A Single-Chip FPGA Implementation of Real-time Adaptive Background Model

Kofi Appiah Andrew Hunter
Department of Computing and Informatics
Faculty of Technology, University of Lincoln
Lincoln, LN6 7TS, UK
{*kappiah, ahunter*}*@lincoln.ac.uk*

Abstract

This paper demonstrates the use of a single-chip FPGA for the extraction of highly accurate background models in real-time. The models are based on 24-bit RGB values and 8-bit grayscale intensity values. Three background models are presented, all using a camcorder, single FPGA chip, four blocks of RAM and a display unit. The architectures have been implemented and tested using a Panasonic NV-DS60B digital video camera connected to a Celoxica RC300 Prototyping Platform with a Xilinx Virtex II XC2v6000 FPGA and 4 banks of onboard RAM. The novel FPGA architecture presented has the advantages of minimizing latency and the movement of large datasets, by conducting time critical processes on BlockRAM. The systems operate at clock rates ranging from 57MHz to 65MHz and are capable of performing pre-processing functions like temporal low-pass filtering on standard frame size of 640X480 pixels at up to 210 frames per second.

1. Introduction

We demonstrate the use of Field Programmable Gate Array (FPGA) for the extraction of accurate background models under variable lighting conditions in real-time. This implementation has a number of uses in embedded systems as well as automated visual surveillance systems. Real-time image processing is difficult to achieve on a serial processor, due to the movement of large data sets and complex operations that need to be performed on the image [9]. Advances in semiconductor technology makes it possible to design complete embedded System-on-Chip (SoC) by combining sensor, signal processing and memory onto a single substrate [13]. New embedded vision systems have emerged as a result of this level of integration and are estimated to increase in the coming years. The aim of computer vision systems is to scan objects and make judgements on those objects at rates much faster than human observers, thus the need to identify imaging functions that allow the computer to behave like a trained human operator[17].

Field Programmable Gate Array, a technology which has recently been made available to researchers, is used as an alternative platform for speedup in computer vision and digital image processing. The potential uses of FPGAs in areas like medical image processing, computational fluid dynamics, target recognition, embedded vision systems, gesture recognition and automotive infotainment have been demonstrated in [2] [4] [10] [11] [13]. Digital Image processing or computer vision algorithms can be broken down into three major stages [8]: early processing, implemented by local pixel-level functions; intermediate processing, which includes segmentation, motion estimation and feature extraction; and late processing, including interpretation and using statistical and artificial intelligence algorithms. Typically algorithm sophistication is concentrated in the later stages, but processing demands dominate in the early stages.

Vanderlei *et al* [2] used a simplified method in converting Red, Green, and Blue (RGB) values into Hue, Saturation, and Intensity (HSI) components, for extracting their binary region of interest (ROI) for their RAM-based neural network. The simplification was necessary to achieve feasible FPGA implementations. The choice of T1 and T2 (inferior and superior thresholds respectively) as shown in equation 1 is not very clear in their implementation. Elham Ashari [1] used an adaptive thresholding method to separate the foreground and background pixels in gray level images. Hardware implementation results in terms of visual performance, speed, and area consumption of the implementation is given, yet the algorithm is application dependent and requires some training period for perfect segmentation. Jim *et al* [16] used an experimental colour representation for filtering common colours found in road signs. They used RGB values as opposed to HSV to avoid computationally ex-

pensive conversions as outputs of many cameras are in RGB mode.

$$f(x,y) = \begin{cases} 1 & T_1 \leq f(i,j) \leq T_2 \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

From the above analysis it becomes clear that most object and activity recognition systems require some form of object extraction at their initial stages, a requirement which has not easily been achieved on reconfigurable platforms such as FPGA. Background subtraction is the commonly used method for extracting moving targets in a scene. Even though it is robust and less computationally expensive, it requires the maintenance of a background model. The maintained model can lead to accumulated errors if not updated over time. This paper presents three different real-time adaptive background models, which have successfully been implemented on FPGA with minimal use of external memory.

The paper is organized as follows. Section 2 briefly describe some background modelling algorithms used for segmenting moving objects. Their advantages and disadvantages for FPGA implementation are also given. Section 3 gives details of the system showing how the various components and peripherals are connected. This is followed by our background modelling approach in section 4 with details on how it is implemented on FPGA. Section 4 also gives results and analysis of each implementation. Finally, we present a summary of our work and point out future directions.

2. Previous Work

The first stage in processing for many video applications is the segmentation of (usually) moving objects with significant difference in colour and shape from the background. Where the camera is stationary, a natural approach is to model the background and detect foreground object by differencing the current frame with the background. A wide and increasing variety of techniques for background modelling have been described; a good comparison is given by Gutchess *et al* [6].

The most popular method is unimodal background modelling, in which a single value is used to represent a pixel, which has been widely used due to its relatively low computational cost and memory requirements [7] [18]. This technique gives a poor results when used in modelling non-stationary background scenarios like waving trees, rain and snow. A more powerful alternative is to use a multimodal background representation, the most common variant of which is a mixture of Gaussians [5] [15]. However, the computational demands make such techniques unpopular for real-time purposes; there are also disadvantages in multimodal techniques [5] [15] [18] including the *blending effect*, which causes a pixel to have an intensity value which has never occurred at that position (a side-effect of the smoothing used in these techniques). Other techniques rely heavily on the assumption that the most frequent intensity value during the training period represents the background. This assumption may well be false, causing the output to have a large error level.

2.1 Grimson's Algorithm

Grimson *et al* [15] introduced a multimodal approach, modelling the values of each pixel as a mixture of Gaussians. The background is modelled with the most persistent intensity values. The algorithm has two variants, colour and gray-scale: in this paper, we concentrate on the gray-scale version. The probability of observing the current pixel value is given as:

$$P(X_t) = \sum_{i=1}^{k} \omega_{i,t} \eta(X_t, \mu_{i,t}, \sigma_{i,t}) \quad (2)$$

Where $\mu_{i,t}$, $\sigma_{i,t}$ and $\omega_{i,t}$ are the respective mean, standard deviation and weight parameters of the i^{th} Gaussians component of pixel *X* at time t. η is a Gaussian probability density function

$$\eta(X_t, \mu_{i,t}, \sigma_{i,t}) = \frac{1}{\sigma_{i,t}\sqrt{2\pi}} e^{\frac{(X_t - \mu_{i,t})^2}{2\sigma_{i,t}^2}} \quad (3)$$

A new pixel value is generally represented by one of the major components of the mixture model and used to update the model. For every new pixel value, X_t, a check is conducted to match it with one of the K Gaussian distributions. A match is found when X_t is within 2.5 standard deviation of a distribution. If none of the K distributions match X_t, the least weighed distribution is replaced with a new distribution having X_t as mean, high variance and very low weight. The update equations are as follows:

$$w_{i,t} = w_{i,t-1} + \alpha(m_{i,t} - w_{i,t-1}) \quad (4)$$

where α is the learning rate and

$$m_{i,t} = \begin{cases} 1 & \text{if there is a match} \\ 0 & \text{otherwise} \end{cases} \quad (5)$$

$$\mu_t = \mu_{t-1} - \rho(X_t - \mu_t) \quad (6)$$

$$\sigma_t^2 = (1-\rho)\sigma_{t-1}^2 + \rho(X_t - \mu_t)^T(X_t - \mu_t) \quad (7)$$

Only the matched distribution will have its mean and variance updated, all others remain unchanged. For

$$\rho = \alpha\eta(X_t|\mu_t, \sigma_t) \quad (8)$$

The first B distributions (ordered by ω_k) are used as a model of the background, where

$$B = arg_b\, min(\sum_{k=1}^{b} \omega_k > T). \quad (9)$$

The threshold T is a measure of the minimum portion of the data that should be accounted for by the background.

2.2 Temporal Low-Pass filtering Algorithm

Aleksej [12] introduced a method to avoid false alarms due to illumination, using a temporal filter to update the background model, while a global threshold value *T* was used to extract target regions. The background update he used is of the form

$$B(k,l,n) = \frac{(p-c)}{p} B(k,l,n-1) + \frac{c}{p} I(k,l,n)$$

where *c* is the number of consecutive frames during which a change is observed and is reset to zero each time the new value becomes part of the background; *p* is the adaptation time or insertion delay constant. The moving target is extracted on a pixel level with the following relation:

$$f(k,l,n) = \begin{cases} 1 & |I(k,l,n) - B(k,l,n)| > L \\ 0 & \text{otherwise} \end{cases} \quad (10)$$

where *f(k,l,n)*, *B(k,l,n)* and *I(k,l,n)* are the respective foreground, background and the grayscale intensity value of pixel *(k,l)* for the n^{th} frame and *L* is the global threshold value.

The low-pass filtering algorithm is attractive for two reasons. First it is very simple and hence updating the background information is computationally cheap and memory consumption is minimal. The use of single global threshold value as well as a single mode makes it unattractive for scenes with varying lighting intensities. In contrast, Grimson's algorithm [15] is robust to outdoor environments where lighting intensity can suddenly change, and it handles multimodal backgrounds such as moving foliage (cyclical motion) without manual initialisation. Unfortunately, the use of floating-point numbers in all its update parameters makes it computationally expensive, and unsuitable for hardware implementation [1].

3 Overview of Setup

The hardware system we present here is composed of a Panasonic NV-DS60B digital video camera, a display unit and an FPGA prototyping board. The camera is interlaced and runs at 50Hz, thus effectively transmitting at 25Hz in 24-bit RGB values of size 768×567 in PAL format. The output of the camera is connected directly to the RC300 prototyping board via the S-video input. The RC300 board is packaged with Xilinx Virtex II XC2v6000, 4 banks of ZBT SRAM totalling 32MBytes (thus 4 banks x 2M x 36bits) and two DVI output ports. The outputs of the processed image and the background are displayed on different VGAs for visual inspection. This constrains the available memory resource.

The inability to read from and write to a single bank of RAM in a single clock cycle calls for the use of all available memory banks. Two banks hold camera data, whilst the other two hold background update information. A control unit controls the RAM bank the camera data is written to and which bank the VGA reads from. The two banks reserved for camera data are swapped after every full frame is acquired. Similarly the RAM bank for the display unit is swapped. The architecture is such that, when data from the camera is being written to bank ♯1 the processing unit reads captured camera data from bank ♯2 and processes it with stored background data from bank ♯3. Concurrently, background data is written to bank ♯4 after update. Figure 1 shows this architecture in detail, which is the same for all three implementations.

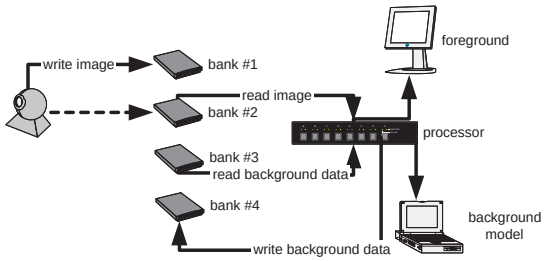

Figure 1. The RAM switching architecture

4 Our Approach

We present here a novel hybrid background modelling algorithm that combines the attractive features of Grimson's algorithm [15] and the temporal low-pass filtering [12], with appropriate modifications to improve segmentation of the foreground image, and to allow an efficient implementation on a reconfigurable hardware platform such as Field Programmable Gate Array (FPGA).

Following Grimson [15], we maintain a number of clusters, each with weight w_k, where $1 \leq k \leq K$, for K clusters. Rather than modelling a Gaussian distribution, we maintain a model with a central value, c_k of 11-bits (8 bits integer part and 3 bits fractional part). We use an implied global range, $[c_k - 15, c_k + 15]$, rather than explicitly modelling a range for each pixel based on its variance as in [15]. The weights and central values of all the clusters are initialised to 0.

A pixel $X = I(i,j)$ (where X is 11-bit fixed-point) from an image I is said to match a cluster, k, if $X \geq c_k - 15$ and $X \leq c_k + 15$. The highest weight matching cluster is updated, if and only if its weight after the update will not exceed the maximum allowed value (i.e. $w_k \leq 64$, given the data width of the weight as 6 bits). The update for the weight is as

follows:

$$w_{k,t} = \begin{cases} \frac{63}{64} w_{k,t-1} + \frac{1}{64} & \text{for the matching cluster} \\ \frac{63}{64} w_{k,t-1} & \text{otherwise} \end{cases} \quad (11)$$

The central values of all the clusters are also updated as follows:

$$c_{k,t,i,j} = \begin{cases} \frac{7}{8} c_{k,t-1,i,j} + \frac{1}{8} X_{i,j} & \text{matching cluster} \\ c_{k,t-1,i,j} & \text{otherwise} \end{cases} \quad (12)$$

Where $c_{k,t,i,j}$ is the central value for cluster k at time t for pixel (i, j)

If no matching cluster is found, then the least weighted cluster's central value, c_K is replaced with X; its weight is reset to zero. The way we construct and maintain clusters make our approach gradually incorporate new background objects. This is similar to [12] and hence the insertion delay is $2^3 = 8$ frames in our case.

The K distributions are ordered by weight, with the most likely background distribution on top. Similar to [15], the first B clusters are chosen as the background model, where

$$B = arg_b \, min(\sum_{k=1}^{b} \omega_i > T). \quad (13)$$

The threshold T is a measure of the minimum portion of the data that should be accounted for by the background. The choice of T is very important, as a small T usually models a unimodal background whiles a higher T models a multi-modal background.

We classify a pixel as foreground pixel based on the following two conditions:

1. If the intensity value of the pixel matches none of the K clusters.

2. If the intensity value is assigned to the same cluster for two successive frames, and the intensity values $X(t)$ and $X(t-1)$ are both outside the 40% mid-range $[c_k - 6, c_k + 6]$.

The second condition makes it possible to detect targets with low contrast against the background, while maintaining the concept of multimodal backgrounds. A typical example is a moving object with grayscale intensity close to that of the background, which would be classified as background in [15]. This requires the maintenance of an extra frame, with values representing the recently processed background intensities.

4.1 Gray-Scale Background Modelling

It is not always necessary to maintain a multimodal background. Many vision systems for handheld devices may have limited memory and may not require the maintenance of multi-modal backgrounds to operate correctly. For these reasons we demonstrate how we can model background scenes in real time on FPGA using a unimodal background. The algorithm used is similar to that described above with $K = 1$ and no associated weight. The central value is updated on each processed frame and hence the background data can suffer from high deviation.

4.1.1. FPGA System Design. The implementation of the unimodal grayscale background modelling algorithm is made up of six distinct processes all running in parallel. The first of these is the image capture block which acquires pixels in RGB format, at camera rate of 25Hz and converts it to 8-bit grayscale in a single cycle. This block has iterative mechanisms to acquire all pixels from the camera. Thus after acquiring pixel (i, j) this block iterates several times for pixel $(i + 1, j)$.

This iteration is necessary and possible as the clock rate of the design is much higher than the camera transmission rate. The successfully acquired pixel is sent to the memory write block. This block takes two cycles to write the camera data to external RAM. The first cycle is used to enable memory write for either RAM♯1 or RAM ♯2. The data is written to the appropriate RAM, which is dependant on the RAM Control signal (RAM switch) in the second cycle. Figure 2 is a pictorial representation of the memory writing pipeline in two clock cycles.

Two display units (VGA) are use for visual inspection to display the background model and moving targets respectively. The two units have the same vertical blanking and hence it is possible to generate different outputs with a single data. The background data read from RAM♯3 (or RAM♯4 depending on the RAM control) is used to extract the foreground image, which is sent to the foreground display unit (FDU). Simultaneously, the background data is sent to the background display unit (BDU). This has been made possible by reading pixels six cycles ahead of their display time.

Out of these six cycle, the first cycle is used to enable memory read for RAM ♯2 and ♯3 (or RAM♯1 and ♯4), camera and its corresponding background data is made available in the second cycle. The 8-bit grayscale intensity value is converted to 11-bit fixed-point value in the third cycle. The fourth cycle is used to build a cluster around the 11-bit central value maintained as the background value. This is followed by the fifth and final stage, which estimates if the 11-bit grayscale value falls in this cluster. If the value falls out of range of the cluster, the grayscale value is sent to the foreground display unit else zero (intensity value of zero) is sent.

The background value is simultaneously sent to the background display unit. The fifth cycle is also used to enable memory write for RAM♯4 (or RAM♯3). The final and sixth cycle displays the

values on the display units and writes the updated background value to the external RAM. It should be pointed out that the first six pixels of the display units are incorrect on reset.

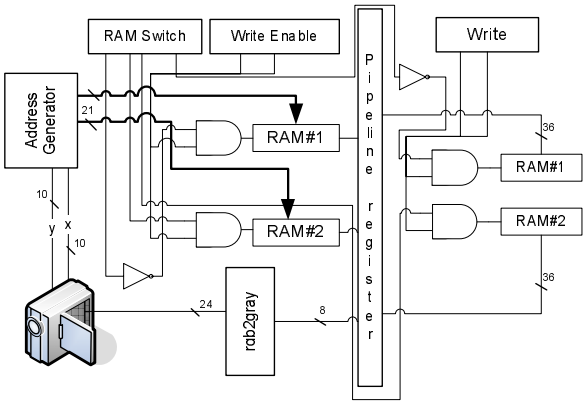

Figure 2. The the 2 stage memory write pipeline architecture

4.1.2. Results and Analysis. The design described above runs at 64.81MHz as reported by the Place and Route (PAR) tool. At this speed, the design is able to process camera data at more than real-time. The total latency from the time the camera sends the first pixel to the time the the pixel is available for display on display units is approximately 0.2μsec. This is because it takes 5 clock cycles to write a pixel from the camera, which runs at 25Hz. It takes 6 clock cycles to update the background data and extract the region of interest for a given pixel. Thus at a clock speed of 64.81MHz, the latency is approximately 0.2μsec. Table 1 summarizes the resource utilization of this implementation. When the pipeline is full the FPGA produces result for a pixel every clock cycle. Thus running at 64.81MHz, the output of a pixel is ready every 15.429ns. At this frequency we can effectively process 210fps of RS-170/NTSC (640$\times$480) frame size and 146fps of CCIR/PAL (768$\times$576) frame size.

4.2 RGB Colour Background Modelling

Many colour segmentation algorithms have been proposed in the past for skin lesion images, intrusion detection and feature extraction. Very few of these have successfully been implemented on hardware platform for speed-up. This is due to a lack of resources to meet the real-time processing needs of these very complex and computationally expensive colour segmentation algorithms. The need for such algorithms running on dedicated systems like FPGAs is mentioned by Neuenhahn [14]. Real-time colour image segmentation could be particularly useful in many applications, especially in biomedical image analysis [3].

Resource	Total Used	Per.
Flip Flops	1,112 out of 67,584	1%
4 input LUTs	1,762 out of 67,584	2%
Block RAMs	0 out of 144	0%
bonded IOBs	366 out of 824	44%
Occupied Slices	1,156 out of 33,792	3%
SSRAM (NTSC)	16 out of 256 Mbits	4.3%
SSRAM (PAL)	11 out of 256 Mbits	6.3%

Table 1. Resource utilization of the unimodal grayscale implementation, using *XC2v6000*, package *ff1152* and speed grade *-4*.

In this section we demonstrate how this challenging task in image processing can be achieved with simplified yet robust algorithms, running at real-time on reconfigurable computing platforms with minimal resources. The architecture we present here is similar to that in section 4.1, but instead of using an 8-bit grayscale intensity value this architecture relies on 24-bit RGB values. Again, the entire system is fitted on a single FPGA chip with 4 banks of SRAM.

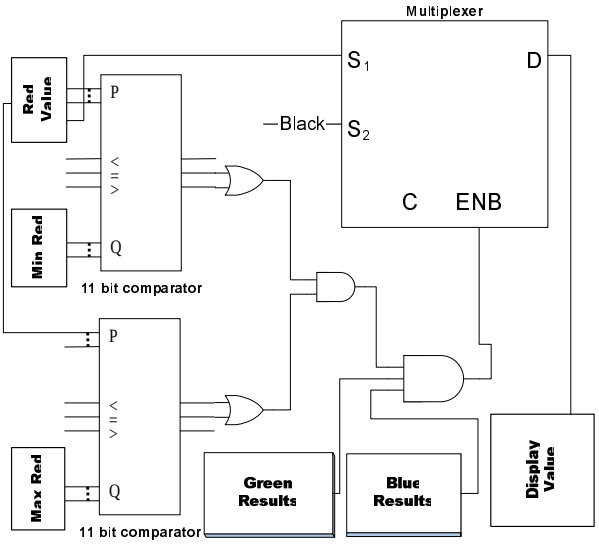

Figure 3. The fifth stage of the six-stage pipeline

4.2.1. FPGA System Design. Similar to the grayscale implementation, this architecture has six processes running in parallel. These are: the RAM control; pixel reading from camera; writing camera data to RAM; processing stored data for the output device and updating background data; writing to the output device; and writing the update background

data back to RAM.

These processes take 12 clock cycles in total. The longest process, the background updating and target extraction process, takes 5 cycles. The first cycle generates memory addresses and reads the corresponding background and camera values from the respective RAM blocks. The length of the camera data is 24bits (8 bits for each RGB channel) and that of the background is 33bits (11bits for each channel). The second cycle makes the addressed data available for processing. This is followed by extending the 3×8 bit RGB values into 11 bit fixed point values in the third cycle. The fourth cycle is used to build a cluster of width 30, each for the Red, Green and Blue colour components. The fifth cycle is used to decide if all the colour components of the observed pixels belong to the RGB clusters. The result of this defines the output to the VGA. If all colour components fall in their corresponding clusters, then the pixel is displayed as a background pixel else displayed as a foreground pixel. The last pipeline stage is depicted in figure 3, for the Red colour component, as the other two components have a similar structure.

4.2.2. Results and Analysis. This design also runs at an impressive speed of 64.40MHz, about 0.4MHz less than that of the grayscale implementation. The percentage of the resources consumed by this implementation is the same as that of the grayscale implementation, with the exception of 4 input LUTs (approximately 3% in this case). This can be attributed to the increase in the number of pipeline registers. The total latency from the time the camera sends the first pixel to the time the pixel is available for display on display units is also 0.2μsec. The percentages of external RAM consumed by the design for processing a PAL and NTSC frame sizes are 13% and 19% respectively. When the pipeline is full the FPGA produces result for a pixel every clock cycle, thus running at 64.40MHz, the output of a pixel is ready every 15.526ns. At this frequency we can effectively process 209fps of RS-170/NTSC (640×480 colour) frame size and 145fps of CCIR/PAL (768×576 colour) frame size.

4.3 Bimodal Gray-Scale Background Modelling

Multimodal background modelling is required to model scenes with non-stationary background objects, so reducing false positive alerts. To successfully implement our algorithm on a Field Programmable Gate Array, access to $17n$ bits of background data is required every clock cycle, where n is the total number of background clusters and 17 bits is the total number of bits require for the cluster's weight (6 bits) and central value (11 bits). Having access to 36 bits in a clock cycle, we can effectively implement a multimodal background with $n = 2$. We demonstrate the success of this implementation and how easily any n can be increased based on the available resources.

Code	Description
000	cluster 0 highest wgt. & data out of range
001	cluster 0 highest wgt. & data in range
010	cluster 0 second wgt. & data out of range
011	cluster 0 second wgt. & data in range
100	cluster 1 highest wgt. & data out of range
101	cluster 1 highest wgt. & in of range
110	cluster 1 second wgt. & out of range
111	cluster 1 second wgt. & in of range

Table 2. Econding scheme for determining which cluster camera data belongs

4.3.1. FPGA System Design. The design of the bimodal background is common with all the other architectures; it has six different processes running in parallel. The interesting part of the design is the eight stage pipeline for extracting moving targets and updating the background. The first two stages are used for reading the camera and its corresponding background data from external RAM. The 8-bit grayscale value from the camera is converted into 11-bits fixed point value in the third stage. To avoid deeply nested conditions as well as iteration in sorting the weights of the background clusters, different registers are used for the results of the sorted weights. This makes it possible to sort the weights in a single clock cycle in the fourth stage in the pipeline. Clusters are also built on the various central values in this stage.

The fifth stage is used to determine if the camera data belongs to the highest weighted cluster. We use an encoding scheme to know which cluster the camera data belongs to. Table 2 gives the encoding scheme used for the bimodal implementation and can easily be used for multimodal implementations. The weight and the background values are updates as in equations 11 and 12. If the camera data does not match the highest weighted cluster, the appropriate code is set and the camera data is compared with the second weighted cluster in the sixth stage. The pipeline stages will increase depending on the number of clusters used in the implementation. This causes a delay in terms of setup and propagation time for the pipeline registers but the speed gain as compared to a deeply nested conditional statement is very significant.

In the seventh and final stage the lowest weighted cluster's central value is replaced with the 11bit fixed point camera data and its weight set to zero if the data does not belong to any of the clusters. The foreground bit-map for the FDU is extracted in this same stage. For the BDU, a grayscale value is displayed if the

Resource	Total Used	Per.
Flip Flops	1,766 out of 67,584	2%
4 input LUTs	3,347 out of 67,584	4%
Block RAMs	57 out of 144	39%
bonded IOBs	366 out of 824	44%
Occupied Slices	2,124 out of 33,792	6%
SSRAM (NTSC)	37 out of 256 Mbits	14.4%
SSRAM (PAL)	25 out of 256 Mbits	9.7%

Table 3. Resource utilization of the bimodal grayscale implementation, using *XC2v6000*, package *ff1152* and speed grade *-4*.

pixel belongs to the highest weighted cluster or none of the clusters, and a red value is displayed if the pixel belongs to the second-weighted cluster. This colour display mode can also be used in a multimodal model. The updated background and weights of the clusters are sent to the background updating processing unit to be written to external RAM. It takes 8 cycles to get the first correct pixel displayed after reset.

4.3.2. Results and Analysis. This design runs at 57.00MHz, about 7.81MHz less than that of the unimodal implementation. This has only been possible with the use of the encoding scheme, as an earlier implementation could only run at maximum speed of 25MHz due to the use of deeply nested condition for evaluating the cluster that the camera data belongs to. To reduce noise due to the camera jitter, morphological opening is conducted on the foreground extracted. This is entirely conducted on the BlockRAM available on the FPGA. Effectively, this reduces the latency by a factor of 9 per pixel as it takes only one cycle to access data from the dual-port BlockRAM as compared to two cycles for the external RAM. Table 3 gives a summary of the resource utilization in this design. At 57MHz an output is ready every 17.543ns, when the pipeline is full.

5. Experimental Results

We evaluate the performance of our approach against that of [15] using $K = 3$, thus 3 cluster in our case and 3 distributions in [15]. We use eight randomly selected video sequences, four each from outdoor and indoor scenes. Manually marked frames are used as reference images for the sequences. Our result is based on pixel-wise errors against the reference image, in terms of true positive(TP), true negative(TN), false negative(FN) and false positive(FP) pixels.

There are approximately 50 frames in each sequence. Table 4 clearly shows the superiority of our algorithm against that in [15] in terms of sensitivity.

Scene	Our Approach (%)		
	SENS.	SPEC.	PPV
In1	84.88	98.94	84.01
Out1	83.18	99.87	94.79
In2	76.26	97.28	62.23
Out2	76.87	99.31	92.50
In3	76.72	96.51	47.82
Out3	81.39	99.56	58.03
In4	86.39	89.14	30.62
Out4	56.35	99.06	48.53
	Grimson's (%)		
	SENS.	SPEC.	PPV
In1	82.20	99.10	85.68
Out1	80.99	99.85	93.50
In2	68.70	97.72	63.93
Out2	64.33	99.22	89.75
In3	68.54	98.30	62.68
Out3	75.58	99.70	65.40
In4	85.38	91.71	36.38
Out4	45.96	99.38	53.91

Table 4. Pixel errors evaluation results

The algorithm does produce more false positive errors; this is the side-effect of our approach in detecting targets with low contrast against the background. However, in our target application false positive errors of the type reported are more acceptable than false negative errors, as subsystem tracking stages can discard distracters such as shadows.

The evaluation parameters used in table 4 are defined as follows: Sensitivity (SENS.) is the proportion of positives that are correctly classified and it's expressed as $\frac{TP}{(TP+FN)}$, Specificity (SPEC.) is the proportion of negatives that are correctly classified and it's expressed as $\frac{TN}{(TN+FP)}$ and Positive Predictive Values (PPV) is the proportion of cases classified as positives that are actually positive and it's expressed as $\frac{TP}{(TP+FP)}$.

Figure 4 gives sample images of the outputs generated by the two algorithms. These are output results of seven out of the eight sequences used in table 4. Visual inspection of the images shows where our approach outperforms that of Grimson's. Grimson's algorithm suffers from the foreground aperture problem, but our approach with its frame-level processing suffers minimally from this problem.

6. Summary and Conclusions

We have demonstrated how a single chip FPGA can effectively be used in modelling robust multimodal backgrounds in real-time. We have presented architectures for modelling unimodal backgrounds using grayscale intensity value, RGB colour values and bimodal backgrounds with grayscale values. The novelty detector used is motivated by [12] and [15]

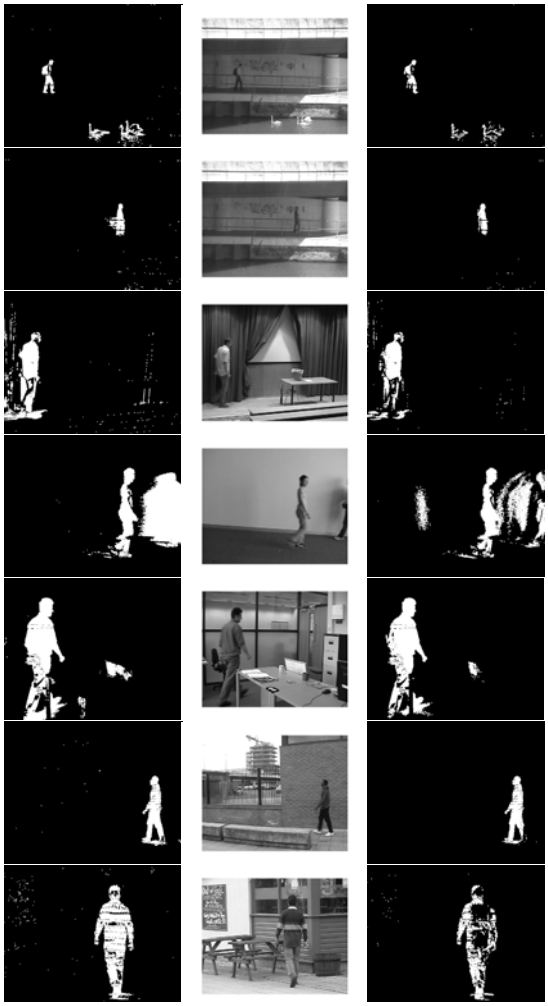

Figure 4. Sample outputs of the algorithms: Left shows our approach, middle is the original frame and the right is Grimson's algorithm

with modifications to enable the extraction of targets with low contrast. The processing speed and resources used in the implementation make room for other imaging algorithms for tracking and activity interpretation.

7. References

[1] E. Ashari. *FPGA Implementation of Real-Time Adaptive Image Thresholding*. SPIE–The International Society for Optical Engineering, December, 2004.

[2] V. Bonato, A. Sanches, and M. Fernandes. *A Real Time Gesture Recognition System for Mobile Robots*. International Conference on Informatics in Control, Automation and Robotics, Portugal, August, 2004.

[3] R. Cucchiara, C. Grana, S. Seidenari, and G. Pellacani. *Exploiting color and topological features for region segmentation with recursive fuzzy C-means*, volume 11. 2002.

[4] P. Ekas. *Leveraging FPGA coprocessors to optimize automotive infotainment and telematics systems*. Embedded Computing Design, Spring, 2004.

[5] A. Elgammal, D. Harwood, and L. Davis. *Non-parametric Model for Background Subtraction*. Proceedings of the 6th European Conference on Computer Vision, Dublin, Ireland, 2000.

[6] D. Gutchess, M. Trajkovic, E. Cohen-Solal, D. Lyons, and A. K. Jain. *A Background Model Initialization Algorithm for video Surveillance*. IEEE, International Conference on Computer Vision, 2001.

[7] I. Haritaoglu, D. Harwood, and L. Davis. *W^4: Who? When? Where? What? A real time system for detecting and tracking people*. IEEE Third International Conference on Automatic Face and Gesture, 1998.

[8] J.Batlle, J. Martin, P. Ridao, and J. Amat. *A New FPGA/DSP-Based Parallel Architecture for Real-Time Image Processing*. Elsevier Science Ltd., 2002.

[9] C. T. Johnston, K. T. Gribbon, and D. G. Bailey. *Implementing Image Processing Algorithms on FPGAs*. Proceedings of the eleventh electronics New Zealand Conference, ENZCON'04, Palmerston North, Nov, 2004.

[10] M. Leeser, S. Miller, and H. Yu. *Smart Camera Based on Reconfigurable hardware Enables Diverse Real-time Applications*. Proceedings of the 12th annual IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM'04), 2004.

[11] B. Levine, B. Colonna, T. Oblak, E. Hughes, M. Hoffelder, and H. Schmit. *Implementation of a Target Recognition Application Using Pipelined Reconfigurable Hardware*. Military and Aerospace Applications of Programmable Devices and Technologies International Conference, 2003.

[12] A. Makarov. *Comparison of Background extraction based intrusion detection algorithms*. IEEE Int. Conference on Image Processing, 1996.

[13] S. McBader and P. Lee. *An FPGA Implementation of a Flexible, Parallel Image Processing Architecture Suitable for Embedded Vision Systems*. Proceedings of the International Parallel and Distributed Processing Symposium, IPDPS'03, 2003.

[14] M. Neuenhahn, H. Blume, and T. G. Noll. *Pareto Optimal Design of an FPGA-based Real-Time Watershed Image Segmentation*. 15th Annual Workshop on circuits systems on signal processing, November, 2004.

[15] C. Stauffer and W. E. L. Grimson. *Adaptive background mixture models for real-time tracking*. IEEE Conference on Computer Vision and Pattern Recognition, 1999.

[16] J. Torresen, J. W. Bakke, and L. Sekanina. *Efficient Image Filtering and Information Reduction in Reconfigurable Logic*. Proceeding of the 22nd NORCHIP Conference, Norway, November, 2004.

[17] R. Williams. *Increase Image Processing System Performance with FPGAs*. Xcell Journal, Summer, 2004.

[18] C. Wren, A. Azarbayejani, T. Darrel, and A. Pentland. *Pfinder: Real-time tracking of the human body*. IEEE Transactions on Pattern Analysis and Machine Intelligence, 1997.