Computer Science Journal of Moldova, vol.**11**, no.3(33), 2003

A system for object detection and tracking in video streams based on motion detection

Ennio Cortellini

Abstract

This paper discusses some of the existent techniques for building robust object detection and tracking systems using motion detection methods. The first part of the article briefs on the tasks of the motion analysis, focusing on object detection and tracking and reviews the most recent approaches to the motion analysis problems. In the second part, there is a proposition for architecture of an automatic surveillance system with a presentation of its current status of implementation.

1 General concepts and tasks in motion analysis

1.1 Introduction

Motion analysis has been for the last years one of the most challenging tasks for the image processing community. Only in image sequences (which is what any video stream finally represents) can we analyze and study dynamic processes, i.e. real-world processes. This is why the analysis of image sequences represents such an important objective, not only in the science and engineering world but also in day-to-day activities and tasks.

Two important problems in image sequence analysis are object detection and object tracking.

The visual content that is involved with the image concept is usually seen as a hierarchy of abstractions. The primitive information in this hierarchy is offered in the form of brightness or color information of the individual pixels. The next abstraction step (and further processing) leads us to such abstractions as edges, shapes, connected regions. The final step (which is also one of the purposes of the image processing) should extract from an image abstractions equivalent to the human concepts of objects and inter-object relationships.

Object detection is about verifying whether an object is present or not in the input image sequence. Its results should offer also the possibility of a precise localization of that object. The problem that the object tracking refers to is the monitoring of the object's spatial and temporal changes (motion kinematics and dynamics) in a video stream. The result of the object tracking might be used as input to an intelligent system that determines the significance of the gestures (in case of a human subject) or motion of that particular object. The two processes are closely interrelated and cooperate in a real system.

The applications of the object detection and tracking are multiple and varied, but some of the most important are operator-independent visual surveillance, video compression, video database indexing, medical imaging and robotics.

1.2 Object detection and tracking approaches

There is a wide variety of methods for accomplishing these tasks, which have been developed in the last decades. They all use as the main features to be extracted from the image either the visual information (e.g. shape, color), the motion information (e.g. velocity fields) or a combination of the both. While the methods from the first class are usually extensions of the methods used for still pictures, the techniques that deal with motion detection are focusing on the extraction of the motion-specific information included in the image sequences. The remaining of this section will focus on the extraction of motion information from a sequence of images and will review some of the techniques and algorithms proposed for this task.

The key issue of the analysis of object motion in image sequences is that the very concept of the movement is *geometrical* in nature, while the effect of the motion is perceived as a change in the *intensity* of the pixel values in the image. It follows that the essence of motion estimation problems is the way that this intensity variation or the changes of some other higher-level abstraction is associated with the movement of the objects in the image.

1.2.1 Velocity vectors

The velocity vectors (also referred to as displacement vectors) are the vectors describing the motion of the different objects between two consecutive images from a sequence. As with any other movement feature that can be extracted, the velocity vector can be inferred from the spatial and temporal gray value changes. These can be computed easily by applying local operators, which compute the spatial and temporal derivatives. However, the local operators are only sensitive to the extent of their mask limits, which generates the aperture and correlation problems.

The aperture problem is caused by the "locality" of the operation: only the normal component of the velocity vector can be determined. The aperture problem is not present in the case of corner edges in the operation's mask, when the exact velocity vector can be determined.

The correspondence problem is the general version of the aperture problem and it refers to the impossibility to find the corresponding points in two consecutive images of the sequence.

There are two main classes of methods that deal with determining the velocity vectors: *differential* and *correlation* methods.

Differential methods

These methods are based on the concept of *optical flow*, which is an extension of the common flow concept found in fluid dynamics. The *continuity equation* for the density of a fluid:

$$\frac{\partial \rho}{\partial t} + \nabla (u\rho) = \frac{\partial \rho}{\partial t} + u\nabla\rho + \rho\nabla u = 0 \tag{1}$$

can be converted to the *simple continuity equation* for the optical flow:

$$\frac{\partial g}{\partial t} + u\nabla g = 0 \tag{2}$$

where g represents the intensity of a pixel.

In the one-dimensional case, it follows that the velocity is:

$$u = -\frac{\partial g}{\partial t} / \frac{\partial g}{\partial x}, \tag{3}$$

provided that the spatial derivative is non-zero. As it can be easily seen, in the one-dimensional case the aperture problem does not appear. The velocity vector can be unambiguously determined.

The consideration of the n-dimensional case leads to a scalar equation that contains n unknowns, and thus the velocity cannot be unambiguously determined. However, the aperture problem can be solved in the two-dimensional case by using a least squares approach to solving the equation (as shown in [8]).

Nevertheless, as it is proven in [8], the simple continuity equation for the optical flow is only valid in the presence of strictly constant illumination conditions. These conditions are not only dependent on the variations of the light source, but also on the reflective properties of the surface of the objects. This makes the method inapplicable for outdoor systems or any system that doesn't perform its task under strict illumination control.

Correlation methods

The correlation methods used for velocity vector estimation are based on the fact that we can solve the correspondence problem by choosing features that uniquely identify the objects in the image. Thus, we can determine the displacement vectors just by identifying some selected features in the next image from the sequence. If the features don't depend upon the illumination condition, then the estimation of the velocity (displacement) vectors is very robust.

Such features have been proposed and applied [8]. One of these is the *monotony operator*, which is defined by:

$$\begin{aligned} &e(k,l) = 1 \text{ if } g(m,n) > g(m+k,n+l), \text{ for any } k,l \in \mathit{mask} \\ &e(k,l) = 0 \text{ else} \end{aligned} \tag{4}$$

The mapping produced by this operator is still ambiguous, since the operator provides the same result if the pixels (other than the center pixel) are permuted within the mask. The computation of the monotony operator greatly reduces the computational time (because only the points with useful features are included into the correlation analysis procedure).

Various algorithms have been proposed for computing the velocity vectors, which combine the two methods in an iterative fashion. Usually these algorithms are working in two steps:

1. The velocity vectors are detected with a local displacement detector

2. A correction procedure is applied to correct the errors in the initial velocity vector field, the errors issued by the aperture and the correspondence problems.

A classical algorithm combining the two steps is described in [1] and proceeds as follows:

In the first step, the partial derivatives with respect to space and time are computed from two consecutive source images:

$$g_x = \frac{\partial g}{\partial x}, g_y = \frac{\partial g}{\partial y}, g_t = \frac{\partial g}{\partial t} \tag{5}$$

The components of the velocity vector are defined as follows:

$$u = \frac{dx}{dt}, v = \frac{dy}{dt} \tag{6}$$

The second step is an iterative procedure, which corrects these components. The iteration ends upon meeting a criterion for stopping. The correction procedure is based on the minimization of a global error, which is the combination of two distinct errors. The influence of these errors on the global error is described by means of a parameter α. The formulas that produce the values at the $(n+1)$-th iteration from the local mean values and the results of the previous iteration are:

$$u^{(n+1)} = \overline{u}^{(n)} - \frac{g_x(g_x\overline{u}^{(n)} + g_y\overline{v}^{(n)} + g_t)}{\alpha^2 + g_x^2 + g_y^2} \tag{7}$$

$$v^{(n+1)} = \overline{v}^{(n)} - \frac{g_y(g_x\overline{u}^{(n)} + g_y\overline{v}^{(n)} + g_t)}{\alpha^2 + g_x^2 + g_y^2} \tag{8}$$

These formulas have the advantage that they share a part of the terms so that the computation time and resources are greatly reduced.

The methods using the optical flow for estimating and analyzing motion perform well under strictly controlled illumination conditions. However, while the information obtained from the computation of the velocity vectors is very consistent and fully describes the motion of the objects involved into a motion scene, the assumptions it makes about the invariance of pixel values due to the changes in light intensity does not hold true for real-world systems. Also, in the case of object detection and tracking tasks not all the information about the motion contained in the velocity vectors is needed. It is often enough to determine the regions of significant changes in the image sequence as a first abstraction of the visual information. This information can then be used as an input to a classifier or object detector module, whose feedback can be used in determining the accuracy of the first estimation of the "significant" regions. This kind of approach is currently used in the most motion detection-based systems for object detection and tracking.

1.2.2 Lower level motion analysis techniques

As stated before, the quantitative approaches such as the computation of the velocity vectors are computationally intensive, sensitive to error and, above all, result in more information than is usually needed. Such is the case of the motion detection problem.

The approaches presented in this section are intuitive and implementable. They proved to be very useful as preprocessing techniques that yield the basic information for more sophisticated methods.

One of the most elementary approaches to the motion detection and analysis in dynamic images is the analysis of *difference images.*

Difference Images

The analysis of the difference images relies on the simple observation stated in the beginning of this part: the movement of an object produces a change in the intensity of the pixels belonging the region where the movement takes place. This simple fact is the basis for all the algorithms for motion estimation from this category.

Given the model of a dynamic image (or a sequence of images) as a space-time function, $I(x, y, t)$, the difference image is computed straightforward using the relation:

$$D(x, y, t_1, t_2) = I(x, y, t_2) - I(x, y, t_1) \tag{9}$$

It is important to notice that this difference image may contain negative values at some points.

Although the operation poses no computational problems and has a high potential for parallel implementations, the resulted amount of information regarding motion is also very small and difficult to be interpreted. It also provides practically no higher-level information with regards to the content of the scene or whether there is a change in the scene or just fluctuations of the camera or the illumination conditions.

There are a few remarks that can be made on the result of this simple calculation [14]:

1. A difference image can be used as an approximation of the time derivative of the image function (more precisely, it is an approximation of the derivative at the midpoint of the time interval $[t_1, t_2]$).

2. Difference images have the visual characteristics of the edge images. This is applicable if the difference in pixel intensities can be modeled as small shifts of the image plane.

3. The difference image will not always contain well-defined regions of changes, but instead a rather incomplete information with regards to where the change occurred will be presented. This happens when the object involved into motion has the same color/texture as the background.

4. Also, the difference images contain information related to the direction of movement.

Segmentation of the difference images

Image change detection is usually performed by a segmentation procedure, which can be described by:

$$O(x,y,t_1,t_2) = \begin{cases} I(x,y,t_2), & \text{if } |D(x,y,t_1,t_2)| \geq T_d \\ 0, & \text{otherwise} \end{cases} \tag{10}$$

The operation described is a usual segmentation procedure using a global threshold. It involves computing the difference image, which is then transformed into a binary image by means of applying a threshold, the result being afterwards multiplied by $I(x,y,t_2)$ (which is the current image in the stream). As a result of this operation, only the significant changes are retained for further analysis.

The critical step in this method is the choice of the threshold. It is obvious that choosing a too high threshold value will ignore some of the significant changes, while a too low value will result in the generation of many false alarms. A correct threshold would have to reflect the changes in the illumination conditions, the possible fluctuations of the camera and the characteristics of the scene.

Threshold determination methods

A number of techniques have been proposed for computing the threshold, but the majority of them make assumptions that do not hold true in the case of the difference images (for example in many cases it is assumed that the histogram of these images is bi-modal).

An example of a more generic and accurate way of determining a global threshold is presented in [4]. This technique tries to model the spatial properties of the signal and/or of the noise and is based on the following idea: although there is no a priori knowledge of the number, location and size of the significantly changed regions in an image, it can be assumed that these characteristics will remain constant for a range of threshold values. This observation leads to the conclusion that if such a range of threshold values is found, then the segmented regions would very unlikely be a result of the noise influence. Thus, one of the values in the found range can be used as threshold.

Another class of methods is concerned with computing thresholds locally, for each pixel or for a "window" of a determined size. These methods are behaving more correctly than the global thresholding techniques especially in cases when the pixel intensity variation is dependent on their specific location. (Such is the case of a scene that contains a computer monitor, for example. The pixels corresponding to the image of the monitor would have a high variation frequency and would yield a bigger difference in intensities.)

There is another simple observation that can be made regarding the analysis of motion in difference images. It is the fact that in the image resulted from subtracting two consecutive images there will be two different regions recorded: one that corresponds to the actual position of the object and one corresponding to the previous position of the object, where in the second picture the background is revealed. This lead to the idea that better results might be obtained if a *reference image* could be computed, which would contain only the background (also known as *background image*). This image, subtracted from the current image would allow having a clearer representation in the difference image of the changed region. This idea, combined with the local thresholding methods is the basis for methods based on *background modelling or estimation.*

Background modelling and estimation techniques

These techniques are usually trying to obtain a reference image, that is, a representation of the background. The result is typically an image where the pixel values are a combination of the values of pixels at the same location over a longer period of time. The algorithms must decide when and from which images the reference image is to be re-estimated. This is crucial because the background has, on one hand, to account for the changes in illumination conditions usually caused by time or atmospherical conditions, and, on the other hand, it has to avoid incorporating changes caused by objects from the foreground. The maximum difficulty arises when the objects/persons, which are moving, have a very slow motion, this causing usually the background to incorporate the moving objects characteristics.

The background image is computed as:

$$B_N(x,y) = F(I_1(x,y), I_2(x,y), ..., I_{N-1}(x,y)), \tag{11}$$

where $I_1(x,y)$ represents the pixel value at position (x,y) in the i-th image of the sequence.

F is a function that actually models the background of the scene. The main difference between the background estimation techniques is represented by the actual computing relation that this function implies.

The problem of background estimation is, in fact, a particular case of the more general problem of estimating an unknown state from a set of process measurements. Because of the noisy nature of the electrical measurements of the sensors that provide the information and the statistical nature of this noise, *stochastic* methods have been successfully used to address this issue.

The most basic forms of the estimation function are represented by weighted averages or median computation. That is, the background estimation function is, in the first case:

$$F(I_1(x,y), I_2(x,y), ..., I_{N-1}(x,y)) = \sum_{i=1}^{N-1} \alpha_i I_i(x,y) \tag{12}$$

In the case of the median, the value of the function represents the median value of the set of pixel values at the same location in the image over a period of time or N samples.

The median filtering is one of the most currently used methods for background estimation. A rigorous discussion of the noise filtering capabilities of this method compared with the average filters is given in [9]. Intuitively, it is important to notice that the output value of a median filter is actually a value that the pixel at that location has really reached, while an average filter outputs a value that the pixel intensity might never take.

Other filters that have been successfully used are: the Kalman filter or the AR (Auto Regressive) filter.

The Kalman filter is essentially a set of mathematical equations that implement a predictor-corrector type estimator that is optimal in the sense that it minimizes the estimated error covariance. The estimation is made using some kind of feedback control: the filter estimates the process state at a time and then obtains the feedback in the form of (noisy) electrical measurements.

A method using this filter is presented in [13] and proposes the following formula:

$$B_{N+1} = B_N + (\alpha_1(1 - M_N) + \alpha_2 M_N)D_N, \tag{13}$$

where:

M_N is a binary mask of hypothesized moving objects in the current frame,

D_N is the difference image between the current frame and the current background image,

α_1, α_2 are the gains computed on the base of estimates of the rate of change of the background.

As the equation shows, the background is adapted by examining two information sources: D_N, which is the result of the direct observation and M_N, which is provided as a feedback by the object tracking module and includes information on a higher-level of abstraction in it.

Another method is one that uses an AR filter is used, which models the background according to the formula:

$$B_{N+1} = (1 - \eta)B_N + \eta I_N, \tag{14}$$

where $0 \leq \eta \leq 1$ represents the learning rate constant. These methods are computationally efficient and provide as much information as needed for a further higher-level processing: the regions of significant changes. As it can be seen from the equation (13), the changes in these regions of interest are afterwards filtered from the current frame when the background is updated (this is what represents, in fact, the M_N mask).

After the background is estimated, a difference image is computed, which is segmented using local thresholds.

One of the advantages exhibited by the background estimation techniques is that even the motion of homogenous structured objects will be detected in the difference image. Another advantage is that even the stationary objects that are not present in the background at first can be detected.

A few problems arise when the moving objects are slow and thus the background acquires a part of the object, resulting in false alarms. Using a conditionally lagged background model can usually alleviate this problem. In this model, a conditionally lagged background image ($B_N^{cond}(x,y)$) is defined, which is updated according to the following scenario:

For every pixel in the current image that has been classified as a foreground pixel, $B_N^{cond}(x,y)$ is set to the continuously updated model, $B_N(x,y)$. Otherwise, the $B_N^{cond}(x,y)$ value is not updated. If after some T time steps, the difference between the conditionally lagged background model and the image - $D_N^{cond}(x,y)$ - is less than the magnitude of the difference image $D_N(x,y)$, the value of $B_N(x,y)$ is reset with the value of $B_N^{cond}(x,y)$. Otherwise (the magnitude of $D_N(x,y)$ is less then the magnitude of $D_N^{cond}(x,y)$), the value of $B_N^{cond}(x,y)$ is set to the value of $B_N(x,y)$. This mechanism makes the decision of foreground/background classification dependent on the $\left|D_N^{cond}(x,y)\right|$, instead of $|D_N(x,y)|$ and avoids the integration of the objects into the re-estimated background.

2 The system description

We have applied the above considerations in the construction of a video surveillance system, using a personal computer and a video web camera attached to it. The program controls an area photographed by the static video camera. When there is a major modification in the pictures taken by the camera, the program saves those pictures and records the event into a log file.

The application is running on an Intel platform, using the Linux operating system, and has low hardware requirements: Pentium 200

MHz, 32 MB RAM and about 500 KB free hard disk space, excluding the disk requirements for the saved pictures. The tested web camera is a Creative Video Blaster WebCam 2, connected to the computer through the parallel port.

2.1 General presentation of the application

The software used by the system has 2 processes, one that takes care of the graphical user interface, and the other one that deals with the image processing. The two processes communicate through shared memory, controlled by a semaphore. Using the graphical user interface, the user can control the program: start the image processing engine, modify some parameters such as using one of the implemented algorithms, enable or disable the saving of the reference image and the threshold image in JPEG format, enable or disable the saving of the images with major modifications and, also in the GUI, the user can see the images taken by the video camera.

2.2 The image processing algorithms used by the application

2.2.1 The difference algorithm

It is used a simple algorithm that computes the difference between the reference image and the current image. As a preprocessing step, on the current image a median filter with a kernel of 3 by 3 pixels is applied. If the dispersion computed for the histogram of the difference image is bigger than a certain threshold, the current image is considered different from the reference image.

2.2.2 The motion detection algorithm (using segmentation and objects area)

The algorithm computes the pixels of the reference image as the median value of the pixels from a sequence of images; the number of the pictures in the sequence is a parameter that can be set by the user.

The value of the pixels from the threshold image is calculated as a proportional value of the dispersion from the median value of the same sequence. Once again, the difference from the reference image and the current image is made. The zones marked as modified are segmented using the threshold image. After that, a pixel connectivity analysis is made, grouping them in compact regions. Applying two successive morphological operations (a closing and an erosion) eliminates isolated pixels and stray detections.

If the area of the biggest region exceeds a given percent of the image area (as a result of the tests, the chosen percent was 2 °/∘), the program decides that there is a major modification. The computation of the median value, used for the filter as well as for the estimation of the reference image, is made with the **qsort** sorting algorithm.

Independent of the used algorithm, the program creates a log text file containing the hour when the modification has been detected as well as other useful information. If the saving the images option is checked, all the images that have been detected as modified are saved on the hard disk.

2.2.3 A presentation of the application

A dataflow chart of the program is presented in image 1.

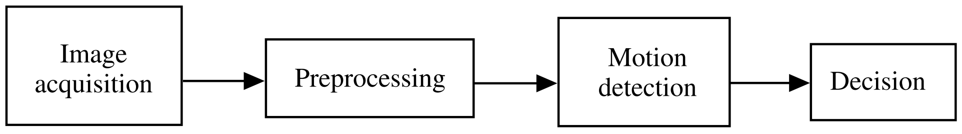

Figure 1. A dataflow chart of the program

Image acquisition

The acquisition of the images is made through a kernel module compatible with Video for Linux standard. The module makes a special file named /dev/video, making the interface between the hardware device and the software application. In order to gain access to this kind of file, the ioctl system call must be used.

Preprocessing and motion detection

For motion detection there were two algorithms implemented, both of them based on the calculation of a difference image. Both algorithms execute, with some modifications, the following steps:

1. estimate the reference image.
2. take the current image from a sequence of images.
3. computes a measure of the difference between the reference image and the current image.
4. if this measure exceeds a certain threshold, consider there is a major modification in the current image.
5. a re-estimation of the reference image is made after a number of processed images.
6. if there are more images to process, no error has occurred and the user did not stop the process, go to step 2.
7. else, free all the used resources and exit the program.

The differences between the algorithms appear on the estimation of the reference image and on the computation of the measure of the difference between the two images.

The first algorithm considers the first image from the sequence as the reference image and computes the measure of the difference as the standard deviation evaluated for the pixels from the histogram of the difference image. The histogram is represented by a table with 256 positions, on the i-th position being the number of pixels with the value i. The histogram is made by incrementing each position in the table for every pixel from the image with the value equal with the index in the histogram.

The computation of the difference image is very simple and consists in the evaluation of the absolute value of the difference between the pixels from the two images. The dispersion of a statistical sequence is defined as in equation (15):

$$\sigma^2 = \frac{\sum\limits_i (x_i - \overline{x})^2 \times y_i}{N}, \tag{15}$$

where $\overline{x}$ represents the mean value of the sequence and N is the total amount of the population. The computation of the dispersion can be made going through the histogram two times, once for computing the mean value and the other time for computing the deviation of each value from the mean value. Using an equivalent equation (2), only one lookup in the histogram is necessary.

$$\sigma^2 = \overline{x^2} - \overline{x}^2 \tag{16}$$

The second algorithm estimates the reference image (the background) as shown in equation (17)

$$B_{xy} = \text{ median } (I^1_{xy}, I^2_{xy}, ..., I^N_{xy}), \tag{17}$$

where I^i_{xy} represents the value of the pixels at coordinates x, y from the i-th image from the taken sequence.

Further on, in order to realize a segmentation of the zones that include changes from the image, a local threshold computation method is used, for each pixel separately. A threshold image is determined as a result of equation (18).

$$T_{xy} = k * \text{MAD}(I^1_{xy}, I^2_{xy}, ..., I^N_{xy}) \tag{18}$$

where MAD denotes the *median absolute deviation* and k is a coefficient of proportionality.

After the background and the threshold images are created, the algorithm computes, for the current image, a difference image as in the first algorithm. The difference image is used to obtain a binary image by applying the threshold image as in equation (19)

$$D_{xy} = \begin{cases} 1, \text{if} \left| I^i_{xy} - B_{xy} \right| > T_{xy} \\ 0, \text{otherwise} \end{cases} \tag{19}$$

For better results, isolated pixels are taken out from the binary image. This is achieved by applying a morphological closing followed by an erosion operation.

The next step in the segmentation of the difference image is connectivity analysis of nonzero pixels. This is done by a "fill in" algorithm, executed by a single passing through the binary image, generating a label for each different zone. This algorithm requires an equivalence list for the labels generated. The equivalence list has the property that, on the i-th position, it is recorded the value of the label equivalent with label i, or i, if there are no equivalences for i.

The "fill in" algorithm goes through the following steps:

1. The image is bordered with a black line on each side, since the border information is noisy.

2. For every pixel in the image, it is analysed the similarity with the left and upper neighbours.

   2.1 If the current pixel is similar to one of the neighbours, it is added to the same region (it receives the same label as the similar neighbor).

   2.2 If it is similar to both neighbours, but they have different labels, an equivalence is registered between the two labels in the equivalence list.

   2.3 If no similarity is found, the pixel receives a new label. Since no information on the possible size of the equivalence list is available at the beginning of the algorithm, the list grows dynamically.

3. In order to generate unique labels for each zone, a unification must be done having the following steps: each label is replaced with the last label from its equivalence list and different ascending labels are generated for each set of equivalent labels; finally, the regions are labeled according to the newly generated values. The measure for difference is the area of the biggest segmented zone. The computation of the area represents, in fact, counting the pixels from that zone.

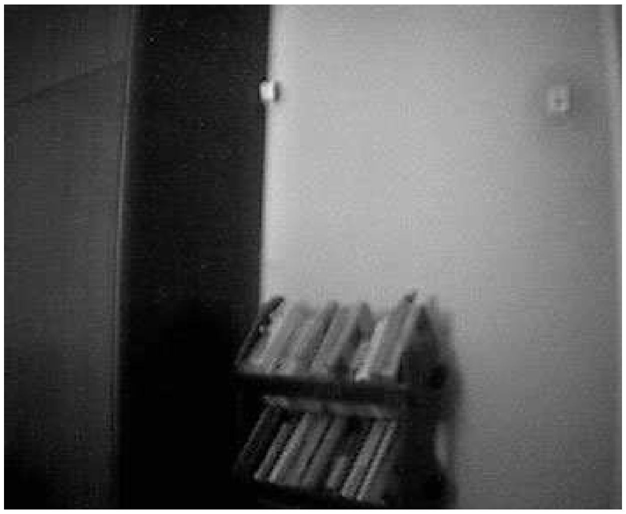

Figure 2. Background image

3 Results and conclusions

Picture 2 and picture 3 represent the background and the threshold images generated by the program.

When the program detects a modification in the image acquired from the video camera, the system alarms and writes a line in the log file and saves the image. An example is figure 4. As you can see, there is a hand in the upper left corner in the image.

In the log file there is the following line recorded:

```
Mon April 8 18:59:09 2002 area: 9663 > 1843 ./log/visitors/A000003.jpeg
```

There can be seen the information about the date and the hour the event happened, the number of pixels from the surface that is changed and the name of the file where the image was saved.

In the current version of the application only the detection algorithms are implemented. After a massive testing of the program, the methods proved to be stable and very fast. Therefore, the system is

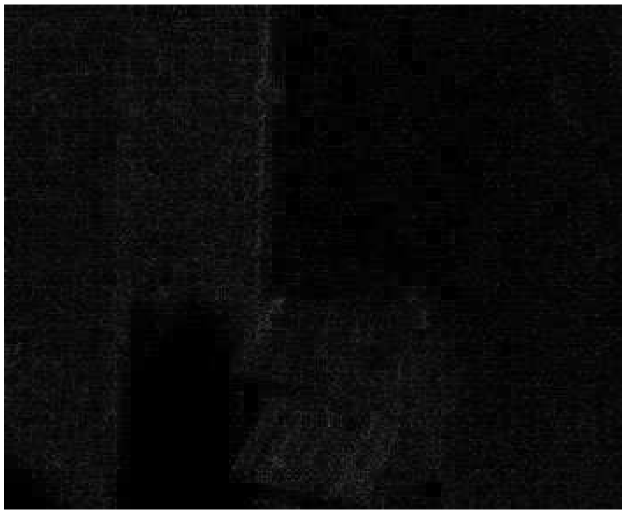

Figure 3. Threshold image

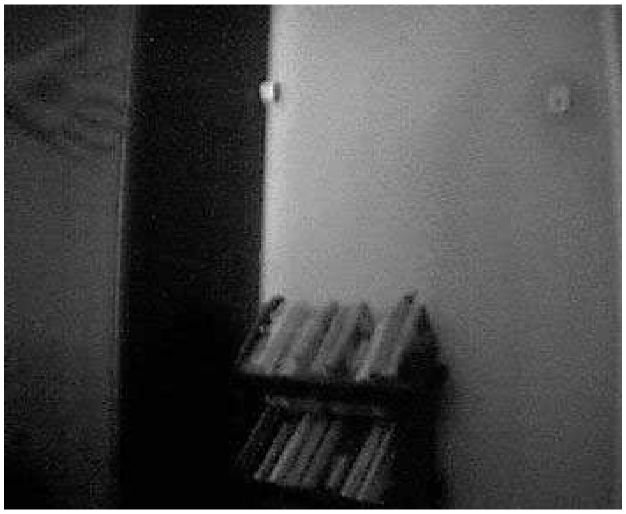

Figure 4. Image that determined the system to alarm

suitable for real time applications and the techniques used can be implemented on embedded devices.

Further development will cover object tracking and classification. For object tracking, an active contour model based system will be implemented, as explained in [13]. Further possibilities of improvement will be studied on the background estimation algorithm.

References

[1] H. Bässmann, P. Besslich: *Ad Oculos. Digital Image Processing.* Student Version 2.0, Thomson Publishing, 1995.

[2] F. Dufaux, F. Moscheni: *Segmentation-based motion estimation for second generation video coding techniques,* in L. Tomes and M. Kunt, editors, Video Coding: A second Generation Approach, pp. 219 – 263, Kluwer Academic Publisher, 1996.

[3] C. Diehl, M. Saptharishi, J. Hampshire, and P. Khosla: *Collaborative surveillance using both fixed and mobile unattended ground sensor platforms.* In SPIE Proceedings on Unattended Ground Sensor Technologies and Applications, vol. 3713, pp. 178–185, 1999.

[4] T.J. Elli, P.L. Rosin, P. Golton: *Model Based Vision for Automatic Alarm Interpretation,* IEEE AE System Magazine, March 1991.

[5] W.E.L. Grimson, L. Lee, R. Romano, C. Stauffer: *Using adaptive tracking to classify and monitor activities in a site.* In CVPR98, pp. 22–31, (1998).

[6] I. Haritaoglu, D. Harwood, and L. Davis: *W 4 : Who? When? Where? What? A real time system for detecting and tracking people.* In IEEE International Conference on Automatic Face and Gesture Recognition, pp. 222–227, 1998.

[7] T. Huang, G. Osawara, S. Russell: *Symbolic Traffic Scene Analysis, Using Dynamic Belief Networks,* AAAI Workshop on AI in IVHS, Washington D.C., 1993.

[8] B. Jähne: *Digital Image Processing. Concepts, Algorithms an Scientific Applications,* Springer-Verlag, 1991.

[9] A.K. Jain: *Fundamentals of Digital Image Processing,* Prentice Hall, 1989.

[10] T. Kailath, A.H. Sayed, B. Hassibi: *Linear Estimation. Upper Saddle River,* NJ USA, Prentice Hall, (2000).

[11] Klaus-Peter Karmann, Achim von Brandt: *Moving Object Recognition using Adaptive Background Memory,* in V. Capellini (ed.) Time-Varying Image Processing and Moving Object Recognition, 2, Elsevier, Amsterdam (1990).

[12] M. Kass, A. Witkin, Terzopoulos, D. Snakes: *Active Contour Models,* International Journal of Computer Vision 1, (1988).

[13] D. Kollar, J. Weber, J. Malik: *Robust Multiple Car Tracking with Occlusion Reasoning,* in Proc. Third European Conference on Computer Vision, Stockholm, Sweden, May 2 - 6, 1994. Eklundh J. - O. (ed.), Lecture Notes in Computer Science, Springer - Verlag, Berlin, Heidelberg, New York (1994).

[14] R.J. Schalkoff: *Digital Image Processing and Computer Vision,* Wiley, 1989.

[15] G. Welch, G. Bishop, L. Vicci, S. Brumback, K. Keller, D. Colucci: *High-Performance Wide-Area Optical Tracking The HiBall Tracking System Presence,* Teleoperators and Virtual Environments, 10 (2001).

[16] G. Welch, G. Bishop: *SCAAT: Incremental Tracking with Incomplete Information.* In T. Whitted (Ed.), Computer Graphics (SIGGRAPH 97 Conference Proceedings ed., pp.333-344). Los Angeles, CA, USA (August 3-8): ACM Press, Addison-Wesley (1997).

Ennio Cortellini,

Received May 27, 2003

MET Department, University of Teramo,
Italy
E–mail: *ennio.cortellini@inwind.it*