I3A
Universidad de Zaragoza
C/ María de Luna num. 1
E-50018 Zaragoza
Spain

Internal Report: 2003-V06

Video-Sensor for Detection and Tracking of Moving Objects[1]

E. Herrero, C. Orrite, A. Alcolea, A. Roy, J.J. Guerrero, C. Sagüés

If you want to cite this report, please use the following reference instead:
Video-Sensor for Detection and Tracking of Moving Objects. E. Herrero, C. Orrite, A. Alcolea, A. Roy, J.J. Guerrero, C. Sagüés, *IbPRIA, Pattern Recognition and Image Analysis*, LNCS 2652, pages 346-353, 2003.

[1] This work was supported by CICYT project COO1999AX014.

Video-Sensor for Detection and Tracking of Moving Objects*

E. Herrero[†**], C. Orrite[†], A. Alcolea[†], A. Roy[†], J.J. Guerrero[§], C. Sagüés[§]

[†] Aragon Institute of Engineering Research
[§] Department of Computer Science and Systems Engineering
University of Zaragoza
María de Luna, 1, 50018, Zaragoza, Spain

Abstract. In this paper we present a complete chain of algorithms for detection and tracking of moving objects using a static camera. The system is based on robust difference of images for motion detection. However, the difference of images does not take place directly over the image frames, but over two robust frames which are continuously constructed by temporal median filtering on a set of last grabbed images, which allows working with slow illumination changes. The system also includes a Kalman filter for tracking objects, which is also employed in two ways: assisting to the process of object detection and providing the object state that models its behaviour. These algorithms have given us a more robust method of detection, making possible the handling of occlusions as can be seen in the experimentation made with outdoor traffic scenes.

1 Introduction

Detection of moving objects is an important problem in applications such as surveillance [1], object tracking [2], and video compression [3]. There exist a lot of related approaches. So, Haar-wavelet transform is used to describe an object class in terms of a dictionary of local, oriented and multi-scale intensity differences between adjacent regions [4] and it is applied to detect pedestrians in driver assistance systems. The AMOS method [3] is an active system that uses low-level segmentation and a high-level object tracking, although it needs an initial segmentation made manually by the user.

Nevertheless when detecting moving objects, methods based on difference are more often used, although they have also some drawbacks. Thus, the difference map is usually binarized by thresholding at some predefined value but, as known, that threshold is critical, since a too low threshold will swap the difference map with spurious changes, while a too high threshold will suppress significant changes. There are several thresholding techniques specifically designed to be effective in these cases [5], but they do not take into account the relation between frames in order to eliminate noise and they are, in general,

* This work has been supported by the CICYT project COO1999AX014
** To whom correspondence should be sent, email: jelias@posta.unizar.es

computationally expensive. Additionally these approaches have some difficulties with small or slow-moving objects. In this sense, to make more robust the detection of changes, intensity characteristics of groups of pixels at the same location may be compared using an statistical approach [6].

All these works give good results, however in cases of occlusions or cluttered images their performance get worse. To solve this problem, some approaches employ techniques based on estimation or optical flow. In this context, some authors use a Kalman filter with snakes in order to track non-rigid objects [7]. In this case, the system detects and rejects spurious measurements, which are not consistent with previous estimations of motion. A Kalman filter and a neural system is used to avoid the gross errors in motion trajectories [8]. In other case, Kalman filter along with XT-slices (spatial-temporal) are used to analyze the human motion [9]. Sometimes the filter is used to recover lost regions when tracking vehicles in a road [2], or even, groups of filters each one specialized in a motion model are proposed in [10].

Our video-sensor is based on difference of images including long time information robustly filtered by the median of a set of images. This makes the method less sensible to the threshold, and changes of illumination have less influence. The pure segmentation algorithms work well in a few applications, but they fail in many cases. As commented, to solve these fails, researchers have used these algorithms together with estimation tools. In our video sensor we complement the idea of robust difference with a Kalman filter as an assistant to improve the system performance. Thus, the prediction provided by the Kalman filter is used to search on the difference map when the segmentation has failed. Besides that, the Kalman filter provides state information to control the object behaviour, avoiding problems when occlusions or slow moving objects are present.

The paper is organized in four sections. In the first one, we explain the detection of moving objects based on the robust difference of images. Secondly, we present the tracking algorithm working in two ways: assisting to detection, and providing object state. In the third section, we show the different experiments carried out and the obtained results. Finally, the conclusions are exposed in fourth section.

2 Detection and segmentation task

To search the object of interest, the proposed method analyzes changes over a sequence of images, instead of just between two images. This is carried out using the difference between a reference frame and current frame. The reference frame is obtained from a set of previous images in the sequence. The new frame is obtained from current frame and a shorter subset of neighbor images.

To obtain a noise-free reference frame we should use some smoothing. Linear filters suppress Gaussian noise but perform very poorly in case of noise patterns consisting of strong and spike-like components. This is the usual situation in a sequence of images where gray level of background pixels stays approximately constant except in a few, corrupted by noise. In these cases, the noise can be

effectively rejected using a rank value filter. In particular, the median filter has become very useful in robust estimation in presence of outliers, in relation to other traditional methods like root mean square.

The reference frame M_k is obtained by a temporal median filter of an input sequence of n images [11], where every frame has $m \times p$ pixels. This noise-free reference frame, is given by:

$$M_k = \begin{bmatrix} median(1,1,k) & \cdots & median(1,p,k) \\ \cdots & \cdots & \cdots \\ median(m,1,k) & \cdots & median(m,p,k) \end{bmatrix} \quad (1)$$

Being $median(i,j,k) = median\{F(i,j,k-n-l+1), \ldots, F(i,j,k-l)\}$ the median of gray level (F) in the image, where $i = 1, \ldots, m$ and $j = 1, \ldots, p$. Besides, k denotes the current time, n is the number of images used to obtain the reference frame and it represents the horizon of background filtering, and l is the number of images used to obtain the current frame.

The parameter n should be properly selected to eliminate the noise. Thus, if n is high enough, we will obtain a reference frame even if there are moving objects in the initial images. This reference frame is updated with every new image and it takes into account the illumination changes in such a way that the object motion detection is not disturbed.

Similarly the current frame (N_k) is computed from a set of (l) previous images. This set represents the horizon of motion filtering, which is related with the minimum velocity to be detected. The "l" parameter should also be properly selected: high enough to eliminate noise, but not too high because fast small objects could be lost. Finally, the detection of the moving blobs is made by the definition of a $MOVIL_k$ frame, which is obtained from the thresholding difference between the current and the reference frames as:

$$MOVIL_k(i,j) = \begin{cases} 1 \text{ if } |M_k(i,j) - N_k(i,j)| > \sigma \\ 0 \text{ otherwise} \end{cases}, \ \sigma \text{ is the threshold.} \quad (2)$$

3 The tracking task

With the robust method exposed above, we have the moving blobs which correspond to the objects of interest. Sometimes, this method can fail because of illumination problems, poor contrast, etc, and certain *assistance* is required to reduce the effect of these problems.

We have been working with the problem of tracking to match lines in a navigation system [12], using the Kalman filter. As known, this filter is a mathematic equations set, which provides a very efficient least squares solution using a dynamic model. It results very powerful in several aspects. For example, it gives future estimate from past information, it can be used with maneuvering targets and it can manage different dynamic models in according to object behaviour [10]. Although in these works linear models are used, some authors work with a

non-linear motion model to segment lines using the Extended Kalman filter [13] but our video-sensor proposes a tracking of objects based on standard Kalman filter.

To track moving objects we have chosen a state vector ($\mathbf{x}$) which is composed by four parameters: x and y positions and v_x and v_y velocities, which define the state of our objects. A constant velocity model with zero-mean random acceleration has been considered.

3.1 Kalman filter: Working as segmentation assistant

The main mission of the filter is to track objects that have been detected by the previous task in order to avoid their loss. The threshold used in the process of image difference (Equation 2) may cause the loss of pixels of low contrast corresponding to moving objects. Besides, as commented in section 2 a morphological filtering has been used, which may eliminate some blob corresponding to "good" but small, far away placed or partially occluded moving objects.

The Kalman filter gives a predicted position and its covariance, in such a way that the full system (in the *Recovering phase*) may look for corresponding pixels in the difference image (Fig. 1). If these pixels are found, then their centroid is used as measurement of Kalman filter.

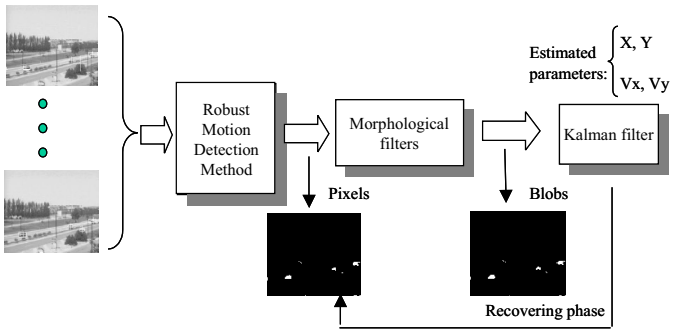

Fig. 1. Block diagram of the detection of moving objects

3.2 Kalman filter: Controlling the state of the object

The second novel use of the Kalman filter is the control and assessment of the state of the object (s_i). To model the behavior of the moving objects, six states and seven transitions have been defined (Fig. 2). The states are **Init**, **Localized_with_blob**, **Localized_without_blob**, **Stopped_&_localized**, **Lost**, and **Overlapped**. Five transitions are related to the evolution of moving objects and two are related to time conditions.

Transitions The transitions related with the evolution of the moving object are:

- T1: This transition is fired when the blob associated to a moving object is detected after the morphological filter.
- T2: This transition is fired when the blob is not detected but the corresponding pixels are detected at the difference image (*Recovering phase*).
- T3: This transition is fired when the Kalman filter estimates the position of the moving object, but neither its blob can be detected after the morphological filter nor corresponding pixels can be found at the difference image.
- T4: This transition is fired when a moving object overlaps with other moving object. So, only one blob is detected after the morphological filter which is associated with the closest object.
- T5: This transition is fired when the object velocity supplied by Kalman filter gets down a certain value.

The transitions related to time conditions are

- TT1: Time transition from **Stopped_&_localized** state when the time at that state is higher than t_{Stop} time.
- TT2: Time transition from the **Lost** state when the time at that state is higher than t_{Lost}.

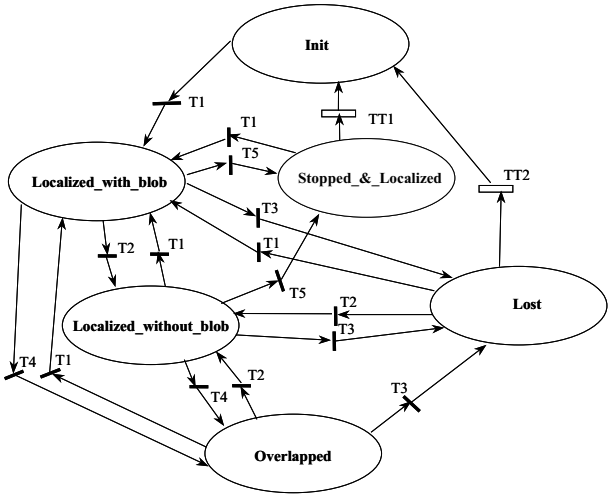

Fig. 2. Block diagram of states of the moving objects. The ellipses indicate the states and the transitions indicate the conditions to jump between states.

States An explanation for the different states follows:

- **Init.** This is the initial state, where the system looks for a new moving object. From this state there is just one output transition to **Localized_with_blob** state (T1). This happens when a new large enough blob is detected, being not close to the influence zone of an overlapping. Likewise, there are two input transitions from **Stopped_&_localized** (TT1) and **Lost** (TT2) states. When some of them is fired the component corresponding to the object is deleted.
- **Localized_with_blob.** In this state, the robust method is able to detect the blob because the blob is large enough. The centroide of this blob is used as measurement for the Kalman filter. When an object comes to this state from **Init**, a new component is created.
- **Localized_without_blob.** In this case, the detected blob after the morphological filtering is very small and it is eliminated. However some corresponding pixels are found at difference image around the position estimated by Kalman filter. So, the centroid of these pixels will be used as measurement for the Kalman filter (*Recovering phase*, Fig. 1).
- **Lost.** This is the state of the object whose blob has not been detected neither after the morphological filter nor in the *Recovering phase*. This normally happens when the moving object is occluded by a static object. In this state, Kalman filter continue estimating for t_{Lost} time without measurement.
- **Stopped_&_localized.** As told, the velocity of the object is given by the Kalman filter. According to this value, it is possible to deduce when the object is stopped. If the object remains in **Stopped_&_localized** state during a time $t > t_{Stop}$, it will be deleted and will evolve to **Init** state.
- **Overlapped.** This is the case in which a moving object is occluded by other moving object, and therefore both objects will evolve to this state. While this happens both objects will have the same measure due to the fact that only one blob is detected.

4 Experiments and discussion

Due to the limited extension of this paper we present some images showing the algorithm working in different situations. In this sense, four example sequences are depicted in Fig. 3. Comments about this figure are included in the legend.

In these images, bounding boxes on the object of the image indicate that the corresponding blob has been detected. Likewise, the size of crosses is proportional to the estimation covariance, in such a way that we may have little cross when corresponding pixels are detected and a large cross when they have not been detected. In the last case, there is no measurement for the Kalman filter.

5 Conclusions

In this paper we have presented a complete chain of algorithms to detect and track moving objects using a static camera. The proposed system performs robust motion detection and object tracking even with illumination changes, using

no special hardware requirements. The motion algorithm is based on image difference between two median filtered frames. In contrast to other methods of difference, which need to take a background free of other moving objects, the smoothing of reference and current frames allows to detect moving objects even though there are moving objects at the initial background. The detection and segmentation algorithms are complemented with a Kalman filter to track and match different moving objects along the sequence. The Kalman filter is also used in two ways: Assisting to the motion detection, and providing information to model the behaviour of the objects. This results in a much better method of detection which also makes possible the handling of occlusions.

References

1. P. Remagnino, T. Tan, and K. Baker, "Multi-agent visual surveillance of dynamic scenes," *Image and Vision Computing*, no. 6, pp. 529–532, 1998.
2. J. Badenas, J. M. Sanchiz, and F. Pla, "Motion-based segmentation and region tracking in image sequences," *Pattern Recognition*, no. 34, pp. 16–18, 2001.
3. D. Zhong and S. Chang, "Amos: An active system for mpeg-4 video object segmentation," *Int. Conference on Image Procesing*, pp. 647–651, 1998.
4. C. Papageorgiou and T. Poggio, "A trainable system for object detection," *International Journal of Computer Vision*, vol. 38, no. 1, pp. 15–33, 2000.
5. P. Rosin, "Thresholding for change detection," *Sixth International Conference on Computer Vision, Bombay, India*, pp. 274–279, January 1998.
6. A. Jain, R. Duin, and J. Mao, "Statistical pattern recognition: A review," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 22, pp. 4–37, January 2000.
7. N. Peterfreund, "Robust tracking of position and velocity with kalman snakes," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 21, pp. 564–569, June 1999.
8. R. Mecke and B. Michaelis, "A robust method for motion estimation in image sequences," *AMDO 2000, Palma de Mallorca, SPAIN*, pp. 108–119, 2000.
9. Y. Ricquebourg and P. Bouthemy, "Real-time tracking of moving persons by exploiting spatio-temporal image slices," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 22, pp. 797–808, August 2000.
10. P. Tissainayagam and D. Suter, "Visual tracking with automatic motion model switching," *Pattern Recognition*, vol. 34, pp. 641–660, 2001.
11. C. Orrite, J. E. Herrero, and A. Alcolea, "Fast robust motion detection by temporal median filtering," *Proc. of the IX Spanish Symposium on Pattern Recognition and Image Analysis, Castellón , SPAIN*, May 2001.
12. J. Guerrero and C. Sagues, "Tracking features with camera maneuvering for vision-based navigation," *Journal of Robotic Systems*, vol. 15, no. 4, pp. 191–206, 1998.
13. G. Foresti, "A line segment based approach for 3d motion estimation and tracking of multiple objects," *Int. Journal of Pattern Recognition and Artificial Intelligence*, vol. 12, no. 6, pp. 881–900, 1998.

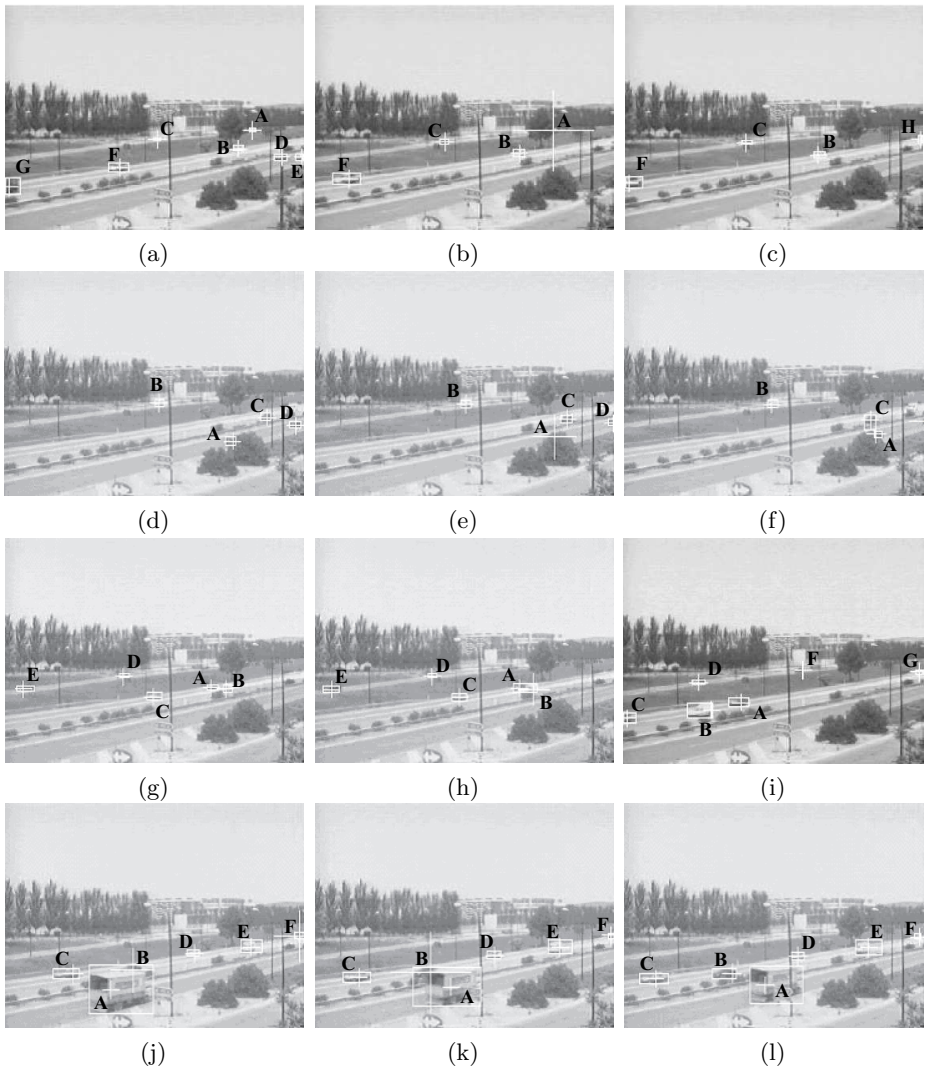

Fig. 3. EXP.1 (a) The "A" object is in the **Localized_with_blob** state; (b) it evolves by the T3 transition to the **Lost** state; (c) finally, it evolves by TT2 to **Init** state. **EXP.2** (d) The "A" object is in the **Localized_with_blob** state; (e) it evolves by the T3 transition to the **Lost** state; (f) finally, it evolves by T1 to **Localized_with_blob** state. **EXP.3** (g) The "A" and "B" objects are both in the **Localized_with_blob** state; (h) both objects evolve by the T4 transition to the **Overlapped** state; (i) finally, the "A" object evolves by the T1 transition to the **Localized_with_blob** state, and the "B" object evolves by TT2 to the **Init** state. **EXP.4** (j) The "A" and "B" objects are both in the **Localized_with_blob** state; (k) the "A" object evolves by the T4 transition to the **Overlapped** state, while the "B" object evolves to the **Lost** state; (l) finally both objects evolve by the T1 transition to the **Localized_with_blob** state.