Intrusion Detection Project Report

Lorenzo Magnanelli

***Abstract*—This report describes a proposal for the "Development of a computer vision system aimed at intrusion detection" project. The solution includes both the first (blob detection and classification) and second (identification of true objects) tasks.**

I. Introduction

I developed a software system that, based on automatic video analysis, can detect objects not belonging to a static scene (background) and establish which of such objects are persons. The software extracts features of moving objects and exploits them to pursue the proper classification. Below are described the main concepts, including: selective background initialization and update, blob detection and features extraction, classification, and identification of true objects.

II. Change Detection Algorithm

Among the various change detection algorithms, I choose and implement Background Subtraction as follows:

$$C_t(i,j) = \begin{cases} 255, & \text{if } d(F_t(i,j), B_t(i,j)) > T \\ 0, & \text{otherwise} \end{cases} \tag{1}$$

The change mask C_t has values 255 or 0 depending on whether or not the distance between frame F_t and background B_t is greater than a predefined threshold T.

In doing so, I avoid ghosting (false positives), foreground aperture (false negatives), and stationary objects (false negatives) problems linked to two/three-frame difference algorithms. On the other hand, this procedure introduces camouflage (false negatives) that can be handled with morphological operators, as described further in the report. Furthermore, the video we want to analyze doesn't show an initial sequence with no moving objects and illumination condition changes over time. There is a subject already in the first frame that keeps moving, while ambient lighting is slowly darkening. Therefore, I have to deal with background initialization and updating.

A. Background Initialization

Firstly, I try a temporal mean and median across a few initial frames to obtain a scene without moving objects. Despite some posterization, a good background is obtained considering a median of at least 60 frames. Instead, mean interpolation would require a longer sequence to achieve a similar result: even with 100 frames, a rectangular shadow can be seen on the bottom right of the image due to the transition of the subject not being correctly managed.

Since it's required to generate the background using as few frames as possible, I implement a "selective" initialization as suggested by [1] and [2]. First, I initialize the background

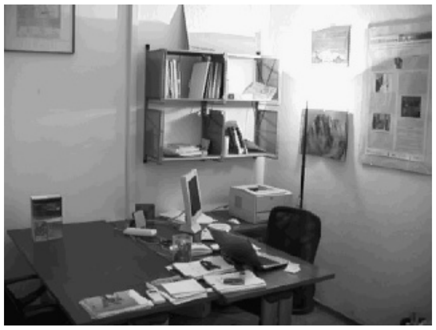
Fig. 1. Background estimation from a median over the first 60 frames.

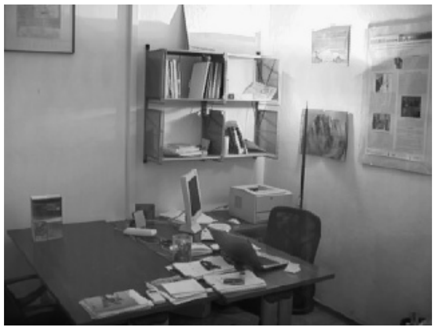
Fig. 2. Background estimation from a mean over the first 100 frames.

B as the first frame. I also set *init_mask* C_0 by thresholding the difference between the second and the first frame. Then, I consider the two-frame difference of consecutive frames, and as soon as a pixel in *mask_t* C_t belongs to the background while in *init_mask* it is foreground, the background and *init_mask* are updated.

```
if $C_0(i,j) = 255$ and $C_t(i,j) = 0$ then
    $B(i,j) \leftarrow F_t(i,j)$
    $C_0(i,j) \leftarrow 0$
end if
```

This procedure is repeated until the number of foreground pixels in *init_mask* goes below a chosen threshold. The masks are filtered using "conservative" morphological operators (opening + dilation + closing). At that point, background pixels corresponding to remained foreground pixels in *init_mask*, get their values from the last frame:

```
if $C_0(i,j) = 255$ then
    $B(i,j) \leftarrow F_t(i,j)$
end if
```

With this approach, I get a pretty good result exploiting just 22 frames. Thus, less than 2 seconds of video (12 fps) are used to initialize the background.

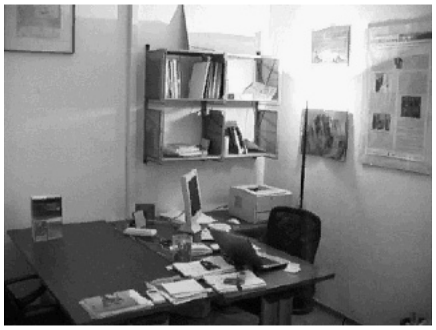

Fig. 3. Background estimation from selective initialization (22 frames used).

B. Background Updating

I implement "selective" background updating by [1] to manage the slow light change in the scene.

$$B_{t+1}(i,j) = \begin{cases} \alpha \cdot F_t(i,j) + (1-\alpha) \cdot B_t(i,j), \text{if } C_t(i,j) = 0 \\ B_t(i,j), \text{otherwise} \end{cases} \quad (2)$$

Where α is called "adaptation rate" and represents the speed of adaptation of the background model to changes occurring in the monitored scene. The change mask C_t is obtained by background subtraction and "conservative" morphological operators (just dilation in this case).

III. Blob Detection

Connected component labeling and blob analysis are executed on the change mask given by the background difference of each frame. Before doing that, I improve the mask by applying a sequence of morphological operators:

1) 3x3 Opening, to delete small blobs;
2) 21x21 Closing, to fill holes in the foreground region;
3) 7x7 Opening, to consider just larger blobs;

Each operator uses a circular symmetric structuring element to achieve smoother contours. Now I can compute labeling and blob analysis.

A. Blob Analysis and Classification

Once I have found the blobs, I extract some features. For each label, firstly I calculate the contours, then compute area, perimeter, compactness, and barycentre coordinates (the latter feature is a result of connected component function). With regards to object classification, I rely on some characteristics of the video. Since the subject is shown in close-up view and has a quite different form-factor with respect to the objects he moves, compactness has been selected to classify "person" or "other". I fine-tune a value and classified as "person" those blobs whose compactness is greater than such threshold.

B. Outputs

The graphical output is performed by showing the labeled blobs mask for each frame, but those deployed for initialization purposes. I transform the gray-scale labeled image into pseudo-colors by a label mapping to HSV color space. In practice, I build hue, saturation, and brightness channels separately, stack them together and convert back to BGR. Saturation and brightness values are set, while hue is computed by a linear stretching of the intensities of the gray-scale image. (It is present also a function able to show labeled contours. The process is similar to what is described above with a little change. Indeed, first, I obtain a gray-scale labeled image of contours only, second, pseudo colors are applied, and eventually, they are transferred to the frame). The graphical output is shown whether in real-time during the execution of the algorithm or saved into a file.

As for the text output, all requested information is saved into a CSV file.

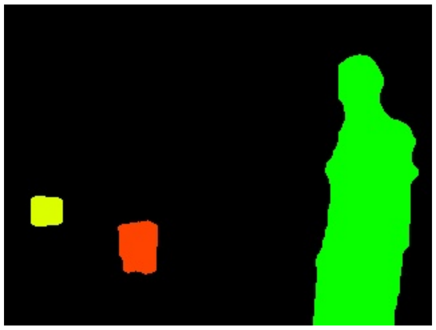

Fig. 4. Labeled blob graphical output example.

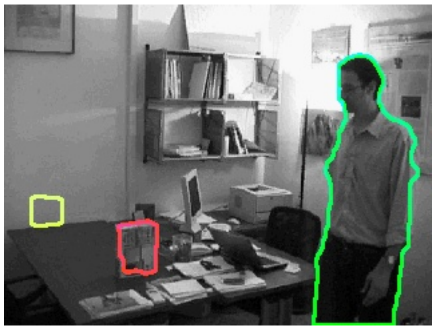

Fig. 5. Labeled contour graphical output example.

TABLE I
TEXT OUTPUT EXAMPLE

410	3						
1	10627	529	2631	270	150	person	True
2	470	84	1506	31	154	other	False
3	970	129	1709	100	181	other	True

IV. Identification of True Objects

The second task is about classifying true and false objects among those detected by the algorithm. Blobs related to present objects in the scene are true, while those missing are false. For instance, in the second part of the video, the subject picks up a box and removes it from the scene. That has to be labeled as false. To achieve this goal, I take into account blob contours. I compute 'real' edges and count how much they belong to the detected objects. Thus, if an object is not present, it won't have a real contour in the frame (unless a similar body was hidden behind it). I implement this process by determining Canny edges and comparing them with change mask contours, for each blob:

```
$cntr_{real}(i,j) \leftarrow cntr_{detected}(i,j)$ **and** $cntr_{canny}(i,j)$
$cntr_{percent} \leftarrow \mathbf{sum}(cntr_{real})/perimeter$
**if** $cntr_{percent} > thr$ **then**
    **return true**
**else**
    **return false**
**end if**
```

References

[1] A. Lanza, "Change detection algorithms," 11 2013, pp. 27 – 29.

[2] E. Rivlin, M. Rudzsky, R. Goldenberg, U. Bogomolov, and S. Lepchev, "A real-time system for classification of moving objects," vol. 3, 08 2002, pp. 688 – 691 vol.3.