SOC FOR REAL – TIME OBJECT TRACKING IN 3D SPACE

By

Rares Raducu

Bachelor of Engineering, Ryerson University, 2011

A project

presented to Ryerson University

in partial fulfillment of the

requirements for the degree of

Master of Engineering

in the Program of

Electrical and Computer Engineering

Toronto, Ontario, Canada, 2014

Author's Declaration

I hereby declare that I am the sole author of this project. This is a true copy of the project, including any required final revisions, as accepted by my examiners.

I authorize Ryerson University to lend this project to other institutions of individuals for the purpose of scholarly research.

I further authorize Ryerson University to reproduce this project by photocopy or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.

I understand that my project may be made electronically available to the public.

SoC for Real – Time Object Tracking in 3D Space

Master of Engineering 2014

Rares Raducu

Electrical and Computer Engineering

Ryerson University

Abstract

With the rapid growth of workplaces, there is also an increase of the risk employees are exposed to. A high percentage of the injuries suffered annually are work related and many of those are due to the decrease in alertness of employees as they get tired. As a result, many of the injuries are fall related, touching a hot object etc. Therefore, safety in the workplace can be increased if employees can be monitored continuously and warning them when they come close to a restricted (hazardous) area.

The current paper presents an in-depth analysis of existing approaches to 3D object tracking which can be deployed to monitor workers and try to prevent injury. The final goal is to design, implement and test an SoC capable of performing real time object tracking in 3D which can send the coordinates to a standalone computer to be displayed and processed.

Acknowledgement

The completion of this work would have never been possible without the consistent help and support of my family, professors, friends and colleagues.

I would like to express my sincere gratitude to my supervisor, Dr. Lev Kirischian, for the support, patient guidance, enthusiasm, and immense knowledge he has provided me throughout my time as his student.

I would also like to thank my friends and colleagues in the lab: Victor Dumitriu, David Diaz and Milad Mehrabi for their advice and sharing of their knowledge throughout my studies.

Lastly and most importantly, I would express my deep sense of gratitude to my parents who have always believed in me, stood by me like a pillar in times of need and who gave up their own dreams to help me achieve mine.

Dedication

This work is dedicated to family and friends who have always loved and supported me unconditionally.

Contents

1. Thesis Introduction 1

1.1Motivation 1

1.2Objectives 2

1.3Contributions 3

1.4Thesis Organization 4

2. Related Works 5

2.1Introduction 5

2.2Radar and Lidar based 3D tracking 6

2.3Stereo vision 7

2.3.1 Human binocular vision 8

2.3.2 Rectification 10

2.3.3 Object tracking 13

2.4Summary 18

3. Architecture organization of the 3D Object tracking system 19

3.1Introduction 19

3.2Concept and theory analysis 21

3.2.1 Lidars and Radars 21

3.2.2 Computer vision 22

3.2.3 Determination of approach 26

3.3System architecture 31

3.4Summary 33

4. Implementation of the 3D Object Tracking System 34

4.1Introduction 34

4.2System implementation 35

4.2.1 Color transformation 36

4.2.2 Average block 39

4.2.3 Frame Storage 43
4.2.4 Subtraction block 45
4.2.5 Blob detection 47
4.2.6 Coordinate calculation and filtering 51
4.2.7 Transfer block 54
4.3Summary 55
5. Experimental Results Analysis and Discussion 56
5.1Introduction 56
5.2Experimental setup 57
5.3Timing Analysis 58
5.4Area and power consumption 60
5.5Summary 63
5.6Conclusion ..... **Error! Bookmark not defined.**
Bibliography 65

List of Figures

Figure 2. 1: Anatomy of human eye 8
Figure 2. 2: Field of view 9
Figure 2. 3: the rectification process: a) distorted images from left and right sensor, b) fixed lens distortion, c) rectified (coplanar) images 10
Figure 2. 4: HSV color space 16
Figure 2. 5: Algorithm to convert RGB to HSV 17
Figure 2. 6: RGB to YUV conversion matrix 17
Figure 2. 7: Gaussian filters 18

Figure 3. 1:Stereo vision coverage area [37] 24
Figure 3. 2: Overview of the design 30

Figure 4. 1: Acquisition block 35
Figure 4. 2: Color Transformation Block 36
Figure 4. 3: Flow chart for color transformation 37
Figure 4. 4: Average block 39
Figure 4. 5: Block diagram for average block 40
Figure 4. 6: Buffer used to compute the average of 8x8 blocks 41
Figure 4. 7: Storage block 44
Figure 4. 8: Subtraction block 46
Figure 4. 9: Block diagram for subtraction block 46
Figure 4. 10: Blob detection 48
Figure 4. 11: Block diagram for blob detection 49
Figure 4. 12: Coordinate calculation and filtering 51
Figure 4. 13: Block diagram for coordinate calculation 52
Figure 4. 14: Filter design 53
Figure 4. 15: Transfer block 54

Figure 5. 1: Overview of MARS platform 58
Figure 5. 2: Timing: last coordinate is sent before front porch 59
Figure 5. 3: Summary of the whole design 62

Figure 5. 4: Summary of Bayer Pattern converter 62

List of Tables

Table 2. 1: resource comparison 13

Table 4. 1: Input signals of the color transformation block 38
Table 4. 2: Output signals of the color transformation block 38
Table 4. 3: Different pixels in a block 40
Table 4. 4: Input signals of the average block 42
Table 4. 5: Output signals of the average block 43
Table 4. 6: Input signals of the storage block 44
Table 4. 7: Output signals of the storage block 45
Table 4. 8: Input signals of the subtraction block 47
Table 4. 9: Output signals of the subtraction block 47
Table 4. 10: Input signals of the blob detection block 50
Table 4. 11: Output signals of the blob detection block 50
Table 4. 12: Input signals of the filtering block 52
Table 4. 13: Output signals of the filtering block 53
Table 4. 14: Input signals of the transfer block 55
Table 4. 15: Output signals of the transfer block 55

Table 5. 1: Current consumption comparison 61

List of Abbreviations

AI	Artificial Intelligence
BRAM	Block RAM
CC	Clock Cycle
DSP	Digital Signal Processing
FIFO	First In First Out
FPGA	Field Programmable Gate Array
FPS	Frames Per Second
HDL	Hardware Description Language
HSV	Hue Saturation Lightness Value
Hsync	Horizontal Sync signal (VGA)
IP	Intellectual Property
Mbps	Megabit per second
MBps	Megabyte per second
PC	Personal Computer
uC	Microcontroller
CamShift	Continuously Adaptive MeanShift
RAM	Random Access Memory
RGB	Red Green Blue
SoC	System – on Chip
UART	Universal Asynchronous Receiver – Transmitter
VGA	Video Graphics Array
Vsync	Vertical Sync signal (VGA)

1. Thesis Introduction

1.1 Motivation

With the rapid development of the work place there is an ever growing hazard to human health and well-being. Between 2002 – 2004 in Canada alone, there have been 465 deaths and close to 300,000 hours of compensated time loss. With more than 66% of those injured being employed in trades, transport, primary industries, processing or manufacturing [1]. Since then, the numbers have increased dramatically. In 2012 there have been 144,865 cases of injury reported in British Columbia alone, for a total of 2.9 million days of work lost [2].

Out of the injuries reported in 2003, close to 19% were cooks or chefs, and over 17% were people working in primary industries such as mining [3] [1] for a total of 36% of the injuries happening to those working in hazardous environments. Furthermore, out of the total injuries reported in Canada in 2003, more than 22% of the injuries were falls, close to 20% were accidental contact with sharp objects, 12% were accidentally struck or crushed by objects and 6% were accidental contacts with hot objects, liquids or gases [4]; representing a total of almost 60% of the total injuries suffered that year. These are types of injuries which can be avoided through continuous supervision of employees and by making them aware of the fact that they are entering or approaching a possible hazardous area of the workplace.

Most surveillance systems used today are not capable of tracking motion in 3D and those which are capable of doing so are very expensive due to the sheer amount of raw data that needs to be processed. Therefore, there is a real need for an inexpensive system which is capable of detecting and tracking motion in 3D which can be used to warn anyone who is approaching a hazardous area in the workplace (such as a hot stove, a hole in the ground etc.), thus helping to prevent injury in the workplace.

Furthermore, such a system can also be used to assist with the automation of processes such as spacecraft docking on the space station, satellite docking, in-flight refueling systems – which is currently done manually by the pilots of the two planes and requires a lot of skill and patience to be done – and many other processes.

1.2 Objectives

The main objective of this work is to create a System – on Chip implementation on an FPGA-based device capable of tracking objects in 3D space, in real time. The following steps need to be taken to achieve the above goals:

1. Research and analyze the existing available approaches in implementation of stereo vision rectification, object tracking and motion tracking in 2D and 3D;

2. Determine possible designs for effective SoC implementations which are capable of tracking objects in 3D

3. Develop the architecture for the system;

4. Design and implement the SoC based on the Xilinx Virtex FPGA platform;

5. Analyze the experimental results associated with the performance parameters;

6. Verify the approach, identify any potential limitations during the test and verification process introduce alternatives for future improvements of the design.

1.3 Contributions

The following contributions were made during the course of the work to meet the objectives stated in section 1.2:

1. Extended literature research in the area of stereo panoramic vision, object tracking and motion detection/tracking.

2. Developed the architecture for the SoC capable of tracking a moving object in 3D (process the images coming from a stereo panoramic camera setup) in real time.

3. Designed the SoC based on Xilinx Virtex 4 FPGA

4. Performed in-depth analysis of the experimental results and observation and recommendations were suggested and discussed for future expansion and implementation of the system.

1.4 Thesis Organization

The remaining organization of this thesis is as follows:

1. Chapter 2 presents an in depth analysis of algorithms currently used to perform object tracking, motion detection/tracking in 2D or 3D and stereo rectification.

2. Chapter 3 presents the proposed architecture of the system which is capable of performing real time tracking of motion in 3D space.

3. Chapter 4 presents an in – depth description of each of the components needed to perform object tracking in 3D space.

4. Chapter 5 presents the observations and experimental results of the proposed system. Additionally, this chapter discusses solutions for further improvement and expansion of the system.

2. Related Works

2.1 Introduction

3D object tracking has many real world applications and as a result, there has been much research in the field in the past few years. There are numerous ways which have been used to perform object tracking in 3D, each of which has its strengths and weaknesses. Some of the most popular methods used to track objects in 3D or in 2D are through use of radar [5], sonar, lidar [6] or through digital image processing (or computer vision) [7]. In this paper

Radar technology has been developed secretly before and during World War 2. It uses radio waves to determine the location, altitude, speed and direction of travel of objects. While it is extensively used in the military, it has many other applications such as air traffic control, radar astronomy, aircraft anti-collision, etc.

Another method which is extensively used for 3D object tracking is through the use of Lidar. While it works in a similar manner to traditional radar, it uses laser beams instead of radio waves.

Computer vision refers to the process of acquiring, processing, analyzing, and understanding images or a stream of images from the real world in order to produce numerical or symbolic information in the forms of decisions [8]. The main objective of computer vision is to copy the human vision system and implement it in an automated environment. As a scientific discipline, computer vision focuses on developing artificial systems which are capable of extracting features or information from images or a stream of images.

Object tracking using computer vision has been one of the most researched topics of the 21st century. Despite the fact that most research has focused on 2D object tracking, this should not be an issue since 3D object tracking is only an extension of this topic. 3D tracking can be obtained by combining the 2D tracking results to extract depth information.

2.2 Radar and Lidar based 3D tracking

Historically, multi-target radar tracking has focused on tracking targets at large distances in the presence of clutter, which it can reliably perform. However, when it comes to tracking objects which are closer to the emitter, radar technology has limitations since an object can have multiple reflection points which cause multiple measurements on the same object. As a consequence there will be large errors in position estimation [5].

Radar systems have a transmitter which emits radio waves. When these waves come into contact with an object, they are reflected or scattered in many directions. It is the waves which are reflected back, towards the transmitter that allow the radar system to detect the object. These reflected waves are picked up by the receiver and are analyzed. If the object which reflected the waves was moving, then the reflected waves have a slight change in frequency due to the Doppler Effect.

Some materials reflect radio waves better than others. Thus, metals (which are electrical conductors) reflect radio waves the best. This is why radar systems are extensively used in the military.

It is possible to use the multiple reflection points of the close range object to generate a more detailed representation of said object. However, this can be done under the assumption that all the reflections of the radio wave were due to a single object. Therefore, it is very hard to track multiple objects at close range [5].

There have been efforts to improve radar technology in such a way to make it possible to track multiple objects in close range. This is necessary for applications such as vehicle collision avoidance. For example, Adam in [9]proposes the use of a Kalman filter in order to track multiple objects in close range using radar technology. This method improved the tracking ability; however, the technology is still not suitable for tracking multiple small objects in close proximity.

Efforts were made to extend on the above idea in order to improve tracking abilities. For example it was proposed to use an Extended Kalman filter which takes measurements in spherical coordinates and then relate them to state vector in Cartesian coordinate system. Through this method researchers were able to track 2 moving objects travelling at constant velocities. Errors were in the range of 15%. However, during experiments, the two objects started moving from different positions. Kalman filter has properties which allow it to track and predict the location of moving objects and therefore would pose a problem if trying to track multiple objects which start from the same location [10].

Despite extensive research efforts, radar is far from being a viable method of accurately tracking objects in close range. Furthermore, radar is an active tracking method which requires generating radio waves in order to track moving objects. This also implies that the system power consumption goes up considerably.

Another, more popular method of tracking object in 3D is through the use of lidar. Lidar works in a similar manner as radar does, the only difference being it employs the use of laser beams instead of radar waves to detect objects. As a result, lidar is not only more suitable in detecting moving objects than traditional radar, but it can also easily differentiate between moving and stationary objects.

One of the main applications for lidars is for vehicle collision-avoidance systems and this is due to their high accuracy in tracking objects in short range [11]. Furthermore, they can operate regardless of the ambient lighting conditions.

However, the main issue with lidar systems comes from the way they operate. The laser beam must scan the environment, horizontally, line by line. This means not only that it has moving components which can be less viable than electronic components, but also that they are slow at acquiring a data from their surroundings compared to other sensors (such as cmos cameras).

In order to improve scanning speeds, multilayer lidars were introduced. These bring the advantage of scanning the frame horizontally and vertically in the same time [12]. However this increases the bandwidth of the data which needs to be processed. Based on experimental data obtained in [6], a single layer lidar which scans a scene at 10Hz requires 370ms on a 2GHz modern processor.

2.3 Stereo vision

Computer stereo vision is the processing of digital images obtained from two cameras in an effort to obtain 3D information. For example, by comparing information about an object from two different vantage points, 3D depth information can be extracted by examination of the relative positions of objects in the two images.

Computer vision strives to emulate the human vision system as close as possible in order to successfully automate everyday tasks such as parking enforcement, security systems, face recognition, etc.

2.3.1Human binocular vision

When the rays of light enter the eye through the cornea and the lens, they are brought to focus. These rays travel to the back of the eye where they are projected on the retina (which contains 100-120 million rods and about 6-7million cones).

Cones are highly perceptive to color and detail, while rods are more sensitive to light intensity. The highest concentration of cones is found in the fovea, which a region on the retina which is specialized in object recognition. The concentration of cones decreases dramatically as we move outside of the fovea. This is the reason why humans have poor vision in the peripheral region. On the other hand, rods are more prominent on the periphery of the retina and are used for night vision, motion detection and detection of large objects.

When light comes into contact with the rods and cones a series of chemical reactions occur, which generate electrical impulses in the optic nerve. These impulses are sent to the brain where they are processed.

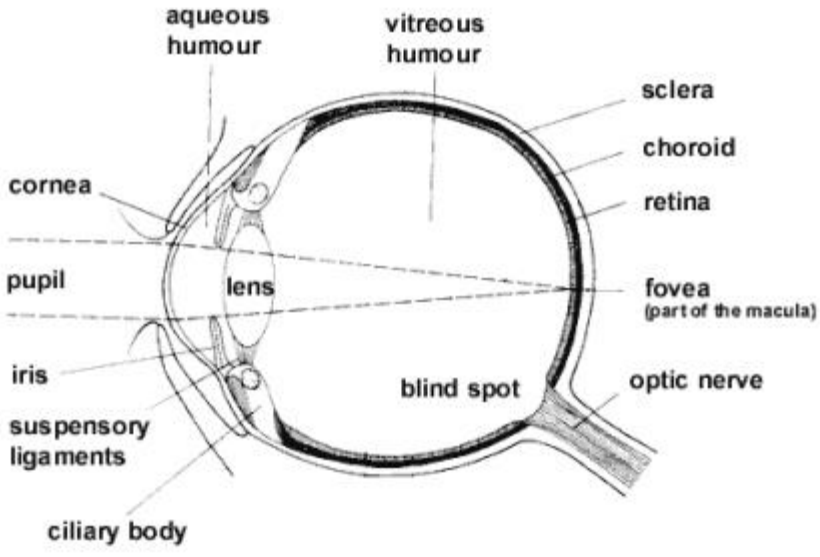


Figure 2. 1: Anatomy of human eye

Each eye acquires individual images of a scene from two slightly different perspectives. This difference is due to the fact that the eyes are placed at a horizontal displacement of about 60-80mm of one

another (this separation is called baseline). When the two images are processed in the brain, it uses the difference between them to obtain depth information as well as the object's spatial location.

The normal human visual field extends to approximately 60 degrees nasally (toward the nose, or inward) from the vertical meridian in each eye, to 100 degrees temporally (away from the nose, or outwards) from the vertical meridian, and approximately 60 degrees above and 75 below the horizontal meridian.

As a result of the anatomy of the human eye, people are only capable of seeing in 3D only in the central visual field and is only capable of detecting motion, light or large objects in peripheral region. The field of view is illustrated in Figure 2. 2.

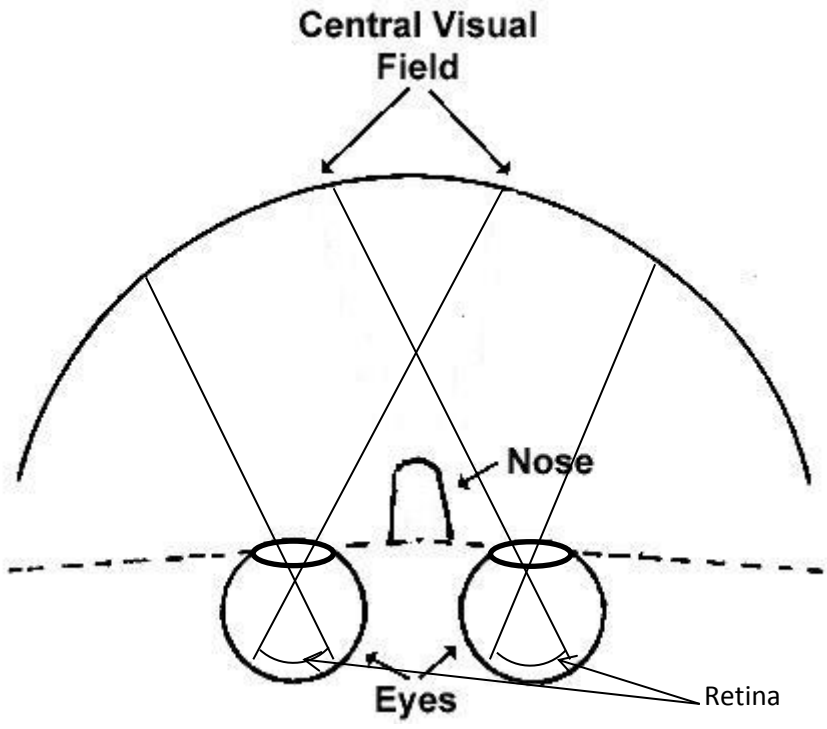


Figure 2. 2: Field of view

A basic understanding of the human vision is required for computer stereo vision because this science tries to emulate the human vision in order to automate different tasks.

2.3.2Rectification

Similar to the human binocular vision, in traditional stereo vision, two cameras, displaced horizontally from one another are used to obtain two differing views on a scene. By comparing these two images, the relative depth information of the scene can be calculated in the form of a disparity map. Disparity is the difference between the horizontal coordinates of the same object as seen by the two cameras [13] [14], and is inversely proportional with the distance between the camera assembly and the observed object [15].

Therefore, in order to compute the disparity map, the system must be able to recognize the same object in both frames and calculate the horizontal difference between the positions of the object in the two frames. In real camera systems however, the images sensor are not completely coplanar and as a result, there is a difference between the vertical coordinates of the objects as well. This is illustrated in Figure 2.3 a) [15].

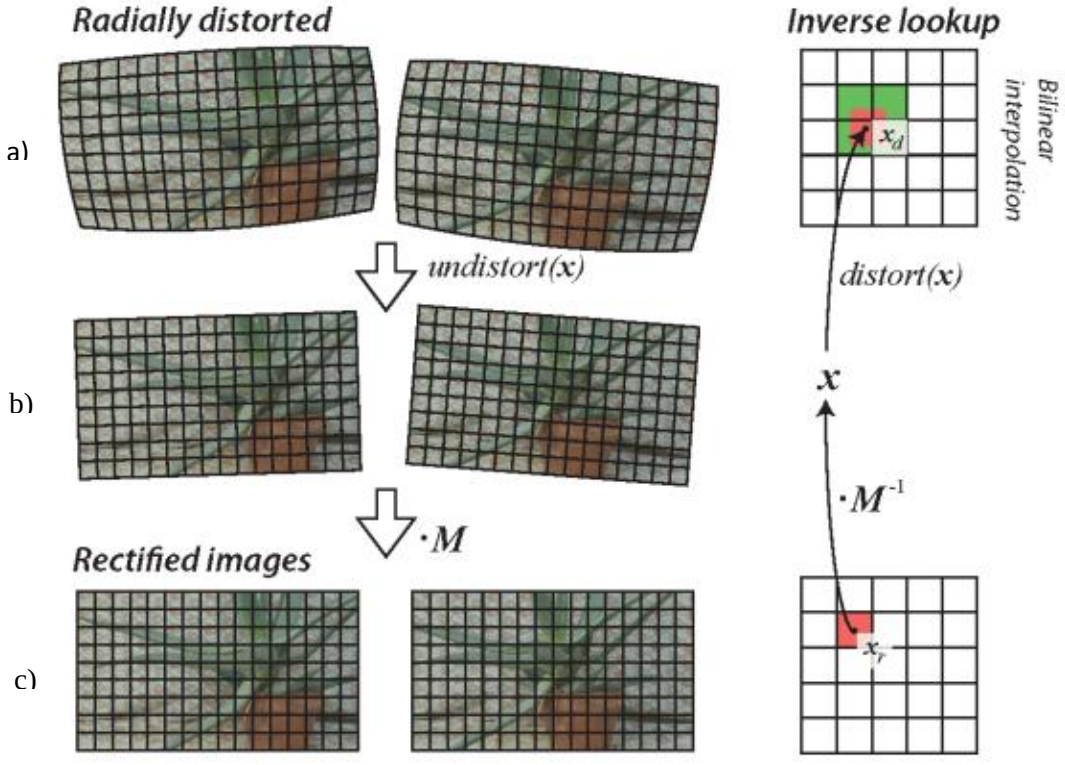


Figure 2. 3: **the rectification process: a) distorted images from left and right sensor, b) fixed lens distortion, c) rectified (coplanar) images**

Stereo image rectification is the process of transforming the images to align them horizontally and obtain perfectly aligned epipolar lines [13]. All the objects in a rectified pair of images have the same vertical position. This is done in order to simplify the already complex process of recognizing the object in both frames [14].

Figure 2.3 illustrates the different stages the rectification process needs to go through before obtaining the final result. As can be seen in a), the images are tilted and have radial distortion (caused by the lenses). The first step is to remove the lens distortion and finally the images are shifted in such a way that they become coplanar and their epipolar lines are parallel (as seen in c)).

Extensive research in the area has led researchers to perfect a standardized approach of performing rectification. Thus, a set of image pairs is used to recursively compute the intrinsic parameter s of the cameras (focal length, baseline etc.) [13].Once the intrinsic parameters are known, they are used to compute a 3x3 matrix – known as transformation matrix – which is used to compute the location of each pixel in the rectified final image [14] [16].

Rectification is performed in a succession of steps as follows:

1. The correspondence problem. The correspondence problem consists in taking a set of points from the image coming from one camera and finding the corresponding set of points in the image coming from the second camera. This is an iterative process, having as input frame pairs coming from the two cameras and outputting a series of points which will be used to find the intrinsic parameters of the camera.
2. Computing the intrinsic parameters of the cameras.
3. Finding the transformation matrix.
4. Apply the transformation matrix to images in order to rectify the images.

While the process is the same, the difference is in the implementation. The preferred approach for many researchers is to transfer all the necessary frames in order to perform the rectification on a PC using software such as MATLAB or using the OpenGL libraries [17]. Once the matrix is known, some continue to use software to perform the rectification stage as well [15], while others prefer to do this real time with the use of an FPGA [17].

The advantage of using software to compute the transformation matrix comes from the extensive support from the active open source community, making it a trivial task to process the frames in order

to obtain the intrinsic parameters of the cameras and find the transformation matrix. Furthermore, this approach also takes advantage of the processing power of modern PCs allowing for computations to be performed very quickly.

However, for many embedded applications, the advantages are overwhelmed by the bandwidth required to transfer the frames necessary for the matrix computation and by the fact that this approach requires continuous access to a PC, which is not desirable for all embedded applications. Moreover, these kind of devices are not suitable for mobile devices or devices which need to be deployed in hazardous environments with lots of objects which can interfere with the wireless communication (such as mines, indoor etc.). Therefore, some researchers prefer to perform the necessary processing on their embedded application [14].

This approach eliminates the need for access to a PC which lowers the overall cost and power consumption of the system. Furthermore, by eliminating the need for a communication interface needed to transfer data to and from the PC, the effective range of a mobile device is increased dramatically while simplifying the hardware and the design process. The result is a well-rounded, standalone embedded application capable of performing all the necessary processing without the aid of any external devices [14]. These kinds of devices can be deployed in any environment since they do not need to communicate with an external device in order to operate properly.

However, since the embedded application needs more processing power, its complexity and cost increases. As a result, there is a tradeoff between the complexity of the algorithm which needs to be employed and the cost and power consumption of the embedded device. Therefore, by decreasing the complexity of the computation algorithm, the cost and power consumption is reduced and as a consequence the accuracy of the rectification process is reduced, potentially increasing the error in the rectification process [14]; on the other hand, by using a complex algorithm for solving the correspondence problem, then the cost and power consumption of the device increase while decreasing the error of the rectification process [18].

For example, in [15] the rectification process was simplified by selecting the images coming from the left camera as reference and applying the algorithm to the images coming from the right camera. As a result, the resources necessary to perform all 4 steps were almost halved. However, as a direct result, the rectified images were tilted and had visible lens distortion.

Moreover, in [18], the researchers used a complex algorithm to solve the correspondence problem and extract the intrinsic parameters of the cameras. As a result, while their results were better than [15] they used a bigger FPGA and 5 times more resources, as seen in Table 2.3.1.

	[18] Jin S., et all	[15]Greisen P. et all
FPGA used	Virtex 4	Spartan 3
BRAMs	322 of 336	28 of 28 [19]
Slices	51,191 of 89,088	9,468 of 26,624
LUTs	60,598 of 178,176	15,969 of 26,624
Slice Flip Flops	53,616 of 178,176	10,788 of 26,624

Table 2. 1: resource comparison

2.3.3Object tracking

Object tracking has many applications in the field of intelligent robots and AI, human – computer interaction, vehicle tracking, biomedical image analysis etc. As a result it has been one of the most researched topics in computer vision in the past few years [20] with much advancement in terms of new algorithms and real time working systems [21]. Unfortunately it is also a very complex task which requires processing of large quantities of data and therefore most research focuses on implementation on stand-alone computing platforms [22] or expensive DSPs [23].

No matter the algorithm, object tracking (in 3D) is performed in 3 steps as follows:

1. Image segmentation – separates the moving object from the background
2. Blob detection – finds the pixels which are a part of the detected moving object
3. Centroid calculation – finds the coordinates of the centroid of the selected blob

Image segmentation in computer vision is the process of breaking down the image into blocks (or segments) [24]. More precisely, image segmentation is the process of labeling each pixel in an image, such that all the pixels with a certain label share a certain characteristic. Segmentation is important in object tracking because it helps distinguish between the pixels of a moving object (which are of interest) and those of the background (which are not of interest).

The difference in object tracking algorithms comes from how each of them performs the image segmentation stage. Thus, there are 2 methods of doing this. The first method is detecting the object

based on a certain feature of the object (such as shape, size, color etc.) which was previously extracted and stored in a database of known objects [25]. The second method, which is the most widely used approach for real time tracking systems is to subtract the current frame from a previously stored frame [20] [23].

There are many approaches when it comes to object tracking. However some are more suitable for the current application then others. For example some researchers propose a method of tracking objects which were first detected based on previously known features [26]. However, this requires additional storage space to store known objects which need to be detected. Furthermore, despite the fact that it does not require a lot of processing power, is not suitable for tracking objects which are not in the database. Moreover, in [27] [21]researchers tried to make the object detection and object tracking stage cooperate with each other and even though he obtained better results than traditional object detection and tracking methods, the additional storage space needed for its operation outweighs the benefits it brings.

In [25] Schulz proposes a method of tracking objects based on color. This method is a very simple one which can be implemented on inexpensive hardware. However, it brings no real benefit other than that. The fact that it relies on color to track objects means that it cannot track objects of different colors or that it will track parts of the static background which is in fact not moving but is of the correct color.

An interesting approach is presented by Nasrullah in [28].He proposes a combination between object detection and frame subtraction which yields really good results. Thus, he uses the subtraction between frames I(t) and I(t-1) to find moving objects and from I(t+1) he continues to search an area around the last known location of the previously detected moving object in previous frames. The advantage of this method is that it performs less overall computations than just continuously subtracting the previous frames while also restricting the movement of the object, not being able to track fast moving objects (which usually are not of interest anyway). However, it can only track one moving object.

Another popular method is the MeanShift algorithm. It was first introduced by Fukunaga in 1975 and since then it has been a favourite for many researchers due to its simple calculations and the theory behind it [29]. Thus nowadays, there are many variations of the algorithm [8], [30], [29] and [31]. The algorithm itself is a nonparametric density gradient statistical method which is used for image segmentation and clustering. But ultimately its popularity comes from its effectiveness in tracking moving objects.

The basic idea behind the algorithm is to use the features (color, intensity and edge histograms) of the tracked object – which were extracted from previous frame – to create a confidence map in the new image, and use MeanShift algorithm to find the pixels with the highest probability of being part of the object in the current frame. The search area is selected around the location of the object in the old frame.

CamShift is an extension of the traditional MeanShift algorithm [8], [30]. Cam shift also utilizes histogram-based object model representation, and iteratively performs a mean shift procedure on the back-projected images generated from new incoming video frames to find the object's location while adjusting the size of the search window at the same time. However, because CamShift can automatically adjust the object scale during tracking, it allows this algorithm to track objects effectively even with target deformation [30], thus making it more successful at tracking objects of interest than the MeanShift algorithm.

In [31] Lu proposes a variant of the MeanShift algorithm which can be implemented on FPGA. The result was that he was able to track an object of interest in real time at 60 fps while using only 3284 logic elements. However, the downside of his approach comes from limitations of the MeanShift algorithm. MeanShift (and a result CamShift) was designed to track a single object of interest at a time, making it unsuitable for applications where tracking of multiple objects is important. While it can be extended to be able to track multiple moving objects the resources required to do so make it an option which is less viable than other methods.

The most widely used approach is to subtract the current frame from a previously stored frame. This stored frame can be a dynamically reconstructed background [32], or the frame before the current frame [17] and [28] or a combination of both [7].

Background subtraction is achieved by taking the absolute difference between the pixels of the current frame and the pixels of the background [26]. However, changes in illumination, weather and shadows make segmentation a difficult problem. The effect of these problems can be reduced by introducing a background learning algorithm which continuously updates the background image. However, a good algorithm requires a lot of processing power. Therefore, a successful model should provide acceptable adaptive background and satisfy the timing requirements [7].

Subtraction of consecutive frames, as the name suggests, is achieved by taking the absolute difference between the pixels of two consecutive frames. If the absolute result of the subtraction is smaller than a

predefined threshold, then the pixel is labeled as part of the background. Otherwise it is part of the moving object. This method is more accurate in environments where illumination and weather changes are a problem [28]. However, the problem with this approach is that it cannot track objects which become stationary within the frame. Furthermore, the result is almost never a homogenous blob but instead, it will have edges and will be empty in the middle [7].

Through the extensive research which has been performed, it was observed that many of the researchers discourage the use of RGB color space due to its high susceptibility to noise and changes in lighting [30], [31] & [33]. It seems that the most reliable color space is HSV [34].

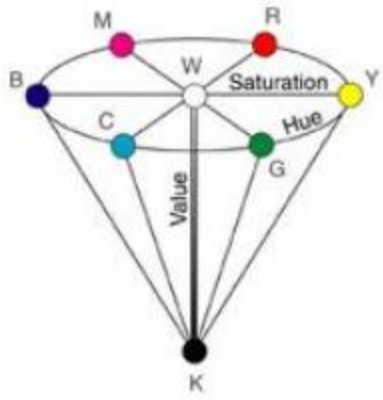


Figure 2. 4: HSV color space

max= max (R, G, B) and min =min (R, G, B)
S=(max-min) /max, V=max
To compute the H value, first compute,

$$R' = \frac{\max - R}{\max - \min} \quad B' = \frac{\max - B}{\max - \min} \quad G' = \frac{\max - G}{\max - \min}$$

If R=max and G=min H=5+B'
elseif R=max and G≠min H=1-G'
elseif G=max and B=min H=R'+1
elseif G=max and B≠min H=3-B'
elseif R=max H=5+G'
otherwise H=5-R'
Then to convert H into degrees Hex=H*60.

Figure 2. 5: Algorithm to convert RGB to HSV

In the HSV color space, each pixel is represented by hue, saturation and lighting intensity value. Thus, it separates the color value (hue) from the other components which are more susceptible to noise and are highly dependent on lighting conditions. The HSV color space is often times represented as a cone, with the hue being the angle from the line representing red as shown in Figure 2. 4 [33]. The algorithm used to convert a pixel from RGB to HSV is shown in Figure 2. 5: Algorithm to convert RGB to HSV [34].

Another proposed color space is YUV. While it is not as insensitive to lighting changes and to noise as HSV, it is an improvement over the traditional RGB. In YUV, the Y component stores the luminance of the pixel while U and V store the color information and some luminance information. Conversion from RGB to YUV is done as shown in Figure 2. 6 [34], [30] & [31].

$$\begin{bmatrix} Y \\ U \\ V \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ -0.147 & -0.289 & 0.436 \\ 0.613 & -0.515 & 0.100 \end{bmatrix} x \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

Figure 2. 6: RGB to YUV conversion matrix

While it adds some insensitivity to lighting changes, YUV is still susceptible to noise. However, the effect of noise can be reduced through some simple noise reductions techniques as the ones shown in Figure

2. 7. While the first filter computes a simple average of a 3x3 block of pixels, the second filter computes a weighted average which gives more importance to the pixel in the center and less importance to the pixels in the corners [28].

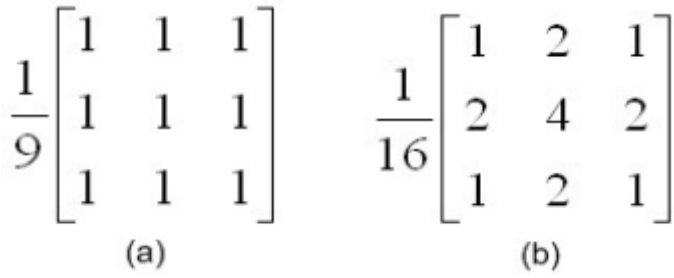


Figure 2. 7: Gaussian filters

These filters take advantage of the fact that in an image, pixels in close proximity to each other are likely part of the same object, and thus should have a similar value. Therefore, in case one of the pixels value was compromised by noise, the noise can be reduced by using the pixels around it.

2.4 Summary

Radar and lidar technology has been used traditionally to detect large objects at long distances. However, with the rapid growth of many industries has driven many researchers to find new applications for the technology. The greatest demand comes from the car industry, where they are being used in collision avoidance systems. Despite the fact that radar technology has issues tracking multiple moving objects in 3D, this is not an issue because collision avoidance systems need only detect objects which are coming close to the vehicle.

Object tracking is one of the most researched topics in computer, vision having many contributors. Extensive research was performed in order to determine the most appropriate approach for the current application. There are many suitable candidates for the required implementation and a thorough analysis will be performed in order to determine the most suitable approach for the 3D object tracking application.

While most researchers proposed their method of performing object tracking, some also made recommendations on approaches which can be used to improve processing results, such as using a different color space than RGB or different noise reduction techniques.

3. Architecture organization of the 3D Object tracking system

3.1 Introduction

Extensive research has been conducted in the field of object tracking and 3D object tracking. With so many different algorithms and approaches which were developed over the years, thorough analysis must be performed in order to determine the most appropriate approach. Therefore, before creating a design for the 3D object tracking system, it is important to know its technical requirements. Based on the requirements, a decision can be made about which tracking option is the most suitable.

As mentioned previously, the main application of the current device is to protect employees in the workplace by warning them when they are getting close to a possible hazard. For example a worker who is working in the kitchen often times is very busy and is required to move large objects from one side of the kitchen to another. These objects are large enough to obstruct his view, thus making it possible for him to touch a hot stove and injure himself.

Therefore, the application must run at real time speeds, which for human vision is approximately 30fps. The tracking accuracy in 2D (horizontal and vertical coordinates) and 3D (depth coordinate) has to be quite high, with an error rate less than 5%. While the start-up delay is not important, the system must have a response time of at least 0.25s to ensure that accidents are prevented.

By continuously monitoring the position of moving objects in 3D, the device, knowing the location of the stove (which never changes), can easily alert the worker and thus prevent the injury. As a result, monitoring must be made in real time and the device should be able to monitor multiple objects of different shapes, sizes and speed of motion and be able to do so at close range (less than 25m).

Another important requirement of the system is to perform its task in a cost efficient manner. This implies both that the overall cost of the system is low and that its power requirements are kept at a minimum.

Summary of the requirements:

a) **SoC which has to minimize system cost and power consumption;**

b) Tracking of multiple moving objects at speeds of at least 30fps;

c) Being able to track these objects in 3 dimensions (vertical, horizontal and depth);

d) Tracking should be done for distances within 25m of the device;

e) **Alert in the case where a moving object comes close to a hazardous area; and**

f) **Be able to cover an area of at least 200 sq feet.**

3.2 Concept and theory analysis

3.2.1 Lidars and Radars

As discussed in Chapter 2, both radars and lidars work in a similar manner. They generate some sort of waves and detect the reflection of the mentioned waves to detect moving or stationary objects. While radar technology was developed to detect object at great distances, efforts were made to make it appropriate for tracking objects at short distances also. However, as shown in the literature report, the results which were obtained show that this approach is not suitable for accurately tracking multiple moving objects in short range [10]. As a result, for this reason alone, radars cannot be suitably used for the mentioned application.

A more suitable approach is to use lidars. Lidar technology enables the tracking of multiple objects in 3D at distances which are less than 50m and in 2D at greater distances. Lidar technology boasts small error rates for tracking objects in close range regardless of the ambient lighting conditions [11], [12]. Moreover, they are capable of covering large areas [6] with an angle of vision which can exceed 180 degrees.

However, the slow scanning speed and large bandwidth of data which needs to be processed raise the question if lidar technology can be used for real time tracking. In an effort to increase scanning speeds and reach real time processing timing, multilayer lidars were used to scan the environment horizontally and vertically in the same time. However, the vertical coverage of such a system is -1.6 to +1.6 degrees [11], which is not enough for the 3D object tracking system.

Furthermore, for the simple reason that lidar technology relies on the actively generated laser light in order to be able to detect and track objects, their power consumption greatly exceeds that of passive systems which do not need to output power in order to track objects.

Another drawback of the approach is the bandwidth of data which needs to be processed in order to track a moving object through the use of lidar technology. Thus, in order to process the data coming from a single layer lidar system which is capable of monitoring an area of 50x50m requires 370ms on a 2GHz Centrino Duo processor [6]. While it is enough to reach real time constraints, it begs the question

if such an approach is suitable for embedded applications. It is quite obvious that in order to achieve similar or better results in an embedded environment, the application needs to be deployed on an expensive multimedia processor.

3.2.2 Computer vision

Object tracking has always been a hot topic in computer vision. Being an extensively researched topic also means that there are many algorithms which were developed to track objects in a string of digital images. While some methods focus on tracking a single object, others are suitable for tracking multiple objects. However, most of the methods focus on tracking objects in 2D (in a single frame) but this should not be an issue since the depth component which is needed for tracking objects in 3D can be obtained through the processing of the disparity between the position of an object as seen in the image pair coming from 2 cameras, horizontally displaced from one another.

With CMOS camera price dropping every month, this approach is one of the most cost efficient. The bulk of the cost will be shared by image storage and processing components of the system. Moreover, while it is not as accurate as lidar technology, the results which can be obtained from such a system are more than acceptable and as a result is a good compromise between system cost and tracking quality.

However, one of the drawbacks of the approach represents the actual 3D tracking area offered by such a system. In order for an object to be tracked in 3D, it must be visible by both cameras. While the coverage area of the system can be quite large, the issue comes from the fact that such a system cannot track objects which are extremely close to the rig, nor can it track objects which are only visible in one camera.

The accuracy of the depth computation can be increased at the cost of a larger minimum distance at which the system can track an object. Thus, if the cameras are placed further apart, then the accuracy increases but at the cost of an increase in the minimum and maximum distance at which objects can be tracked. This is illustrated in Figure 3. 1. Generally, the maximum distance at which an object can be tracked in 3D is the distance from the rig at which the disparity of the object which is tracked becomes 0.

One of the most obvious differences between human vision and computer vision is that the sensor in computer vision is flat while the retina in human vision is round. Furthermore, for the current application, the system will try to mimic the area around the fovea since it is not necessary to identify the moving object but simply track it.

Another issue with computer vision in general is that it performs poorly in environments where lighting conditions change continuously. Furthermore, due to the electrical properties of the sensors, they are prone to error and as a result and different pixels of the same sensor can have different sensitivities to the 3 main colors. As a result, no 2 sensors will output exactly the same values.

However, as outlined in Chapter 2, there are many fixes which can be applied in order to reduce noise and the effects of illumination changes on computer vision in general. Thus, it is highly recommended that the final system have some sort of filter to reduce the noise of the pixels, and that the computations are not performed on the RGB color space.

As already discussed in Chapter 2, there are 2 types of tracking algorithms which are used to track objects. The first type of algorithm revolves around the idea of extracting the features of the object of interest and tracking it in every successive frame using these extracted features. MeanShift is the most popular algorithm which employs this approach and it is the foundation of many other highly utilized algorithms such as CamShift [30], ensemble tracking [35] etc.

The second type of algorithm tries to segment images by computing the absolute difference between the current frame and a previously stored frame. This stored frame can be a dynamically reconstructed background [26], [32], [36] & [30], or the frame before the current frame [28] or a combination of both [7].

Further analysis of the advantages and disadvantages of each of the approaches needs to be performed in order to determine which one is potentially more viable for the current system.

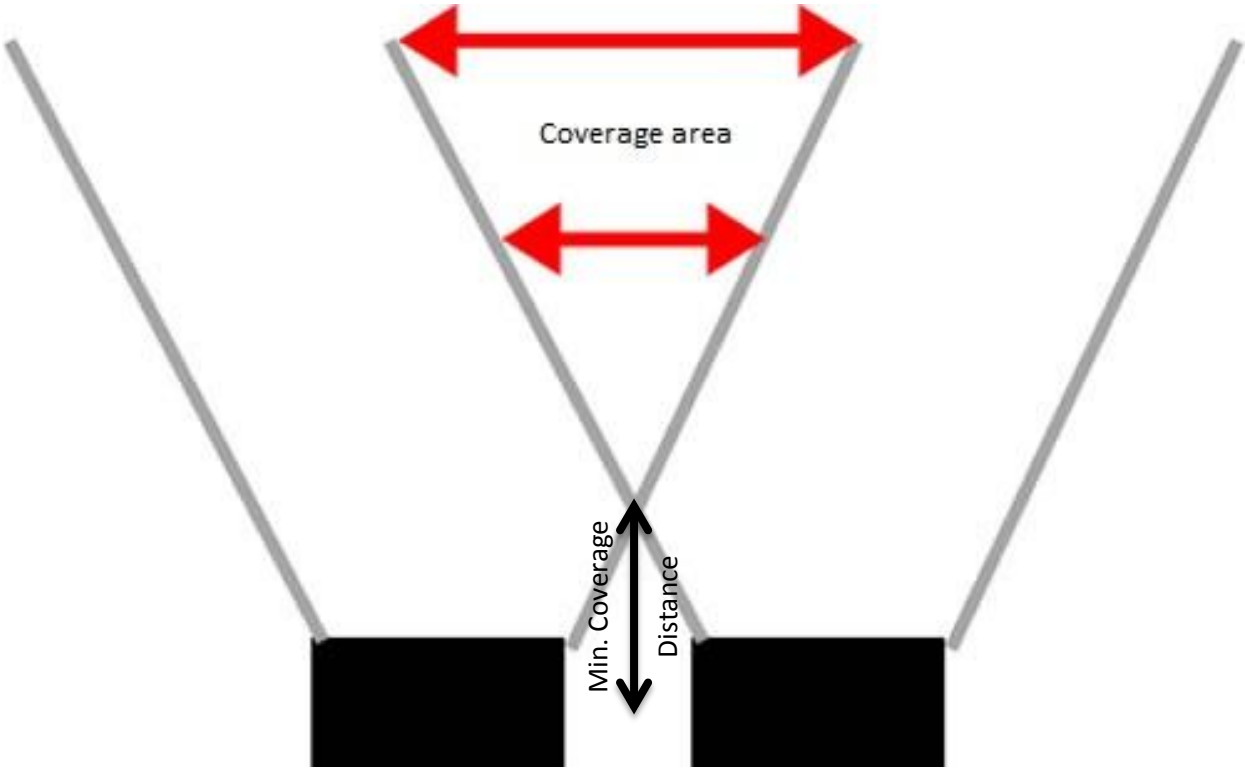


Figure 3. 1:Stereo vision coverage area [37]

Feature extraction

In the case of the MeanShift algorithm, objects are tracked based on features which are stored in a database. These features are stored in the form of color histogram, edge histogram or intensity histogram. However, such a system is limited to tracking objects which are known. Therefore, in order to make it more robust, it was proposed to allow the system to identify which objects are of interest and then extract their features in order to track these objects in future frames [30].

Most researchers focus on the color histogram of the object which offers the best results and is the easier to compute out of the 3. While using all 3 histograms would be ideal and would offer the best results, the extra resources necessary to compute, store and process all three histograms outweigh the results [29], [31].

One of the biggest issues with the MeanShift algorithm is that selecting the object which needs to be tracked is a complex issue. A possible solution to simplifying this could be to manually select the objects

which should be tracked in the frame [38]. However, this is not desirable as the system must be completely autonomous.

Furthermore, it is also complicated to select an appropriate window size. Improper window selection can cause the system to perform poorly. A solution to this problem comes from CamShfit algorithm which allows for the tracking system to dynamically adjust and resize the window.

CamShift performs well in real time processing constraints. Despite the fact that it can reliably track moving objects, it was designed for tracking of a single object. While the algorithm can potentially be extended in order to facilitate the tracking of multiple objects, this implies the use of multiple search windows and will require a more powerful processing unit.

Feature extraction algorithms, due to their properties, perform really well but ultimately their suitability for the current application will be dictated by the processing power of the device and the complexity of the device since it needs additional hardware for storing the features extracted from objects.

Frame subtraction

Another method which is attractive due to its simplicity is to continuously subtract consecutive frames in order to find the differences between them. Assuming that the rig is free of vibration, then it can be concluded that the difference between the frames is the actual moving object. It does not require any complex calculations and can be implemented through the use of a simple subtraction block. Furthermore, its simplicity allows for fast and effective implementation using an FPGA which would allow for impressive frame rates (120fps+) if the process is appropriately pipelined.

However, the downside is that it is very susceptible to noise and change in lighting conditions. This is one of the main reasons in [7] Li had to combine background subtraction with consecutive frame subtraction in an effort to reduce the error rate of the system. By doing this he took advantage of the positive traits of both methods.

Furthermore, while it is easy to find moving objects within the frame, it is a challenge to determine how many objects are present in the frame at a time and calculate their centroids. From this point of view, the feature extraction method has an advantage here. However, even with the added complexity of blob detection, this method requires less complex computations compared to the feature extraction algorithms.

3.2.3Determination of approach

While radar systems are not yet suitable for tracking multiple objects in close range, lidars have the advantage of being very accurate at tracking objects in 3D. However, following the theory analysis, it is evident that their high cost, high power consumption and processing requirements render them not suitable for the current application.

As a consequence, the most viable method is computer vision. While it has its shortcomings, it has many advantages over the use of lidar in terms of power consumption, overall cost and complexity of the system.

Thus, the system will need to be able to acquire digital images from a pair of CMOS cameras. This is not a trivial task since the bandwidth will have to be fairly big in order to achieve real time processing constraints.

Considering a 24 bit camera pair of 1204x768 resolution running at 30 fps the necessary bandwidth to read in the image and process it is:

$$BW = \frac{1024 * 768\ pixel * 24bit/pixel * 30fps}{8\ bit/byte} = 67.5\ MB/s$$

As a result, the current application cannot be deployed on uC because that would require a powerful multimedia processor which are not only expensive but also consume a lot of power. Another option is to deploy the application on an ASIC, but since this is a research project it cannot account for the high cost of the ASIC and as a result it will be deployed on a Xilinx Virtex 4 FPGA.

The main advantage of using an FPGA is that a properly pipelined design can be developed, which will allow for achieving higher than real time speeds. Furthermore, depending on the algorithm which is used, a small FPGA can be employed which will mean that the system will be an inexpensive one.

On the other hand, the development of an FPGA application will be slightly more complicated and time consuming than the development of C code (which can be run on a multimedia processor). However, the high efficiency of an FPGA system means that the cost efficiency will greatly exceed that of a multimedia processor. As a result an FPGA system is the recommended approach for the current application.

The use of an FPGA will simplify many things including the acquisition of images from the camera since this can be done in real time without the need of any additional external hardware.

As discussed in Chapter 2, in order to simplify the process of computing a disparity map, rectification must be performed on the frames coming from the 2 cameras in order to ensure that the two images are coplanar and do not have lens distortion. For the purposes of the 3D object tracker project it is assumed that the images do not have lens distortion and as a result only basic rectification will be needed. This should be enough for the current system, which is a proof of concept.

Most CMOS cameras output the pixel values in RGB on 24 parallel lines. However, it is not necessary to use all 24 bits for further processing. Instead it is sufficient to use the green component. But as shown in Chapter 2, the RGB color space is not reliable enough because it is susceptible to noise and lighting changes. As a result of the literature research YUV or HVS color spaces are recommended instead.

While HSV offers the best performance due to separating the color information from the saturation and light intensity, the transformation itself is complicated to calculate. Since the system needs to be designed as a pipeline it will be very hard to design a pipelined version of the HSV converted which can assure the computation can be performed in 1cc. Not only does it require performing 2 divisions and 1 multiplication per pixel, but these operations are also float point. As a result it is too complicated to be implemented on an FPGA with no multimedia blocks.

While the computation of YUV transformation also requires float point multiplication and division, the process is much less complicated. Furthermore, with slight modifications the transformation can be performed with the use of a few shifts and additions. As a result, the process can be implemented in a 2 stage pipeline which can be used to process 1 pixel per cc.

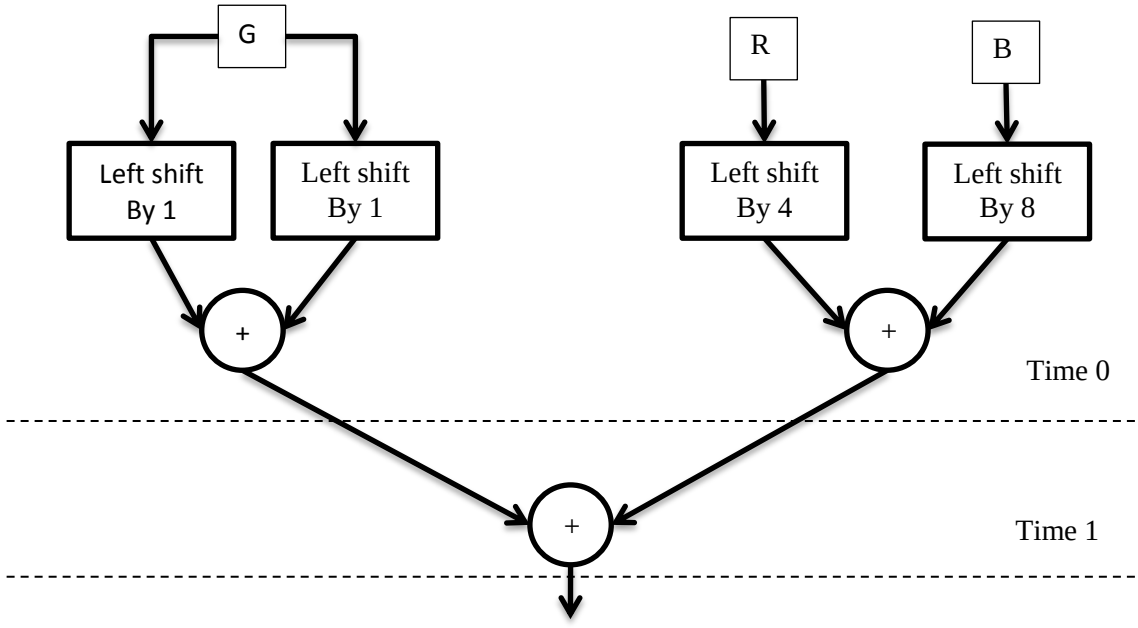
As seen in Figure 2. 6 the transformation needed to calculate the Y component is

$$Y = 0.299\,R + 0.587G + 0.114B \qquad \text{Equation 3. 1}$$

Green is the main component followed by red and finally blue. By keeping this intact and rounding the fractions to the closest divisor of 2, the equation can be simplified to

$$Y = 0.25\,R + 0.625G + 0.125B \qquad \text{Equation 3. 2}$$

Therefore, the computation can be performed in two stages through the use of 4 barrel shifters and 3 adders as follows:



Equation 3. 3: Color space converter

Since the Y component contains the most detail information and is the least susceptible to lighting changes, it will be the only component used for further processing.

Moreover, in order to reduce the effects of noise on pixels and in the same time reduce the storage requirements of the system, the image will be divided in blocks of 8x8. The average value of every such block will be calculated and used as a new image. The resulting image will be of smaller resolution than the original but it will be less noisy.

CamShift algorithm is very accurate at tracking moving objects based on their extracted features. While the algorithm itself can be extended in order to be able to track multiple objects, the process of selecting the object which needs to be tracked is complicated and outweighs the benefits the algorithm

brings. Furthermore, in order to implement it on an FPGA, it is necessary to create an instance of the tracking component for each object that needs to be tracked (i.e. 2 instances to track 2 objects, 5 instances to track 5 objects etc.) which means that in a real life environment the design will scale uncontrollably.

Furthermore, by extracting the object features, the algorithm performs more computation than is needed. The 3D tracker should track any moving object which is of a certain dimension. Because it is extracting the object's features, the CamShift algorithm can be compared to an object recognition algorithm.

A more appropriate algorithm which can be used is the subtraction of consecutive frames. The most important feature of this object is its simplicity. However, for the simple reason that it can detect the motion of any object coming within the frame (independent of shape, size or speed) it is more appropriate for tracking motion than CamShift for the purposes of the current project.

Therefore, the system needs to be able to store two full frames (one coming from each camera) which are necessary for the subtraction algorithm. Two dual ported BRAMs will be needed to perform the storing and reading operation needed for the system to run in real time. In order to ensure that there is no data collision, a read will be performed first and the write stage. Furthermore, the read and write will be delayed and such the read and write are not done on the same cell of the BRAM.

The subtraction operation will be performed on pixels coming from the average and from the storage block. In order to improve the noise reduction, the subtraction block will compute the absolute difference between the inputs and will output whether or not it is greater than a known threshold. Essentially marking the pixels as either being part of the background or of the moving object. Once this is known, a blob detection algorithm can be used to compute the centroid of the object.

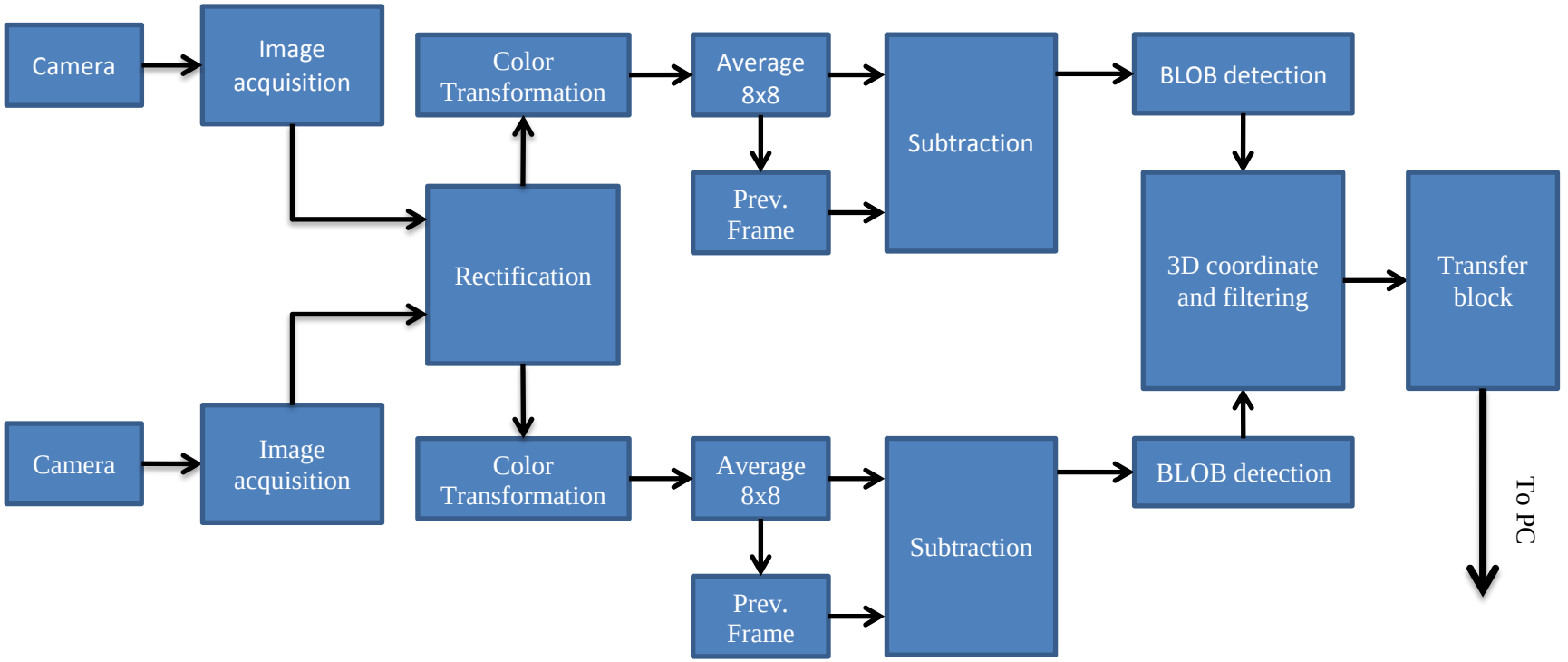


Figure 3. 2: Overview of the design

3.3 System architecture

An overview of the system is shown in Figure 3. 2.

The images coming from the camera are read into the FPGA by the image acquisition blocks. These blocks need to also generate all the signals required for proper VGA operation, in addition it also generates an address number for each visual pixel which is needed to determine the location of the object which is being tracked. Once the images are acquired, a basic rectification is performed. Since the assumption is that there is no lens distortion and that the images are displaced vertically, rectification stage will remove a few lines from both cameras.

The actual processing of the 3D object tracker starts with the color transformation block. As mentioned before, the purpose of this block is to reduce the effect of lighting changes and of noise and operates as described in section 3.2.3.

The color transformation block outputs the value of the Y component in the YUV color space. This signal is fed into the average block which tries to further the noise reduction process by dividing the image in 8x8 blocks and finding the average value of the pixels in each block.

This can be done in real time by having a buffer with enough size to store 8 times fewer entries than a full line (because we need only store the addition of the 8 pixels of each block in the line) and enough bit resolution to be able to store the addition of all 64 pixel values. As an example, assuming a full line has 1024 pixels and each pixel is 8 bits, then the buffer has to be able to store 128 entries of 14 bit each.

As pixels values are coming in, they are added to the value stored in the buffer corresponding to which block the current pixel is part of (which is 0 if the current line is a multiple of 8 i.e. the first line of the 8x8 block). This is done until the last pixel value of the block is added to the sum. Once this happens, the result is shifted 6 times (division by 64) and the result is output for further processing.

The output of the average block is sent to 2 other blocks: storage block and subtraction block. The purpose of the storage block is to output the value of the pixel in the previous frame corresponding to the address of the pixel output by the average block. In order to do so, it must be able to store a full frame of values. In order to avoid data collisions, the block outputs the previous frame pixel before replacing it in memory with the new value.

The subtraction block receives as inputs pixel values from both the storage block and the average block. These pixels are synchronized using the address of the pixel in order to make sure that they are the previous value and the current value of the same pixel. These two values are compared to each other to make sure that they are similar within a certain threshold. If they are not similar then it is concluded that the pixels are part of the moving object. Otherwise they are part of the background.

This decision is input to the next block – the blob detection blob – which accumulates all the pixel addresses of the pixels which are part of the moving object. These are stored in order to compute the centroid of the object. While the accumulation of data can be done in real time, the computation can only be performed after all the data has been accumulated. As a result, the processing is done after the current frame has finished and before the start of the new one. This period of time is indicated by vsync value of 0.

The output of both blob detection blocks (one for each camera feed) is sent to the next block. The 3D coordinate and filtering block must compute the 3D coordinates of the object. In order to smoothen out the transitions and eliminate jumps, the 3D coordinates are averaged on a window of 16 frames. After the first 16 frames have elapsed and after each consecutive frame a new 3D coordinate will be output and sent to the transfer block.

The transfer block has to send the coordinates to the PC for processing.

3.4 Summary

Object tracking is a complex task which requires processing large quantities of data. In order to be performed in real time and in a cost efficient manner, it was decided that the system be implemented on an FPGA.

Considering the constraints of the 3D tracker, a top-down approach was employed in order to come up with the overview of the system. All the approaches found during the literature research were analyzed and the best solution for the purposes of the project was selected.

4. Implementation of the 3D Object Tracking System

4.1 Introduction

The objective of this chapter is to build on the decisions taken in Chapter 3 and come up with a design for all components presented in Figure 3. 2. This is necessary as a preliminary step to the implementation stage of the project. The actual implementation of each component (HDL code) will be attached in the appendix section of the current paper.

4.2 System implementation

As already mentioned in section 3.4, the camera outputs color information in Bayer Pattern. The conversion was performed in the past and the current system merely re-uses the IP. This IP can be identified as the Image Acquisition stage shown if Figure 3. 2 and is tasked with acquiring the images from the CMOS sensors, converting the Bayer Pattern to RGB color space and with generating all the necessary synchronization signals needed for the VGA display. As such, the acquisition block is shown in Figure 4. 1. It is the one which takes the input from the two cameras, converts the Bayer Pattern to RGB, and generates all the necessary control signals for the rest of the system.

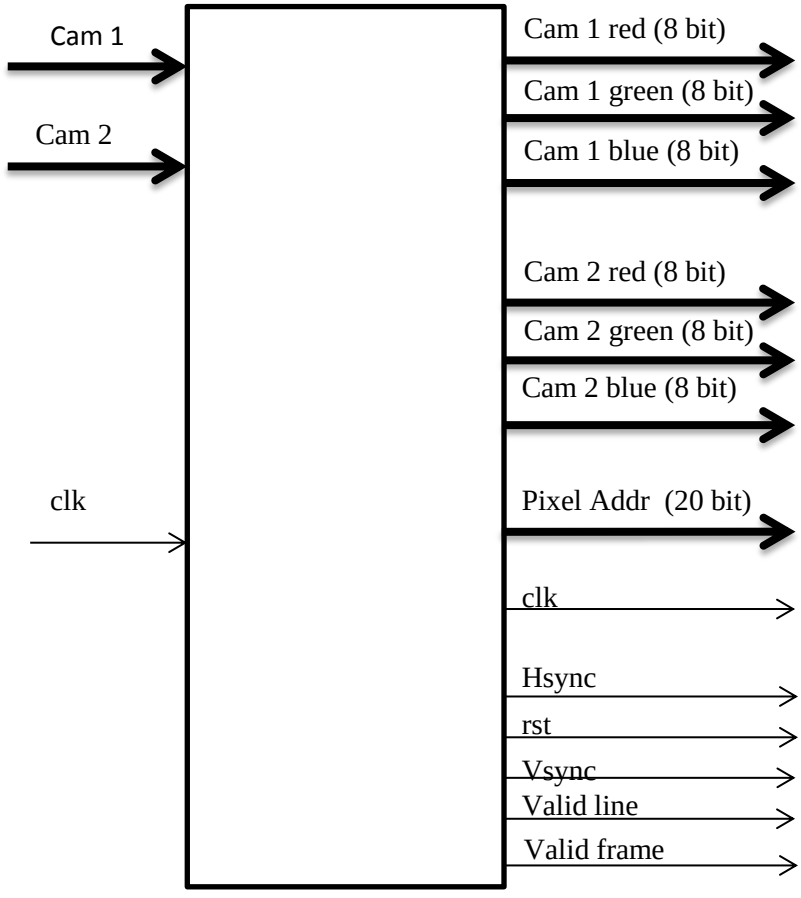


Figure 4. 1: Acquisition block

Thus, it must output the RGB values for the two cameras, the pixel address (which is the address of a specific pixel within the frame), the clock, vertical synchronization signal(vsync), horizontal synchronization signal (hsync), reset signal, line valid signal (which indicates the portion of the line which has valid display data, without the front porch and back porch), and frame valid signal (which

indicates the portion of the frame with valid display data, without the front porch and back porch). The output signals are described in Table 4.1.

The following sections describe the components in detail.

4.2.1Color transformation

The color transformation block transforms the pixel value from RGB to YUV. However, only the Y component needs to be used in the frame processing. This saves a lot of space on the FPGA by reducing the memory which is required to store a full frame (in the storage block), but also by reducing the amount of signals which need to be routed. This allows for the use of a smaller FPGA, saving power and cost – which is beneficial to the scope of the project.

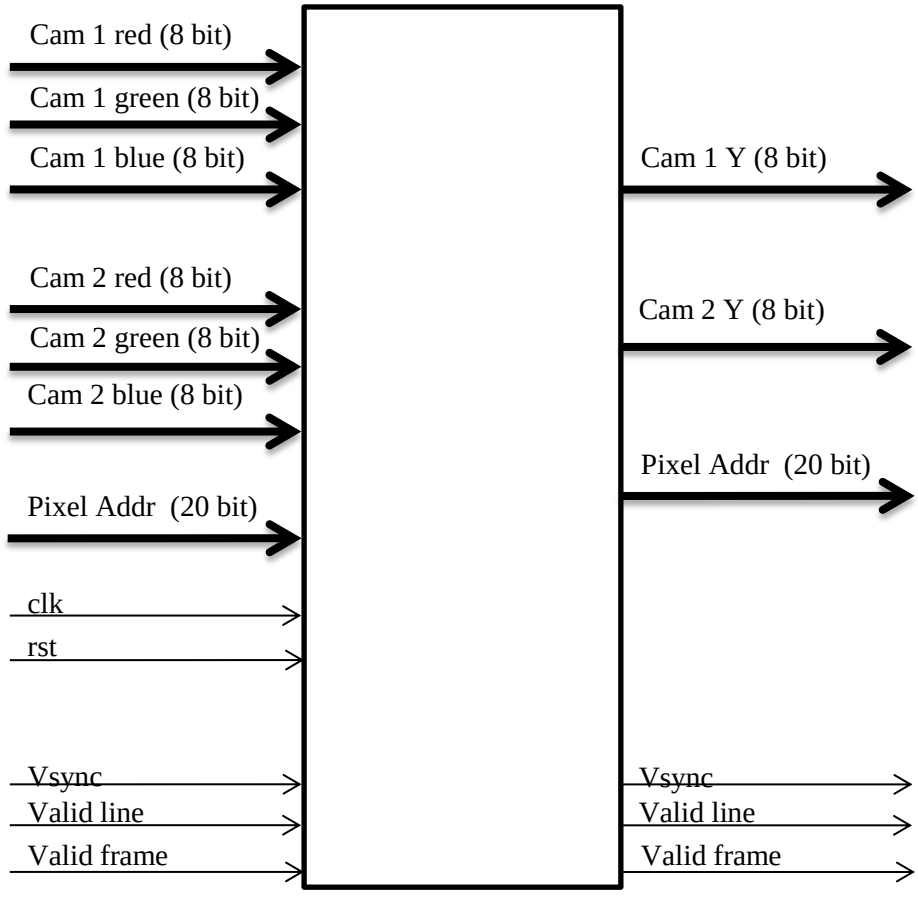


Figure 4. 2: Color Transformation Block

In order to make a more compact system and make the design more efficient by reducing the number of duplicate registers and control logic that is needed, there will be only 1 color transformation block for both cameras. The block will perform color transformation as illustrated in Figure 4. 3.

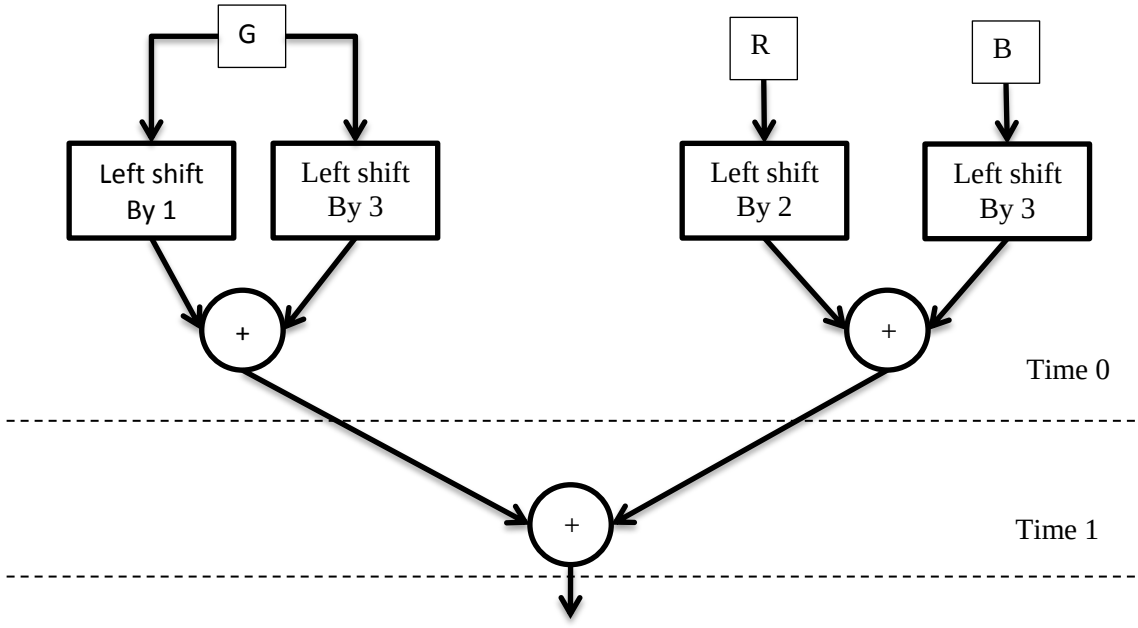


Figure 4. 3: Flow chart for color transformation

As the RGB values are read in, the first step is to calculate the fractions of the colors shown in **EQ 3.2 .** Therefore, the G component is shifted 1 bit to the right (to get 0.5 G) and also by 3 bits to the right (to get 0.125G. The results are added together for a total of 0.625G. In the same time, the R component is shifted 2 bits (0.25R) and the B component is shifted 3 bits (0.125B).). The shifts are performed without delay, by taking the appropriate signal wires and appending '0's to them. The addition; however, will require 1cc to complete. Once the first stage of addition is finished, the results are added together again to obtain the final value which needs to be output.

Therefore, this block needs to take as input the signals described in Table 4. 1.

The incoming pixels are read 1 per cc while the valid line signal is asserted. As a result there is no need for a special latch signal. For each incoming pixel, the block calculates the Y component for the YUV color space in the manner described in Chapter 3. Once processing is done it will output the Y component. Because of the pipelined design, it will output a new value on each cc after the 2cc delay needed to fill up the pipeline. The output signals are shown in Table 4. 2.

Name	Description	Size (Bits)
Left cam red	Red value of the pixel coming from left camera	8
Left cam green	Green value of the pixel coming from left camera	8
Left cam blue	Blue value of the pixel coming from left camera	8
Right cam red	Red value of the pixel coming from right camera	8
Right cam green	Green value of the pixel coming from right camera	8
Right cam blue	Blue value of the pixel coming from right camera	8
Pixel address	Address (location within the visual frame) of the pixel	20
Clock	The system clock (65 MHz)	1
Rst	Reset signal	1
Vsync	Vertical sync	1
Valid line	When this signal is '1', the incoming pixel is part of a valid line in the visual frame	1
Valid frame	When this signal is '1' it indicates that the incoming line must be displayed	1

Table 4. 1: Input signals of the color transformation block

Since there is a 2cc delay (because of the pipeline) all the control signals must be delayed by 2cc before they can be used by further logic. As a result, the Vsync, valid line and valid frame signals are delayed inside using registers and are output for the next stage in the processing pipeline.

Name	Description	Size (Bits)
Left cam Y	Y value of the pixel coming from left camera	8
Right cam Y	Y value of the pixel coming from left camera	8
Pixel address	Address (location within the visual frame) of the pixel	20
Vsync	Vertical sync	1
Valid line	When this signal is '1', the incoming pixel is part of the visual frame	1
Valid frame	When this signal is '1' it indicates that the incoming line must be displayed	1

Table 4. 2: Output signals of the color transformation block

4.2.2Average block

The average block has multiple purposes. First of all it reduces the frame size by dividing the image in 8x8 blocks and finding an average value for each block. By combining them, the resulting image is 64 times smaller than the original image and in the same time less noisy. As a result, the pixel address signal has a smaller resolution (14 bit). The block should also forward the control signals which are delayed to make sure the system is still in sync.

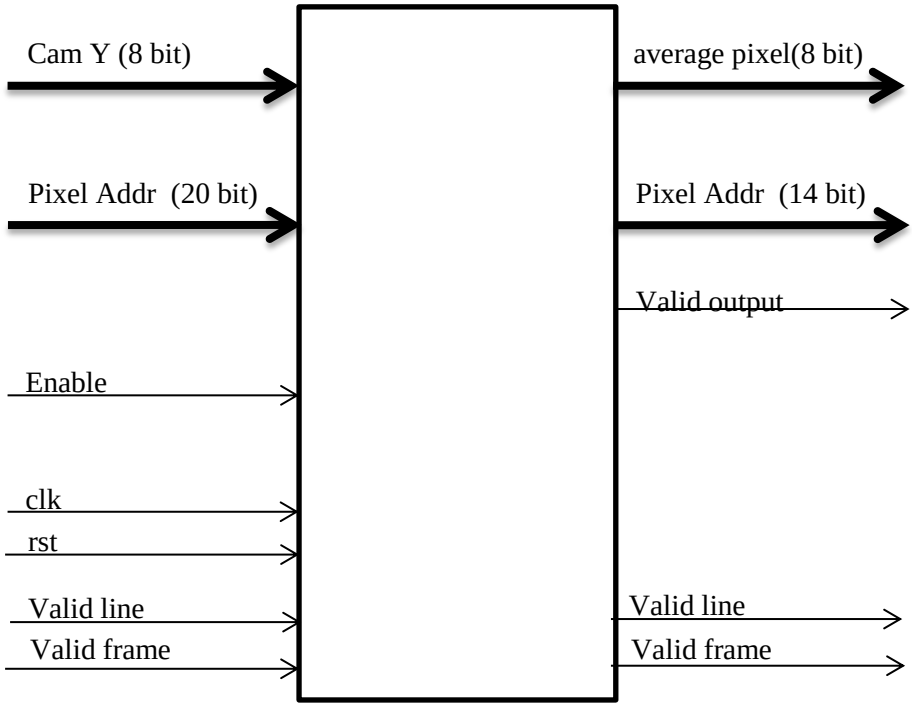


Figure 4. 4: Average block

The average block must run in real time. In order to be able to compute the average of an 8x8 block with each pixel coming in line by line is a complicated matter and needs storage capabilities in order to store the sum of the pixels in the lines that have passed.

The block diagram of the average block is shown in Figure 4. 5. The control block controls the operation of the average block using the address of the pixel. It splits the address in line address (19 downto 10) and in pixel address (9 donto 0) and continuously monitors them. The logic behaves differently depending on the positon of the pixel in the 8x8 block. Since the average block essentially reduces the resolution of the system 64 times, the address of any block in the frame will be given by the address bits

19 downto 13 and 9 downto 3. This means that the temporary sum of each block can be stored in the memory block at the address represented by 9 downto 2.

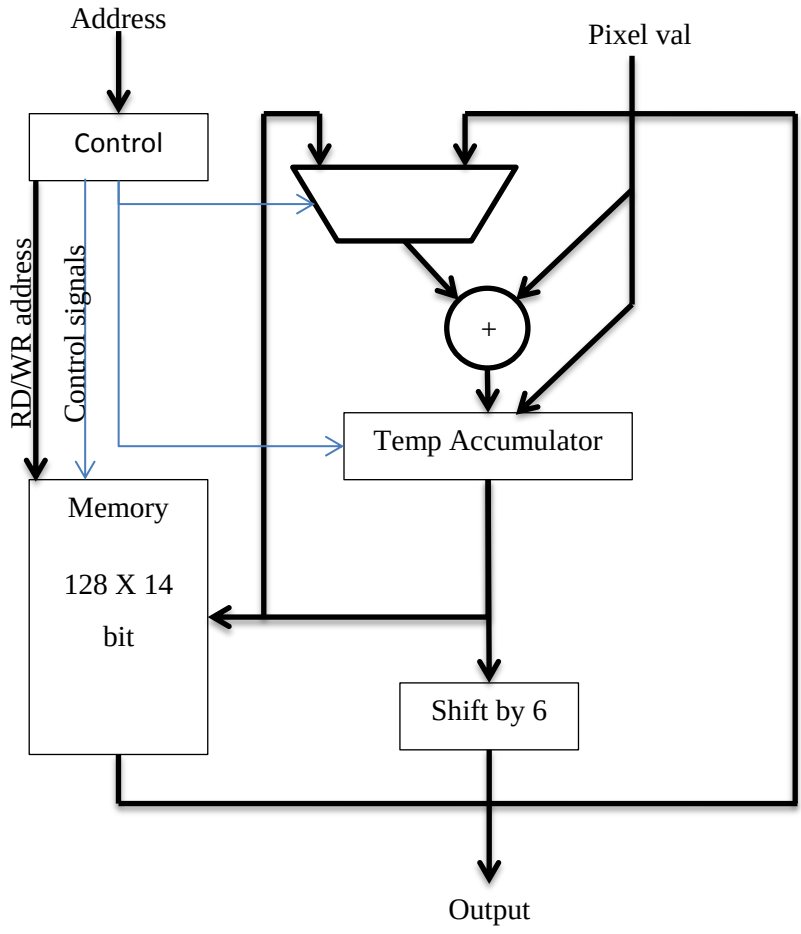


Figure 4. 5: Block diagram for average block

Address (12 downto 0)	Address (2 downto 0)	description
“000”	“000”	The first pixel in the block
“xxx”	“000”	The first pixel on line “xxx” in the block
“111”	“111”	The last pixel in the block
“xxx”	“111”	The last pixel on line “xxx” of the block
“xxx”	Other	All other pixels

Table 4. 3: Different pixels in a block

Furthermore, the control block can differentiate between different pixels in the block by simply monitoring the address field as shown in Table 4. 3.

As a result the average block must behave differently depending on which pixel in the block it is reading in:

a) When the first pixel in the block is read in, it is stored in the temp accumulator.
b) Otherwise, if it's just the first pixel on a specific line, it is must read the temporary sum for the current block from memory and add the new pixel value to it.
c) Otherwise, if it's the last pixel in the block, it must add the incoming pixel to the sum in the temporary accumulator and output the result
d) Otherwise, if it's the last pixel on any other line, it must add the incoming pixel to the sum in the temporary accumulator and store the result in memory at the address indicated by address (9 downto 3) fields.
e) Otherwise if it's any other pixel in the block it must be added to the temporary accumulator.

For example consider the buffer of 128 entries (each of which can store 14 bits) shown in Figure 4. 6. Consider as an example that the pixels of the first line of the frame are coming in 1 by 1. Each pixel of the block is added to a 14bit temporary register (which initially is 0). When the 8^{th} pixel comes in, it is added to the register and after that the result is stored in the 1^{st} location of the buffer and then the register is reset to prepare for the next incoming value. This continues in a similar manner until the whole first line has been split in 128 bins.

When the second line comes in, the previous value is restored and is added to the first pixel of the second line. The result is stored in the temporary register again. And again, all the pixel values are added until the last pixel of the first block on the second line is added to the temporary register. Once this is done the register is again stored in the first position of the buffer. This continues in a similar manner up until the 8^{th} pixel on the 8^{th} line is added to the temporary register. Once this is done, the result is shifted to the right 6 times (division by 64) and the result is output.

1	2	3	4	5	6	….	126	127	128

Figure 4. 6: Buffer used to compute the average of 8x8 blocks

As a consequence, there is an accumulation period (of 7 lines) in which the average block does not output any value. Once the 8^{th} line comes in, a value will be output every 8 cc until all 128 values of the whole block average line is output.

The block uses the pixel address bytes as control. As mentioned previously, the resolution of the camera is 768 X 1020. This means that 10 bits are needed to represent the address on a line and another 10 bits are needed to represent the vertical address. As a result, the last 3 bits (2 downto 0 or Pixel address) of the address are used to indicate when all 8 values of a line have been added together and bits 12 downto 10 are used to indicate whether the current line is the first one, the last one or anything in between.

It is important to differentiate between the first line and the last line in a block because this is how the average block knows when it should reset the temporary register and when to shift its value and output it.

However, since the image is resized, the address of each pixel also has to be shrunk. As a result, the block must remove the above mentioned bits and forward them along with the average value. For convenience purposes the pixel address is also split in its horizontal and vertical component. This is done solely for convenience purposes and is not necessary.

All the input signals are presented below:

Name	Description	Size (Bits)
Cam Y	Y value of the pixel	8
Pixel address	Address (location within the visual frame) of the pixel	20
Enable	Enables the block's operation	1
Clock	System clock	1
Rst	Resets all the internal registers to a known state	1
Valid line	When this signal is '1', the incoming pixel is part of the visual frame	1
Valid frame	When this signal is '1' it indicates that the incoming line must be displayed	1

Table 4. 4: Input signals of the average block

Again, since the pixel values are coming in every cc, there is no need for a latch signal, but instead the valid line signal is used to indicate which pixels are part of the visual frame.

Name	Description	Size (Bits)
Average pixel	The result of the average block	8
Pixel address	New address of the pixel	14
Valid output	Latch indicating that the output is valid	1
Valid line	When this signal is ‘1’, the incoming pixel is part of the visual frame	1
Valid frame	When this signal is ‘1’ it indicates that the incoming line must be displayed	1

Table 4. 5: Output signals of the average block

The outputs are shown in Table 4. 5. Since there no longer is an output every cc, there is a need for a signal which indicates whether or not the output is valid. Otherwise, the block works in a similar manner as all other blocks and has to delay the control signals in order to make sure the system stays synchronized.

4.2.3Frame Storage

The subtraction algorithm relies on the ability to store a previous frame for comparison with the current one. Frame storage block stores the incoming pixels in a 12KB BRAM. In the same time it must output the pixel indicated by the pixel address signal.

The frame storage has to perform 2 tasks. First, every time it receives a valid latch from the average block (the latch signal is driven by the valid output signal of the average block) the block must output the pixel which is stored at the address indicated by the pixel address signal. Secondly, after it has outputted the pixel value, it must also store the new pixel at the same address. These two operation could be done in the same time since the BRAM is dual ported; however, in order to eliminate any possibility of data collision, they are not performed in the same time. First it outputs the old value and then writes the new value.

Furthermore, to ensure that the pixel value coming from the average and the pixel value which is output from memory are synchronized, the storage block outputs both the pixel value in the current frame and the pixel value from the previous frame (i.e. forwards the input it receives from the average block also).

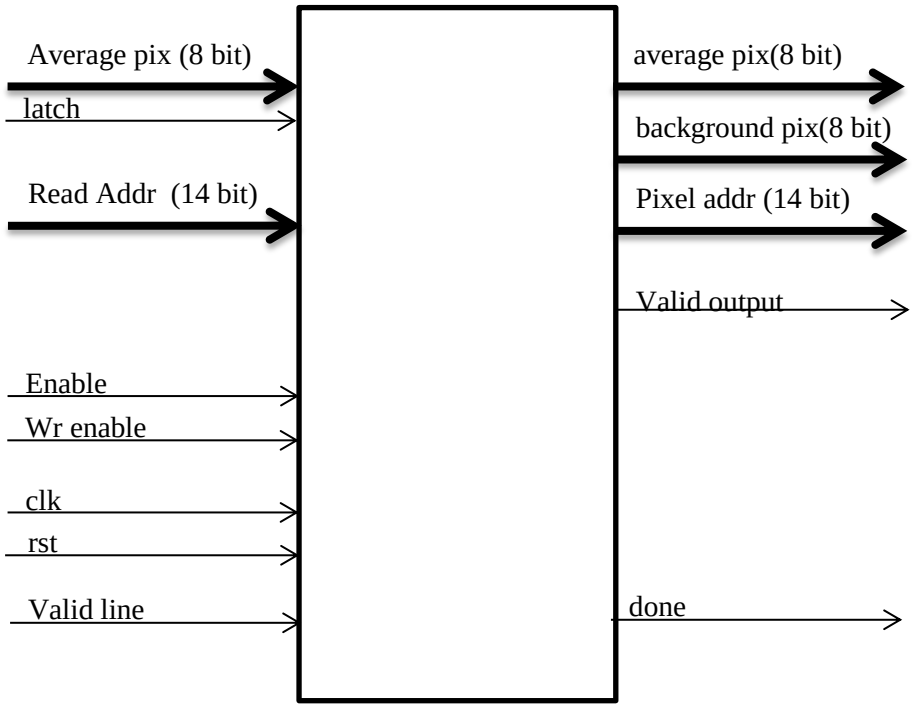


Figure 4. 7: Storage block

Name	Description	Size (Bits)
Average pixel	Pixel value coming from the average block	8
Read address	Address (location within the visual frame) of the pixel	14
Clock	The system clock (65 MHz)	1
Rst	Reset signal	1
Latch	Indicates that the input is valid and needs to be latched in	1
Enable	Enables the block	1
Write enable	The incoming value is stored in the memory when this signal is 1	1
Valid line	When this signal is '1', the incoming pixel is part of the visual frame	1

Table 4. 6: Input signals of the storage block

Input values are latched in every time the latch signal is asserted. On the following cc, the block outputs the value of the pixel in the current frame, the value of the pixel from the previous frame and the

address of the pixel. It also indicates whenever it is done processing a full frame. This signal is used later in the pipeline to initiate the blob detection block.

Name	Description	Size (Bits)
Average pixel	Pixel value of the current frame (coming from the average block)	8
Background pixel	Pixel value of the previous frame	8
Read address	Address (location within the visual frame) of the pixel	14
Valid output	Indicates that the output is valid and needs to be latched in	1
Done	Indicates that the storage block has finished processing a full frame	1

Table 4. 7: Output signals of the storage block

4.2.4Subtraction block

The subtraction block performs the subtraction between the pixel in the current frame and its corresponding one in the previous frame. As a result, it takes the outputs from both the average block (which gives the value of the current pixel) and the output of the frame storage block (which gives the value of the corresponding pixel in the previous frame). These 2 are compared and the block outputs a 1 if the pixel is part of the object or a 0 if it is part of the background.

The control signals are also delayed as they are needed for the next stage.

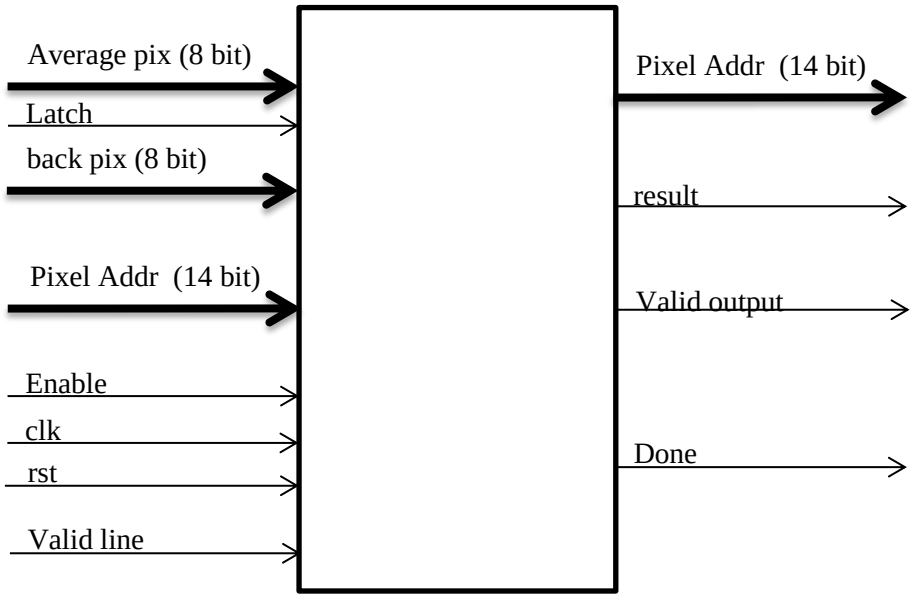


Figure 4. 8: Subtraction block

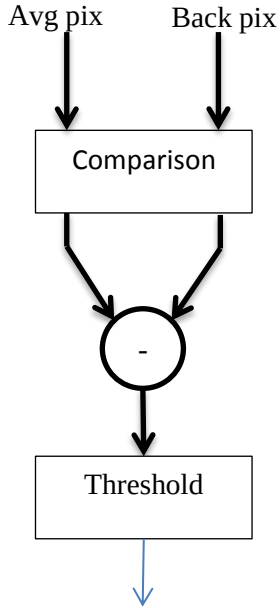


Figure 4. 9: Block diagram for subtraction block

The average block operates in a simplistic manner. Whenever the latch signal is asserted, the average pixel, background pixel and pixel address are latched in. The pixel values are compared to see which

one is greater and a subtraction is performed. If the result of the subtraction is greater than a threshold, then it is concluded that the pixel is part of the moving object and the block outputs 1 on the result line along with the pixel address and it pulses the valid output signal. If the result is less than the threshold then it outputs 0, the pixel address and pulses the valid output. Once the block has finished processing a full frame it pulses the done signal. This process is illustrated in Figure 4. 9.

Name	Description	Size (Bits)
Average pixel	Value of the pixel from the current frame	8
Back pixel	Value of the pixel from the previous frame	8
Read address	Address (location within the visual frame) of the pixel	14
Clock	The system clock (65 MHz)	1
Rst	Reset signal	1
Latch	Indicates that the input is valid and needs to be latched in	1
Enable	Enables the block	1
Valid line	When this signal is '1', the incoming pixel is part of the visual frame	1

Table 4. 8: Input signals of the subtraction block

Name	Description	Size (Bits)
Read address	Address (location within the visual frame) of the pixel	14
Result	The result of the subtraction block	1
Valid output	Is pulsed whenever the result is valid	1
Done	Is pulsed whenever a full frame has been processed	1

Table 4. 9: Output signals of the subtraction block

4.2.5Blob detection

The blob detection block takes as input the output of the subtraction block and must output the coordinates of the blob in the image. The blob detection algorithm is a simple one and assumes that all pixels which were identified as not being part of the background (subtraction block output 1) are part of the same object (i.e. the system can track only 1 moving object). Therefore, as values are latched in

from the subtraction block, the blob detection block finds the left top corner and bottom right corner of the object and based on these computes the size of the object and finally the centroid coordinates.

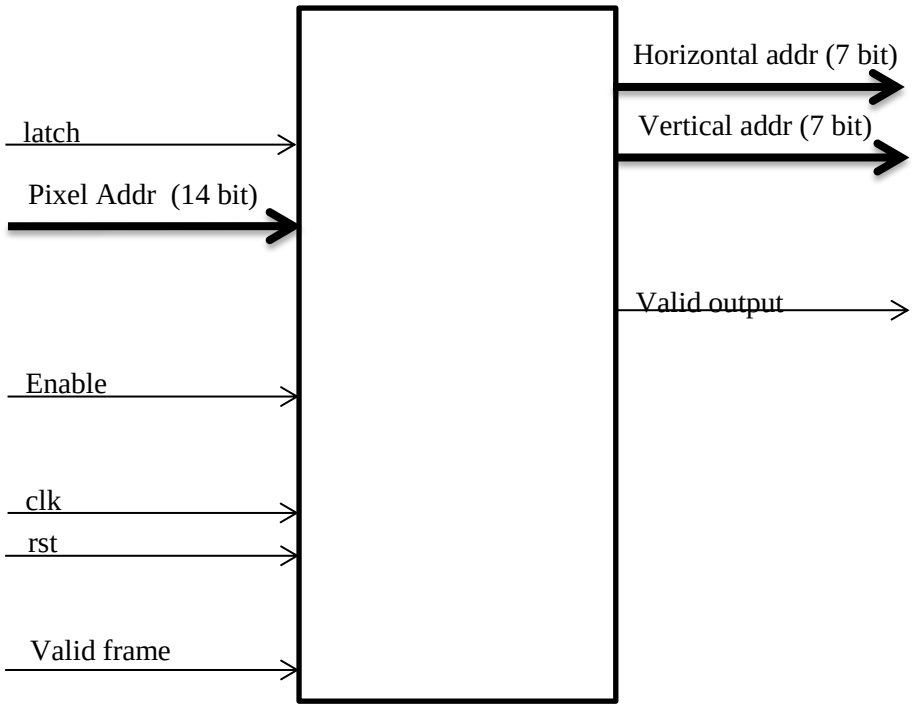


Figure 4. 10: Blob detection

Once the centroid coordinates are found, they are sent to the next block for filtering. Please note that the latch signal is actually driven by the result of the subtraction block, not by its latch signal.

The assumption that all there is only 1 moving object within the frame at a given time greatly simplifies the blob detection block. As a result, in order to find the centroid of the block the object is surrounded by an imaginary box and the top right and bottom left coordinates of this box are calculated.

The right and left boundaries of the box are found by analyzing the pixel address (9 downto 3) whiles the top and bottom boundaries are found by analyzing the line address (19 downto 13). The calculations are performed while in the period of time when the visual frame is being output by the cameras and the results are output when the visual frame has finished (frame valid signal is 0).

Both the vertical and horizontal coordinate are calculated in a similar manner. To calculate the horizontal coordinate, the reset value of the right is 0 while the reset value of the left boundary is 127.

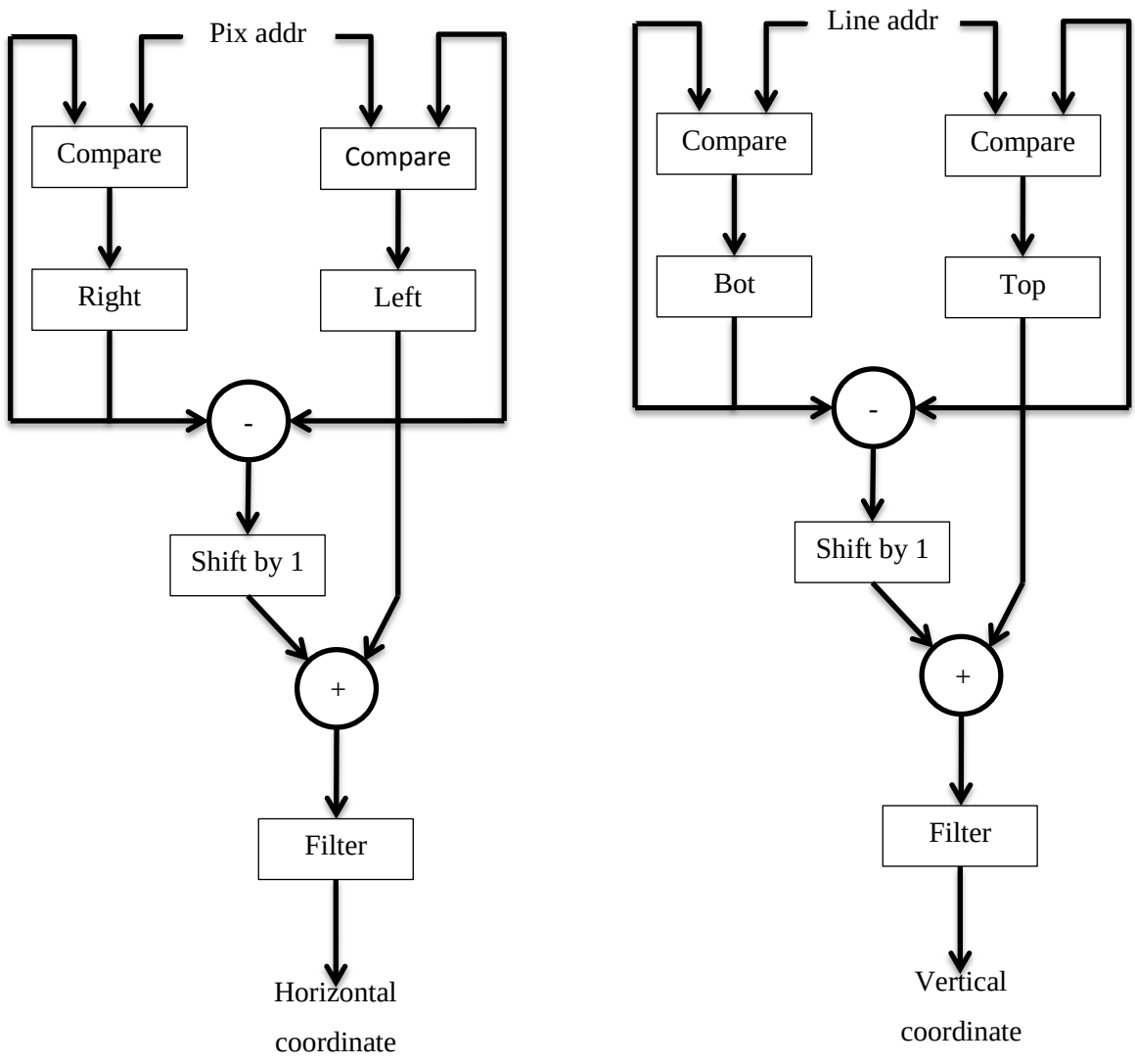


Figure 4. 11: Block diagram for blob detection

Addresses are latched in using the output of the subtraction block. As a result, only the addresses which have the subtraction result as 1 (part of the object) will be latched in.

Every time a new address is latched in, the pixel address is compared with the right and left boundary and will replace the old values if the new value is larger or smaller, respectively. This is done for a complete visual frame. When the valid frame signal becomes 0, the horizontal coordinate is found by subtracting the right and left boundaries. The result is halved (shift by 1 bit) and added to the left

coordinate. The result is sent to a simple 16 value arithmetic filter and then is output to the next block. The filter block diagram is shown in Figure 4. 14.

Name	Description	Size (Bits)
Pixel address	Address (location within the visual frame) of the pixel	14
Clock	The system clock (65 MHz)	1
Rst	Reset signal	1
Latch	Indicates that the input is valid and needs to be latched in	1
Enable	Enables the block	1
Valid line	When this signal is '1', the incoming pixel is part of the visual frame	1

Table 4. 10: Input signals of the blob detection block

Name	Description	Size (Bits)
Horizontal addr	The horizontal coordinate of the centroid	7
Vertical addr	The vertical coordinate of the centroid	7
Valid line	Is pulsed whenever the result is valid	1

Table 4. 11: Output signals of the blob detection block

As already mentioned, there are 2 pipelines in the system. The first one starts with the color transformation and ends with the subtraction block and is performed while the cameras are outputting pixels which are part of the visual frame. The second pipeline starts with the blob detection calculation and ends with the transfer block. This pipeline is executed after the cameras have finished outputting the pixels of the visual frame (during the vertical front porch).

4.2.6Coordinate calculation and filtering

The next block in the pipeline calculates the 3D coordinate. The horizontal and vertical coordinates are taken from the left camera while the depth coordinate is found by subtracting the right horizontal coordinate from the left one.

There are 2 filtering stages. The first one is performed on the horizontal and vertical coordinates coming from the blob detection block. Filtering is done by accumulating 16 values (from 16 consecutive frames) and outputting the average value. The pipelined design is shown in Figure 4. 14.

As seen in Figure 4. 14, when the coordinates come in, they are latched in the 16 value FIFO and in the same time they are added to the subtraction result. This works because the oldest value of the FIFO will be 0 until it gets filled up and as a result, the addition stage is simply adding the 16 values together up until the point when the FIFO is filled. Once the FIFO is filled; however, the result of the subtraction will always be the previous value (the sum of the 16 values currently in the FIFO) minus the oldest value. Therefore, the new addition result will always be the sum of the newest 16 values. Before outputting the result, it is shifted 4 times to the right (division by 16).

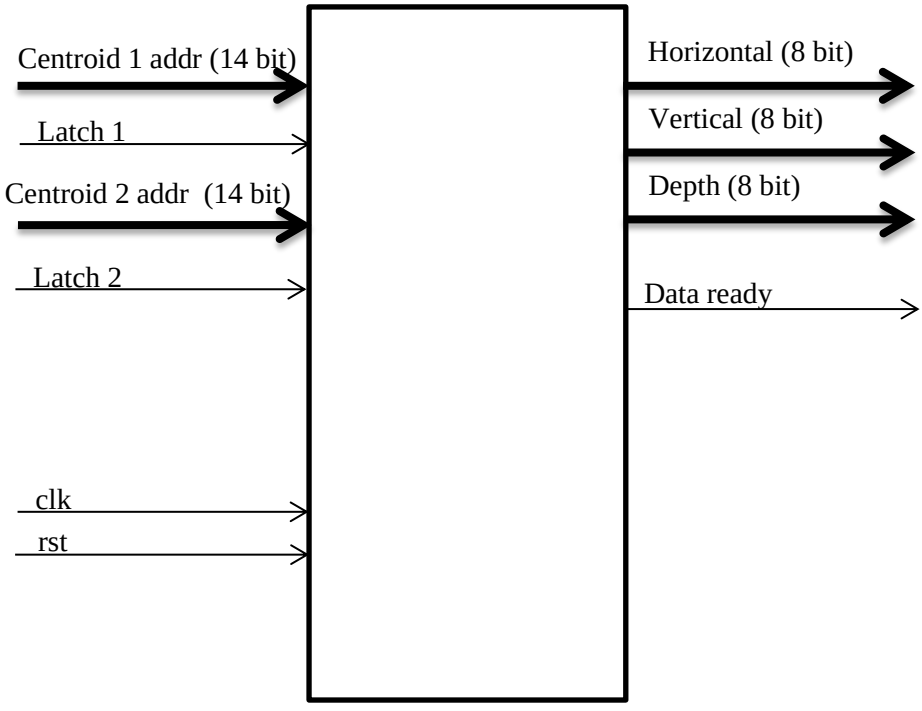


Figure 4. 12: Coordinate calculation and filtering

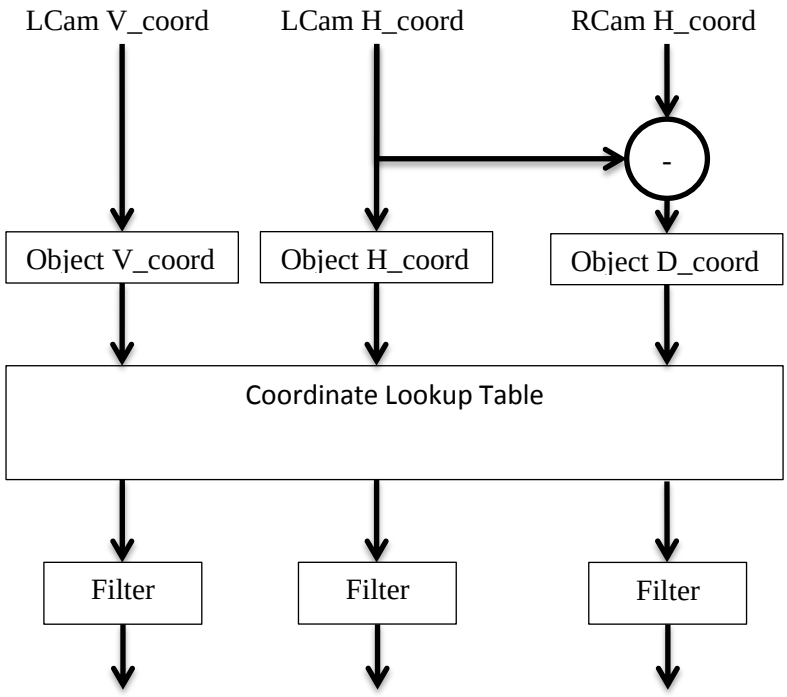


Figure 4. 13: Block diagram for coordinate calculation

Name	Description	Size (Bits)
Cenroid 1 addr	Address of left centroid	14
Cenroid 2 addr	Address of right centroid	14
Clock	The system clock (65 MHz)	1
Rst	Reset signal	1
Latch 1	Indicates that the left centroid is valid	1
Latch 2	Indicates that the right centroid is valid	1

Table 4. 12: Input signals of the filtering block

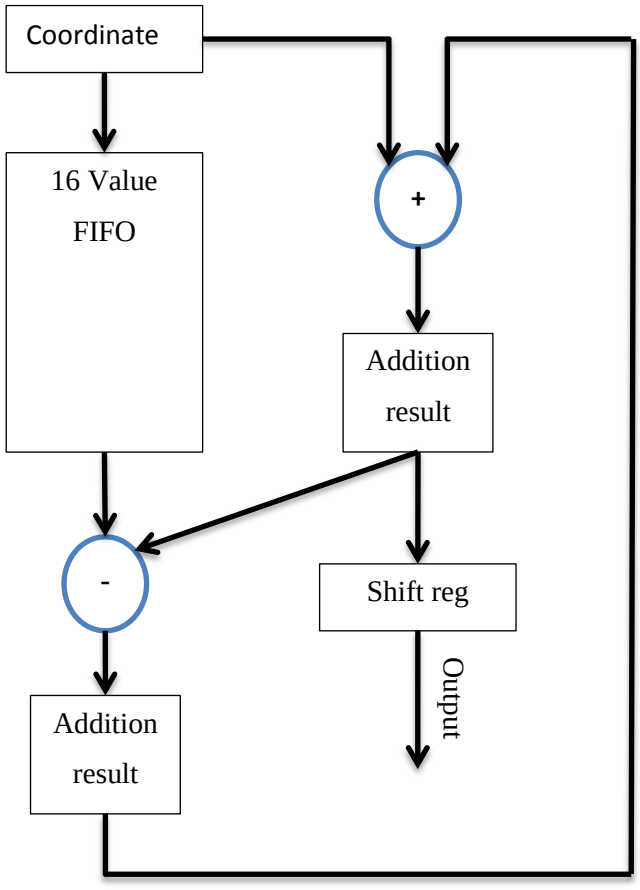


Figure 4. 14: Filter design

Name	Description	Size (Bits)
Horizontal addr	The horizontal coordinate of the centroid	8
Vertical addr	The vertical coordinate of the centroid	8
Depth addr	The vertical coordinate of the centroid	8
Data ready	Is pulsed whenever the result is valid	1

Table 4. 13: Output signals of the filtering block

4.2.7Transfer block

The results from the Blob detection blocks from both cameras are transferred to the pic microcontroller, which in turn sends them to the PC through UART.

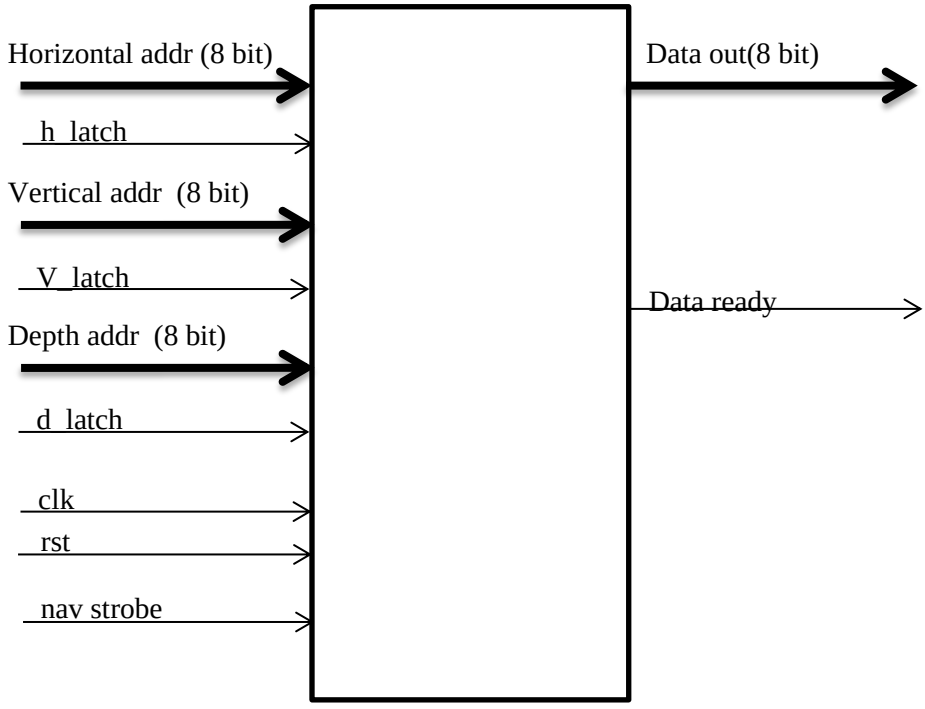


Figure 4. 15: Transfer block

Once the coordinates are received from the filtering block, they are stored in a small, 3 value buffer. When all coordinates are received, the first byte (the horizontal coordinate) is output and the data ready signal is asserted. The next step is to wait for the uC to read in the value and pulse the nav strobe. Once this is done, the data ready signal is set to 0, and the block waits for the nav strobe to be de-asserted before outputting next value. This process is repeated another 2 times for the vertical and depth coordinates respectively.

Name	Description	Size (Bits)
Horizontal addr	Horizontal address of object	8
Vertical addr	Vertical address of object	8
Depth addr	Depth address of object	8

Clock	The system clock (65 MHz)	1
Rst	Reset signal	1
Nav strobe	Feedback signal coming from the uC	1
H_latch	Indicates that the horizontal coordinate is valid	1
V_latch	Indicates that the vertical coordinate is valid	1
D_latch	Indicates that the depth coordinate is valid	1

Table 4. 14: Input signals of the transfer block

Name	Description	Size (Bits)
Data out	Data which needs to be sent to the uC	8
Data ready	Valid data is on the line	1

Table 4. 15: Output signals of the transfer block

4.3 Summary

In this chapter, the design of the system was described in – depth. The system itself was designed as an 8 stage pipeline in order to ensure that each frame is processed before the next one starts. This ensures not only real time operation but also that there is no delay and the response time of the system is fast enough to prevent injury.

The next step is to test the implementation of the system and make sure that it meets the technical requirements presented in Chapter 3.

5. Experimental Results Analysis and Discussion

5.1 Introduction

This chapter presents and discusses the experimental results of the design presented in Chapter 4. The main objective of the conducted experiments was to collect and analyze the performance parameters data of the system such as timing characteristics, power consumption and the occupied area and used resources to investigate the behavior of the proposed design.

This chapter is concluded by recommending ways of improving the results.

5.2 Experimental setup

In order to implement the system and perform the required analysis on the results, an appropriate platform is required. The platform is composed from operational hardware and the configuration/verification tools.

The system was deployed on the MARS platform. An overview of the system is shown in Figure 5. 1. The MARS platform is a custom board which consists of the following elements which are used for the purposes of the 3D object tracking system:

a) Video processor based on the XILINX XCV4LX160 FPGA. This element is the one which performs all the image processing: i) Synchronous reception of the video streams which are needed for the processing; ii) Image processing as outlined in Chapter 4 of this thesis; iii) Send the 3D coordinates for display on the PC
b) Two Cypress CY7C1462AV33 SRAM banks used for video frame buffering
c) One PIC18F8410 microcontroller which is used to facilitate the data transfer between PC and FPGA.

The system configuration/verification tools consist of the following:

Programmers:

- Xilinx Platform USB cable
- Microchip MPLAB ICD3

Software:

- Xilinx ISE 14.2
- Xilins Chipscope Pro 14.2
- Microchip MPLAB X IDE 2.1
- Microsoft Visual Studio Express 2012

Equipment:

- HP 54620C Logic Analyzer
- BK Precision Digital Multi Meter and Power supply

- DELL PC

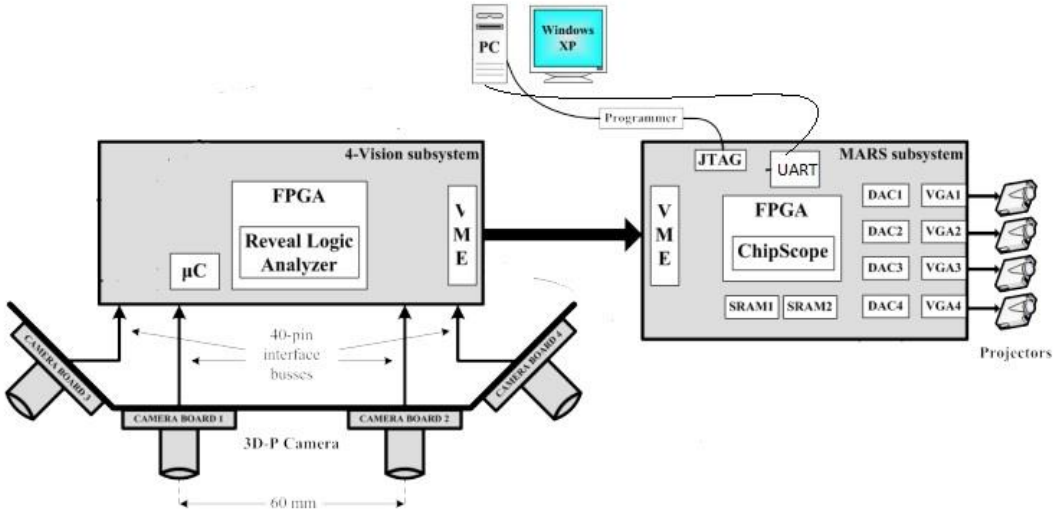


Figure 5. 1: Overview of MARS platform

Communication between FPGA and uC is done through a custom interface. The system is receiving the pixel values from the acquisition block and performs processing in real time. Once the valid frame signal becomes 0, the blob detection block starts computing the centroid of the detected blob and sends the coordinates to the transfer block. Once the coordinates from both cameras are received, the transfer block will assert the data ready signal. This signal is configured to generate an interrupt on the uC and forces the uC to read in the first 8 bit coordinate. Once this is done, the uC asserts the data strobe line to indicate that the data has been read. This is performed 4 times in order to transfer all 4 coordinates.

Communication between uC and PC is done through UART.

5.3 Timing Analysis

As already mentioned, the system was implemented employing an 8 stage pipelined design. This allows the processing to be performed in real time, as the pixels of the new frame are read in by the FPGA using the acquisition block. The actual result is merely delayed by 8 cc (the size of the delay before the pipeline is filled). As a result, all the processing is finished before the frame enters the vertical front porch thus having no issue performing all computations in real time.

The task which is the most time consuming is the transfer from the FPGA to the uC and from the uC to the PC. The reason behind this is that the uC is a basic and operates fairly slowly. However, this should not be a problem since the FPGA runs the all the hardware blocks in parallel and thus can start processing a new incoming frame even if the previous coordinates were not sent yet. However, timing analysis using Chipscope shows clearly that the FPGA finishes sending the 3D coordinates during the vertical sync period (as expected).

The baud rate is limited to 35000 because the uC is run using a 20 MHz oscillator. Furthermore, to ensure correct transmission, the uC appends a header byte and a checksum to the coordinates before sending. Thus, in order to send 6 bytes with 1 start bit and 1 stop bit and no parity will take roughly 2ms.

The cameras are running at 60Hz which means that they are acquiring a new frame every 33ms. Since 2ms is considerably smaller than 33ms, it can be safely assumed that the system is able to process the frame and transfer the data before a new frame is done processing.

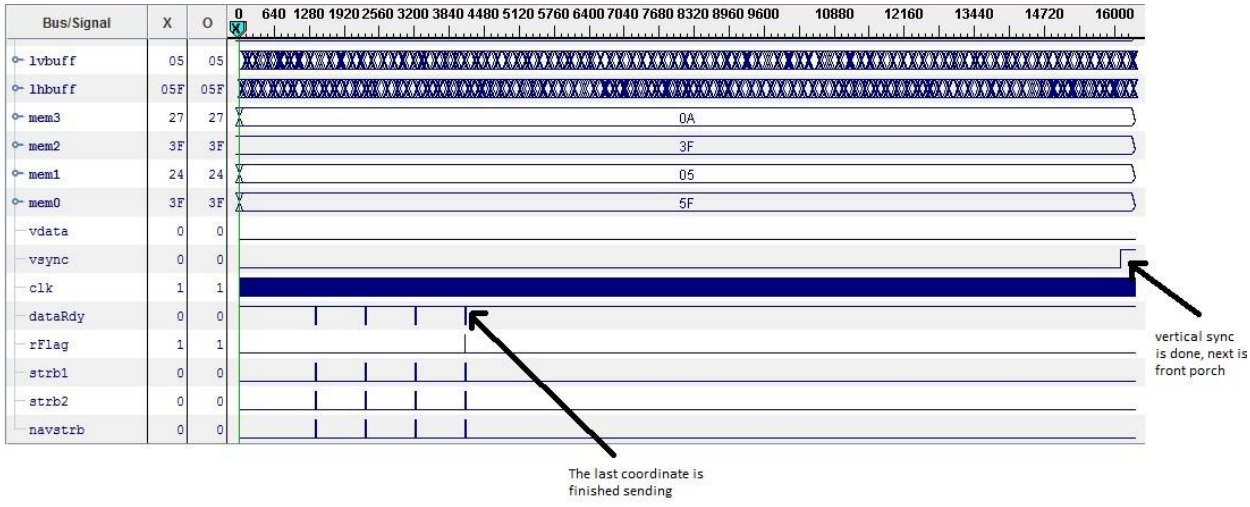


Figure 5. 2: Timing: last coordinate is sent before front porch

5.4 Area and power consumption

When it comes to FPGAs power consumption is highly dependent on the size of the FPGA as well as to how many logic elements are being used in the design (the area of the design). There are two types of power consumption in FPGAs:

- Static power consumption (also known as quiescent or standby power). Is the power consumption of the FPGA when no logic is switching (the FPGA is not processing anything). This consumption is dependent to transistor leakage current, temperature and the cube of the supply voltage.
- Dynamic power consumption: is the power consumption created by switching activity that occurs inside the FPGA. It depends on switching frequency, the square of the supply voltage and the load capacitance.

As supply voltage is dependent on the technology behind the FPGA, the consumption of the current system can be reduced through the use of a more modern FPGA, which requires 1.8V or even less for the internal logic. However, this is not under the control of the developers of the application but rather in the control of the FPGA manufacturer.

Dynamic power consumption can be reduced by optimizing the code. If the HDL code is as small as possible then a lesser number of transistors is used to implement the design and as a result there are less components which are switching when the FPGA is processing.

However, there is not much that can be done to reduce the static power consumption of an FPGA and therefore it is paramount that the smallest FPGA possible is selected for the purposes of the current project.

Figure 5. 3 shows what is the requirement of the current system. However, as mentioned earlier, the acquisition block of the current design converts the input from the cameras from Bayer Pattern to RGB only to be converted to YUV later. This is a waste of resources especially since the Bayer to RGB requires storage, which in turn takes a lot of space.

Figure 5. 4 shows the resources needed by the Bayer Pattern converter alone. As can be seen, a large amount of resources can be saved by using a different kind of CMOS sensors, or by using the Bayer

Pattern colors to perform the processing. However, even as is the system requirements are small and as a result a smaller FPGA can be used (XC4VLX60, which is 3 times smaller than the current one).

	Whole system		Bayer Patter to RGB only	
	FPGA board	Camera board	FPGA board	Camera board
System not programmed (mA)	0.9	0.4	0.9	0.4
System programmed, not running (mA)	1.3	0.4	1.18	0.4
System programmed, running (mA)	1.35 ~ 1.4	0.4	1.2	0.4

Table 5. 1: Current consumption comparison

As can be seen in Table 5. 1, the largest portion of the consumption is due to the static power consumption; with 0.3 mA being consumed by the Bayer Converter and only 0.15 ~ 0.2 mA by the 3D object tracker.

Device Utilization Summary				[-]
Logic Utilization	**Used**	**Available**	**Utilization**	**Note(s)**
Number of Slice Flip Flops	2,426	135,168	1%	
Number of 4 input LUTs	2,703	135,168	1%	
Number of occupied Slices	2,142	67,584	3%	
Number of Slices containing only related logic	2,142	2,142	100%	
Number of Slices containing unrelated logic	0	2,142	0%	
Total Number of 4 input LUTs	3,082	135,168	2%	
Number used as logic	2,630			
Number used as a route-thru	379			
Number used as Shift registers	73			
Number of bonded IOBs	256	768	33%	
IOB Latches	32			
Number of BUFG/BUFGCTRLs	13	32	40%	
Number used as BUFGs	11			
Number used as BUFGCTRLs	2			
Number of FIFO16/RAMB16s	30	288	10%	
Number used as RAMB16s	30			
Number of DCM_ADVs	3	12	25%	
Average Fanout of Non-Clock Nets	3.05			

Figure 5. 3: Summary of the whole design

Device Utilization Summary				[-]
Logic Utilization	**Used**	**Available**	**Utilization**	**Note(s)**
Number of Slice Flip Flops	980	135,168	1%	
Number of 4 input LUTs	1,554	135,168	1%	
Number of occupied Slices	1,054	67,584	1%	
Number of Slices containing only related logic	1,054	1,054	100%	
Number of Slices containing unrelated logic	0	1,054	0%	
Total Number of 4 input LUTs	1,789	135,168	1%	
Number used as logic	1,554			
Number used as a route-thru	235			
Number of bonded IOBs	255	768	33%	
IOB Latches	32			
Number of BUFG/BUFGCTRLs	8	32	25%	
Number used as BUFGs	6			
Number used as BUFGCTRLs	2			
Number of FIFO16/RAMB16s	14	288	4%	
Number used as RAMB16s	14			
Number of DCM_ADVs	3	12	25%	
Average Fanout of Non-Clock Nets	3.08			

Figure 5. 4: Summary of Bayer Pattern converter

5.5 Summary

As shown, the implementation, while it is small enough to be deployed on a small FPGA, it is suitable to process each frame in real time (finish processing current frame before the new one begins). The most time consuming process is the one which transfers the coordinates from the FPGA to the uC. This is due to the fact that the uC is a very basic one, and it is run at 20MHz. However, even as is, the system is able to transfer the coordinates before the start of the new frame (as seen in Figure 5.1).

6. Conclusion

Due to its compact size, the system can easily be implemented on a smaller (cheaper) FPGA. As can be seen in Figure 5.2, the whole system only needs 30 BRAMs. This means that the system can be deployed on the smallest FPGA available in the Virtex 4 family (XC4VLX15). This not only will reduce the system overall cost but it will also dramatically reduce the power consumption because the static power consumption will be a lot less since it is about 12 times smaller than the VLX160 which was used in the current test. As a result the system will consume less than 1.35A, which is in orders of magnitude less than what is consumed by standalone computers or multimedia processors.

Furthermore, due to the pipelined design, the system was able to perform all the computations in real time. As a result, the current frame was processed and the results were transmitted before the beginning of the front porch area of the next frame.

Due to limitations of the experimental setup, the system was not able to extract depth information for distances greater than 7m. This was due to the small displacement between the cameras. In order to improve this in order to track objects at up to 25m, the cameras should be set further apart from each other. This will increase the disparity resolution, while in the same time causing the min coverage area to increase (see Figure 3.1 for illustration).

Both effects are desirable, since the system was never meant to track objects which are really close to the cameras. The system will typically be deployed on a wall or in a corner of the workplace, and as a result, the min coverage area can realistically go up to 0.5m and the system will still be effective.

Another big advantage for the system is the fact that it only needs to transfer 3bytes of data to the PC for further processing and display of results. The bandwidth is dramatically less that what is needed to transfer the frames and perform the whole processing on the standalone computer.

Bibliography

[1] Statistics Canada, "Statistics Canada," 05 09 2014. [Online]. Available: http://www.statcan.gc.ca/pub/82-003-x/2006007/article/injuries-blessures/t/4060686-eng.htm.

[2] WorksafeBC, "WorksafeBC," 05 09 2014. [Online]. Available: http://www.worksafebc.com/publications/reports/statistics_reports/assets/pdf/stats2012.pdf.

[3] Statistics Canada, 05 09 2014. [Online]. Available: http://www.statcan.gc.ca/pub/82-003-x/2006007/article/injuries-blessures/4149017-eng.htm.

[4] Statistics Canada, "Statistics Canada," 05 09 2014. [Online]. Available: http://www.statcan.gc.ca/pub/82-003-x/2006007/article/injuries-blessures/t/4060687-eng.htm.

[5] L. Hammarstartd, M. Lundgren and L. Svensson, "Adaptive Radar Sensor Model for Tracking Structured Extended Objects," in *Aerospace and Electronic Systems*, 2012.

[6] E. Richter, P. Lindner, G. Wanielik, K. Takagi and A. Isogai, "Advanced occupancy grid techniques for lidar based object detection and tracking," in *12th International IEEE Conference on In Intelligent Transportation Systems*, 2009.

[7] G. Li, Y. Wang and W. Shu, "Real-time moving object detection for video monitoring systems," in *Second International Symposium on In Intelligent Information Technology Application*, 2008.

[8] R. Klette, Concise Computer Vision: An Introduction Into Theory and Algorithms, Springer Publishing Company, 2014.

[9] C. Adam, R. Schubert and W. G., "Radar-Based Extended Object Tracking Under Clutter using Generalized Probabilistic Data Association," in *Proceedings of the 16th International IEEE Annual Conference on Intelligent Transportation Systems*, 2013.

[10] Z. Muhammad, I. Ali and M. Ali, "3-D Object Tracking in Millimeter-Wave Radar for Advanced Driver Assistance Systems," in *Global Conference on Signal and Information Processing*, 2013.

[11] T. Miyasaka, Y. Ohama and Y. Ninomiya, "Ego-motion estimation and moving object tracking using multi-layer lidar," in *In Intelligent Vehicles Symposium*, 2009.

[12] K. Cho, S. H. Baeg and S. Park, "Real-time 3D multiple occluded object detection and tracking," in *ISR*, 2013.

[13] J. Rodrigues and J. Ferreira, "FPGA-based rectification of stereo images," *2010 Conference on Design and Architectures for Signal and Image Processing (DASIP),,* pp. 199 - 206, 2010.

[14] B. Maldeniya, D. Nawarathna, K. Wijayasekara, T. Wijegoonasekara and R. Rodrigo, "Computationally efficient implementation of video rectification in an FPGA for stereo vision applications," *2010 5th International Conference on Information and Automation for Sustainability (ICIAFs),* pp. 219 - 224, 2010.

[15] P. Greisen, S. Heinzle, M. Gross and A. P. Burg, "An FPGA-based processing pipeline for high-definition stereo video," *EURASIP Journal on Image and Video Processing,* pp. 1-13, 2011.

[16] A. Darabiha, "Video-rate stereo vision on reconfigurable hardware," *Doctoral dissertation, Department of Electrical and Computer Engineering, University of Toronto,* 2003.

[17] B. Dietrich, "Design and implementation of an FPGA-based stereo vision system for the EyeBot M6," University of Western Australia, 2009.

[18] S. Jin, J. Cho, X. Dai Pham, K. M. Lee, S. K. Park, M. Kim and J. W. Jeon, "FPGA design and implementation of a real-time stereo vision system," in *IEEE Transactions on Circuits and Systems for Video Technology*, 2010.

[19] Xilinx, "Application Note: Spartan-3 FPGA Family," 05 09 2014. [Online]. Available: http://www.eng.utah.edu/~cs3710/xilinx-docs/xapp463.pdf.

[20] J. U. Cho, S. H. Jin, X. Dai Pham, J. W. Jeon, J. E. Byun and H. Kang, "A real-time object tracking system using a particle filter," in *2006 IEEE/RSJ International Conference on InIntelligent Robots and Systems*, 2006.

[21] P. Zhao, H. Zhu, H. Li and T. Shibata, "A Directional-Edge-Based Real-Time Object Tracking System

Employing Multiple Candidate-Location Generation," in *IEEE Transactions on Circuits and Systems for Video Technology*, 2013.

[22] Y. Huang, N. Sang, Z. Hao and W. Jiang, "Eye Tracking Based on Improved CamShift Algorithm," in *2013 Sixth International Symposium on In Computational Intelligence and Design (ISCID)*, 2013.

[23] N. K. Quang, Y. S. Kung and Q. P. Ha, "FPGA-based control architecture integration for multiple-axis tracking motion systems," in *2011 IEEE/SICE International Symposium on In System Integration (SII)*, 2011.

[24] B. Lee, L. Lee and L. Lee.U.S.A. Patent 618,543, 2003.

[25] D. Schulz, W. Burgard, D. Fox and A. B. Cremers, "Tracking multiple moving targets with a mobile robot using particle filters and statistical data association," in *IEEE International Conference on In Robotics and Automation*, 2001.

[26] S. Prasad and S. Sinha, "Real-time object detection and tracking in an unknown environment," in *In Information and Communication Technologies (WICT), 2011 World Congress on*, 2011.

[27] C. C. Huang and S. J. Wang, "A cascaded hierarchical framework for moving object detection and tracking," in *In Image Processing (ICIP), 2010 17th IEEE International Conference on*, 2010.

[28] S. Nasrullah and D. A. Khan, "A novel algorithm for real time moving object tracking," in *In Image Information Processing (ICIIP), 2011 International Conference on*, 2011.

[29] L. Zhang, D. Zhang, Y. Su and F. Long, "Adaptive kernel-bandwidth object tracking based on Mean-shift algorithm," in *Fourth International Conference on In Intelligent Control and Information Processing*, 2013.

[30] X. Chen, X. Li, H. Wu and T. Qiu, "Real-time object tracking via CamShift-based robust framework," in *International Conference on In Information Science and Technology*, 2012.

[31] X. Lu, D. Ren and S. Yu, "FPGA-based real-time object tracking for mobile robot," in *International Conference on In Audio Language and Image Processing*, 2010.

[32] D. Yuren, Z. Aijun and Y. Feng, "Moving object detection for video monitoring systems," in *International Conference on In Electronic Measurement and Instruments*, 2007..

[33] S. Bold, M. A. Jeong, S. M. Jeon and S. R. Lee, "FPGA Based Real Time Embedded Color Tracking Mobile Robot," in *International Conference on In IT Convergence and Security*, 2013.

[34] M. H. Asmare, V. S. Asirvadam and L. Iznita, "Color space selection for color image enhancement applications," in *In Signal Acquisition and Processing*, 2009.

[35] S. Avidan, "Ensemble tracking," in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2007.

[36] W. Zhiqiang, J. Xiaopeng and W. Peng, "Real-time moving object detection for video monitoring systems," *Systems Engineering and Electronics,* pp. 731-736, 2006.

[37] 05 09 2014. [Online]. Available: http://www.shortcourses.com/images/b7ch3/distance.jpg.

[38] S. Wong and J. Collins, "A proposed FPGA architecture for real-time object tracking using commodity sensors," in *19th International Conference In Mechatronics and Machine Vision in Practice*, 2012.