term moving object segmentation and tracking using spatio-temporal consistency," in *International Conference on Image Processing (ICIP)*, 2001.

[12] D. William and M. Shah, "A fast algorithm for active contour and curvature estimation," *Computing Visualization Graphics Image Processing: Image Understanding*, vol. 55, pp. 14–26, 1992.

[13] J. Allen, R. Xu, and J. Jin, "Object tracking using camshift anlgorithm and multiple quantized features spaces," in *Australian Computer Society*, 2004.

[14] R. Hess and A. Fern, "Discriminatively trained particle filters for complex multi-object tracking," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009.

[15] H. Firouzi and H. Najjaran, "Robust pca-based visual tracking by adaptively maximizing the matching residual likelihood," in *International Conference on Computer and Robot Vision (CRV)*, 2013, pp. 52–58.

[16] G. Phadke and R. Velmurugan, "Improved mean shift for multi-target tracking," in *IEEE International Workshop on Performance Evaluation of Tracking and Surveillance (PETS)*, 2013, pp. 37–44.

[17] S. Benhimane and E. Malis, "Homography based 2D visual tracking and servoing," *Int'l J. Robotics Research*, vol. 26, pp. 661–667, 2007.

[18] S. Holzer, S. Ilic, and N. Navab, "Multilayer adaptive linear predictors for real-time tracking," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, pp. 105–117, 2013.

[19] L. Xiaohui and L. Huchuan, "Object tracking based on local learning," in *IEEE International Conference on Image Processing (ICIP)*, 2012.

[20] M. Maska, P. Matula, O. Daněk, and M. Kozubek, "A Fast Level Set-Like Algorithm for Region-Based Active Contours," *Proceedings of the 6th International Symposium on Visual Computing*, vol. LNCS 6455, pp. 387–396, 2010.

[21] O. Bernard and D. Friboulet, "Fast medical image segmentation through an approximation of narrow-band B-spline level-set and multiresolution," in *IEEE International Symposium on Biomedical Imaging: From Nano to Macro, 2009. ISBI '09.*, 2009, pp. 45–48.

[22] Y. Shi and W. Karl, "Real-time tracking using level sets," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2, 2005, pp. 34–41.

[23] H. Yilmaz and M. Shah, "Contour-based object tracking with occlusion handling in video acquired using mobile cameras," *IEEE Trans. Patt. Analy. Mach. Intell.*, vol. 26, pp. 1531–1536, 2004.

[24] J. Gambini, D. Rozichner, M. E. Buemi, M. Mejail, and J. Jacobo-Berlles, "Occlusion handling for object tracking using a fast level set method," in *SIBGRAPI*, 2008, pp. 61–68.

[25] Y. Shi and W. Karl, "A real-time algorithm for the approximation of level-set-based curve evolution," *Image Processing*, vol. 17, no. 5, pp. 645–656, May 2008.

[26] P. Perona and J. Malik, "Scale-space and edge detection using anisotropic diffusion," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 12, no. 7, pp. 629–639, 1990.

[27] J. Weickert, *Anisotropic Diffusion in Image Processing*, 1st ed. Teubner-Verlag, 1998.

[28] J. Popoola and A. Amer, "Performance evaluation for tracking algorithms using object labels," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2008.