ABSTRACT

Object detection systems involve three components: a video source, a processing unit, and a video display or alarm. The video source and display may vary greatly, but the video processing is generally done in software on a PC server. This research project focused on moving as much video processing as possible into hardware in order to reduce the size, cost, and level of user interaction needed for a robust security system.

The challenge of implementing the video processing of such a system in hardware is that present object detection software takes advantage of the sheer power of modern processors, using very complicated and computationally expensive algorithms. In order to be implemented in hardware, these complex algorithms needed to be simplified without greatly affecting their effectiveness in detecting objects.

In this research project, a Field-programmable Gate Array (FPGA) on an Altera DE2-70 Development Education board was used to develop the custom hardware required to perform this video processing. The Altera Nios II embedded processor system was used to perform all hardware interaction tasks necessary on the DE2-70 board and the custom hardware was constructed as modules inside the Nios II system. The theory of object detection was broken up into three portions: background subtraction, noise filtering, and finally, object detection. The background subtraction was accomplished in software on the Nios II processor, but in very simple code fast enough for an FPGA to efficiently execute. The noise filtering and object detection modules were then written in the Verilog hardware description language.

The final product successfully identified single objects in the video feed while occupying only 16% of the FPGA's logic elements and 14% of memory. The video processing was accomplished within 99,216 cycles per frame, or 5,952,960 cycles per second at 60 frames per second. The Nios II/f processor used can complete 51 million instructions per second, making this system require approximately 10% of the computational ability of the FPGA.

Further research should address the process of background subtraction. The current process involves capturing a single frame from the live video feed. This is prone to noise and requires user input. In addition, this process is the last portion of the project still done in software. Also, the project currently only identifies one object. If multiple objects are in the frame, they are seen as a single object. Future research should include a method to identify multiple objects.

1. INTRODUCTION

1.1 Purpose

It is often impractical or impossible to have security systems continuously monitored and analyzed by humans, whether because of fiscal reasons or because there is simply too much material to monitor at once. Although certain types of automated systems exist to track and/or detect security threats, there is a need for more robust systems to accurately and automatically identify and track foreign bodies. If additional restrictions, such as the need for a self-contained system, are applied, the additional issues of processing speed and memory management become especially important. There is a need for an automatic system to detect foreign bodies accurately and maintain effective processing speeds when not aided by larger systems.

The goal of this project is to design and implement an FPGA-based system that uses an off-the-shelf camera to detect and track foreign bodies in real-time, and provides the user with an alert accordingly.

1.2 Background

For more than forty years, video feed has been used in some way as a security measure, whether to identify criminals, or simply discourage theft. Closed circuit television (CCTV) and other similar measures are now so widely employed as to be considered commonplace in America.

However, closed circuit television and most similar security measures require human oversight to be effective; from human review of recorded data to someone actually needing to be present to oversee the feed live, the necessity of monitoring the data has been a traditional limitation on CCTV.

In order to overcome this limitation, a movement tending towards automatic monitoring of video feed has risen, and while these advancements allow for greater efficiency in the realm of automatic security systems, a significant number of shortcomings exist within most systems. According to WECU Surveillance, a company which provides surveillance equipment to include camera systems with automatic detection and tracking, the majority of tested systems have less than 50% probability of detection, and similarly poor false alarm rates [1]. Even considering this, upper end systems are almost prohibitively expensive, with the central controller alone costing in excess of $13,000. While this may be appropriate for larger organizations with more money to invest in such measures, smaller establishments have an interest in less costly, but still accurate, systems to provide automatic detection.

While many capabilities should be considered when implementing an automatic security system, such as detecting abandoned objects or object removal, this project will focus on detecting object intrusion, or the entrance of foreign objects upon the camera's view. In order to meet the needs of such a system, certain key characteristics must be considered during design. First, the system must be able to receive and analyze video feed from a camera source. Second, it must compare this feed to a previously saved image to detect any foreign intrusions. Lastly, it must provide some sort of feedback to the user highlighting the intruding object.

1.3 Objective Tree

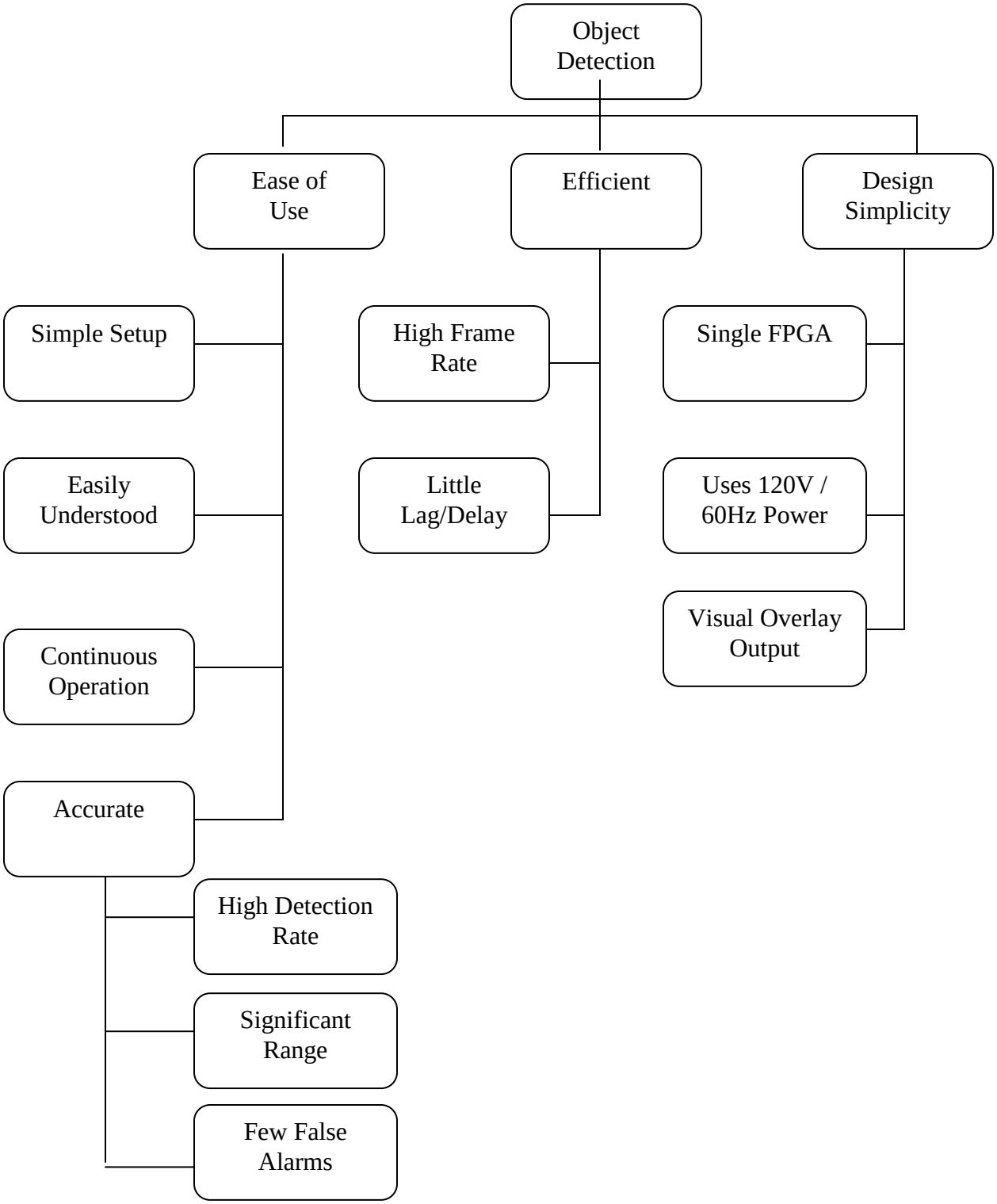

Figure 1.1: Objective Tree

1.4 Requirements

a. The system's display speed will not be less than 30 frames per second.

This is one half of the slowest average frame rate at which the human eye perceives video as continuous. While this is less than ideal, it is fast enough to not be inconvenient or significantly hamper the system.

b. The system will operate with a delay of less than 1/30 second.

The system must operate with a small enough delay when computing to complete analysis of a given frame before reception of the next.

c. Upon detection of a foreign body, the system should provide explicit feedback.

In order to be useful, the system must not only identify and detect foreign bodies, it must alert the user to their presence, too.

d. The system will operate using the computing ability of a single FPGA.

Because one of the main concerns is the efficiency of the algorithm to be used, limiting the system to a single field programmable gate array allows the entire system to be effectively self contained. Furthermore, this limits the cost of the system.

e. The system will function independently, without attachment to any other source except for external power.

This restriction helps make the system more mobile and less dependent upon other sources to be effective; it allows more freedom in placement of the security system.

f. The system will operate effectively at ranges of at least 25 ft in low light conditions, and 100 ft in normal lighting..

The system should be able to identify and detect bodies a significant distance from the camera. This number was based upon a standard range for security cameras.

1.5 Scope

This project will focus on the design and realization of the software algorithms for an automatic surveillance system; physical construction and implementation of a security system will not be considered. No sort of advanced background formation will be considered; a simple screen capture will be used. Testing will focus on functionality to accurately identify and track intruders.

2. THE DESIGN

2.1 Design Overview

The Intruder Detector consists of three major physical components: (1) an off-the-shelf camera, (2) an FPGA processing unit, and (3) a display. These physical components are interconnected through the processing unit as in Fig. 2-1 below.

Figure 2-1: Block diagram for Intruder Detector, illustrating the major systems.

The first module, the camera, is a commercial off-the-shelf camcorder that sends video data in an analog NTSC format via a composite video connection.

The second module, the FPGA, controls all data transfer and processing. The blocks inside the CPU block are software code; all other components are strictly hardware based. The data from the camera is collected by the Video-In Decoder component in the Nios II system loaded onto the FPGA. This decoder captures the video input one pixel at a time and sixty frames per second. Each pixel is passed through the system individually. From the Video-In Decoder, the video is passed to a Grayscale Re-sampler that converts the full color video to eight bit grayscale. The video is then passed to a video clipper and scaler that reshape the image to the proper dimensions for the built-in touch-screen display. This video is then stored in memory quadrant one by an SRAM memory controller.

The second section of the FPGA processing occurs in C software inside the CPU block of Figure 2-1. Running off of a touch-screen interrupt, the Menu Drawing function either draws the menu on the screen or executes one of the menu options selected by the user. These menu options include capturing a new background image or switching between the live, background subtracted, and noise filtered video streams. The Background Capture function simply copies the contents of the first video quadrant into a character array. The Background Subtraction function is not interrupt driven, but instead runs continuously. This function takes the difference of the live video stream and stored background pixel data, compares that difference against a user-set threshold, and stores either a white or black pixel into the third memory quadrant depending on whether the threshold was met or not.

The third section of the FPGA processing includes the Erosion and Dilation filters as well as the Object Detection block. The Erosion filter is a five pixel by five pixel AND gate that deletes objects smaller than the area of the AND gate. This has the side effect of reducing the size of larger objects by clipping the object edges. The Dilation filter corrects this by using a five pixel by five pixel OR gate to expand the object back to its original size. These two filters in conjunction with the threshold from the Background Subtraction function greatly reduce the effects of noise on the video feed. After the Dilation filter, the Object Detection block sweeps through the image looking for white pixels and determining the location of the outermost pixels.

The last section of the FPGA processing includes the Color Re-sampler with Box Drawing and the Character Buffer. The Color Re-sampler takes the location of the outermost pixels from the Object Detection block and draws a red box around those coordinates. The Character Buffer stores the menu text from the Menu Drawing function in the CPU. The Alpha Blender combines these two inputs and the VGA Controller outputs the final image onto the display.

The third module, the display, is a small, touch-screen VGA display attached to the Altera DE2-70 board.

2.2 Design Details

2.2.1 Video Device

This piece of the system presented two main options: a traditional camera or a thermal camera. Both options are commercial, off-the-shelf products. A traditional camera provides a more familiar image to process, while the thermal camera provides a simpler image due to the nature of warm objects standing out against a cool background. Although the simpler image of

the thermal camera would allow for much simpler noise filtering and object detection algorithms, thermal cameras were considered prohibitively expensive, and a traditional camera was chosen.

2.2.2 Processor Selection

There were two major choices for what sort of processing to use for the video tracking algorithms. Either a microprocessor or an FPGA could be used, and the FPGA was chosen because of speed reasons. A microprocessor would not be fast enough for video processing, but an FPGA could utilize custom architecture to negate the low clock speed of these devices. In addition, an FPGA was chosen over a program running on a full computer due to portability concerns. An FPGA could be added to existing systems (such as night vision systems in helicopters) with no further input required from the user. A full computer system would require input from the user which may interfere with their primary task (such as flying a helicopter).

2.2.3 FPGA Selection

There were two major design decisions for FPGA selection: brand and product line. Some brands include Altera, Xilinx, and Digilent. Altera was chosen due to the familiarity of the Altera DEx embedded development board kits and Altera Quartus II Software Suite. Within the DEx line, the DE2-70 and DE4 development boards were available for use. The DE2-70 board contains more familiar hardware than the DE4 board, but the Nios II processor handles most direct hardware interfacing to compensate for that difference. However, support for the DE4 board's Ethernet interface is virtually non-existent, and the interface could not be implemented in a manner that was fast enough for transmission of video data. Because the DE4 board lacks any sort of dedicated video input, the Altera DE2-70 development board with composite video inputs was chosen.

2.2.4 Interface Selection

The interface selection process depended mostly on performance concerns. Since Altera did not provide the necessary documentation with instructions for incorporating the Ethernet interface for video input, the initial implementation could only reliably receive around ten packets per second. This speed is not nearly fast enough to transmit video data. Because of these limitations, the dedicated composite video input interface was chosen instead. The composite video input interface has pre-built modules for the Nios II system reducing the amount of custom components required in the project.

2.2.5 Nios II System

The Nios II/f CPU handles the hardware interfacing portion of the system. The Nios II/f is a RISC 32-bit CPU that runs on a 50 MHz system clock and is capable up to 51 Dhrystone MIPS. The CPU implements the Avalon Stream, a pipeline system that incorporates separate hardware functions as modules. In this case, each stage of the pipeline consists of a single pixel of the video stream. The CPU governs the movement of data from one module to another along the stream, memory access, inputs and outputs such as the switches and touch-screen, and system clock access to all modules.

Non-custom Nios modules included in the system include the CPU, SSRAM and SDRAM controllers, pushbutton and slider switch input registers, Video-In Decoder, Chroma Re-sampler, Video Clipper and Scaler, Pixel Buffer, Character Buffer, Alpha Blender, and VGA Controller. These modules form the portion of the system that performs an unaltered video pass-through from the camera to the display. The custom modules below perform the video processing that allows object detection.

2.2.6 Menu Drawing Function

The Menu Drawing Function simply switches the video stream to a black background stored in memory quadrant two and draws the text for the menu buttons on the screen. There are six menu options to choose from: switch to live video feed, switch to background subtraction feed, switch to noise filtered feed, or perform a background capture and then switch. If the screen has already been drawn, the function executes the chosen menu option and removes the menu from the screen.

2.2.7 Video Stream Switching

The input to the VGA Controller is controlled by a video direct memory access (DMA) controller. This DMA stores the beginning address of the selected video stream in memory. In order to change the stream that is displayed on the LCD, the memory address for the requested stream is simply copied into the DMA's register.

2.2.8 Background Capture Function

The Background Capture function is a nested For loop that copies data for each pixel from the first memory quadrant (live video stream) into a 400 by 240 character array. A character array was chosen because a character in C code is composed of eight bits, which is the same number of bits with which the Grayscale Re-sampler encodes the pixel color data.

2.2.9 Background Subtraction Function

The Background Subtraction function takes the difference between the stored background from the Background Capture function above and the live video stream, pixel for pixel. A simple If statement is used to ensure that the result is an absolute value. Another If statement is used for implementing the user threshold. The threshold is set by eight switches on the DE2-70 board. The switches represent an eight bit binary number. If the difference between the pixels is larger than the threshold, the function recognizes that pixel as a foreground pixel and outputs a white pixel. If the difference is smaller than the threshold, the function recognizes the pixel as a background pixel and outputs black. Changing the threshold can help to account for changes in lighting or more subtle differences in color.

2.2.10 Erosion Filter

The Erosion Filter consists of a five pixel by five pixel AND gate that sweeps across each frame. Since the pipeline structure of the Nios system processes a single pixel at each stage, a buffer of 1605 pixels was constructed to store the necessary pixel data. This number is derived from four rows of 400 pixels each, which is the width of the display, and then a fifth row of just the required five pixels to fill the bottom row of the AND gate. Since the Background Subtraction function stores the background pixels as black and the foreground pixels as white, the AND gate has the effect of deleting any foreground data smaller than five by five pixels. Any object smaller than that box is deleted. This virtually eliminates noise when combined with the customizable threshold in the Background Subtraction function, but has the side effect of trimming the edges of objects since only the core pieces of the object will fill the AND gate block.

Because of the use of AND gates inside the module and the creation of the buffer, a delay must be placed on the output of the module in order to avoid distortion of the video stream. Each level of AND gates requires one cycle of delay, and each pixel in the buffer also requires one cycle of delay. This totals 1608 cycles of delay.

2.2.11 Dilation Filter

The Dilation Filter is similar to the Erosion Filter, but it uses OR gates instead of AND gates. This reverses the side effect of the Erosion Filter trimming the edges of objects. The OR gates have the effect of filling the five by five box with white pixels when any of the pixels in the box is white. Since noise was already eliminated in the Erosion Filter, this process has no ill effects.

Because the Dilation Filter requires the same size buffer and number of gates as the Erosion Filter, it also requires the same output delay, adding another 1608 cycles of delay.

2.2.12 Object Detection Algorithm

The Object Detection Algorithm is very simple. Because the only white pixels at this point in the Avalon Stream are actual objects, the algorithm only needs to determine where the outermost pixels of those objects are. To do this, the algorithm simply compares the location of all white pixels against the stored locations of the outermost pixels on the width and length axes, updating as necessary. On the last pixel of the frame, these four pixel locations are output to the Color Re-sampler for drawing a box around the objects.

2.2.13 Color Re-sampler with Box Drawing

The Color Re-sampler is also very simple. The Re-sampler compares the current pixel location to the outermost pixel locations received from the Object Detection Algorithm. When the current pixel location is inside a line connecting the outermost pixel locations, the output is replaced with a red pixel. The end effect is that a red box with a line width of one pixel is drawn around the object.

2.2.14 Display

The display receives the video output from the Color Re-sampler and text from the Character Buffer and combines the two on a small embedded screen for the user to see. The display is also a touch-screen that gives the user control over which video stream to view and when to take new background captures.

3. DESIGN VERIFICATION

3.1 Test Results

Figure 3.1: Live Feed with no Foreign Objects

Figure 3.1 above shows the default state of the system: Live feed with no foreign objects. At this point, no background has been captured. Because of this, the entire background capture is black and the system thinks that there is an object covering the entire screen. Once a background has been captured, the red box disappears.

Save Image & Return to Live Feed	**Save Image & go to BGND Subtract Feed**	**Save Image & go to Noise Filtered Feed**
Return to Live Feed	**Go to BGND Subtract Feed**	**Go to Noise Filtered Feed**

Figure 3.2: Menu Screen

Figure 3.2 above shows the possible menu options that appear when the touch-screen is pressed. Before the system can be operated, this menu must be brought up and one of the Save Image options must be selected.

Figure 3.3: Live Feed with Foreign Object

Figure 3.3 above shows a live feed with a foreign object. This highlights one flaw in the system: if the background capture is taken when the object is already in the frame it is not detected. The background capture has to be done when the frame is empty or it is useless.

Figure 3.4: Live Feed with Foreign Object, Detection On

Figure 3.4 above shows a live feed with a foreign object when a proper background capture has been completed. As evident, the foreign object has been detected and a box has been drawn to highlight its presence.

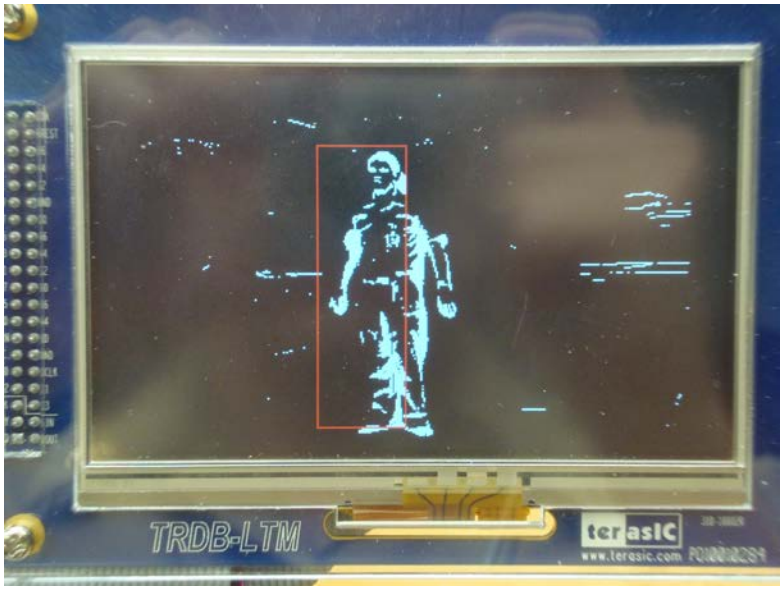

Figure 3.5: Background Subtraction Feed with Foreign Object

Figure 3.5 above shows the background subtracted feed with the same foreign object from Figure 3.4. An important aspect of this image is the amount of noise evident in the horizontal white lines present in the image. Because of this, noise filtering is required for reliable results.

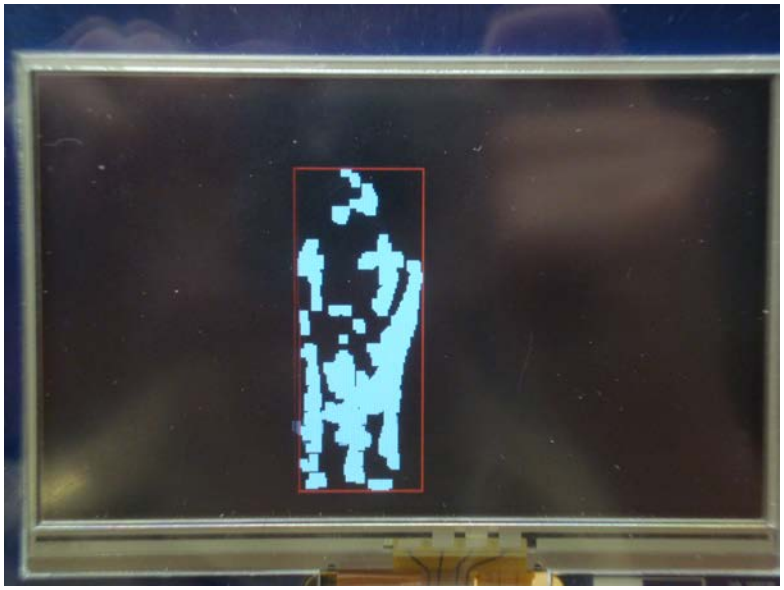

Figure 3.6: Noise Filtered Feed with Foreign Object

Figure 3.6 above shows the noise filtered feed with the same foreign object. The noise present in Figure 3.5 has been completely eliminated and the object, while its integrity has been degraded, is still completely detectable by the system.

3.2 Requirements Verification

Efficiency	*The system's display speed will not be less than 30 frames per second.*
	Successfully met. The system's display speed is the full frame-rate for LTM touch-screen, keeping the video input from the camera at 60 frames per second. Since this speed is set by the VGA Controller and Video-In Decoder, the system would operate at the required speed regardless of delay.
	The system will operate with a delay of less than 1/30 second.
	Successfully met. Each pixel requires one cycle, and there is an initial delay of 3216 cycles due to the filters. With 96,000 pixels, this totals 99,216 cycles per frame. At 60Hz, this is 5,952,960 cycles per second, which is much less than the 51 million instructions per second that the Nios II/f is capable of.
Ease of Use	*Upon detection of a foreign body, the system should provide explicit feedback.*
	Successfully met. Upon detection of a foreign body, the system overlays a red box around the intruding object.
	The system will operate effectively at ranges of at least 25 ft in low light conditions, and 100 ft in normal lighting.
	Partially met. As long as the object is large enough at a longer distance, the object will be detected. However, the effect of the lighting conditions on detection rate is dependent upon the camera used. With the visual spectrum camera that the system was tested on, low light conditions break the system. However, with a thermal or night vision camera, the detection rate should remain high.
Design Simplicity	*The system will operate using the computing ability of a single FPGA.*
	Successfully met. The system uses only the Cyclone II FPGA on the Altera DE2-70 board. The hardware components take up a total of 16% of the Cyclone II's logic elements and only 14% of the total memory available.
	The system will function independently, without attachment to any other source except for external power.
	Successfully met. Except for the camera and software used to load the system onto the DE2-70 board, the system requires no external sources other than power.

4. SUMMARY AND CONCLUSIONS

In the process of designing and creating this project, several conclusions were reached. Primarily, one of the most significant factors on the ability of a system to detect objects when performing background subtraction is the effect of noise and small variations within an image. While certain filtering methods can be utilized to minimize these effects, they are imperfect. In testing, changes imperceptible to the human eye were observed by the camera, due to vibrations, or the strobe of fluorescent lighting. Furthermore, changes in lighting over time, becomes a noteworthy consideration in areas subject to environmental light, as the filter required to overcome a major change in ambient light conditions would likely filter out intruding objects.

For this reason, a major recommendation for further work would be research into and work on a more robust background creation algorithm. Rather than having the user dictate a single frame within memory to be the background, using a process to take a series of images and average them together would be much more useful. This would eliminate minor noise within the background image itself, and could be used to create a constantly-updating background image that would adapt to changes in lighting over time.

In addition, some parts of the system are still done in software, most importantly the background subtraction. If the background subtraction could be moved to a custom Nios module, the very little bit of CPU processing power could be used for more robust background capture methods. In order to facilitate this, the method of memory management should be reassessed. Because the background subtracted and noise filtered feeds are true black and white, but still represented with eight bits of color data, a huge amount of memory is being wasted to store these values. In addition, the background of the menu is taking up a large amount of memory and it is also pure black. Determining more efficient methods of organizing the memory in the system would allow for the background subtraction to be moved into hardware once the synchronization problem is solved.

One last consideration for future development on the project would be the inclusion of multiple object detection. Currently, the system only recognizes one object at a time. One possible solution to this is to implement additional registers to store pixel data of the outlying pixels of objects and a method to determine when the edge of an object is reached. This will raise the problem of how to determine when groups of pixels are actually one object, but there are numerous software algorithms already developed that could possibly be implemented in hardware.

Important technical lessons learned in the design process include not only more advanced understanding of video processing itself, but always more precise use of the Nios system used. For example, the creation of custom Nios modules was fundamental to a hardware implementation of the system, and something we had little experience with beforehand. Additionally, Synchronization of the Nios stream was also vital to being able to properly manipulate and display the video data, and another major lesson learned.

Important non-technical lessons learned focus on the importance of proper documentation of work and the value of seeking assistance when faced with a difficult problem. Speaking to an informed subject about a technical problem helps one to put a problem to words, and often think about it in a different way than originally considered.

5. REFERENCES

[1] "Automatic Detection and Tracking Hardware," WECU Surveillance, October 2010, http://www.wecusurveillance.com/adat (accessed October 19, 2010).

[2] Fatih Porikli and Oncel Tuzel, "Human Body Tracking by Adaptive background Models and Mean-Shift Analysis" *Mitsubishi Electric Research Laboratory*, TR-2003-36, July 2003.

[3] T. Chen, H. Haussecker, A. Bovyrin, R. Belenov, K. Rodyushkin, A. Kuranov, V. Eruhimov, "Computer Vision Workload Analysis: Case Study of Video Surveillance Systems" *Intel Technology Journal*, **9**, pp. 109-118, 2005.

[4] "DE2-70 Development and Education Board," Altera, 2010, http://www.altera.com/education/univ/materials/boards/de2-70/unv-de2-70-board.html (accessed October 19, 2010).

APPENDIX A: PROJECT MANAGEMENT PLAN

A.1 Work Breakdown Structure

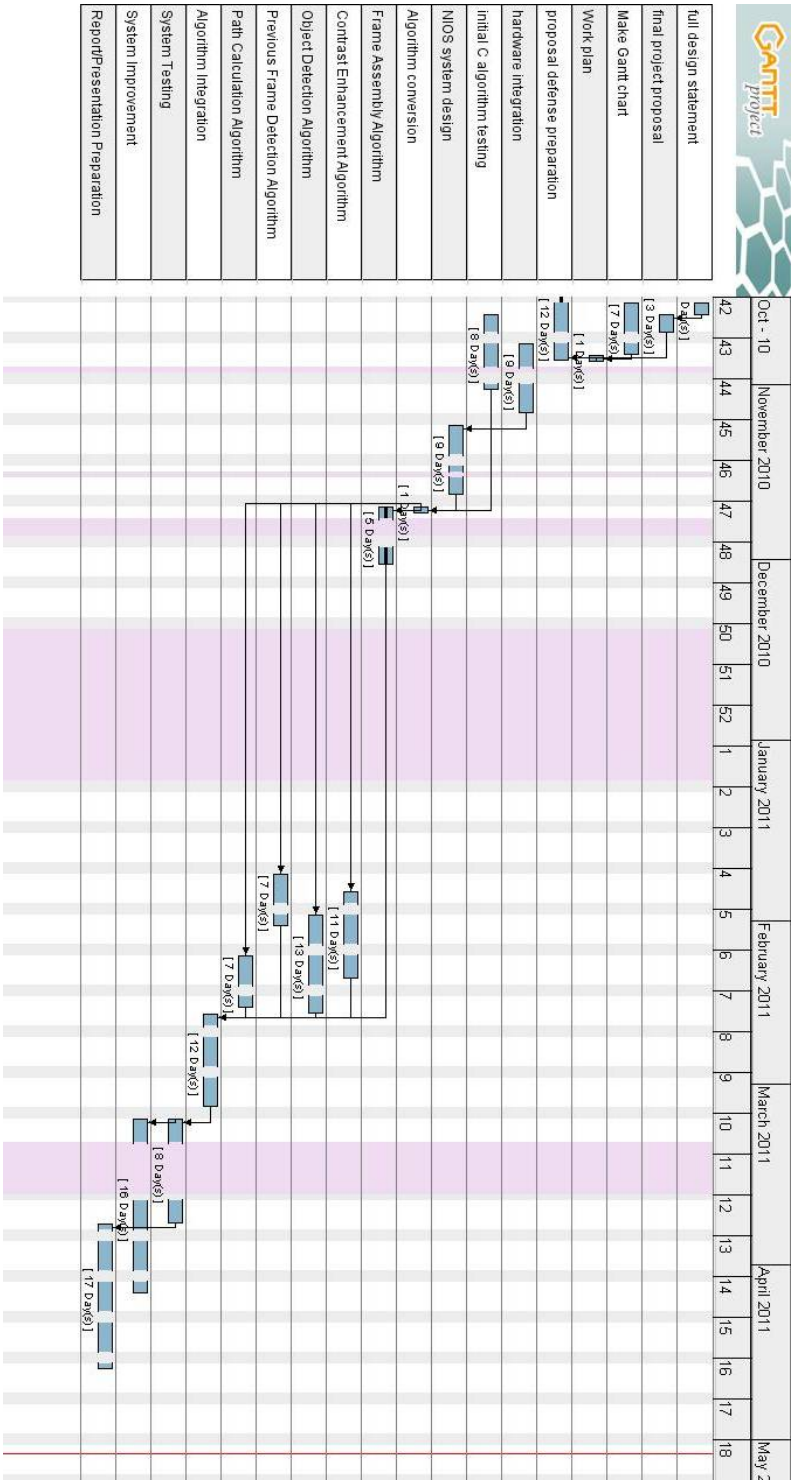

Figure A.1: Project Gantt Chart

A.2 Member Contributions

Project Task	Primary Contributors
Interrupt Handling	MIDN 1/C Adams
Menu Drawing	MIDN 1/C Adams
Background Capture	MIDN 1/C Adams
Video Stream Switching	MIDN 1/C Adams
Background Subtraction	Dr. Ngo and MIDN 1/C Adams
Erosion and Dilation Filters	Dr. Ngo and MIDN 1/C Holcomb
Object Detection	MIDN 1/C Holcomb
Color Re-sampler with Box Drawing	Dr. Ngo and MIDN 1/C Holcomb

A.3 Development Costs

Object Name / Price	Description
Commercial Camera $100	Off-the-shelf commercial camera; no specific requirements, cost of camera in actual system will vary depending on user requirements for quality. Camera for project was already in possession of Electrical and Computer Engineering Department and did not require additional funds.
Altera DE2-70 Board $329	FPGA with development kit; in actual production only the FPGA itself will be necessary, greatly reducing cost. DE2-70 board was already in possession of Electrical and Computer Engineering Department and did not require additional funds.