Automatic Control of Video Surveillance Camera Sabotage

Conference Paper · June 2007

DOI: 10.1007/978-3-540-73055-2_24 · Source: DBLP

Pedro Gil-Jiménez
University of Alcalá
49 PUBLICATIONS **1,162** CITATIONS
SEE PROFILE

Philip Siegmann
University of Alcalá
46 PUBLICATIONS **418** CITATIONS
SEE PROFILE

Javier Acevedo
University of Alcalá
39 PUBLICATIONS **516** CITATIONS
SEE PROFILE

Automatic Control of Video Surveillance Camera Sabotage

P. Gil-Jiménez, R. López-Sastre, P. Siegmann,
J. Acevedo-Rodríguez, and S. Maldonado-Bascón

Dpto. de Teoría de la Senal y Comunicaciones
Universidad de Alcalá, Alcalá de Henares, 28805 (Madrid), Spain
{pedro.gil,robertoj.lopez,philip.siegmann,
javier.acevedo,saturnino.maldonado}@uah.es

Abstract. One of the main characteristics of a video surveillance system is its reliability. To this end, it is needed that the images captured by the videocameras are an accurate representation of the scene. Unfortunately, some activities can make the proper operation of the cameras fail, distorting in some way the images which are going to be processed. When these activities are voluntary, they are usually called sabotage, which include partial o total occlusion of the lens, image defocus or change of the field of view.

In this paper, we will analyze the different kinds of sabotage that could be done to a video surveillance system, and some algorithms to detect these inconveniences will be developed. The experimental results show good performance in the detection of sabotage situations, while keeping a very low false alarm probability.

1 Introduction

Automatic video surveillance systems are typically used to monitor particular places, trying to detect suspicious activities such as vandalism or intrusion [1,2,3,4,5]. These kinds of systems are normally composed of a number of videocameras and an image processing block, which is generally implemented over a computer or specialized hardware. The robustness of a video surveillance system depends heavily on both the image processing algorithms and the quality of the images received.

When it comes to the video quality, some extern parameters can influence it, such as illumination, weather conditions, or voluntary actions. In systems where these are crucial aspects, some dispositions must be taken to prevent, or at least, detect these situations. In this work, we focus in voluntary actions, which are typically called sabotage, and some algorithms have been designed to detect them. There are not many works in the literature that deal with this problem. In [6], for instance, a camera dysfunction detector is implemented, designed mainly for surveillance systems inside vehicles.

Figure 1 shows the block diagram of the system implemented. Essentially, the system is composed of a videocamera, where the images captured are processed by a group of image processing algorithms, such as motion detection, or object tracking, running over a PC. To perform the sabotage detection, another block has been added, which analyzes the images before the video surveillance block does. If no sabotage is detected, the system works as usual. However, if the block detects any kind of sabotage, an alarm is triggered to warn the surveillance staff. In this state, it makes no sense the operation of the video surveillance block, so it can be deactivated until the normal state is reached again.

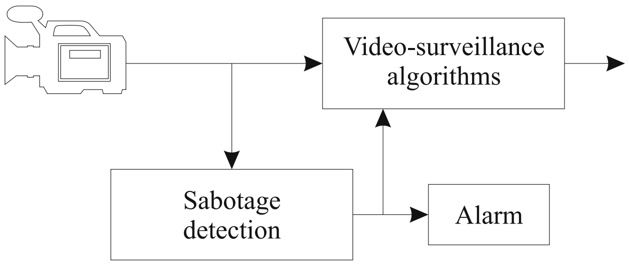


Fig. 1. Block diagram of a video surveillance system with sabotage detection

2 Camera Sabotage

In this paper, we will consider sabotage as any voluntary action on a videocamera that changes the physical parameters of the camera, affecting, therefore, the images received and processed by the video surveillance algorithms. These actions include:

- **Image occlusion:** The lens of the videocamera can be occluded with an opaque object, preventing the system to analyze the surveyed scene.
- **Image defocus:** The focus setting of a videocamera can be altered, changing the image captured, specially the edges of the objects in the scene.
- **Change of the field of view:** The field of view can be changed easily displacing the camera, in such a way that the new image is partially or totally distinct to the previous one.

In Fig. 2 we can see some examples. Figure 2(a) shows an image of a scene without any kind of sabotage. In Fig. 2(b), the lens has been partially occluded with a blank sheet of paper, hiding part of the scene to the camera. In Fig. 2(c), the lens setting of the camera has been changed to defocus the image, distorting specially the edges of the objects. Finally, in Fig. 2(d) the camera has been shifted to the right, to partially change the field of view. In any of these situations, common video surveillance algorithms, such as background subtraction, or background modeling, would not be able to operate correctly, and so, the output of the system would no longer be valid. Although the video surveillance system will not be able to solve this problem by itself, some algorithms can be designed to detect these situations and alert the surveillance staff, who can finally analyze the situation, and perform the corresponding actions.

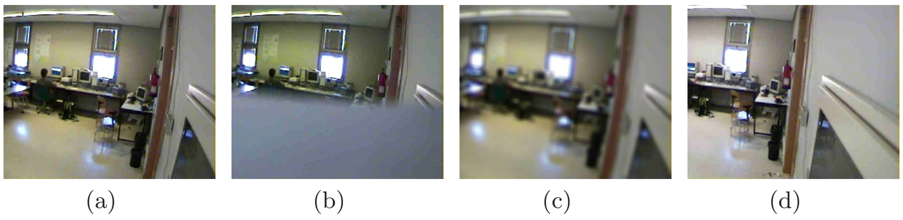


Fig. 2. Some examples of video camera sabotage. (a) No-sabotaged image. (b) Partial occlusion example. (c) Defocus example. (d) Displacement example.

3 Sabotage Detection

The algorithm which has to be designed for the detection of sabotage activities must fulfill two different constrains. Firstly, it must be robust enough to ensure a high detection probability, while maintaining the false alarm probability as low as possible. Secondly, the algorithm should be computationally simple to carry out with the real time requirements.

We have to consider also the scenario where the system is working. As in many other video surveillance algorithms, the fundamental concept is the comparison between the current frame and some model built from previous images, which is usually called the background model. In the sabotage case, two different situations must be distinguished. On the one hand, when the cameras are not altered, normal motion of objects or people in the scene must not be considered as sabotage, as it must be interpreted by the video surveillance algorithms themselves. On the other hand, voluntary alterations of the parameters of the cameras do must be considered as sabotage. Although both situations are completely different to each other, from the algorithm's point of view they can be quite similar, since both are reflected as a change between the current frame and the background model.

To allow the anti-sabotage system to discern between each situation, we compute a model of the background using a supervised sequence of the scene where no sabotage exists. Although many techniques have been proposed for the computation of background models which comprise the whole image, in this work we use the edges of the scene instead of the whole image. This yields us two advantages. Edges are more robust against illumination changes than smooth parts of the image when comparing a frame with the computed model of the background. But also, by using only the edges in the comparisons to perform the sabotage detection, a smaller number of operations are done by the algorithm, reducing the computational complexity, and so, the time required to perform the sabotage detection.

3.1 Background Model

As it has been explained above, the background model only includes the information at the edges of the objects belonging to the background. The background model is a binary matrix of the same size than the original images, where pixels set to true belong to the edges of the background, and vice versa. To ensure that an edge belongs to the background, and not to a moving object, the edges are computed for M consecutive frames,

$$B_n = \sum_{i=nM}^{(n+1)M-1} D_i \ , \tag{1}$$

where D_i is the edge matrix computed for frame i, whose pixels take the value 1 if it belongs to an image edge, and 0 otherwise. Furthermore, to allow the system for slow changes of the background, the matrix B is computed for each M new frames, as it is indicated in (1), and the background model updated according to:

$$P_n = \alpha P_{n-1} + (1-\alpha) B_n \ , \tag{2}$$

where α is a parameter between 0 and 1 that establishes the influence of the new frames with respect to the last background model. Finally, only those pixels which have been computed as edges for an enough number of times (in our case we chose $P_n \geq M/2$) are considered to belong to the background model.

In Fig. 3 we can see an example of this computation. Figure 3(a) shows a frame of the sequence recorded, while from (b) to (d) it is shown the background model state for $M = 30$ and $\alpha = 0.3$ after 30 ($n = 1$ iteration), 60 ($n = 2$) and 120 ($n = 4$) frames respectively. It is obvious that the model becomes more stable as the number of frames used in the computation increases, and after about 3 iterations the model remains almost unchanged as long as the background remains stationary. From now on, we will use this model with the above mentioned settings to perform the sabotage detections, as it is explained in the following sections.

3.2 Occlusion Detection

When an object covers partly or totally the lens of the camera, part of the image is no longer visible, and a number of pixels, or even all the pixels, of the background model disappear as well. This occlusion must be differentiate from partial hidings due to moving objects. To this end, the entropy of the pixels belonging to the background model is computed:

$$E = -\sum_k p_k \cdot \ln\left(p_k\right) \ , \tag{3}$$

where p_k is the probability of appearance of the gray level k in the image. Since the model is not altered until new M consecutive frames arrive, causing its update, this computation is performed only once each M frames. For each new frame, the same entropy is calculated again. Let suppose now that an object is placed near the lens, so that we can occlude part of the image. In this situation, a loss of information will occur in the zone of the image that has been occluded, resulting in a decrease of its entropy.

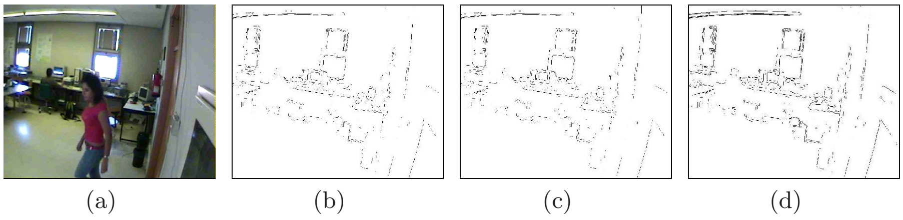


Fig. 3. Computation of the background model. (a): An image of the original sequence. (b-d) Background model after (b) 30 frames (1 iteration), (c) 60 frames (2 iterations), (d) 120 frames (4 iterations).

Although this would be enough in the case of a total occlusion for the detection of the sabotage, in partial occlusions this effect can be masked by the entropy of the rest of the image, which probably remains unchanged. To allow the system to detect partial occlusion as well as total ones, the image is divided into several sub-images, or blocks, of equal size, and the process is performed over each sub-image independently. A sabotage will be detected when the current entropy for at least one of these blocks is lower than a given threshold.

3.3 Defocus Detection

The defocus of the lens of a camera implies the degradation of the edges, so that most of them disappear from the image [7]. Therefore, we can easily count the number of pixels in the background model and, for each new frame, the same parameter can be computed. If the camera has been defocused, this number for the current frame will be much lower than that of the background model, and this reduction can be used to detect that the camera is not correctly focused.

3.4 Displacement Detection

The detection of shifts on the field of view is computed using the zero-mean normalized cross correlation (*ZNCC*), as in classical block matching algorithms, but in this case, to speed up the algorithm, only for the pixels of the background model:

$$ZNCC_i\left(m,n\right) = \frac{\sum\limits_{x,y}\left(I_{i-1}\left(x,y\right) - \mu_{I_{i-1}}\right)\left(I_i\left(x+m,y+n\right) - \mu_{I_i}\right)}{\sigma_{I_{i-1}}\sigma_{I_i}}\,, \qquad (4)$$

where I is the image captured by the camera. In the computation, only the pixels belonging to the background model are considered. A block matching algorithm returns two parameters [8], the optimal value in both axes (m,n) of the translation between the current frame and the background model, and the value of the *ZNCC* for that displacement, and both parameters are used to detect

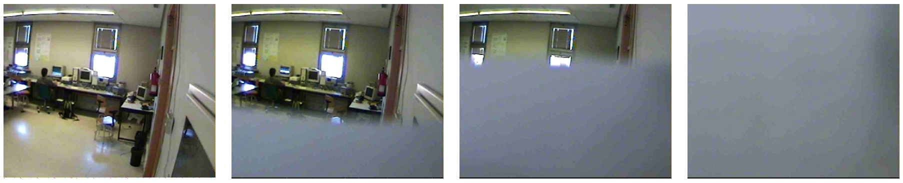

(a) (b) (c) (d)

Fig. 4. Different frames of the occlusion example

a possible sabotage. The second parameter can be used to estimate the reliability of the matching. If this value is lower than a given threshold, we interpret that as an error in the matching, which can be caused for many reasons, such as a partial occlusion, too large a displacement, noise, etc. In this case, the algorithm waits for the next frames to come, to see if the problem still exists.

The first parameter must be analyzed more carefully. A small displacement, of the order of about 5 pixels, could be caused by vibrations or blows, and should not be considered as sabotage. Bigger displacements however do must be considered as sabotage. A problem can arise if the camera is displaced slowly so that the system could consider it as a consequence of vibrations or noise for each frame, and no sabotage will be detected. To solve this inconvenience, the system computes both the relative displacement and the accumulated displacement, and whenever one parameter surpasses the threshold a sabotage is detected.

4 Experimental Results

The proposed system has been implemented using C++ language as an optional block to the video surveillance system our group is currently developing, so that the changes in the original system needed to incorporate the algorithms described in this paper are minimal. The system processes 5 gray level images per second, running in real time over a PC.

The system has been tested in indoor and outdoor environments, with images that include fast (lights on and off) and gradual (different hours of the day) illumination changes, and background motion (specially people and cars). During the recording of the sequence, the camera has been intentionally sabotaged to test the response of the system. These actions include the three kinds of sabotage analyzed in this work, as has been described previously.

Figure 4 shows some images of an occlusion example. These images have been extracted from a longer sequence, which comprises enough time for the background model to stabilize, and a few more frames to register the sabotage action. In Fig. 5 the evolution of the entropy for the last 8 frames of the sequence is displayed. As it has been described before, the whole image has been divided into 9 blocks, and the 9 graphics of Fig. 5 correspond with these 9 blocks. In the graphics it is also marked in dotted lines the reference entropy computed from

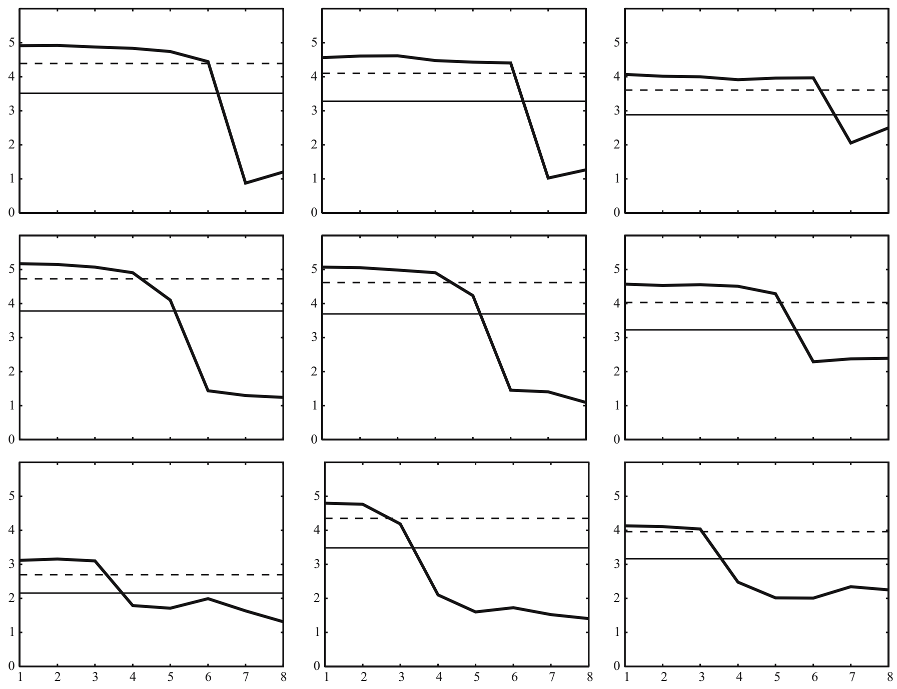


Fig. 5. Occlusion detection for the 9 blocks of the image. Horizontal axis shows the number of sequence from 1 to 8. Vertical axis measures the entropy of the corresponding frame and block.

the background model, and in solid lines the threshold for each block, which is computed as 0.8 times the reference entropy. From a look at this figure, it can be seen that, between frames 3 to 7 the entropy of all blocks go below its corresponding threshold, and so, an occlusion has been detected. We can also see that, since the lens of the camera has been occluded from bottom to top, that is, the sheet of paper have been moved upwards, the blocks in the bottom line reach the threshold, and detect the occlusion, before the top line's do.

Some frames of a defocus sabotage example is shown in Fig. 6. Again, these images correspond to a sequence longer enough to allow the background to stabilize. For each sub-image, the original frame is displayed at the left, and its corresponding edge mask at its right. The results of the defocus sabotage detector are displayed in Fig. 8(a), along with the reference in dotted line, which is the number of pixels that compose the background model, and the threshold in solid line, which is computed as 0.4 times the reference described above. In this case, the threshold is reached at frame 7, and at this point, a defocus sabotage is then detected.

Finally, and example of a displacement sabotage can be seen in Fig. 7, and the results of the sabotage detector in Fig. 8(b). The measured parameter is the value of the normalized correlation. While there is no camera motion, this

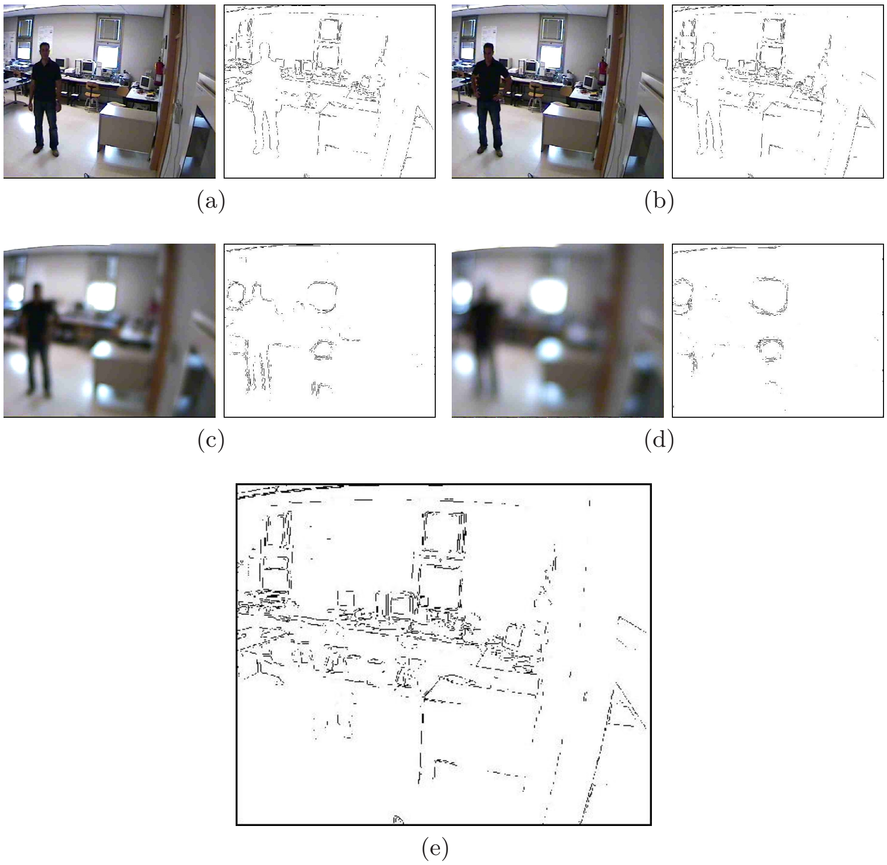


Fig. 6. (a-d) Different frames of the defocus example. For each sub-image the original frame is displayed on the left, and the computed edge mask on the right. (e) Background model.

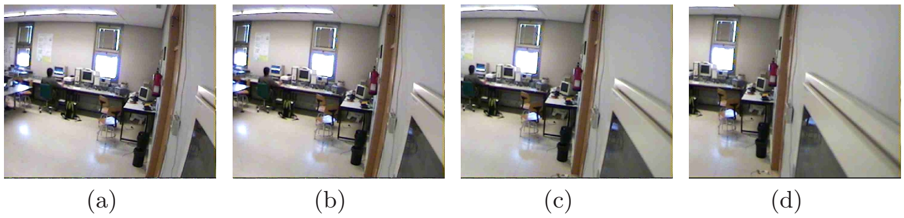
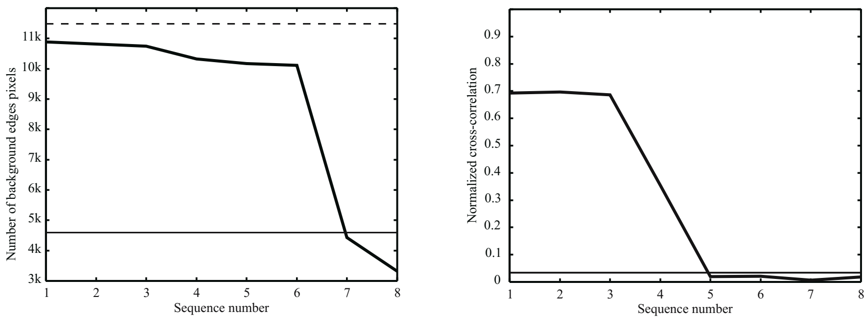


Fig. 7. Different frames of the displacement example. (a) Original position of the camera. (b-d) Successive displacement of the camera.



(a) Defocus detection. (b) Displacement detection.

Fig. 8. Defocus and displacement detection examples

parameter takes values around 0.7, which is mainly due to camera noise and vibrations. However, when the camera in shifted to change its point of view, the correlation decreases to values under 0.01. The threshold for the detection of camera displacements has been set to 0.03. This value has been drawn in Fig. 8(b), that allows us to see that, in this case, the displacement sabotage is detected in frame 5.

5 Conclusion

This paper describes the main kinds of sabotage that could be done to a video surveillance system, and how this is reflected into the images captured, and to what extent could the operation of a video surveillance system be affected. The paper also proposes some algorithms to trigger an alarm when a sabotage has been detected. The algorithms fulfill the typical video surveillance constrains, that is, robustness to ensure a high detection and low false alarm probabilities, and simplicity to accomplish with the real time requirements.

The future work is basically related with the testing of the system on more complex sequences, such as adverse outdoor conditions (night, rain, fog or wind among others), crowed scenarios, etc. Although the basic idea of the system is to keep it as simple as possible to reduce the computation time, more complex algorithms could be designed whenever additional hardware is available to go on maintaining the real time requirements.

Acknowledgments

This work was supported by the project of the Ministerio de Educación y Ciencia de España number TEC2004/03511/TCM.

References

1. Foresti, G.L., Mhnen, P., Regazzoni, C.S.: Multimedia video-based surveillance systems. Requirements, issues and solutions. 1 edn. Kluwer Academic Publishers, 101 Philip Drive, Assinippi Park, Norwell, Massachusetts 02061 USA (2000)
2. Di Stefano, L., Neri, G., Viarani, E.: Analysis of pixel-level algorithms for video surveillance applications". In: CIAP01. (2001) 541–546
3. Toyama, K., Krumm, J., Brumitt, B., Meyers, B.: Wallflower: Principles and practice of background maintenance. In: ICCV (1). (1999) 255–261
4. Boult, T., Micheals, R., Gao, X., Eckmann, M.: Into the woods: Visual surveillance of noncooperative and camouflaged targets in complex outdoor settings. In: Proceedings of the IEEE. Volume 89. (2001) 1382–1402
5. Dawson-Howe, K.: Active surveillance using dynamic background subtraction. Technical Report TCD-CS-96-06, Dept of Computer Science, Trinity College, Dublin, Ireland (1996)
6. Harasse, S., Bonnaud, L., Caplier, A., Desvignes, M.: Automated camera dysfunctions detection. In: Image Analysis and Interpretation, IEEE (2004) 36 – 40
7. Pentland, A.: A new sense for depth of field. IEEE Transactions on Pattern Analysis and Machine Intelligence **9** (1987) 523–531
8. Roma, N., Santos-Victor, J., Tom, J.: A comprative analysis of cross-correlation matching algorithms using a pyramidal resolution approach. World Scientific (2002)