Moving object detection and tracking system : a real-time implementation

Fernando de la Torre Frade, Elisa Martínez Marroquín,
Mª Eugenia Santamaría Pérez, Jose Antonio Morán Moreno

Department of Communications and Signal Theory.
Escola de Telecomunicacions La Salle. Universitat Ramon Llull.
Paseo de la Bonanova nº 8, 08022 Barcelona. Spain.
Phone (343)2902426 Fax (343) 2902416 Email:ftorre@els.url.es

RÉSUMÉ

Parmi les paramètres plus importants lors de l'optimisation des ressources humaines, on peut compter le nombre de personnes qui entrent ou sortent d'un local, leur temps moyen de permanence et leurs vitesses instantannées et moyennes. Nous presentons un système de detection et suite d'objects en mouvement avec une caméra stationnaire pour calculer ces paramètres. Notre système est capable de decerner et suivre un objet en mouvement sur une scène réelle sans en avoir aucune information préalable. Il s'agit d'une méthode independante du model. Nous proposons une nouvelle variable ainsi qu'une sélection à une limite multiple pour la détection de mouvement qui travaille à partir de deux images-difference pour assurer un algorithme robuste. En outre, nous proposons une nouvelle méthode de mise a jour de l'image de référence et de résolution des problèmes de collision et d'occlusion.

ABSTRACT

The number of people getting in and out some place (cashier, department store,…) in addition to the average time of a person inside or the average and instaneous speed, are important parameters to allocate optimally human resources. A moving object detection and tracking system with a static camera has been developed to estimate these parameters. The system presented is able to locate and to track a generic target in motion in a real scene without any previous information about it. We propose a non-model based system of tracking. In this paper, a new dynamic and multiple threshold selection to perform motion detection is applied. It works with two difference images in order to make the algorithm more robust. A new image reference updating and new way to solve the occlusion or collision problem is proposed.

1 Introduction

The number of people getting in and out some place (cashier, department store, underground,…) in addition to the average time that a person spends inside or the average and instantaneous speed with which he/she moves,… are important parameters to allocate optimally human resources. On the other hand, these parameters make it possible to detect anomalous situations like people sleeping, make statistics about the place where people spend more time or not to waste recording tape in the vigilance cameras if there is nobody in the scene.

A moving object detection and tracking system with a static camera has been developed to estimate these parameters. The system presented is able to locate and to track a generic target in motion in a real scene without any previous information about it. The problem of tracking targets in complex scenes can be solved with correspondence based methods taking out an appropiate set of features in order to establish a correspondence between them in different instants of time. We propose a non model based system of tracking.

2 System overview

The proposed system has two sub-systems (Fig. 1).

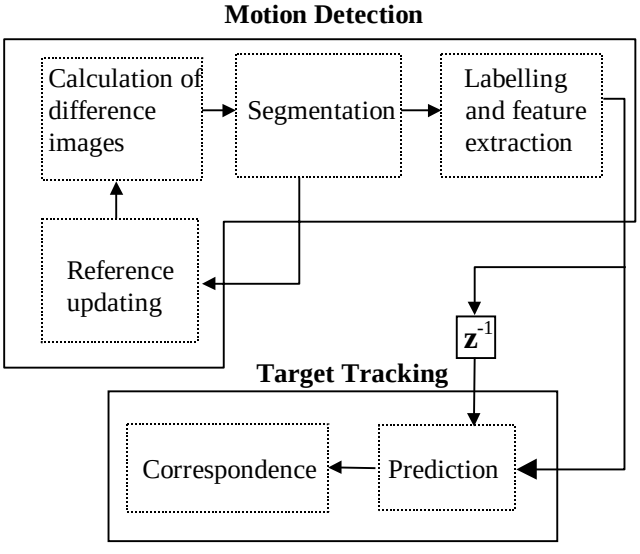

Fig. 1 System overview

The *Motion detection* algorithm has a sequence of 2 frames/seg. (2 Hz) as an input and gives a binary image containing the objects in motion as the output. Every connection region (blob) represents an object with motion. The motion detection system goal is to try to have a biunivocal correspondence between blobs and objects, less in cases of occlusion or collision.

The *Target Tracking* algorithm solves the problem of correspondence. That is, it finds the correspondence between the blobs of the frame k-1 and the frame k.

3 Motion detection

This section describes a method for finding candidate areas for moving objects in each image. The motion detection is based on differences between the image with the moving objects and the image which represents the stationary background (*Reference image*, RI). The main goal in this block consists on generating one blob for every object in motion. That is, to minimize the problem of *fragmentation*, that happens when a single object can generate two or more blobs. It's not always possible to get it, for example, in situations of *collisions or occlusion*, when two or more objects are so close that they generate a single blob in the binary image. These kinds of situations are solved in the tracking system when disocclusion is produced. As it can be observed in Fig. 1 the algorithm has four steps:

3.1 Calculation of difference images

At this stage we get an image which is easy to segment with a process of variable and multiple threshold selection at next step. To make the algorithm more robust, and less influenced by the thresholds selection, it works with two difference images.

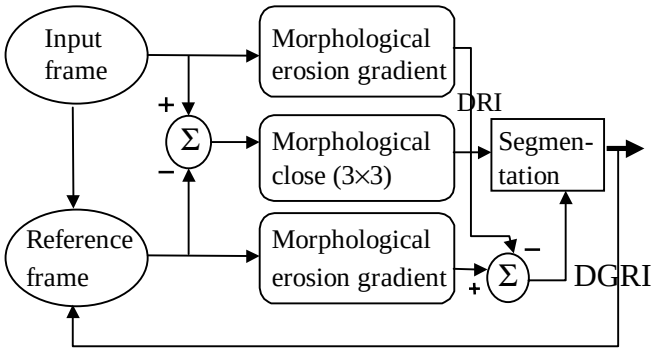

Fig. 2 Block diagram of Motion detection.

The first one (DRI) was the difference between the input frame and the reference frame. After, a morphological close with a structuring element (s.e) 3×3 (it's supposed that the background is brighter than the targets, else an open-close should be applied) is applied in order to remove noise and homogeneous the image locally. The other difference image (DGRI) will be the difference between the morphological erosion gradient with a s.e 5×5 of the input image and the reference image. The block diagram is represented in Fig. 2.

During all the paper, morphological filters will be applied to perform the pre or post-filters, since its computational load is less than conventional linear processing and it preserves the edges. Furthermore, the morphological gradients are more robust against noise. The philosophy was to work with different representations of the same scene, to make the process of segmentation more independent from the parameters election, if it doesn't represent an important computational load. It is possible to combine (add) these images weighted, but we used them independently because it's easier to obtain a low threshold with a connectivity filter in the DGRI image.

3.2 Segmentation

The aim of segmentation process is to group pixels together into regions which correspond to areas of interest within the scene. In our case, these areas are moving objects, and the final result gives a binary image. The major restriction in this project was counting and tracking moving objects in real time. Therefore, a simple, fast segmentation technique is required. That's the reason to use thresholding to segment the scene, but it should be robust front the selection of the threshold and should minimize the problem of fragmentation. The segmentation process works with a dynamic and multilevel thresholding.

Given the DGRI and the DRI (both have a histogram similar to decreasing exponential) the segmentation is done in three steps:

a) Obtain the thresholds of DGRI and DRI images. There were four thresholds.

DRI:

1- ThOtsu : Threshold of Otsu [2].

2-ThMean : Calculate the mean and variance of the histogram from the gray level with more probability (m.p) plus one to the first level different from zero beginning from the back. After, going across the histogram from m.p to the histogram will be less than the mean and the absolute difference of two consecutive values will be smaller than the standard deviation.

3-ThRe(Reconstruction) : Similar to ThMean, but now it goes across the histogram until two consecutive values of it will be smaller than 5 times the mean (t.m). After getting this threshold it checks that all the rest of the values until the end of the histogram are smaller than 3 times the mean, else it does again the same process, but now begin with the level that is bigger than 3 t.m.

4-ThUp (Updating) : it goes across histogram from m.p to get a value of histogram smaller than 7 t.m.

DGRI just calculate the ThRe threshold.

b) In the second step we obtain the motion objects thresholding the DGRI and DRI images. First we threshold DGRI with ThRe (1 represents movement, otherwise 0), after the pixels with less than 2 neighbourhood pixels at 1 are eliminated. To compact all the edges a morphological close is made with a s.e (5×5). The DRI image is thresholded with the minimum between ThMean and ThOtsu. Later and OR is made with these two thresholded images, after an erosion (3×3) is applied in order to eliminate spurious pixels.

c) At this stage a procedure similar to opening by reconstruction of erosion is used [6]. Where the binary image obtained at step b will be the marker image (MI) and the DRI image thresholded with Thre will be the reference image (IR). The idea is that it just reconstructs the objects in motion, which had a representation (marker) in MI, and the reconstruction with a smaller threshold permits the minimization of the fragmentation problem. The algorithm is :

$flag = 1$
$While(\ flag)\{$
$flag = 0$
$\forall p \in MI\{$
$if\ (MI(p) = 0\ and\ IR(p) = 1)$
$if\ (MI(p)\ has\ more\ than\ 1\ neighbourhood\ pixels\ at\ 1)$
$\{MI(p) = 1;\ flag = 1;\}$
$\}\}$

Relaxation methods [5] were also applied to segment the difference images, they produce good results, but they are usually very time-consuming.

3.3 Labelling and feature extraction

Once the image is segmented, it's necessary to assign a different identification to every blob, in order to know the number of blobs, or to find out the parameters like centroid or area. First, the contour is extracted (similar to chain code), after the inner is emptied. It is optional to calculate some useful characteristics of a binary object such as static moments, … with the chain code or Fourier Descriptors in order to do some geometrical filter.

3.4 Reference updating

The system must be robust front the change in lighting conditions, that's the reason to update the reference image (RI). An Update Image (UI) is created thresholding the DRI with ThUp (1 if is bigger than ThUP). Later an erosion (3×3) and a dilation operation (5×5) is applied in order to eliminate noise and connect the regions respectively. If the pixel in the UI is 0 then this pixel in the RI will be equal to the pixel in the Input Image, otherwise it would be the same as before. The RI is updated every three frames, when the targets move on average a distance bigger than they do.

4 Target tracking

Once the moving objects are extracted, their trajectory is found with a comparison between the real image and its prediction, based on the previous history. This Target Tracking block has two sub-systems : *prediction* which predicts the trajectory based on the centroid of the object and it is implemented with the recursive least square (RLS) to adaptively adjust the weights of the predictor filter. The other sub-system resolves the problem of the *correspondence*, that is to find the correspondence between the centroids of the objects in the frame k-1 and frame k.

4.1 Prediction

Let's assume the smoothness of motion trajectory, if we know the target location at times k, k-1, k-2, … then the location where the target will be at time k+1 can be predicted. The prediction error is reduced if the weights of the predictor filter are adaptively adjusted [4]. The basic idea of RLS forward linear predictor is to find a group of transversal filter weights with which the predictor minimizes the weighted prediction errors square sum. That is to find a group of filter weights that minimize the cost function (4.1)

$$E(k) = \sum_{i=1}^{k} \lambda^{k-i} e(i)^2 \qquad (4.1)$$

where e(i) is the prediction error given by (4.2). λ is a constant (forgetting factor) between $0<\lambda\leq 1$ and allows us to weight more recent errors (those closer to time k) more heavily than errors in the distant past.

$$\begin{aligned} e(i) &= x(i) - \mathbf{h}^{\mathbf{T}}(k)\mathbf{x}(i-1) \\ \mathbf{h}(k) &= [h_1(k),\ h_2(k), \ldots,\ h_N(k)] \qquad (4.2) \\ \mathbf{x}(i-1) &= [x(i-1),\ x(i-2), \ldots,\ x(i-N)] \end{aligned}$$

where x(i-1) is the centroid of a blob in time i-1 and **h**(k) are the predictor weights. The prediction algorithm calculates the prediction centroid for every blob in the previous image. This is made with the RLS forward predictor of third order with a λ=0.95. Notice that another advantage of the RLS algorithm is that it doesn't need a priori statistical information of the centroids.

4.2 Correspondence (corr.)

Once the prediction centroids of all the objects in the previous image are calculated, it's possible to build a predicted image by shifting each blob of the previous frame to its prediction centroid. The intersection between the expected image (predicted one) and the input image (actual) is made, building the cooccurrence matrix (A) [3]. The rows represent the blobs in the new image, while columns represent the blobs in the expected image. The element [i,j] of this matrix represents the intersection between a new blob i and the expected one j. With the analysis of this matrix and the definition of a path coherence function [1] it's possible to track the objects inside the scene. The proposed algorithm of target tracking classifies the blobs of the input image in five classes :

1-*Simple corr.* between expected blob j and new one i.
If $A_{ij}\neq 0$ and $A_{kj}=0$ $A_{il}=0$ for (k=0,…, NEB, k≠i),
for (l=0,…, NNB, l ≠j) and it isn't an occlusion blob.
2- *Fusion and Split of blobs*
If $A_{ij}\neq 0$ and $A_{kj}=0$ (k=0,…, NEB, k≠i) fusion of predicted blob j and new blob i.
If $A_{ij}\neq 0$ and $A_{il}=0$ (l=0,…, NNB, l≠j) split of predicted blob j and news blobs i.
3- *In, out blobs.*
If $A_{ij}=0$ (j=0,…,NNB) and [Cx(i) or Cy(i)]< border then persons=persons+1
If $A_{ij}=0$ (i=0,…,NNE) and [CPx(i) or CPy(i)]<border then persons=persons-1

4-*Corr. of occlusion or collision blobs.*

If the number of persons changes between two successive frames and there aren't in or out blobs there's a situation of collision or occlusion. Go back the trajectory until it arrives to frame (k-n), before the occlusion. Then link the blobs of the frame (k-n) with the blobs of frame k which minimize a directional coherence function similar to [1] :

$$\psi = \left(2 - \frac{\overline{X_{ik-n}X_{i(k-n-1)}} \cdot \overline{X_{i(k-n)}X_{ik}}}{\left\|\overline{X_{ik-n}X_{i(k-n-1)}}\right\| \cdot \left\|\overline{X_{i(k-n)}X_{ik}}\right\|} - \frac{\overline{X_{ik-n}X_{i(k-n-1)}} \cdot \overline{X_{i(k-1)}X_{ik}}}{\left\|\overline{X_{ik-n}X_{i(k-n-1)}}\right\| \cdot \left\|\overline{X_{i(k-1)}X_{ik}}\right\|} \right) \quad (4.3)$$

where X_{ik} represents the centroid of trajectory i at frame k. The $X_{i(k-n)}$ represents the centroid in the frame before occlusion and X_{ik} in the situation of disocclusion. If disocclusion is not produced the blobs link together and this blob is catalogued as occlusion blob. When disocclusion is produced, if the targets maintain the same direction the algorithm will produce a correct linking between the blobs before and after occlusion.

5- *Corr. between blobs without intersection.*

If the predicted blob j hasn't correspondence and it isn't an out blob, it's associated with one which minimizes a path coherence function. The same occurs with a new blob without intersection.

Where :

NNB : number expected blobs ; NEB : number new blobs

CP(i) : prediction centroid of blob i ; C(i) : centroid of blob i

5 Experimental results

The procedure was applied to different sequences of real s images of dimensions 200×150 pixels, with a frequency of two images per sec. The system was implemented in a pentium with a clock speed of 133 Mhz. The average processing time of a frame is 400 ms, the 95% of the time was dedicated to the motion detection algorithm.

In Fig. 3 we can observe a sequence with three people, which produce a situation of occlusion and collision. The segmentation achieved by the system can be observed in Fig. 4, note that it's not always possible to solve satisfactorily the problem of fragmentation, for example in the eighteenth frame the head is splitted from the body. The correspondence of this blob (head) is solved by the tracking algorithm linking it with the blob of the nineteen frame with minimizes a path coherence function. In Fig. 5 the trajectories of the people are found out satisfactorily.

6 References

[1] Sethi I.K. and Jain R. "*Finding trajectories of feature points in a monocular image sequence*", IEEE Trans. On Pattern Anal. Mach. Intell., PAMI-9, Nº 1, pp 56-73, January 1987.

[2] Otsu N. "*A threshold selection method from gray-Level histograms*", SMC-9, Nº 1, pp.62-66, January 1980.

[3] Franchi, G., Gamba, P., Marzzi, A. and Mecocci A. "*Object tracking in complex scenes by means of the correspondence based method*", Proceedings of Melecom 96, vol. 2, pp. 1093-1097, May 1996.

[4] Haykin, S. *Adaptative Filter Theory*. Prentice-Hall, Englewood Cliffs, New Jersey. 1986.

[5] Rosenfeld A. "*Iterative methods in image analysis*". Pattern Recognition Letters, vol.10, pp.181-187. 1978.

[6] Vicent, L. "*Morphological Grayscale Reconstruction in image analysis : applications and efficient algorithms*", IEEE Trans. On Image Process., vol.2, pp. 176-201, April 1993.

Fig. 3 Nine consecutive images from a real scene.

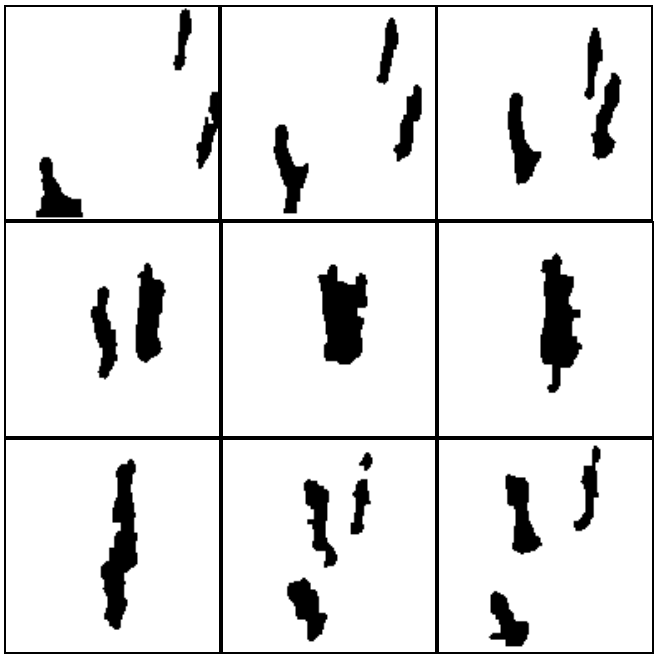

Fig. 4 The blobs extracted from the sequence of Fig. 3.

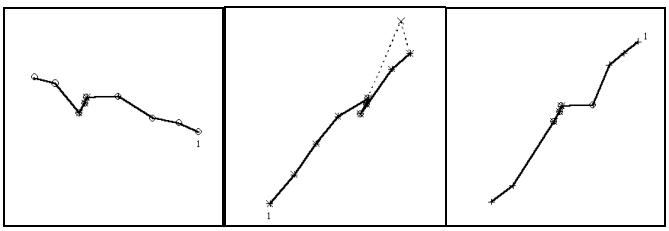

Fig. 5 The trajectories of three people extracted by the proposed system.