Occlusion Handling for Object Tracking using a Fast Level Set Method

Juliana Gambini, Damian Rozichner, María E. Buemi, Marta Mejail, Julio Jacobo Berllés
Departamento de Computación, Facultad de Ciencias Exactas y Naturales
UBA, Buenos Aires, Argentina
{jgambini,drozitch, mebuemi, marta, jacobo}@dc.uba.ar

Abstract

Tracking objects in image sequences is a difficult problem because of the frequent occlusions encountered among the objects and people. In this paper, a method for tracking objects through occlusions based on a level set technique, is presented. Two cases are considered: 1-the object is obstructed by another one of a different color. 2-the object is obstructed by another one of the same color. In the first case, the classical algorithms based on active contours fail to track the object's contour because the curve disappears from the scene. In this proposal, occlusion is detected and the solution is based on keeping the curve in the same place until the object of interest appears in the scene again. In the second case, the classic algorithms based on active contours do not distinguish between two image objects of the same color, so due to the possible changes in the topology, the curve fits the edge of both objects and it is divided. We solve this problem by means of localizing the centroid of the curve.

The experimental results show that these methods significantly improve the tracking with occlusions.

1. Introduction

Video tracking is an important task in image processing and computer science. It is widely used in several applications such as animal behavior, robotic vision, medical diagnosis and surveillance. The development of automatic algorithms for object tracking in images sequences is a challenge. The first algorithm for boundary detection based on curve evolution using variational calculus and energy minimization, was developed by Kass *et al.* [7]. This method has serious limitations in its convergence depending on the shape of the object of interest, it is very sensitive to the initial condition location and it do not allow topology changes. Then, various methods have been developed to improve this first algorithm as distance snakes [4], balloon snakes [3], deformable contours [11] and gradient vector flow snakes [20], which are not robust in noisy images. In [6], these methods are compared.

For the purpose of object tracking in image sequences, the level set approach is a more powerful technique and various models have been proposed [12, 16, 23, 1]. All these methods are based on differential equation solving, they are more robust and they allow topology changes, but the high computational cost of them is a limitation. In [18], a novel approach of video tracking, is proposed. It is based on the evolution of the curve by means of simple operations as switching elements between two lists of pixels and there is no need to solve partial differential equation. This is 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 occlusion.

Handling partial and total occlusion in object tracking in image sequences is a very difficult task. There are many works related to this subject. In [2, 5] the occlusion problem is solved using the fusion of multiple cameras inputs. In [22] a system that overcome the occlusion problem by means of adding information from radar data, is developed. In the article [14] the authors introduce a method for object tracking under occlusion using a model based on localizing objects that appear during the occlusion and then find the depth of them. Khan and Shah [8] present a system for people tracking in presence of occlusion, in fixed camera situation. They first segment the person into classes of similar color. Then, they track these classes from frame to frame using a maximum *a posteriori* probability approach. In [21] a contour-based tracking method, is proposed. The tracking is achieved by evolving the contour by minimizing energy in the gradient descent direction and occlusion is detected using the distance between the objects in the scene.

In this work, a technique for video tracking under occlusions using a fast level set implementation, is proposed. It is designed as an improvement of the method presented in [18].

Two cases are considered:

1. The object is obstructed by another one of a different

color.

2. The object is obstructed by another one of the same color.

In the first case, the original algorithm fails to track the object's contour because the curve disappears from the scene. Then, occlusion by another object with a different color is detected by the algorithm when the length of the curve decreases sharply. In the second case, the object is hidden itself or it is very near of another one of the same color and the algorithm does not distinguish between them. So, due to the possible changes in the topology, the curve fits the edge of both objects and it is divided. We solve this problem by means of localizing the centroid of the curve.

The results are excellent with a low computational cost.

This paper is organized as follows: in section 2, a brief summary of the tracking model and the fast implementation, is explained. In section 3, the problems of occlusion are analyzed and solutions are proposed. The obtained results of applying the algorithm to several videos, are presented in section 4. Finally, in section 5 we give conclusions and future work.

2. Region-based tracking model

In this section, the theory of the region-based tracking algorithm is explained. For more details see [23, 16, 17, 18].

The main objective of this work is video tracking. Given an image sequence, each scene is composed by a set of object regions $\{\Omega_1, \ldots, \Omega_M\}$, $\Omega_i \cap \Omega_j = \emptyset$ if $i \neq j$ and the background $\Omega_0 = \{\Omega_1 \bigcup \ldots \bigcup \Omega_M\}^c$. The boundaries of these object regions are denoted by $\{C_1, \ldots, C_M\}$. We consider each region Ω_i, $i = 1, \ldots, M$ modeled by a distribution $p(\Theta^i(\mathbf{x}) \mid \Omega_i)$, where $\mathbf{x}$ is a pixel and $\Theta^i = \{\theta_1^i, \ldots, \theta_n^i\}$, is the feature vector defined at each pixel. In this way, each object region is characterized by a set of parameters Θ^i. The feature vector is for example the color of a sample of pixels in the region, the mean value of a set of pixels, the estimated parameters of a statistical distribution, etc.

In the region competition approach, the result curve in each frame is the minimum of the energy function ([23, 16, 17, 18]) given by:

$$E(C_0, \ldots, C_M, \Theta_0, \ldots \Theta_M) = \quad (1)$$

$$-\sum_{i=0}^{M} \int_{\Omega_i} \log(p(\Theta^i(\mathbf{x}) \mid \Omega_i) d\mathbf{x} + \lambda \int_{C_i} ds \quad (2)$$

The first term corresponds to the likelihood of the scene and the second term represents the length of each contour C_i, the smoothest contours are favored and λ is a non negative regularization parameter.

On the other hand, the object boundary can be represented by a parametric curve, given by:

$$C(s) = (x(s), y(s))$$

where $0 \leq s \leq S$, $C(0) = C(S)$. In the fronts evolution approach ([9, 10, 15, 16]), the curve evolves under an arbitrary force $F(s)$. Then, a temporal variable t is added and the curve evolution equation, from an initial curve is given by:

$$\begin{aligned} C_t(s,t) &= \overrightarrow{N}_{C(s,t)} F(s) \\ C(s,0) &= C_0(s) \end{aligned} \quad (3)$$

where $\overrightarrow{N}_{C(s,t)}$ is the unitary vector normal of $C(s,t)$ and $C_0(s)$ is the initial curve. Eq. (3) establishes that a curve $C(s,t)$ evolves in the normal direction with a velocity $F(s)$. From the space-scale theory [13, 19] 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 Gaussian filtering the function.

By computing the first variation of the Eq. 2 we obtain the curve evolution formula, 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 represents the region competition and is given by

$$F_d(\mathbf{x}) = \log(p(\Theta^i(\mathbf{x}) \mid \Omega_i / p(\Theta^j(\mathbf{x}) \mid \Omega_j), \quad (4)$$

where Θ^i and Θ^j are the parameters of regions Ω_i and Ω_j, respectively. F_s is given by

$$F_s(\mathbf{x}) = -2\lambda\kappa(\mathbf{x}), \quad (5)$$

where κ is the curvature.

Another formulation of the problem of curve evolution is to represent the curve as the zero level set of a surface ϕ. In this case the evolution equation is given by:

$$\phi_t(\mathbf{x}, t) = |\nabla\phi(\mathbf{x}, t)|(F_d + F_s) \quad (6)$$

In the fast level set implementation [18], the curve is represented by the zero level set of a function ϕ and the boundary contours are represented by a set of pixels which correspond to the inner and outer boundaries, called L_{in} and L_{out} respectively. Then, the curve evolution is carried out by switching of pixels between these two sets of pixels. The contour expands or shrinks in the image until it fits to the object boundary.

For the sake of clarity, we describe here the algorithm for the case that the scene is composed of only one region and the background. Then, each image I_k in the sequence is

composed by a region of interest Ω_1 and the background Ω_0, where $\Omega_0 \cup \Omega_1 = I_k$ and $\Omega_0 \cap \Omega_1 = \emptyset$. Each one of these regions is characterized by the feature vector $\Theta^m,\ m = 0, 1$.

In this work, the curve is represented by the zero level set of a function ϕ given by,

$$\phi(\mathbf{x}) = \begin{cases} 3 & if\ \mathbf{x} \in \Omega_0 \text{ and } \mathbf{x} \notin L_{out} \\ 1 & if\ \mathbf{x} \in L_{out} \\ -1 & if\ \mathbf{x} \in L_{in} \\ -3 & if\ \mathbf{x} \in \Omega_1 \text{ and } \mathbf{x} \notin L_{in} \end{cases} \quad (7)$$

where L_{in} and L_{out} are the inner and outer boundaries of C_1, respectively. They are defined as follows:

$$L_{in} = \{\mathbf{x} | \phi(\mathbf{x}) < 0 \text{ and } \exists\, \mathbf{y} \in N_4(\mathbf{x}) \text{ such that } \phi(\mathbf{y}) > 0\}$$

$$L_{out} = \{\mathbf{x} | \phi(\mathbf{x}) > 0 \text{ and } \exists\, \mathbf{y} \in N_4(\mathbf{x}) \text{ such that } \phi(\mathbf{y}) < 0\}$$

where $N_4(\mathbf{x}) = \{\, \mathbf{y} / |\mathbf{x} - \mathbf{y}| = 1\}$.

Then, $\mathbf{x}$ is an interior pixel of Ω_1 if $\mathbf{x} \in \Omega_1$ and $\mathbf{x} \notin L_{in}$ and $\mathbf{x}$ is an exterior pixel of Ω_1 if $\mathbf{x} \in \Omega_0$ and $\mathbf{x} \notin L_{out}$.

Figure 1 shows an example of the inner and outer boundaries of an object, L_{in} and L_{out}.

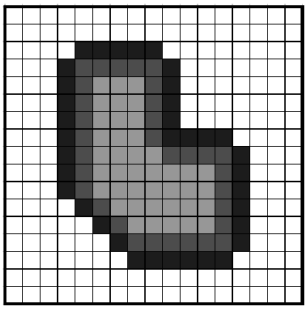

Figure 1. Inner and outer boundaries of an object.

This algorithm begins with the specification of an initial curve determined in a supervised way. The method is based on switching the neighboring pixels between the two lists L_{in} and L_{out} in two cycles. In the first cycle, the curve evolves by the speed $F_d(\mathbf{x})$, which depends on the image data, then the curve fits the object boundary. The second cycle is similar to the first, but the speed $F_s(\mathbf{x})$ for smoothness regularization, is used.

First Cycle:

The following steps are executed N_a times, where $0 < N_a < \max(columns,\ rows)$. This parameter determines the limit of expansion or contraction that may have the curve, in a neighbor of the initial contour.

1. For each $\mathbf{x} \in L_{out}$, if $F_d(\mathbf{x}) > 0$ then, delete $\mathbf{x}$ from L_{out} and add it to L_{in}. Then, $\forall \mathbf{y} \in N_4(\mathbf{x})$, with $\phi(\mathbf{y}) = 3$, add $\mathbf{y}$ to L_{out} and set $\phi(\mathbf{y}) = 1$.
2. After step 1 some of the pixels $\mathbf{x}$ in L_{in} become interior pixels so they are deleted from L_{in} and set $\phi(\mathbf{x}) = -3$.
3. For each $\mathbf{x} \in L_{in}$, if $F_d(\mathbf{x}) < 0$ then, delete $\mathbf{x}$ from L_{in} and add it to L_{out}. Then, $\forall \mathbf{y} \in N_4(\mathbf{x})$, with $\phi(\mathbf{y}) = -3$, add $\mathbf{y}$ to L_{in} and set $\phi(\mathbf{y}) = -1$
4. After step 3 some pixels $\mathbf{x}$ become exterior pixels so they are deleted from L_{out} and set $\phi(\mathbf{x}) = 3$.

As an example, Algorithm 1 shows the steps of the procedure for switch in the contour in the pixel $\mathbf{x}$. Note that $F_d(\mathbf{x}) > 0$ implies that $\mathbf{x}$ is an interior pixel of Ω_1 and $F_d(\mathbf{x}) < 0$ implies that $\mathbf{x}$ is an exterior pixel of Ω_1

Algorithm 1 Switch in the contour in the pixel $\mathbf{x}$.

```
for i = 1, ..., N_a do
    if x ∈ L_out and F_d(x) > 0 then
        Delete x from L_out
        Add x to L_in.
        Set φ(x) = −1
    end if
    if y ∈ N_4(x) and φ(x) = 3 then
        Add y to L_out
        Set φ(y) = 1
    end if
end for
```

In the second cycle, the curve is smoothed by convolution with a Gaussian filter, so the evolution speed is $F_s(\mathbf{x}) = G \otimes \phi(\mathbf{x})$, which imitates the behavior of the curve evolution by the speed given in Eq. 5.

In each frame, the boundary contour of the object is found using an initial curve. In the first frame, the initial curve is given by the user in a supervised way. In the following frames the result obtained by the method in the previous frame, is used as initial curve.

In this work, the feature vector is given by the color of the pixels. It can be a vector of three components in the RGB color system (r, g, b) or a gray value $g \in [0, \ldots, 255]$. The distribution is given by

$$p(\Theta) = 1 - \frac{\|\Theta - \mu\|_2}{\left\|\tilde{\Theta}\right\|_2},$$

where μ is the mean value of the color, and $\tilde{\Theta}$ is such that

$$\tilde{\Theta} = argmax_{\Theta} \|\Theta\|_2 .$$

The algorithm stops when the stopping condition occurs (Eq. 8) or when the maximum number of iterations is

reached.

$$\begin{aligned} F_d(\mathbf{x}) &\leq 0 \quad \forall \mathbf{x} \in L_{out} \\ F_d(\mathbf{x}) &\geq 0 \quad \forall \mathbf{x} \in L_{in} \end{aligned} \tag{8}$$

This algorithm is very efficient, even in applications that require real-time execution, but fails when the object of interest is hidden behind of another object. In the following section this problem is analyzed and solutions are proposed.

3. Occlusion Handling

In this section, the behavior of the algorithm in presence of occlusion is described and the solution for this problem is presented. In video sequences, many objects appear in the scene then, different objects cross paths and cause partial or total occlusions. The object of interest may be occluded by another object of a different color or by another one of the same color. In the first case, obstruction is presented, in the second case, obstruction and confusion occur. In both cases the algorithm fails. In the following, each one of these problems is analyzed and they are solved in a different way.

3.1. Occlusion by another object with a different color

When the object of interest is obstructed by another one of a different color, the algorithm fails in its goal of tracking the object's contour because the curve disappears from the scene. Then, occlusion presence is detected by the algorithm when the length of the curve decreases sharply.

Figure 2 shows the result of applying the original algorithm to an image sequence. The object of interest is the horizontal rectangle and it is obstructed by the vertical rectangle. It is noted that the curve is becoming smaller until it disappears and the algorithm fails.

In order to solve this problem, we introduce a change in the algorithm. The new method detects the occlusion presence and keeps the curve in the same place until the object of interest appears in the scene. Then, it finds the boundary of the object again. In order to detect occlusion presence, we measure the size of the curve by counting the number of pixels from the contour. If the number of pixels of the curve is small enough, the modified algorithm considers that occlusion is occurring. Let

$$k_C^i = \sharp \{\mathbf{x} : pixel / \mathbf{x} \in L_{in}\}$$

be the number of pixels of the curve C in the image, with $i, 0 \leq i \leq N$, where N is the number of images in the sequence. In addition, let

$$\overline{k^i}_C = \frac{1}{i} \sum_{j=1}^{i} k_C^j$$

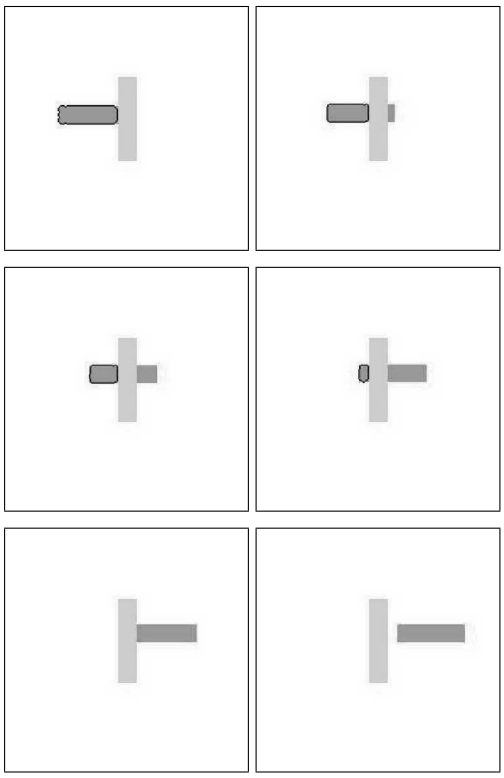

Figure 2. Example to show a case where the original algorithm fails in the presence of occlusion. The horizontal rectangle is the object of interest and it is obstructed by the vertical rectangle.

be the average of the number of pixels of the curve C in the previous i images. If

$$k_C^{i+1} < \mu \overline{k_C^i}, \;\; 0 < \mu < 1 \tag{9}$$

then, the number of pixels of the curve has been reduced dramatically and the algorithm considers that occlusion is occurring. The parameter μ is given by the user and depends on the size of the object of interest.

Figure 3 shows an image sequence where the new method was applied to the image sequence of the figure 2. The presence of occlusion is detected by measuring the number of pixels from the curve, then the contour is expanded in order to find the object border again. It can be seen that the result is excellent.

3.2. Occlusion by an object of the same color

During its trajectory, the object may hide itself behind of another object of the same color. The algorithm does not

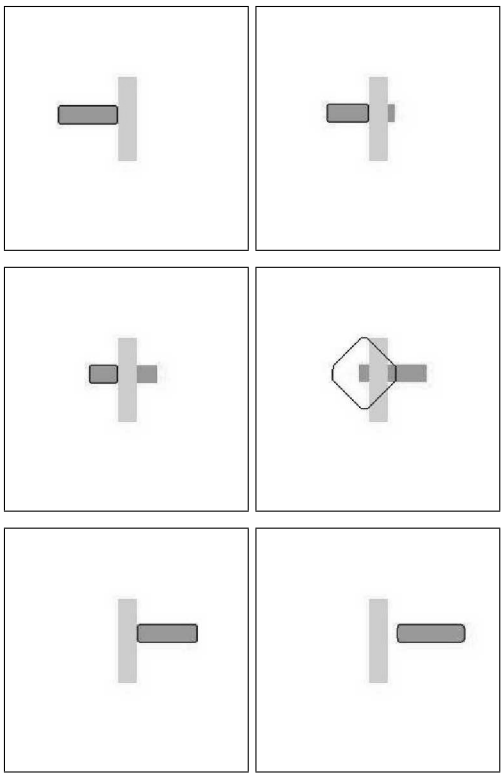

Figure 3. The modified algorithm is applied to the image sequence of Figure 2. It can be seen the excellent result.

distinguish between two image objects of the same color, so due to the possible changes in the topology, the curve fits the edge of both objects and it is divided. Figure 4 shows an example where this problem occurs. The rectangle is the object of interest. When it is hidden by a circle of the same color, the curve is divided into two parts. This example shows that in this case, the original method confuses the object of interest.

In order to solve this problem, a change in the tracking algorithm, is proposed. The main idea is that the method tracks a unique closed curve using its centroid. The first step is to find the different closed curves from the list of pixels, the second step is to discard the curves that do not correspond to the border of the object of interest.

Let C_i be the fitting contour which was found in the i^{th} image, $0 \leq i \leq N$, And let p^i be the centroid of C_i. We know that the curve C_i is composed of k small closed curves $\left\{B_1^i, \ldots, B_k^i\right\}$. Each B_j^i is the fitting contour of an object Ω_j in the image, let q_j^i be the centroid of the curve B_j^i. We consider the curve $B_{\widehat{j}}^i$ whose centroid is closer to the centroid of the main object in the previous image. Then, the fitting contour in the i^{th} frame is given by

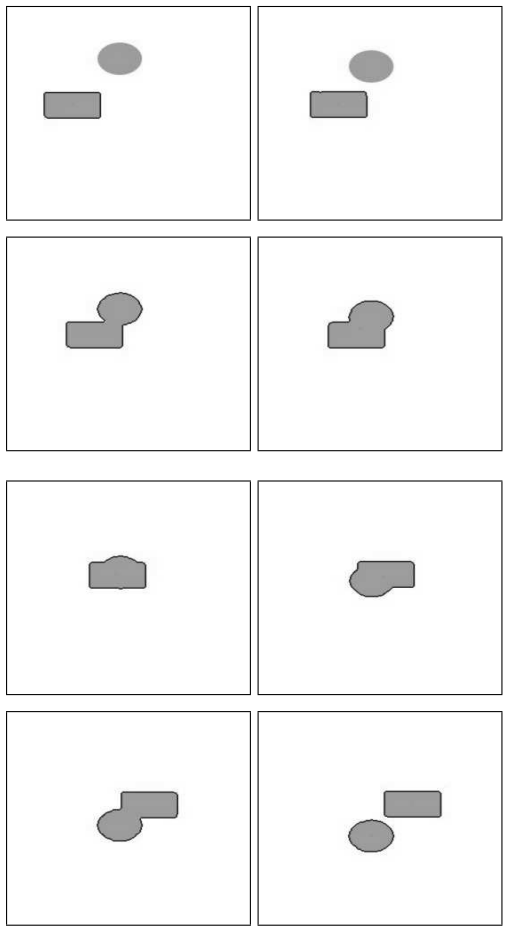

Figure 4. Occlusion by an object which has the same color as the object of interest. It can be seen that the curve is divided and the curve fits to the boundary contour of both objects.

$$C_i = B_{\widehat{j}}^i \tag{10}$$

where $\widehat{j}$ is such that it minimizes the distance between p^{i-1} and $q_j^i,\ j = 1, \ldots, k$.

$$\widehat{j} = \arg\min_j \{\left|p^{i-1} - q_j^i\right|\}. \tag{11}$$

Then, the following steps are applied:

1. For all pixel $\mathbf{x} \in B_j^i$ such that $j \neq \widehat{j}$, delete $\mathbf{x}$ from L_{in} and $\phi(\mathbf{x}) = 3$.
2. For all pixel $\mathbf{x} \in$ interior of B_j^i, set $\phi(\mathbf{x}) = 3$.
3. For all pixel $\mathbf{x} \in L_{out}$ such that $\mathbf{x}$ does not satisfy the L_{out} condition, delete $\mathbf{x}$ from L_{out} and set $\phi(\mathbf{x}) = 3$.

Figure 5 shows the result of applying the proposed algorithm to the sequence of images of Figure 4. It can be seen that the resultant curve only fits the boundary of the rectangle.

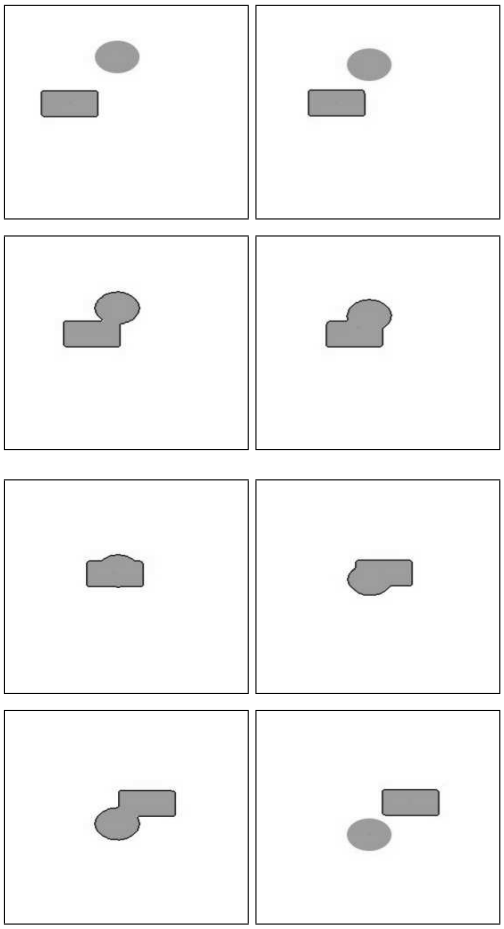

Figure 5. Result of applying the proposed algorithm to the sequence of images of Figure 4. It can be seen that the resultant curve only fits the boundary of the rectangle and the problem is solved.

In the following section, the results of applying these methods are shown.

4. Results

Figure 6 shows the result of applying the proposed algorithm to a real video. The tracking of the bag was done. It can be seen that the result is very good even in the case of complete occlusion and re-emergence of the occluded object. Figure 7 shows the result of applying the original algorithm to a test video taken from http://vision.ucsd.edu/datasets/lfarchive/lfs.shtml. It can be seen that the original algorithm fails. Figure 8 shows the result of applying the proposed algorithm to the same video of Figure 7. It can be seen that the result is very good.

Figure 6. Result of applying the proposed algorithm to a real video. Case of complete occlusion and re-emergence of the occluded object.

Figure 9 shows the result of applying the original algorithm in a case where confusion occurs. The object of interest is very close to another object of the same color and it can be seen that the original method fails. Figure 10 shows that the algorithm proposed in section 3.2 solves this situation. Note that the signpost is the same color as the bag, but it is rejected as a tracked object because it is not close

Figure 7. Result of applying the original algorithm to a real video. Case of partial occlusion.

Figure 8. Result of applying the proposed algorithm to a real video. Case of partial occlusion.

enough to the object of interest.

5. Conclusions and Future Work

A new method for dealing with the occlusion problem in video tracking, has been proposed. Occlusion is very likely to occur and the methods for object tracking fail. This paper proposes a tracking algorithm that detects when occlusion has occurred and solves the problem. Two cases are considered: 1-the object is obstructed by another one of a different color. 2-the object is obstructed by another one of the same color. In the first case, the algorithm fails in tracking the object's contour because the curve disappears from the scene. In this case, the occlusion is detected by the algorithm when the length of the curve decreases sharply. The solution is based on keeping the curve in the same place until the object of interest appears in the scene again. In the second case, the algorithm does not distinguish between two image objects of the same color, so due to the possible changes in the topology, the curve fits the edge of both objects and it is divided. We solve this problem by means of localizing the centroid of the curve. The results are very good even in the case of complete occlusion and re-emergence of the occluded object. Occlusion handling is a very difficult problem, and the presented solution has limitations, for example in case of occlusion by multiple objects. However, the original algorithm does not handle occlusion, so our work is an improvement.

Our ongoing work aims at improving the method robustness and at generalizing it for tracking multiple objects.

References

[1] T. Chan and L. Vese. Active Contours without Edges. *IEEE Trans. Image Processing*, 10(2):265–277, 2001.

[2] T. Chang, S. Gong, and E. Ong. Tracking multiple people under occlusion using multiple cameras. In *British Machine Vision Conference*, September 2000.

[3] L. D. Cohen. On active contour models and balloons. *Computer Vision, Graphics, and Image Processing. Image Understanding*, 53(2):211–218, 1991.

[4] L. D. Cohen and I. Cohen. Finite-element methods for active contour models and balloons for 2-d and 3-d images. *IEEE Trans. Pattern Anal. Mach. Intell.*, 15(11):1131–1147, 1993.

Figure 9. Result of applying the original algorithm in a case where confusion occurs. The resulting curve that provides the method consists of three tight curves.

Figure 10. Result of applying the proposed method to the video of figure 9.

[5] S. L. Dockstader and A. M. Tekalp. Multiple camera fusion for multi-object tracking. In *WOMOT '01: Proceedings of the IEEE Workshop on Multi-Object Tracking (WOMOT'01)*, page 95, Washington, DC, USA, 2001. IEEE Computer Society.

[6] A. Dumitras and A. Venetsanopoulos. A comparative study of snake models with applicationsto object shape description in bi-level and grey-level images. In *EURASIP, Workshop on non-linear Sgnal and Image Processing*. IEEE, 2001.

[7] M. Kass, A. Withkin, and D. Terzopoulos. Snakes: Active contour model. *International Journal of Computer Vision*, 1(1):321–333, Mar. 1988.

[8] S. Khan and M. Shah. Tracking people in presence of occlusion. In *Asian Conference on Computer Vision, Taipei, Taiwan*, January 2000.

[9] R. Malladi, J. A. Sethian, and B. C. Vemuri. Shape modeling with front propagation: A level set approach. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(2):158–175, 1995.

[10] R. Malladi, J. A. Sethian, and B. C. Vemuri. A fast level set based algorithm for topology-independent shape modeling. *J. Math. Imaging and Vision*, 6(2/3):269–290, 1996.

[11] T. McInerney and D. Terzopoulos. Medical image segmentation using topologically adaptable snakes. In *CVRMed*, pages 92–101, 1995.

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

[13] P. Perona and J. Malik. Scale-space and edge detection using anisotropic diffusion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(7):629–639, 1990.

[14] A. Senior, A. Hampapur, Y. Tian, L. Brown, S. Pankanti, and R. Bolle. Appearance models for occlusion handling. *IVC*, 24(11):1233–1243, November 2006.

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

[16] J. Sethian. *Level Set Methods and Fast Marching Methods: Evolving Interfaces in Geometry, Fluid Mechanics, Computer Vision and Materials Sciences*. Cambridge University Press, Cambridge, 1999.

[17] Y. Shi and W. Karl. A fast level set method without solving pdes. In *Acoustics, Speech, and Signal Processing, 2005. Proceedings. (ICASSP '05). IEEE International Conference on*, volume 2, pages 97–100, March 18-23, 2005.

[18] Y. Shi and W. Karl. Real-time tracking using level sets. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 2, pages 34–41vol.2, 20-25 June 2005.

[19] J. Weickert. *Anisotropic Diffusion in Image Processing*. Teubner-Verlag, first edition, 1998.

[20] C. Xu and J. L. Prince. Gradient vector flow: A new external force for snakes. In *CVPR*, pages 66–, 1997.

[21] A. Yilmaz, X. Li, and M. Shah. Object contour tracking using level sets. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(11):1531–1536, 2004.

[22] J. Zhao and D. Yao. Occlusion adaptative object tracking based on video image and radar data. In *International Conference on ITS Telecomunications Proceedings*, pages 943–946, Chegdu, China, June 2006.

[23] S. C. Zhu and A. Yuille. Region competition: unifying snakes, region growing, and bayes/mdl for multiband image segmentation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 18(9):884–900, Sept. 1996.