New Generation Video Surveillance System Based on Image Processing and FPGA Processor towards Autonomous System

A. Benkhalil, S. Ipson and W. Booth
Department of Electronic & Electrical Engineering
University of Bradford, Bradford, West Yorkshire, DB8 9EU, UK
akbenkha@bradford.ac.uk, *+44-1274-234075*

A stand-alone two-camera automatic security system is described. The system utilises a single Field Programmable Gate Array (FPGA) chip which performs a variety of video-rate arithmetic and logical operations on images, grabs and stores image frames and generates all necessary control signals. Results illustrating the highly successful operation of the system are included.

1. INTRODUCTION

Modern digital computing, semiconductor technologies and communication technologies have introduced a complete change of perspective in the design of the architectures of surveillance systems. Recently, several research groups have devoted a great deal of effort to devising prototype video surveillance systems based on robust methods of dynamic scene interpretation. Most of this work relies on complex and expensive hardware such as multiple DSP boards or on high-speed host computer platforms in order to achieve near real-time performance[1], [2], [3].

The research reported in this paper was directed at developing an image processing based surveillance system that does not need operator intervention, or a host computer or expensive hardware. The major aim of the research was to develop a stand-alone automatic security camera system (ASCS) capable of grabbing Comité Consultatif International des Radiocommunications (CCIR) format images, detecting motion and providing high resolution images of moving objects. A further aim was to take advantage of the high logic density of recently introduced FPGAs to implement motion detection and tracking algorithms and control functions in a single chip [4].

The paper is organized as follows. In section 2 an overview of the system is presented. In section 3 algorithms developed to detect motion, to update reference images and to carry out adaptive thresholding are described. Section 4 deals with the functionality of the FPGA including the implementation of the algorithm used for target classification. Section 5 presents some results to illustrate the performance of the system under different operating conditions and finally section 6 draws conclusions and indicates possible future developments.

2. SYSTEM OVERVIEW

A block diagram of our prototype ASCS, which employs two cameras working in a master slave combination, is shown in Fig. 1. The master camera points in a fixed direction and incorporates a 16mm lens to provide a field of view 22° wide in the horizontal direction. It provides greyscale images that are analyzed by the FPGA device every 40ms in order to detect target motion. The slave camera, which is mounted on a robotic arm for mobility, has a 150mm focal length lens to provide a high-resolution view of the target. It is moved to point at a target by the control logic hardware on the basis of information provided by the master camera. In the system currently implemented, the slave camera is driven to view one of the 100 sub-regions into which the field of view of the master camera is divided. The direction chosen is the one that is closest to the direction of the detected target.

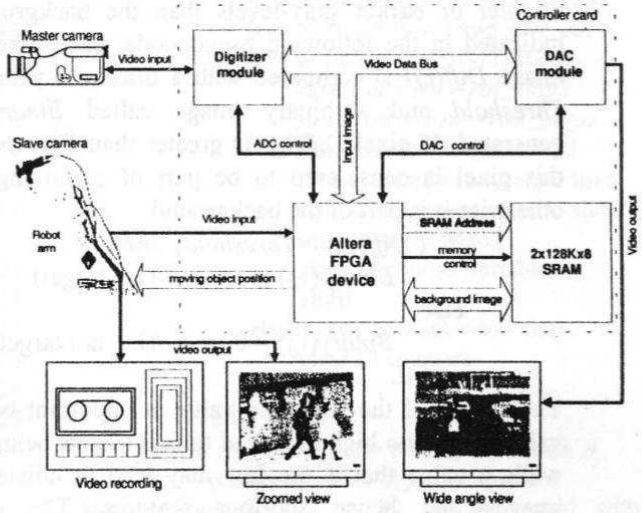

Fig. 1. Block diagram of the ASCS

The FPGA device shown in Fig. 1 is responsible for controlling the digitization and display of each incoming live-video image and the storage of a continually updated background image in the external SRAM. It also performs a variety of video-rate arithmetic and logic operations associated with moving target detection prior to transferring the co-ordinates of a detected target to the slave camera via the robot arm. A detailed description of the functionality of the FPGA is presented later in section 4. The digitizer module shown in Fig. 1 was implemented using a conventional flash Analogue-to-

Digital Converter (ADC) for digitization, a sync separator IC for timing control and a PLL (Phase Locked Loop) device to produce the system's 10MHz global clock from the 15.625 kHz CCIR line frequency signal. The Digital-to-Analogue Converter (DAC) module shown in Fig.1 uses an off-the-shelf DAC integrated circuit that automatically generates all the pedestal signals necessary to produce a standard video signal (CCIR). The system is designed to grab a stream of gray scale images with a resolution of 512(H) × 512(V) × 8 bits.

3. MOTION DETECTION

Since the system was intended to operate at video rates, a simple but efficient motion detection algorithm was chosen for implementation in the FPGA:

$$Diff(i,j) = \left| Live\_im(i,j) - Bg(i,j) \right| \quad (1)$$

where $Live\_im\{(i,j) \in R\}$ is the current image, $Bg\{(i,j) \in R\}$ is the background image and R={(i,j): i=1...M (512), j=1...N (512)}. This technique of subtracting the background image *Bg(i,j)* from the input image *I(i,j)* to produce the difference image *Diff(i,j)*. is called background differencing and it still forms the cornerstone of many of today's more complex systems [1], [2], [5]. The background image should, ideally, not contain moving objects and the system can use either a stored single live image or, more usefully, a continually updated stored image that is computed using a temporal filter as described at the end of this section. The absolute value of *Diff(i,j)* is used in order to detect moving objects which may have brighter or darker gray-levels than the background. As indicated in the following pseudocode, each pixel of the image *Diff(i,j)* is compared with a threshold value called *Threshold* and a binary image called *Binary(i,j)* is generated. If pixel *Diff(i,j)* is greater than *Threshold* then this pixel is considered to be part of a moving object, otherwise it is part of the background.

```
if ( Diff(i,j) > Threshold) then
        Binary(i,j) = 1        //(i.e. target)
else
        Binary(i,j) = 0        //(i.e. not target)
end if.
```

The choice of the threshold value is important because a value that is too high may lead to real targets being missed while a value that is too low may lead to noisier binary images and hence spurious features. The optimum threshold value also varies with the instantaneous illumination level and a suitable compromise has been achieved by continuously adapting the threshold value

$$Threshold = f \times \sum_{k=1}^{m \times n} \frac{Live\_im\ (k)}{(m \times n)} \quad (2)$$

Here, $m \times n$ is the total number of pixels and f is a multiplying factor whose value can be set by the user. A value of f = 1 was found to be appropriate for the surveillance scenes used for testing so the threshold values actually used in these cases were the average gray levels of the live images. Fig. 2 demonstrates the detection of moving features in a sequence of images captured using the master camera.

The background image is continually updated using the temporal low-pass filter [5] defined by:

$$Bg_{k+1}(i,j) = Bg_k(i,j) + G \times \left(Live\_im_k(i,j) - Bg_k(i,j) \right) \quad (3)$$

Here k labels the current frame and G is a gain factor defined by the user. This feature causes the system to adjust the background automatically to take account of changes in the scene caused, for example, by changes in the illumination or by an object moving into the scene but then stopping. In either case the change of scene will become part of the updated background within a time-scale related to the value of G. Image sequence simulations carried out using MATLAB indicated that G = 0.125 is an appropriate value to choose. Then the background image takes about 20 frame periods to adjust to a step change in the scene, which enables new static features to become part of the background in a reasonably short time whilst ensuring that no moving objects are added to the background.

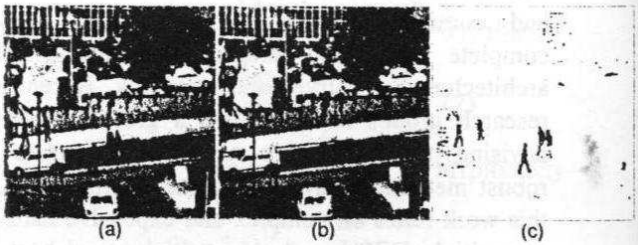

Fig. 2. The detection of moving objects. Here (a) is one of the images in a sequence of images, (b) is the current background image and (c) is the binary difference image after applying the current threshold.

4. FUNCTIONALITY OF THE FPGA

One FPGA chip forms the heart of the single board surveillance system. The functionality of this logic device is shown in Fig.3 by means of a schematic diagram. It can be seen that this chip performs a number of tasks, implemented as individual modules, in order to generate control signals for the slave camera from an analysis of incoming video images. The diagram also shows the data, address and control signal pathways connecting the different modules. Two of the FPGA's modules, frame-grabber control and memory control, provide all the control signals required by the external components in order to synchronize the image processing hardware to the CCIR composite video signal. These modules are implemented using 56 and 29 logic cells respectively, where 1 logic cell is equivalent to about 20 gates. The robot arm control module generates the signals required to move the slave camera and is implemented using only about 10 logic cells. Three image-processing modules are included in the FPGA. The first two of these, background image update and threshold selection, implement the algorithms described in the previous section within 17 and 68 logic cells respectively. Both algorithms require either a multiply or

divide operation (implemented by bit shifting) but the latter also requires an accumulator (implemented as a 20 bit adder) which accounts for the difference in the amount of resources used. The third image-processing module, target detection, is the most complex of all the internal modules and its implementation requires 286 logic cells. It decides whether a connected group of changing pixels is a target of interest or not. Because of its major influence on the workings of the system, the remainder of this section concentrates on describing the algorithm used to achieve the desired functionality.

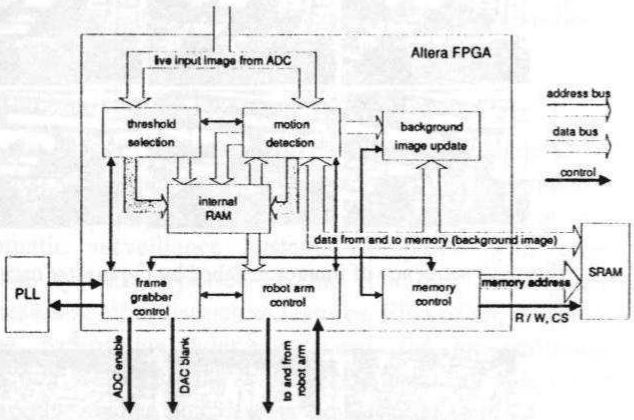

Fig. 3. Schematic diagram of the FPGA part of the overall system design.

The target detection module receives the live and background image data at video rates, pixel by pixel on a row by row basis, and applies differencing and thresholding as described earlier. The result of this process is illustrated in Fig. 4, which shows a small region cropped from Fig. 2(c). A potential target group of pixels, whose rate of change with time is faster than that set by the time constant of the temporal filter, is shown in black in Fig. 4 while background pixels are indicated in white. Moving along each row, a 4-bit counter *col_count* counts the number of successive pixels whose gray levels exceed *Threshold*. If the length of this run of black pixels is less than a user defined width limit *col_udtp1* (appropriate to people) the counter is cleared. If however, this run length is greater than or equal to *col_udtp1* another 4-bit counter, *row_count* checks the height of the identified object and compares it with a height limit *row_udtp1*. The parameter *row_count* counts the number of successive rows in which runs of black pixels, starting at about the same column position, exceed the width limit. This situation is illustrated by position *b* in Fig. 4 for the case where *col_udtp1* is chosen to be four. If *row_count* equals the height limit, then the co-ordinates of the target are saved in internal memory and also sent to the slave camera controller. This situation is illustrated by position *c* in Fig. 4 for the case where *row_udtp* is chosen to be 6.

The logic implemented in the target detection module is defined more precisely by the following pseudo-code. As indicated in this code, a section of the logic is repeated, the first time to detect large objects such as cars and the second time to detect smaller objects such as people. The temporary variables and saved data indicated in the pseudocode are stored in the internal RAM of the FPGA.

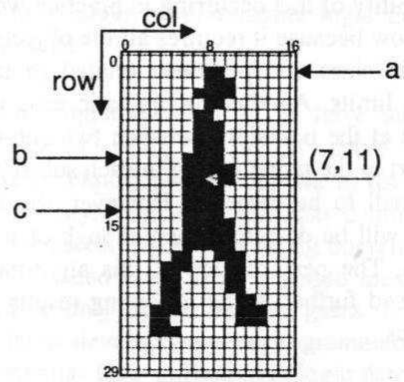

Fig. 4. A small region of Fig. 2(c) showing a detected object.

```
//within each of the 100 sub-regions do the following
1   Clear all counter variables to zero
2   for row =1 to end of current sub-region
3   for col =1 to end of current sub-region
4       if pixel at location [col,row] > Threshold then
5           col_count= col_count + 1
6       else
7           if col_count > or = to col_udtp2 then // (this
              could be a large object i.e. car)
8             col_count = 0
9             if row_count zero then // (first row of object)
10                save_col = col – col_udtp2
11                save_row = row – row_udtp2
12                row_count = row_count + 1
13            else if row_count not zero then
                  // (subsequent rows of same object)
14                if save_row = row-1 and save_col =
                  (col or col + 1 or col – 1) then
15                  save_col = col – col_udtp2
16                  save_row = row – row_udtp2
17                  row_count = row_count + 1
18                  if row_count = row_udtp2 then
19                  send target co-ordinates to slave
                  camera controller and
20                  clear all counter variables
21                  endif
22              else if save_row not = row then
23                  clear all counters
24              endif
25          endif
26      else if col_count > or = to col_udtp1 then //
            (this is a smaller object i.e. person)
    The previous lines 8 to 25 are duplicated with
    col_udtp2 changed to col_udtp1, row_udtp2 changed
    to row_udtp1.
45      end if
46    endif
47  end for
48  end for
```

This relatively simple algorithm detects only one potential target object within each sub-region and ignores any other potential target to the right of the first. This is of no concern in the current system because, as long as one target is detected, a view of that sub-region will be captured showing all targets that are present. However, it is conceivable that the presence of a number of objects,

smaller than the size threshold, to the left of a real target could prevent the detection of the real target. The probability of this occurring in practice would be expected to be low because it requires all the objects present to have specific sizes and positions related to target width and height limits. Another problematic case is when a target occurs at the boundary between two sub-regions, because the part of the target falling in each sub-region may then be too small to be detected. However, the target is moving and it will be detected when enough of it is within a sub-region. The performance of this algorithm in practice is discussed further in the following results and conclusions sections.

5. RESULTS

All the functions described in previous sections have been encapsulated within a single Altera EPF10K10LC84 device using the design and simulation tools provided by the Altera MAX+PLUS II development system. Two Altera library megafunctions, namely a fast 16-bit adder/subtractor and an allocator for the FPGA's internal RAM (EABs) were each used more than once in the design to shorten the design cycle time. It was found appropriate to create each of the six sub-designs shown in Fig. 4 as text files using Altera Hardware Description Language (AHDL). The complete design consists of about 1600 lines of AHDL source code. It uses most of the logic capacity and about 70% of the memory capacity of the specified device.

The complete system was tested by mounting the two cameras side by side, about 11m above ground level, and setting them to view a thoroughfare in our university campus that is used by both pedestrians and road vehicles. In order to illustrate the system performance, two synchronized frame grabbers were used to capture simultaneously image sequences from the master and slave cameras. Fig. 5 shows a typical short sequence of 12 images captured from the master camera over a 6s period on a rainy day. In this sequence one person (indicated by an enclosing black rectangle for ease of identification) can be seen walking down the sidewalk.

Fig. 6 shows higher resolution views of the moving person captured by the slave camera during the same period as it follows him. These images show the direction of the slave camera remaining fixed while the target moves through one of the 100 sub-regions, e.g. B1, B2 and B3, then changing to the next sub-region, e.g. C1, as the target progresses further. Because the slave field of view contains 512 by 512 pixels, there is sufficient resolution to recognize the person if the images are appropriately magnified. In this particular sequence, only one moving person is present and the slave camera accurately tracks the movement. When multiple moving targets are present, however, the system no longer tracks an individual target. It is designed to cause the slave camera to view them in order of their positions starting from the top left and moving from left to right and top to bottom through the list of target coordinates stored in the FPGA's internal memory. The list of targets is updated every frame period.

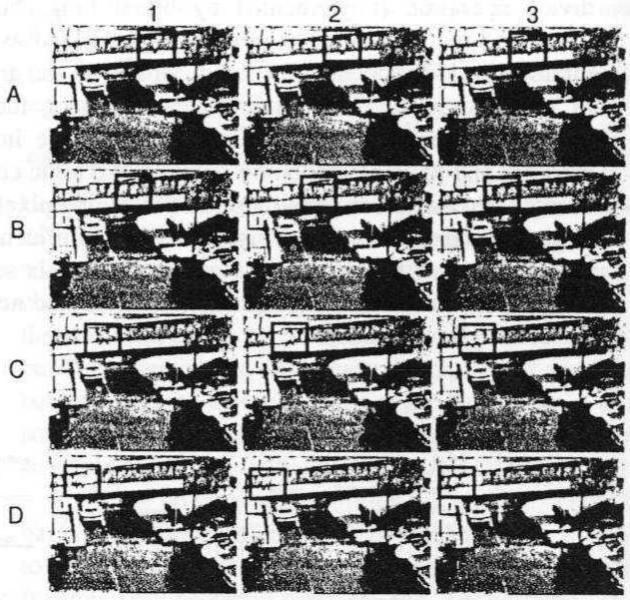

Fig. 5. A sequence of images grabbed by the master camera.

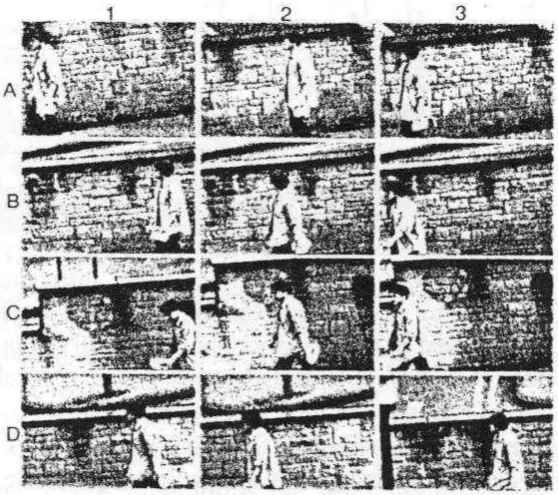

Fig. 6. A sequence of images grabbed by the slave camera.

Results on the detection and tracking of people recorded in one hour under different environmental conditions are shown in Fig. 7. The results demonstrate the ability of the system to work well in different situations. In bad atmospheric conditions, such as strong winds and heavy rain, the system generates a few false alarms but this is considered to be preferable to missing genuine targets. These false alarms arise from the movements of objects similar in size to people such as the branches of trees. On sunny days only a very small percentage of target people are missed, due entirely to their clothing being indistinguishable in gray level from the background. The possibility alluded to in the previous section of targets being missed due to the presence of other smaller objects was not observed in the trials and is much less important than the lack of contrast between some targets and the background.

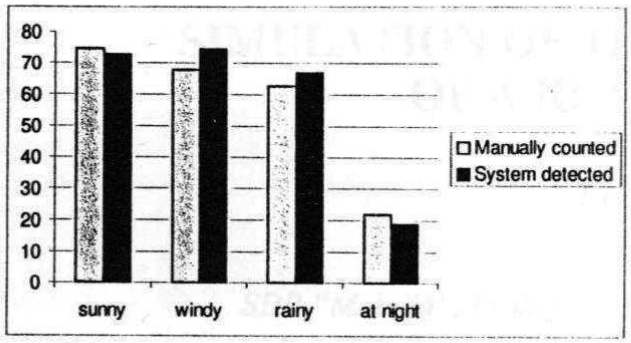

Fig. 7. The number of people detected over a one hour period under different environmental conditions.

6. CONCLUSIONS

All the major objectives of the research, namely the creation of a low cost stand-alone system incorporating a single FPGA controller, have been achieved in our automatic surveillance system. Compared with other similar systems reported in the literature, our system possesses several distinctive features. First of all, its stand-alone nature precludes the need for an additional computer. Secondly, the system operates at full video rates and grabs images and detects /tracks objects at 25 or 30 frames/s depending on the video standard used. Most of the alternative systems are significantly slower than this. Thirdly, the components on the board cost only about $100 which is much less than all the other systems. Fourthly, unlike the other systems, it is specifically designed to provide several high-resolution views of all targets of interest within the scene. Like all of the systems mentioned in the introduction, our system is capable of working effectively under varying illumination and weather conditions.

Improvements to the current system could be made on two fronts. Firstly, the relatively slow speed action of the pan and tilt movement, which is currently provided by a robot arm, must be improved to take full advantage of the very fast detection and tracking performance of our system. This would involve developing a lighter and more accurate opto-mechanical means of changing the direction of the optical axis of the slave camera. By amending the control logic, the slave camera could then be directed to point at the center of each target of interest rather than at a sub-region. This would permit even higher optical magnification to be employed by the slave camera. The second improvement relates to image processing functionality. Although the current system works, it has less sophistication in terms of image segmentation and target classification than computer-based systems. Adding to the existing design an algorithm that fully segments the whole binary image in each frame period would enable all moving objects in the field of view to be located and improved target classification to be added. This would require extra internal memory to store one or two lines of run-length encoded live video data, temporary variables, arrays and the bounds of each segment comprising a group of connected pixels. Additional logic resources would also be necessary. These modifications could be implemented in a larger programmable logic device at very little extra cost.

The successful implementation of a fully automatic surveillance system using a master slave camera combination is an important development in the area of vision-based security. The work has also demonstrated that a number of useful image-processing operations can be performed at video rates using a single inexpensive FPGA containing only 30000 useable gates. It is very clear that the latest developments in programmable logic devices, in particular their million gate logic density and their ability to perform several operations in parallel, afford new opportunities for researching and prototyping intelligent autonomous surveillance systems.

7. REFERENCES

[1] G. L. Foresti, "A real-time system for video surveillance of unattended outdoor environments," *IEEE Trans. Circuits Sys. Video Technol.*, vol.8, pp. 697-704, Oct. 1998.

[2] A. Iketani, A. Nagai, Y. Kuno, and Y. Shira, "Detecting persons on changing background," *IEEE 14th Int. Conf. on Pattern Recognition*, Queensland, Australia, 1998, pp. 74-76.

[3] B. A. Boghossian, and S. A. Velastin, "Real-time motion detection of crowds in video signals," *Inst. Elec. Eng. Colloquium High Performance Architectures for Real-time Image Processing*, IEE, London, 1998, pp. 12/1-12/6.

[4] D.W Trainor, D.J. Ridge, and R. Hamill, "Generic megafunction for high performance processing on FPGAs/PLDs," *IEEE Asilomar Conf. Signals, Systems & Computers*, Pacific Grove, CA, 1997, pp. 1047-1051.

[5] A. Makarov, "Comparison of background extraction based intrusion detection algorithms," *Proc. IEEE Int. Conf. Image Processing*, Lausanne, Switzerland, 1996,vol. 1, pp. 521-524.