MOTION DETECTION AND SURVEILLANCE

Dominic Mircea Kristaly, Delia Ungureanu, Sorin-Aurel Moraru

Automatics Department, "Transilvania" University of Brasov, M.Viteazu Street, no.5, 500174, Brasov, Romania, phone/fax: +40 0268 418836,
kdominic@vision-systems.ro, delia@deltanet.ro, smoraru@vision-systems.ro

This paper is set to present a method of detecting motion that offers the possibility of creating more complex system for surveillance using image capture devices such as webcams, video cameras and others.

A video surveillance system can start recording the images on a magnetic support, call the authorities or sound an alarm in case of detection of motion.

Making use of an Internet connection, video surveillance software can broadcast the captured images and can be administrated remotely.

Keywords: motion, detection, surveillance, histogram, video

1. INTRODUCTION

This paper is set to present a method of detecting motion that offers the possibility of creating more complex system for surveillance using image capture devices such as webcams, video cameras and others.

The hardware and software elements needed for the most simple video surveillance system are: a webcam connected to a computer (through any serial interface – for example USB) and a program that reads the video stream and finds the difference between consecutive frames, therefore detecting motion. The block diagram is shown in figure 1.

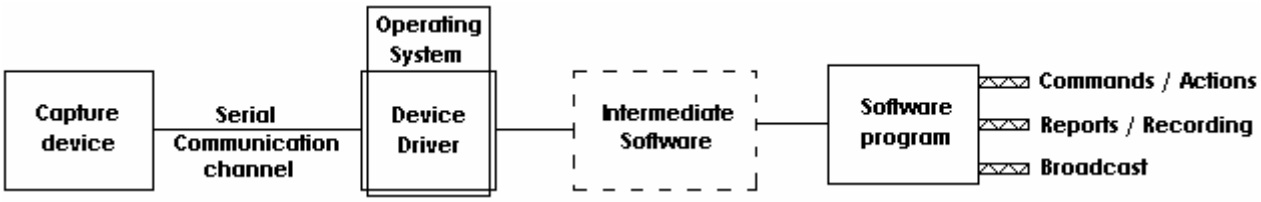

Fig. 1. Block diagram of a generic video surveillance system

The *capture device* provides the raw material – a bi-dimensional matrix of colored dots (pixels) that models the real image. Each pixel in the matrix is represented by three numbers that are the levels of intensity for the colors red, green and blue (RGB), from which any color can be obtained. In most cases, the values can range between 0, for non-presence, and 255, for the maximum intensity. Not all the devices transmit the video stream in the RGB format; some use the YUV coding (obtained from the RGB space through a linear transformation).

The *serial communication* with the computer it can be assured by an USB or firewire (*i.Link* or *IEEE 1394*) cable.

The *device driver* has the role of translator between the hardware subsystem and

the software subsystem. It collaborates with the operating system installed on the computer.

The *intermediate software* it's a supplementary layer of the architecture: it can be a protocol (for example *TWAIN* or *WIA*) or a piece of software that ease the access to the capturing device (Microsoft's *DirectX* or Sun's *Java Media Framework*). If the *DirectX* is used, then the programming can resort to the *DirectX SDK*; if the *JMF* is preferred, the *JMF API* it's the best choice.

The *software program* receives the images, compares it with the previous one (if exists) then stores the image in a buffer (becoming the previous image for the next one); it makes decision, based on the comparison's result, computing outputs (actions, reports, etc.) and can broadcast the stream over a network. For example, if a change is detected, the program can start recording the images on a magnetic support, call the authorities or sound an alarm.

This paper will concentrate on a possible algorithm that can be used by video surveillance software to detect motion.

2. MOTION DETECTION ALGORITHM

The described algorithm takes as input the source image in RGB format. The output will consist in a number of pairs of coordinates that defines rectangular regions, indicating the zones from the input image that had changed from the previous frame. The block diagram is shown in the figure 2.

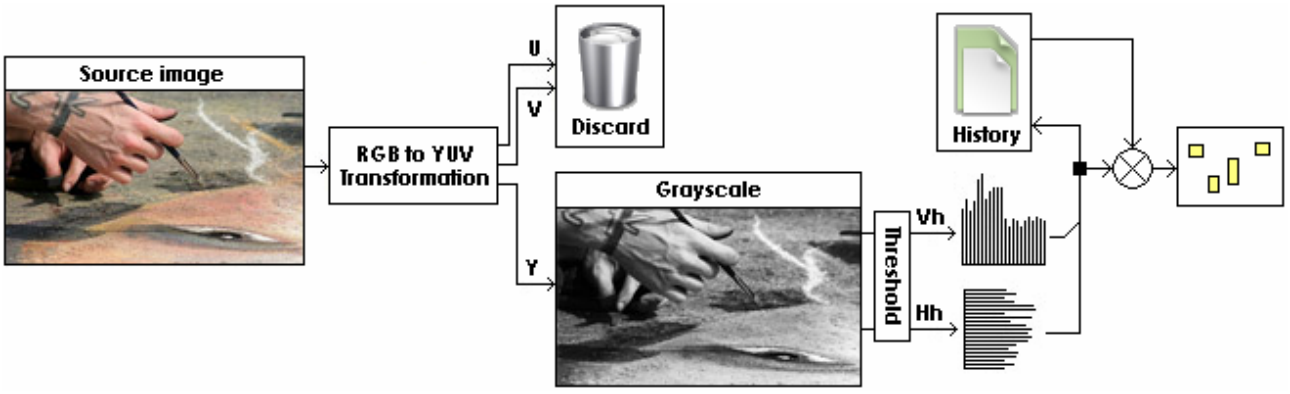

Fig. 2. Block diagram of the algorithm

The algorithm can take as input a YUV image, case in which the RGB to YUV transformation it's not longer needed.

From the YUV image the algorithm discards the color information (chrominance) obtaining the grayscale version (Y component). Based on this version and according to a threshold for light intensity, the image is transformed in a bi-dimensional matrix composed only of 0 and 1. Summing up the 1 or 0 values on columns and rows there are obtained two *spatial histograms*, *Hh* and *Vh* that are compared to the ones obtained from the previous image and then stored to be compared with the histograms of the next captured frame.

Any modification in a new frame will result in some changes between the previous and current spatial histograms (more or less 1/0 values). The registered

modifications permit to approximate the position of the moving object and its velocity. The movements can be recorded to a file or stream.

The algorithm can be modified, for example, to process the grayscale image using a spatial filter such Gray Pyramid to discard any modification that are not relevant (movement of small objects or radio interference)[1].

This version of the algorithm needs a constant intensity of the light to work properly, but in nature this cannot be achieved easily. Adding a supplementary layer of calculation can solve, to some extent, this problem.

In order to reduce the differences in brightness and contrast between the consecutive frames (modifications produced by changing light), the source image can be filtered using a homogeneous point operation – the uniform non-adaptive histogram equalization (UNAHE) that outlines the details in the image.

2.1. Transformations between color spaces

As mentioned in section 1, the images can be represented using the RGB or the YUV color space (Y is called luminance and U and V – chrominance). The luminance (Y component) contains the details of the image in shades of gray; the chrominance (U and V components) contains the information about the colors. The transformation between the two spaces is given by the equations 1 and 2.

$$\begin{bmatrix} Y \\ U \\ V \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ -0.147 & -0.289 & 0.436 \\ 0.615 & -0.515 & -0.100 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} \tag{1}$$

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1 & -0.000039457070707 & 1.139827967171717 \\ 1 & -0.394610164141414 & -0.580500315656566 \\ 1 & 2.031999684343434 & -0.000481376262626 \end{bmatrix} \begin{bmatrix} Y \\ U \\ V \end{bmatrix} \tag{2}$$

In software implementation is better to use a formula that uses only integers and fixed-point operations, to speed-up the calculation. The two transformations above can be written as:

$$\begin{aligned} Y &= \min(|2104R + 4130G + 802B + 4096 + 131072|) >> 13, 235) \\ U &= \min(|-1214R - 2384G + 3598B + 4096 + 1048576|) >> 13, 240) \\ V &= \min(|3598R - 3013R - 585B + 4096 + 1048576|) >> 13, 240) \end{aligned} \tag{3}$$

$$\begin{aligned} C &= Y - 16 \\ D &= U - 128 \\ E &= V - 128 \end{aligned} \tag{4a}$$

$$\begin{aligned} R &= ((298C + 409E + 128) >> 8) \% 256 \\ G &= ((298C - 100D - 208E + 128) >> 8) \% 256 \\ B &= ((298C + 516D + 128) >> 8) \% 256 \end{aligned} \tag{4b}$$

where *min*(*a*,*b*) is a function that returns the minimum value between *a* and *b* , |*a*| returns the absolute value for *a*, *a>>b* means *a* shifted to the right with *b* positions (refers to the binary representation of value *a*), *%* is the symbol for modulus operation (used, in this case, to clip the values into the range 0 to 255).

3. IMPLEMENTATION

The algorithm presented in this paper can be implemented in various ways, and included in any video surveillance software.

For example, a video surveillance system can be implemented as a Java servlet using JMF API (included in a web application). The administration of the system could be done through the Internet (from a JSP – Java Server Pages – site). Also, the captured images can be broadcast with the help of a Java Applet. In this way, the surveillance system will be just a click away.

4. CONCLUSIONS

Video surveillance is a very modern way to secure a perimeter. The cost of such a system can vary from very little to very expensive, making it suitable for a wide range of clients.

Combined with a fuzzy logic engine or an artificial intelligence network the system could detect shapes and object and could predict movements.

5. REFERENCE

[1] Lyon D.A., *Image Processing in Java*, Prentice Hall PTR, 1999.
[2] Pratt W.K., *Digital Image Processing: PIKS Inside*, 3rd Edition, John Wiley & Sons, 2001.
[3] Gonzales R.C., R. Woods, *Digital Image Processing*, Prentice Hall PTR, 2002.
[4] Baciu R., D. Volovici, *Graphic systems*, Ed. Albastra, 1999.
[5] Kristály D.M., *Morgana – Implementing steganography*, Scientific communication session for students, May 2004.