# DEVELOPMENT AND IMPLEMENTATION OF THE METHOD FOR HIGH RESOLUTION OBJECT TRACKING IN 3D SPACE

by

Spiros Jason Hippolyte

Master of Science in Electrical Engineering,
DeVry University, Illinois, U.S.A., 2010.
Bachelor of Science in Electronics Engineering Technology.
Calgary, Alberta, 2004.

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, 2016

**AUTHOR'S DECLARATION FOR ELECTRONIC SUBMISSION OF A Project**

I hereby declare that I am the sole author of this project. This is a true copy of the masters' report, including any required final revisions.

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

I further authorize Ryerson University to reproduce this masters report by photocopying 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 masters report may be made electronically available to the public.

**Abstract**

DEVELOPMENT AND IMPLEMENTATION

OF THE METHOD

FOR HIGH RESOLUTION OBJECT TRACKING IN 3D SPACE

by

Spiros Jason Hippolyte

Master of Engineering

Electrical and Computer Engineering

Toronto, Ontario, Canada, 2016



The means to track objects in 3D space is paramount to computer vision and robotics. Improving upon prior work of the M.A.R.S. project enabled more accurate object tracking and ranging, required investigation into current techniques of stereo depth estimation, object tracking algorithms and the use of FPGA platforms. The research focused on aviation, ground vehicle and robotic applications of stereo computer vision and image processing methods. The implementation of the project design focused on how to obtain greater disparity resolution from the stereo system while minimizing memory resources. The analysis of the optimal method and then the coding and debugging of the optimal solution was performed to insure inter-operability with the existing system and lay the foundation for further expansion of the system. Comparative analysis of Xilinx FPGA platforms and MATLAB simulation of the concept provided data on hardware resources, improved disparity output and the minimal use of memory.

## Acknowledgements

I wish to thank the support and advise of Victor Dumitriu & Rishabh Kumar, for aiding me in the technical matters relating with this master's report.

## Dedication

To my parents who have supported me in all my endeavours. This paper is a testament and is made manifest thanks to their support and love.

# Table of Contents

# List of Tables

# List of Figures

# List of Abbreviations

| | |
|---|---|
| ANN | Artificial Neural Network |
| BRAM | Block RAM |
| CAMShift | Continuously Adaptive MeanShift |
| CC | Clock Cycle |
| DSP | Digital Signal Processing |
| FOV | Field of View |
| FPGA | Field Programmable Gate Array |
| HDL | Hardware Description Language |
| IOB | Input Output Block |
| IP | Intellectual Property |
| Kb | Kilo bit |
| KTL | Karhunen-Loeve transform |
| LUT | Look Up Table |
| PC | Personal Computer |
| RAM | Random Access Memory |
| VGA | Video Graphics Array |
| VHDL | VHSIC (Very High Speed Integrated Circuit) HDL |

# 1 Introduction

## 1.1 Motivation

The full implementation of computer controlled systems for use in the physical world and beyond can be accomplished by developing autonomous systems, allowing its embedded computing system to perform all required processing tasks with additional human guidance of the system. Such a system can be utilized in robotic, satellite and vehicle operations. Many of these functions are linked to image processing for operations in physical environments. This is because to fully operate in a three dimensional environment requires the ability to determine the distance to objects within the operational area. Many autonomous robotic and human operated systems would benefit from a stereo vision system as robotic systems (ex. planetary probes, autonomous vehicles, manufacturing) would have a direct method to know the range to objects within the environment. This is also true of human interfaced systems where the operator needs this information as well in order to manipulate objects in the environment. Accurate range and object dimensional data would allow robotic systems to more readily operate in environments with similar perception to humans and the means to manipulate objects in the environment to perform useful tasks. An example of such tasks are operations in environments hostile to humans (space exploration, nuclear reactor maintenance/fuel handling, mining etc.) where regular commands from a human operator may not be possible so the robotic system must perform a set of decisions and actions to compete its task, which are based on its perception of its operating environment (ex. Moving toward an object of interest, tracking it if moving and holding manipulating it). Given depth perception data, an object's dimensions can be determined by knowing the differences in range from the observer. This is more pronounced for tracking a moving object that possess a velocity and direction of motion from the point of view of the observer. The system of stereo vision described here is modelled from biological systems, mammal (predator) & human vision. This method of image processing would allow the robotic or computer controlled system to navigate environments, track and manipulate objects, by only using a pair of cameras to see and detect the world in a similar way to humans. (Digital cameras focus the light on to photo sensitive transistor array-CMOS; the image formed per unit of time is called a frame). Unlike similar systems that use one camera, LIDAR or RADAR (these detection systems are effective at large distance, from metres to kilometres, but are not effective in detecting object distances of units less one metre to centimeters), a visible or near infra-red 3D camera system is less affected by weather [1] and provides a direct means of depth perception using less complex hardware (LIDAR for 3D measurement uses a rotating scan head with an array of lasers to map out objects and detectors to measure the returned laser light, while RADAR wave-lengths are used for detecting vehicles and precipitation via

the Doppler effect) [2]. In comparison a pair of cameras is far less equipment and uses less power than a LIDAR system or an antenna/dish/emitter and klystron equipment used for RADAR [2]. The range to an object is determined from the stereo vision method by using a pair of cameras and by the focal length of each camera. The distance between the stereo pair (base line), the camera focal length and the relative distance of the object to each camera is used to triangulate the distance to the object of interest.
The stereo vision computer vision process requires the rectification of the separate images into one (the images are aligned to match similar points along the horizontal axis) and compute the disparity between them (the displacement along the horizontal axis of similar points representing the same object).
This information is used via triangulation to find the distance to the object. The position of the object that is determined in the stereo vision process by the relative position within the camera frame, these coordinates within the frame are at the centre of the object, called a centroid. This project was presented originally to augment and improve upon an existing system the Multi-mode, Adaptive, Reconfigurable System or M.A.R.S., which employ a stereo set of cameras to perceive the robotic probe model environment and object tracking capabilities. The original system utilized frame image averaging to reduce memory costs and provide object tracking. By tracking an object from the averaged frames, the tracking and dimensioning of object data is less accurate and so is the ranging data because of this.
Also the applications of the stereo vision system extend to human interface systems (augmented reality) where computing the distance from the operator's field of view to the projected virtual image is critical for successful human machine interfacing [1-31].

## 1.2 Objective

The objective of the project report is to optimize the M.A.R.S., 3D stereo vision system through higher image resolution; therefore, improving object tracking and ranging of the system with further implications for robotic vehicle operations. This will allow robotic systems to more fully operate in a real environment as it must be able to efficiently detect, analyse and estimate the distances to objects within the operating area. This method of image processing would allow the robotic or computer controlled system to navigate environments, track and manipulate objects, by only using a pair of cameras to see and detect the world similarly to humans.

The key objective as defined by the project specification are outlined as follows:

1. Improving distance measurement from computed disparity.
2. Optimize and minimize memory/logic recourses on the FPGA (Field Programmable Gate Array).

Branching from the key objective are the sub objective tasks:

1. To improve the object tracking behaviour of the system by providing higher image detail within the window region of interest.
2. Compute the object centroid and dimensions.
3. Analysis the accuracy of object tracking by comparison of initial to updated object coordinates.
4. Optimize the object tracking algorithm to provide updates on the objects position and distance.
5. Optimize the use of FPGA resources to minimize chip area, time delay of the system.
6. Integrate the new high resolution object tracking stereo vision system into the existing M.A.R.S. and compare its performance.
7. Insure minimal object image artifacts caused by moving complex objects (ex. people) and focus tracking of the object centroid.

## 1.3 Original Contribution

The original contributions in the project is the concept of enhancing an existing stereo vision system, that is outlined as follows. The review of related and relevant research into computer vision, to focus on the best methods to accurately and efficiently measure depth from a digital camera system. From the research to synthase the best architecture for a system to address operating requirements, defining the engineering specifications. The testing and implementation of the system architecture to confirm its correct operation as compared to the project requirements. And finally to analyze the results from the system to measure its performance in comparison to the original system. This project contributes the higher resolution, object tracking VHDL module to accomplish the goal. From a stereo video source (which must provide current object coordinates and dimension relative to each camera data stream) it can directly track the object of interest and produce its coordinates and distance relative to the stereo cameras. The project objective is accomplished by the implementation of a region of interest window, where image data is read at the frame resolution (in this case 640x480 pixels). Also the disparity is computed based on the object edge found within the region of interest window. Achieving the objectives of the project requires implementing a means to directly inject higher resolution camera image data into the system. This is enabled by integrating and utilizing higher camera resolution data to detect the object, update system data, track the object motion and extract the object distance and dimensional data in an efficient way. By using a higher resolution camera scan to augmented a lower resolution system, the data loss and resultant reduction in accuracy due to compression techniques like averaging can be avoided to provide improved object tracking performance [3-20].

## 1.4 Organization

The project report is organized in to the following six chapters after the first, the introduction.

Chapter 2, the literature review provides a synopsis and analysis of research related to the project objective of stereo vision methods and its applications to object detection, tracking and navigation. The example applications include an automotive self driving vehicle, navigating autonomous aerial vehicles and object tracking methods using neural networks as a feedback control system. The concept of stereo vision explained in full via the mathematical model.

Chapter 3, design synthesis explains why the proposed solution was chosen. This fully explorers what was outlined in the contribution section of the introduction and conceptualized in the motivation section. The possible project solutions and its limitations are discussed and where the project solution is fully described as the only logical course of action given the objectives and specifications of the project.

Chapter 4, implementation & test provides details into the logic structures used to build the project solution and how testing methods were established and performed. The VHDL code structure is explained based on its required functions to achieve the project objectives. The methods of testing and tools used are also explained in full (from the design interface software, FPGA hardware platform to the signal waveform display).

Chapter 5, comparative analysis presents and analyzes the results of FPGA resource utilization and object tracking performance. This chapter will direct FPGA logic utilization data from the ISE and explain how the report objectives were satisfied.

The sixth and final chapter is the summary of the report, that provides a summary of the project report and concluding remarks on its findings from the previous chapters. Furthermore, future work proposals will be included.

The references, definitions and index of key word follow afterward.

# 2 Literature Observations

The following synopses of near current research into 3D stereo image processing and its applications is presented to provide background knowledge, observations and analysis in comparison to the report objectives. The process of 3D stereo vision involves the main stages of rectifying the two camera images (image matching is improved by aligning the cameras in the horizontal plane), then the disparity map is created from the images, matching the same object from both left and right camera images. From the disparity map the relative displacement of the object on each of the image are used compute the distance to the object based on the trigonometry of the cameras and the object. These aspects are further reviewed in the multiple synopsis of the research papers and its analysis [3-31].

## 2.1 UAV Tracking

The first paper [3] uses a pair of stereo cameras to track an unmanned aerial vehicle (UAV), triangulate its location to compute landing information for the UAV. This major area in robotics research into UAV has many military and civilian applications. An open problem remains flight operations in GNSS (global navigation satellite system) denied environments. The task of landing a UAV requires high precision motion control and reliability of its sensor system. "So the main idea in this research is enlarge the navigation range during the landing progress and detect the fixed-wing UAC as early as possible" [3]. A novel ground based system was developed (figure 1). On either side of the runway, independent stereo vision units are installed to separately scan the airspace above the runway. When an aircraft is detected, the units can cooperate to estimate the relative position between the UAV and the runway via triangulation. Unlike on-board sensor solutions, the ground-to-air visual system has no payload restriction and allows high computation power. The paper [3] contributes the ground based platform with a large field of view (FOV), eliminating the need for GNSS, utilizing the image processing algorithm Ad-aBoost for target detection and tracking under varying light condition and the system used a middle sized fixed wing aircraft [3].

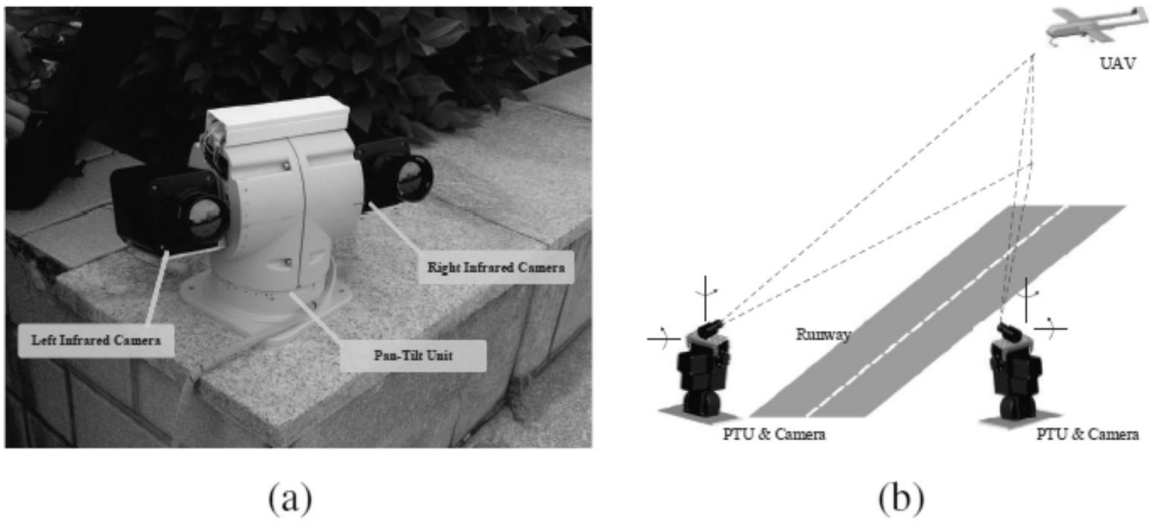


*Figure 1. (a) Previous system single PTU camera tracking, (b) developed dual stereo PTU camera system [3].*

UAV landings have employed GNSS and IMU (inertial measurement unit) to complete this task. Yet in urban environments or low altitude operations the GNSS receiver antenna can lose line of sight of the GPS satellites, therefore the UAV loses its position data. Military grade UAV systems also used millimeter wave track radar, laser pointers and cooled infrared cameras to provide further position data in non-GNSS environments. Previous methods for rotary wing UAVs used monocular vision or dual camera feedback. Fixed wing aircraft landing proves more challenging since tiny errors in guidance could yield system damage. The flight system in the paper [3] is composed of its aircraft platform and ground based visual system and communications. The fixed wing aircraft is a Pioneer by VIGA Tech Company. It is gasoline powered radio-controlled, approximately 2.3 metres in length with a 5kg payload capacity. The model aircraft uses the iFLY-F1A module for autopilot, "a ground control station, a redundant power management module and an engine RPM monitoring module" [3]. The G2 module possess a GNSS and IMU instruments. The ground based visual system components consist of two DFK 23G445 visible light cameras, with an image resolution of 1280x960 pixels at 30fps. It has the GigE interface. A PTU (pan tilt unit) is used to extend the field of view. "Its pan/tilt speeds up to 50 degree/sec with the position resolution of 0.00625 degree" [3]. The PTU-D300E is programmable via Ethernet and RS-232 interfaces. The ground station uses the EOS-1200 embedded vision system for transmitting control command to the UAV and records GNSS data. The communication system utilizes a radio modem, operating at 900 MHz with a range of 22km and data rates from 10 to 23000 bps.
Each vision unit independently operates and transfers image processing data & PTU status to the navigation computer. Then the estimated relative position of the UAV is calculated [3].

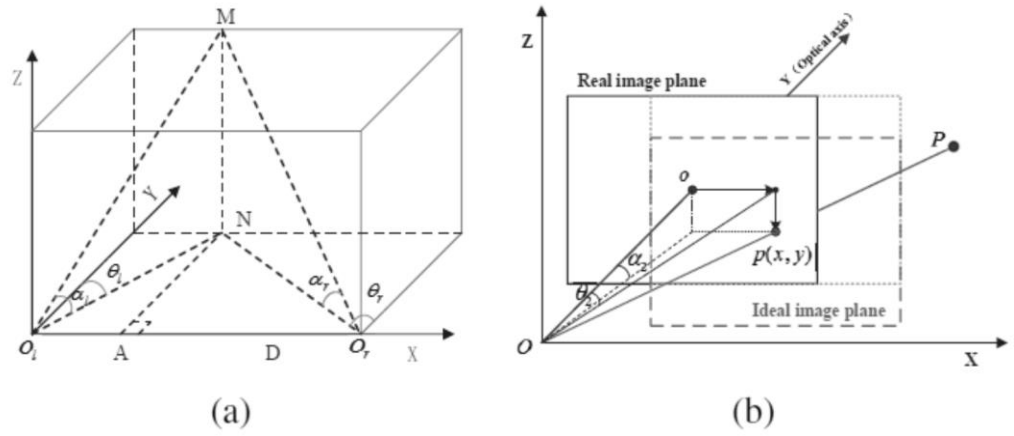


*Figure 2. (a) Theoretical model, (b) Pixel deviation compensation model [1].*

The model of the optical system (figure 2) is based on both cameras at the same baseline, the X axis, with its optical direction shown be the Y axis. The coordinate system origin starts at the left camera.
A tracked aircraft converges the cameras at point M. The tilt and pan of the cameras are shown by α and θ respectively. Pixel deviation is compensated for since point M may not coincide with the centre of the image plane. Measurement error and its analysis is taken into account, since intersection of the camera at point M is not perfect. "The method estimates intersecting point by means of optical axis combined vertical line of two different planes in space" [3]. The optimal point of point M is the vertical line segment centre, the least square method is used to better meet this condition [3]. Error analysis is based on examining the partial derivatives of the equations with respect to the pan & tilt angle and its influence from the gradient. For the landing trajectory, the UAV must keep an angle of attack between 5 to 7 degrees with the runway. Error within 100 m to the landing area produces an error of altitude of 0.02 m, while errors at greater distances are more notable [3]. The experiments conducted tested the MATLAB ground control software for aircraft detection via training samples, also under varying light conditions, approximately two hours about sunset [3]. The aircraft would takeoff (navigation via DGPS), proceed to set waypoints around the flight area, then in position for approach the ground system takes control to guide the aircraft to the runway from distances over 600 m [3]. This paper [3] presents a dual stereo camera system with PTU at the landing zone (runway) to control and guide the aircraft to the ground. This increases the computational tasks compared to a single stereo camera system with fixed FOV, as this report focuses on, in its implementation. With the computer control system on the ground and not on the aircraft, it removes hardware mass/power limitations. This report will focus on FPGA implementation to allow for an aircraft based control system. Having a ground based landing system allows for safer flight operations in controlled airspace [3].

## 2.2 Landing Quad Rotor UAV's

The second paper [4] proposes an autonomous means of controlled landings of UAV and quad rotors utilizing stereo camera and the FPGA to operate the real time neural network architecture. This system can overcome human factors of piloting UAVs given "limited situation awareness and lack of realism" [4] for the most critical phase of flight, landing [4]. "Most of the [v]ision based landing systems detect known visual markers" [4] which is used to compute the vehicles relative position, such as a heli-pad 'H' or high contrast patterns (ex. Checkerboard). The alternative approach is to land with any marker or target, where the flight system determines the safe landing area, attitude and distance of the UAV to the landing zone [4].

The idea is to track the landing area identified by the safe landing area module, measure the displacement & relative position between the landing target and the UAV. This is implemented using C/C++ and MATLAB based on OpenCV, while landing area tracking is performed by an FPGA. Extensive validation of the practical scenarios was performed in RTL simulations, power and area constraints of the system were also evaluated. Algorithmic development and evaluation was first done using MATLAB then in Verilog HDL [4]. The Xilinx XC2V1000 is used to synthesize the system architecture on FPGA. This low end model was selected to match the limited area requirements with a low power platform for the proposed design approach. The algorithm for the proposed method (figure 3) operates as follows: the stereo cameras obtain the images; the left camera image is taken as a reference. Next the visual odometry module estimates the relative position/rotation (Euclidean distance) of the UAV and tracks the landing target. This is enabled by the safe land detection module and the safe landing area tracking/Euclidean distance measurement module. The visual odometry module processes image gain correction, calibration and rectification in the calibration and pre-processing step. Next key points are taken from the left camera image using the FAST algorithm for real time 30fps operation. The object feature/key points detected in the left image are mapped to the right image. The features (part of or property of interest, for example: tip or colour) are searched within a disparity limit (ex. 16 x16 pixel square). "Normalized correlation is computed between feature[s] in [the] left image and all potential features in that box using zero normalized correlation (ZNCC) [4].

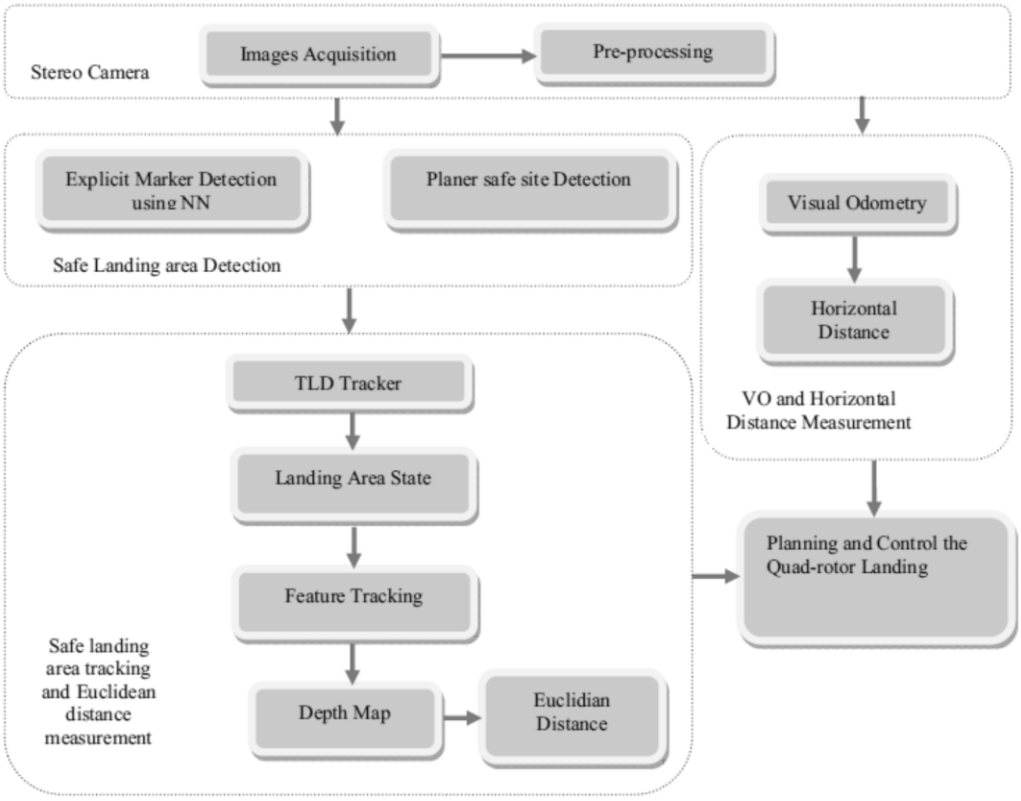


*Figure 3. Algorithm for landing a small UAV [4].*

The Kanade Lucas Tomasi (KLT) tracker is used to track key-point locations between frames. KLT applies the affine distortion (non-rectangular camera pixel matrix transformation representing the distance between images) model to better detect large changes. The disparity is calculated to compute the horizontal displacement between stereo frames, corresponding to a real world point. From the disparity map 3D points are calculated for the triangulation step. Outliers are rejected using RANSAC, rotation and translation is estimated using the 3-point algorithm. The horizontal displacement between landing target and UAV is calculated by the visual odometry pose estimation, via the height data [4].

The objective for the safe landing area detection module is as follows: the landing area is a flat level plane without obstacles, its large enough for the UAV and has sufficient features to be tracked. The safe landing area is tracked frame to frame by the TLD (tracking-modeling, detection) tracker. The TLD output is a bounding box representing a safe landing area called "Landing Area State" [4]. The bounding box is initially set to maximum, if it exceeds its threshold it then creates another box within the first which is less than the threshold, this relates to the UAV altitude. When the landing zone distance decreases the bounding box will increase; therefore, the centre is targeted (second smaller bounding box) and tracked. The IMU provides direct measurement of the UAV angle to calculate the Euclidean distance [4].

The algorithmic level of the automatic identifier & classifier is composed of the following: image acquisition & pre-processing (the UAV front facing camera is used for obstacle avoidance and object recognition), detection and segmentation (extraction of features from the background via neural network), classification (matching of target feature using an ANN, so centroid calculation is not required).
The target templates are based on complete or partial target information, saved in a knowledge base. Template based matching are sensitive to affine moments; extracted features are matched based on reliable features; the KNN based classifier produces minimal distances for similar objects, rejecting similarities amongst different classes [4]. "The neural network performs a series of matching by minimizing the distance between the learnt model features and measurement points" [4]. The KNN function classifier provides power and storage advantages over traditional linear search methods.
KNN performance remains linear over the range of recognition. FPGA based KNN classifier offers parallelism and flexibility in numeric precision. Neural network weights can be easily updated via writes to block RAM elements. The ANNs have its input/output data interface, where each computes the Euclidean norm for the distance. The hardware uses square root approximation (exponentially reduces computation compared to cordic, newton Raphson method square root cores from Xilinx). To optimize the implementation, fixed point C was used to implement quantization noise and the optimum word length was defined to minimize the effect on system performance, while reducing hardware area.
The control block is responsible for data flow control in real time, acquiring data and updating neural weights. Implemented on the FPGA, a group of 8 neurons each with 128 Byte feature vector (weights). The critical path length approaches 85ns in the FPGA, operating at 11MHz, yet still meets performance requirements given the parallel architecture with wider data paths. The control block manages the data during the learning process, the output is used to define a match with the given data set. The results yielded an efficiency of 71% of feature extraction. From 5000 patterns 8% were matched with F.A.R. [4].
The approach for a complete system to detect and land a UAV in unknown locations is the logical future step for this report's future development. Having the image processing and neural network for feature matching (via an onboard expert system) on a low power FPGA presents the versatility of design of hardware and software. Though the paper's [4] results were preliminary, improvements in the learning data of the expert system can produce improved results [4]. The use of the neural network provides flexibility in processing features compared to subdividing an image into blocks to isolate a specified feature. It would also make high resolution image rescan redundant since it can be done at the beginning.
The system in the paper [4] was developed for recognizing relatively static targets at a distance.
For moving targets at close range the neural network would require a training data specific to extracting features from fast moving targets.

## 2.3 Stereo computer vision

The third paper [5] studies and implements a 3D stereo computer vision system. This form mimics human vision by using two cameras. Its applications cover autonomous vehicle, virtual reality, video surveillance systems and 3D television for example. The advantage of stereo vision is acquiring depth perception in contrast to plane vision from a signal camera. Previous video tracking methods have been developed including the following: optical flow, kernel-based tracking, contour tracking, blob tracking, mean shift tracking, Kalman filtering and 3D tracking, etc. [5].

The aim is the study of 3D feature (part of or property of interest, for example: tip or colour) tracking and localization using stereo vision. When the feature to track is defined, the system automatically tracks and localize the given feature in motion, estimating its depth information. The system is designed using two identical cameras, having the same settings. The cameras are installed so its optical axes are parallel to each other and the floor, they're both the same height off the floor. The main system processing includes the following: feature definition, tracking (best matched feature point via minimizing the mean squared error between two blocks in both frames), localization (estimated matching motion vector) and depth computation (computed from the relationship of the camera focal length, position in space, object distance view angle to the camera and light point on the camera lens) (figure 4). The user must indicate a 3D feature point of interest to track using the first frame in the sequences of future frames. Next the system will automatically track the 3D feature in the left video sequence and localize the corresponding 3D feature in motion using the right video sequence. Given the two feature locations, tracking and localization the depth information is computed via triangulation from the object to the cameras [5].

This is based on the pin-hold camera model (extended for stereo vision). The focal length (distance from CCD centre to the image plane), projected pixels on the left & right image plane and its horizontal distances to the respective image plane centre, the horizontal distance from the target point respective to the camera centres, the distance between the two cameras and depth, a measurement from the object feature point to the centre of the two cameras [5].

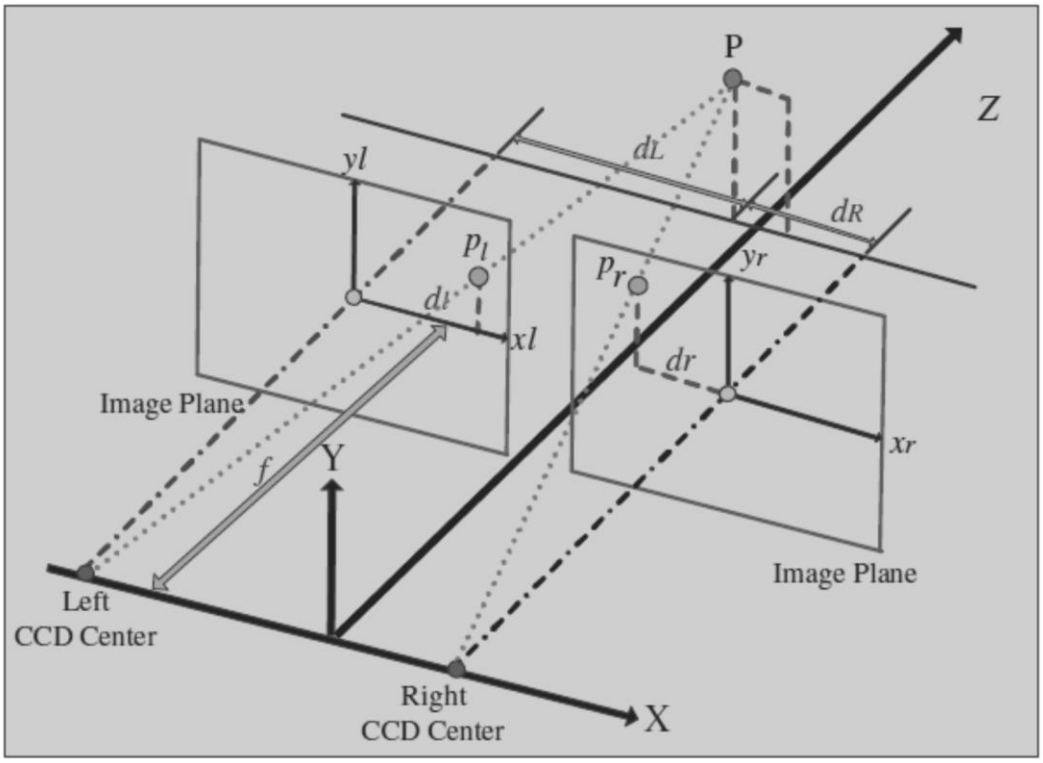


*Figure 4. Stereo vision model presenting the two cameras, with alignment to the object down range [5].*

The system was evaluated for its performance in key parameters are as follows: depth (150 to 250 cm), tracking in plane or stereo vision (with respect to depth computation). Feature tracking was evaluated independently for each camera. Therefore, two motion vectors are evaluated independently. The use of kernel functions effects (computed via M.S.E) and integer vs sub-pixel accuracy. "For the integer-pixel accuracy, the original image resolution (i.e., 640x480 pixels in this study)" [5]. Sub-pixel accuracy magnifies the image by k (e.g., k=2) in two dimensions before tracking and localization; therefore, depth data is evaluated by 1/k of the pixel. The error rate of depth perception is computed given the difference in actual and measured depth. The experiments show that depth measurements at the mid-range (200 cm) were most accurate with the sub-pixel function for stereo vision as compared to plane vision with integer pixel accuracy [5]. These findings and methods directly apply to the project report's implementation as it uses a similar physical set up to emulate human vision. The application of sub-pixel accuracy may require further study for implementation. Indicating an image feature to track in real time would also be an interesting addition; in comparison the project will track any one object via blob detection [5].

## 2.4 CAMSHIFT Improved

In the fourth paper [6] an improved version of the CAMSHIFT algorithm is utilized for stereo vision object tracking is implemented. This involves tracking over successive frame, creating large amounts of video data to be transmitted and processed in real time. Increases in computation power, algorithm design also for implementation single processor systems. While advances in FPGA density, speed and programmability make it a viable alternative [6]. Stereo analysis involves measuring the range to the target object. "The fundamental problem is stereo analysis is finding corresponding elements between the images" [6]. Image illumination, perspective and differences directly effect image correlation. "In visual control systems, real time performance of object recognition with pose has been regarded as one of the most important issues for several decades" [6]. Computational burden is the prime factor for designing vision systems [4]. Related works in 3D vision have focused on improving segmentation of object from the image background, the use of invariant descriptors of local features and colour based trackers. For non-rigid object tracking it is advisable to represent it with probability distributions.
The paper [6] presents a modified CAMSHIFT algorithm. In the original "the current frame is searched for a region, a fixed-shape variable-size window, whose colo[u]r content best matches a reference colo[u]r model" [6]. The search is deterministic and may encounter problems when parts of the background present similar colour or the object is blocked from view [6]. The paper [6] operates on both stereo images and its disparity image. The tracking of stable colour features can overcome distortion of shape and partial occlusion [6].
The proposed approach of the modified CAMSHIFT algorithm is as follows: 1st the region of interest of the probability distribution is the entire image, 2nd an initial location of the Mean Shift search window is selected, this is the target distribution for tracking, 3rd find a colour probability distribution of the region of the Mean Shift search window centre, calculate a disparity probability distribution of the Mean Shift search window, obtain the final probability image combining colour and disparity probability distributions, 4th next iterate a Mean Shift algorithm to find the centroid of the probability image (storing the 0th distribution area & centroid location). And (5) for the next frame, center the search window at the mean location found previously, setting the window size to a function of the 0th distribution area and repeat from the 3rd step [6]. To produce the probability distribution image, the method of histogram back-projection can be used. The first step of CAMSHIFT is computed to generate the initial histogram from the ROI of the filtered image. The hue channel in HSV colour space is used to isolate pure colour. Saturations & intensity values are selected within threshold values. The output pixel characterizes the probability that the corresponding input pixel group is part of the object whose histogram is used creating a probability distribution image where objects are presented in shades of grey. In the modified CAMSHIFT algorithm, the final probability distribution depends on the colour and disparity probability

distribution (its proportional the difference of the disparities of the image and mean disparity of the tracked object and is defined per pixel) [6]. The correspondence problem is present when deriving the disparity image. Given the missing data it must be created using "an epipolar line for which we have disparity information" [6] by the formula $disp(i) = (1 - y)disp(L) + y(disp(R))$. The modified algorithm then follows the next two steps of the original CAMSHIFT algorithm, centroid probability image and marking of the tracked object [6].

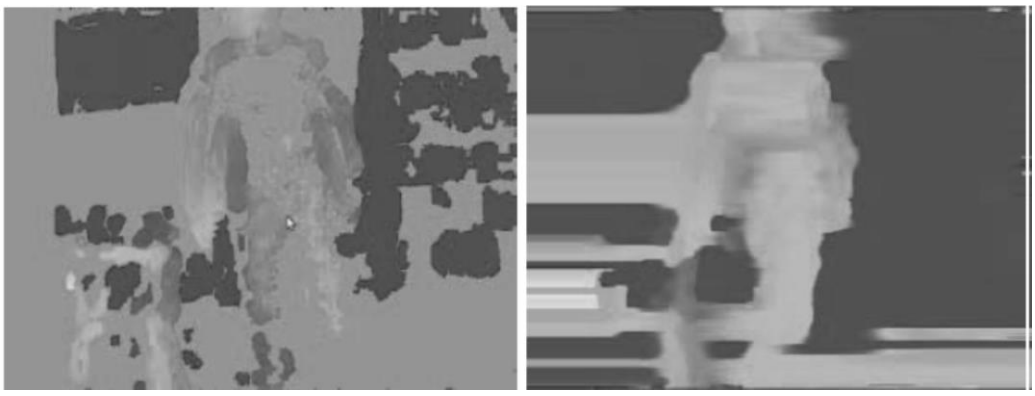

*Figure 5. Disparity image before & after interpolation [4].*

The algorithm is performed as a node in a Robot Operating System framework. The ROS is based on graph architecture where processing nodes receive, post and multiplex sensor, control and other messages. The nodes developed get image and disparity messages, processes them and post messages with object position data. The initial object position is defined in two ways, a rectangular area with the incoming video stream or the node receives a message with the initial position (from node or human input).
The node developed detect people's faces in images and posts messages on their position.
The cameras use its FPGA to calculate the disparity image [6]. This reduces the computational load, allowing higher resolution and refresh rates [6]. The experiments in the paper [6] focused on people moving around the camera while the camera is not fixed. The modified algorithm used a dual processor PC. In contrast the lab set up for this report will have the camera at fixed location with objects in its field of view; furthermore, all processing of the image data will be computed by a FPGA board. It would be an advantage to use cameras that can process the disparity map. In the paper [6] 2000 frames were processed, 30Hz a resolution of 640x480 pixels. The report will make note of frame processing with its findings presented in the analysis section. The report will focus on blob region detection with additional high resolution refinement. The CAMSHIFT algorithms, computational time in comparison to hardware platform (PC vs FPGA system) used is an interesting avenue of research. The report's 3D stereo system will work in controlled lit conditions, object tracking given partial occlusions is an interesting point of

comparison with this paper [6]. The improved CAMSHIFT algorithm works with colour data as part of its pixel/object detection process, while this report will work with gray scale pixels only [6].

## 2.5 AER Object Tracking

The fifth paper [7] explores the object tracking method of using address-event representation (AER) space to exploit on-chip pre-processing by a dynamic vision sensor (DVS) [7]. The improved efficiency is summarized in three aspects: the reduction of data rate of scene dynamics by on-chip pre-processing of visual data (focal plane processing), the second aspect uses high temporal resolution for edge detection, the third aspect is the high dynamic range. "Edge detection using DVS is robust to varying intra-scene illumination due to the pixel sensitivity to local temporal contrast, rather than temporal intensity change" [7]. The object tracking algorithm is applied on a 3D AER that was generated by a stereo vision system. Address-even representation by DVS presents pixels of light intensity changes (on-event increase, off-event decrease) that show activity events (figure 6), while non-moving objects produce no data [7]. Previous work on monocular DVS using AER to detect moving object by an event-based approach greatly reduced the correspondence problem as compared to frame based tracking. "Because there is no temporal quantization of the information by the vision sensor, it generates events for every pixel along the objects path of motion". The object path is held in a pixel address vector, organized by time.
Moving object detection is performed by the sensor, reacting only to changes in the scene reflectance. Also no event buffer is needed and a DSP runs the tracking algorithm. Other methods may support fast object tracking but need more signal processing circuitry [7].

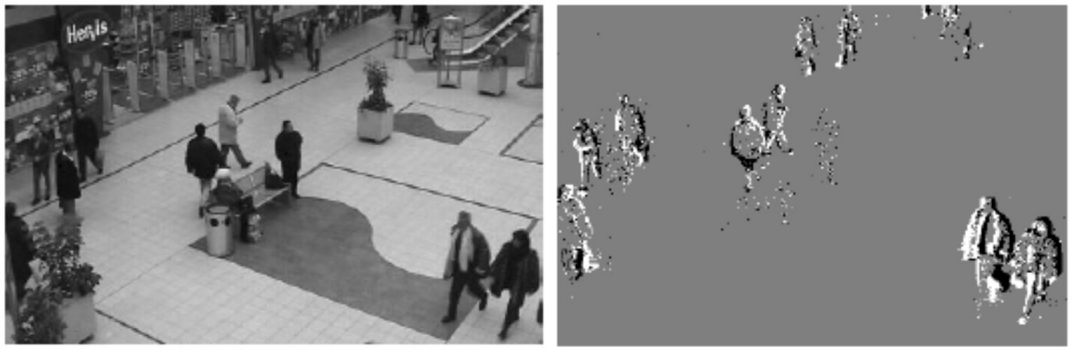

*Figure 6. Conventional camera (left), AER from DVS [7].*

The 3D dynamic vision system hardware is an embedded stereo system made up of the following functional groups: two sensor elements and a buffer unit (multiplexer, FIFO memory, DSP). The DVS is an array of 128x128 pixels (0.35µm CMOS). The sensors generate address events (AE) that is sent to the

multiplexer, then through the memory buffer to the DSP. The AEs are time stamped at 1ms and used for processing. The address event stereo algorithm produces real time depth estimation that are performed in three steps: camera calibration & rectification, stereo correspondence calculation and reconstruction; an adapted area based method is used. The major differences to conventional stereo vision systems created by use of AE are as follows: AEs are accumulated to find significant visual correlations between both cameras, correlation calculations are only performed on relevant images areas. The stereo algorithm has two functional blocks, AE stream partitioning and Integration [7] are necessary to get AEs in an appropriate form. The stereo correspondence calculation algorithm is modified to utilize AE-based processing. With the AE continuous data stream, the "AE stream partitioning" block partitions the timeslots of duration DT that determine the temporal resolution of the 3D sensor system. A typical value of DT on the system is from 5 to 50ms for the depth map [7]. The tracking algorithm works by continuously finding bounding boxes for each object and using the movement sensitivity characteristic of the DVS [7].

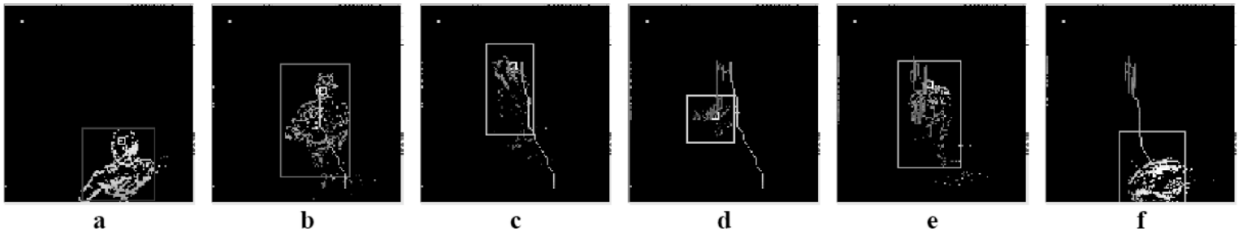


*Figure 7. Six examples over 20 second tracking sequence of AER depth representation of a person moving [7].*

"Any moving object will generate a number of AEs and appears as a cloud of AEs in space-time" [7]. There is no need to extract objects from the background image, making the bounding box easy to generate. The algorithm functions as follows: first at a starting point a bounding box is made around an object by stepwise expanding of the boxes boundaries in pixel coordinates till no AEs lay on the boundary itself. For the 2nd step the first bounding box of the detected object in the sensor is made persistent.
A comparison step is performed to match new bounding boxes to previously ones. After a match is found the bounding box is updated in its size and position using new information from the new bounding box using a mean shift algorithm. In the 3rd step object trajectory can be reconstructed by updating the position of the persistent box, reporting its current location within the field of view [7].
This dynamic vision system was tested by tracking people (detected from their head) moving through a room (figure 7). The vision system was mounted on the ceiling, 2.5m high tilted at 45 degrees.
This would be a slightly different set up in terms of orientation as this report's camera system is based on a platform above the floor with the cameras parallel to it. Also the camera system detects any objects.

This is accomplished by image averaging, rectification and blob detection over the sequence of frames, whereas the DVS methods relies on AER to detect those pixels that have changed light level due to movement. It is a fascinating alternate method of 3D stereo real time object detection that utilizes less computational resources. A full comparison would require placing the DVS on an FGPA SoC, to examine further efficiencies [7].

## 2.6 TLD Stereo vision

The sixth paper [8] reviewed is about the application of a tracking-learning-detection (TLD) algorithm in stereo vision applied to the Nao robot [8]. TLD detects & tracks an object by a continuous series of video frames. The object position is defined as coordinates of its bounding box. To gain data on an objects distance, a second camera is used to create stereo vision. The 3D information (dimension & position) on the object is sent to a forward arm controller to enable the robot's interaction with a tracked object [8]. The TLD tracking method is meant to work in unconstrained environments, the system is composed of three independent components: the tracker (short term, based on Lucas-Kanade method and used to train the detector), the detector "enables incremental update of its decision boundary and real-time sequential evaluation during run-time" [8] and the learning algorithm (P-N learning), it uses the tracker to generate positive (P) and negative (N) examples used to improve the detector model [8]. Before tracking operations begin the bounding box of the object is manually selected by the supervisor. Then the object is tracked, during tracking the object model is created for the detector. The model is based on the first frame and tracker data. When the detector training successfully completes, re-detection of the object is enabled. Any error by the detector or tracker is mutually canceled out to maintain system stability [8]. The tracker used by TLD is Median-shift. It is based on the Lucas-Kanade tracker "which is robust to partial occlusions and also estimates translation and scale" [8]. It tracks points between consecutive frames, estimating a rectangle displacement and scale change using the median of tracked points that are selected at 50% of the most reliable tracked points within the rectangle boundary. Within each frame a new set of feature points are tracked. This is done using the "forward-backward error [that] is defined as a difference between two feature point trajectories, where the first one is defined by forward motion of tracked point and the second one by its backward motion" [8]. This recursive tracking is possible so long as the object is visible. Failure would result from occlusion or other dynamic changes in object appearance. In this case the TLD relies on the detector [8]. The detector in the TLD system uses methods based on the scanning window and randomized fern forest classifier [8]. In the scanning window approach the input image is scanned across all possible positions and scales. Each sub-window has a binary classifier which decides about the presence of the object [8]. The model of the object is defined as a set of image patches representing possible appearances of the object, each is described by a number of

local 2 bit binary patterns. The position, scale and aspect ratio are randomly generated, these features are randomly partitioned into groups of the same size. Each group is a different view of the patch appearance [8]. The main idea of the P-N learning algorithm is to utilize inevitable errors of both the tracker and detector functions. When P-N learning is initialized in the first frame by the initial detector, the Lucas-Kanade tracker initial position is set to the selected path. For each consecutive frame the detector and tracker find location(s) of the object. The distance (close or far) of the detected patch to the tracker trajectory (figure 8) produces the positive & negative learning examples respectively [8].
The stereo vision system advantage provides more depth information unlike single camera systems. With depth data the spatial coordinates of a tracked object can be computed, for the arm controller to move the robot hand towards the object in this paper [8]. Depth is calculated from known values of the focal length of the cameras, the respective horizontal position of the object for each camera and the distance from each other by triangulation via the following equation (1) [8].

Equation 1: $$z = \frac{-2af}{X_L - X_R}.$$


*Figure 8. Representation of L-N learning. The object is tracked & detected. Patches close to the trajectory update the detector [8].*

Integration of TLD into the stereo vision tracking system requires two TLD system per camera to track the same object. The user selects the object for tracking from one camera. The initial object model for the first TLD is made and copied to the second TLD, both TLD system synchronize to avoid divergence in object detection. The output of each TLD provides the object bounding box and its object centroid coordinates (figure 9). This method overcomes the correspondence problem of matching point from each image [8]. For the robotic controller the object data is transformed into Cartesian coordinates.

The controller performs the moving of its robot hand by using object data from the tracking system that is processed by a feed-forward neural network of three inputs and outputs. To train the neural network a sufficient set of training data is required; it is generated by exploiting the inverse model of the system. The inputs of the artificial neural network are polar coordinates of the object position with respect to the robot body & head position. The outputs are joint position values of the robotic arm corresponding to the current object 3D position. The neural network has two hidden layers, each having four neurons.
The training of the neural network is based on defining the transformation matrix from the polar spatial coordinates of the object to the robot arm joint positions. T: (d,α,β) →(s1,s2,s3). One method is to use feedback control, the other is to set the robot hand at the object to get the robot joint positions; therefore, using the inverse transformation, T': (s1,s2,s3) → (d,α,β). This set of input and outputs can be used to train the neural network via the error feedback propagation algorithm. A trained neural network will need to normalize the input/output data between 0 and 1. For example limiting the domain to values of the robotic arm length and the minimum distance range of the stereo vision system [8].

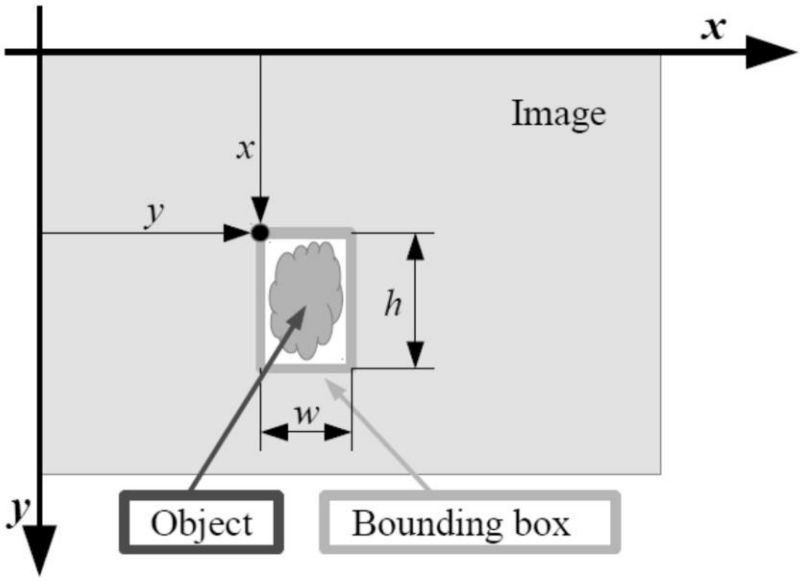


*Figure 9. Bounding box tracking with object parameters (x,y,h,w) forming the region of interest window [8].*

The TLD system takes an accurate approach to matching feature pixels to create a bounding box patch and using the learning algorithm to adjust and refine the process. In this report's system the goal is add further accuracy by rescanning and updating from known object coordinates, which will be explored in later sections. TLD provides a direct method of high resolution edge detection via the TLD learning algorithm. The application to robotics and use of artificial neural networks provides an example of applications of stereo vision and its advantages. Means of feedback control will warrant future

investigation. As will its application on an FPGA to investigate improvements in performance of the system over the original PC platform [8].

## 2.7 Literature Classification

The papers reviewed for this report and specifically the six prime papers selected can be classified by the camera type, image processing method and its application and use of the vision data. Most of the papers presented in the literature review use digital cameras in the visible light spectrum, while the paper [7] used the dynamic visual sensor (DVS) is a modified class of this as its design to sense light intensity. More specialized algorithms were used for paper [5,6,7] such as object feature tracking, modified CAMSHIFT, address-event representation respectively. All the selected papers directly used stereo vision cameras, while a significant amount reviewed, presented mono camera or LIDAR/RADAR object detection and distance measurement systems [9]. For two of the papers [3, 6] there are limitations that are directly and indirectly linked to the reports motivation and objectives. The first paper requires a ground based general computer system using MATLAB for processing. This negates mobile computing applications and power requirements for such (higher cost and can not be embedded into a system platform [3]. The fourth paper utilizes colour vision for object detection, this has the limitation of poor night vision [6]. This was not an original consideration given the project requirements, yet for broader applications and future work that would be a server limitation for vehicle, real world environment operations. For the set of papers selected [3,4] were used for aviation applications for aircraft landing operations. While the rest of papers [5,6,7,8] were applied to general object tracking. Papers [6,7] in particular focused on tracking human in controlled enclosed or dynamic open environments. Stereo vision image processing forms the first stage of a computer vision system. The next stage is in using the visual information to perform tasks. In paper [4, 8] artificial neural networks were employed as a feedback system to refine the task of identifying visual targets (objects or visual features, like symbols). In these cases, identifying the range to a Heli-pad symbol or adjusting the movement of a robotic arm to reach and grab an object [4,8]. In general, all the papers initially reviewed given the research focus and platform requirements spanned signal/stereo camera detection methods. The included object tracking and navigation methods for vehicles and mobile robots. The examination embedded system and its direct application were also reviewed [12-31].

# 3 Design Synthesis

The project's design, its approach and architecture is based on meeting the specified requirement as a result of the limitations of the existing base system performance and research findings presented in the previous chapter of this report. The presented research among others shows the present state of the art means of implementing computer vision systems and image processing techniques to address the problem of utilizing stereo vision to provide depth information. The original stereo camera system is designed to track objects and measure distances within a laboratory environment 3-6 metres, from cameras at a fixed horizontal level and at a vertical height a metre off the ground. Also to detect objects within the cameras field of view.

## 3.1 Research Analysis

The papers analyzed in the literature observations represent research that aligns the best with this report's objective. Dozens of papers were reviewed covering computer vision, artificial intelligence and image processing methods & applications. Single camera methods of object detection (edge detection, textures, colour detection, image segmentation) & tracking and its applications for creating a representative navigation map in two dimensions and identifying obstacles in the environment, was prominent in the research. Without additional sensors such as LIDAR to provide range information, getting accurate depth information will be difficult and entail extensive calibration. LIDAR has its limitations given its scan resolution (number of laser points) and the electromechanical device needed to scan the surrounding area to reflect the laser beam via the rotated mirror, providing a 2D representation around the scanner recording depth information. This presents mechanical reliability and precision issues under real world conditions of poor lighting, visibility due to weather and obstacles in the environment smaller than the LIDAR resolution [2]. These methods even when combined into a system of multiple LIDAR's (multiple sources of laser return to reduce wait time) [2] and cameras still present additional operational limitations due to hardware, computational and power requirements. Therefore, a passive system of detection and ranging is necessary, which is stereo vision.

## 3.2 Requirements

The project's requirements are based on its interoperability with the existing system (Embedded System Reconfigurable Laboratory: MARS platform), its stereo camera field of view, the lab testing environment of a well lit room and the distances involved of within 3 metres to track objects of various sizes.

The project design is based on emulating a natural vision system of nature, stereo vision. It is a direct method that provides image & object details, including distance information on the object of focus.
The stereo vision model is based around the original platform, the 3 metre range tracking and the required base line (horizontal distance between the pair of parallel cameras) and base height of one metre.
The stereo vision model is depicted graphically in figure 4, where details on the camera specifications (focal length, field of view, pixels per camera image) and additional factors that effect the range of objects that can be targeted. Equation (1) provides the computational model. If a low cost and quality camera is used, then there is the possibility of error in the distances measure, so averaging may be need to provide a Euclidean distance of all points measured, as shown in equation (2), where n is the total number of disparities and z is the z-axis position [4].

Equation 2: $$d = \frac{\sum_{i=1}^{n} z_i}{n}$$

Given the current state of the art technology, FPGA's can be used for its lower power consumption, efficient use of chip area, logic programmability and as needed hardware reconfiguration, which provides improved computational efficiency. Furthermore, the use of two cameras provide stereo vision with depth perception at reduced hardware cost compared to a LIDAR 2D system [9]. Also given the current limitations of the existing system mainly its reduction of resolution due to image filtering (mean average of image segments) and FPGA memory the project solution is to employ a stereo vision camera system utilizing a FPGA to provide full resolution and depth information for tracking an object in the field of view.

## 3.3 Design Specification

The stereo vision high resolution system is described by its functional & technical specifications, providing its behaviours and parameters of operation, as presented in the following list.

### 3.3.1 High Resolution Stereo System Block (functional specification):

1. To accept two sets of co-ordinate inputs for both camera streams (left & right) from the system.
2. Accept direct camera data input streams for high resolution detail.
3. Produce both sets of co-ordinates of the tracked object as output.
4. Produce the disparity distance between the left & right cameras, used to compute object distance.
5. Accept data when the valid frame signal input is asserted.

6. Assert a valid frame signal output when processing is complete.
7. Object dimension data can be accepted to determine region of interest.
8. Determine the region of interest that is the probable object location.
9. Operate over a minimal set of image frames.

### 3.3.2 High Resolution Stereo System Block (technical specification):

1. The primary clock#1 must operate at 25.175MHz for VGA 640x480 operation.
2. A reset signal effects the state of the valid frame signal controlling the process state.
3. Process one frame at least every 33 milliseconds, for >30 frames a second.
4. Utilize YUV luminescence for camera data for system compatibility.

The following figures (10-16) presents the entity declaration/symbol, the overall design, process & model timing diagrams for the system.

## 3.4 Design Symbol

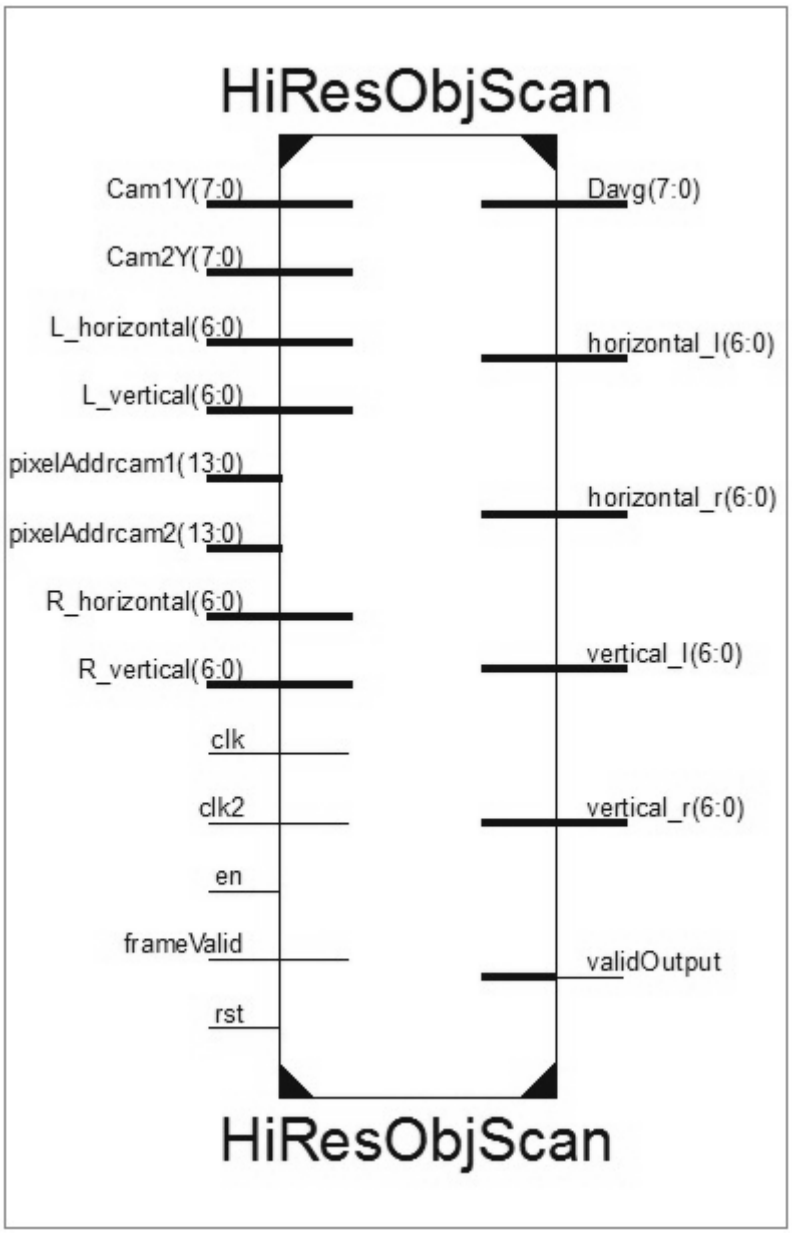


*Figure 10. Top level Design Entity: Hi-Resolution Object Scanner. The input/output flow is depicted from left to right. Note: in testing, additional debug signals and modified camera/pixel bus address sizes were used.*

## 3.5 Entity Declaration

```
entity HiResObjScan is
port ( pixelAddrcam1      : in  STD_LOGIC_VECTOR (19 downto 0);
       pixelAddrcam2      : in  STD_LOGIC_VECTOR (19 downto 0);
                          clk                    : in  STD_LOGIC;
                          clk2                   : in  std_logic;
                           en                          : in  STD_LOGIC;
                            rst                   : in  STD_LOGIC;
                            frameValid      : in  STD_LOGIC;
                            L_horizontal     : in STD_LOGIC_VECTOR (9 downto 0);
                             L_vertical       : in STD_LOGIC_VECTOR (9 downto 0);
                             R_horizontal           : in  STD_LOGIC_VECTOR (9 downto 0);
                            R_vertical             : in  STD_LOGIC_VECTOR (9 downto 0);
                          Davg                   :out std_logic_vector (7 downto 0);
                          horizontal_l           : out  STD_LOGIC_VECTOR (9 downto 0);
                          vertical_l             : out  STD_LOGIC_VECTOR (9 downto 0);
                          horizontal_r           : out  STD_LOGIC_VECTOR (9 downto 0);
                          vertical_r             : out  STD_LOGIC_VECTOR (9 downto 0);
                          validOutput            : out  STD_LOGIC;
                          d1                          : in std_logic_vector (7 downto 0);
                          d2                          : in std_logic_vector (7 downto 0);
                          Cam1Y  : in std_logic_vector (7 downto 0);
                          Cam2Y  : in std_logic_vector (7 downto 0));
end HiResObjScan;
```

| **Signal Name** | **Description of Inputs** | **Size (bits)** |
|---|---|---|
| PixelAddrcam1 | Left Camera pixel address | 20 |
| PixelAddrcam2 | Right Camera pixel address | 20 |
| Clk | Main clock signal (internal) | 1 |
| Clk2 | Auxiliary clock signal (external to camera) | 1 |
| EN | Enable operation | 1 |
| RST | Reset | 1 |
| FrameValid | Frame ready | 1 |
| L_horizontal | Left camera known horizontal object coordinates | 10 |
| L_vertical | Left camera known vertical object coordinates | 10 |
| R_horizontal | Right camera known horizontal object coordinates | 10 |
| R(_vertical | Right camera known vertical object coordinates | 10 |
| D1 | Object dimension (left camera) | 8 |
| D2 | Object dimension (Right camera) | 8 |
| Cam1Y | Left camera data | 8 |
| Cam2Y | Right camera data | 8 |

*Table 1. HiResObjScan Entity Input signals.*

| Signal Name | Description of Outputs | Size (bits) |
|---|---|---|
| Davg | Disparity amount (operational average) | 8 |
| Horizontal_l | Left camera horizontal updated object coordinates | 10 |
| Vertical_l | Left camera vertical updated object coordinates | 10 |
| Horizontal_r | Right camera horizontal updated object coordinates | 10 |
| Vertical_r | Right camera vertical updated object coordinates | 10 |
| ValidOutput | Frame ready, with updated data | 1 |

*Table 2. HiResObjScan Entity Output signals.*

## 3.6 Design Block Diagram

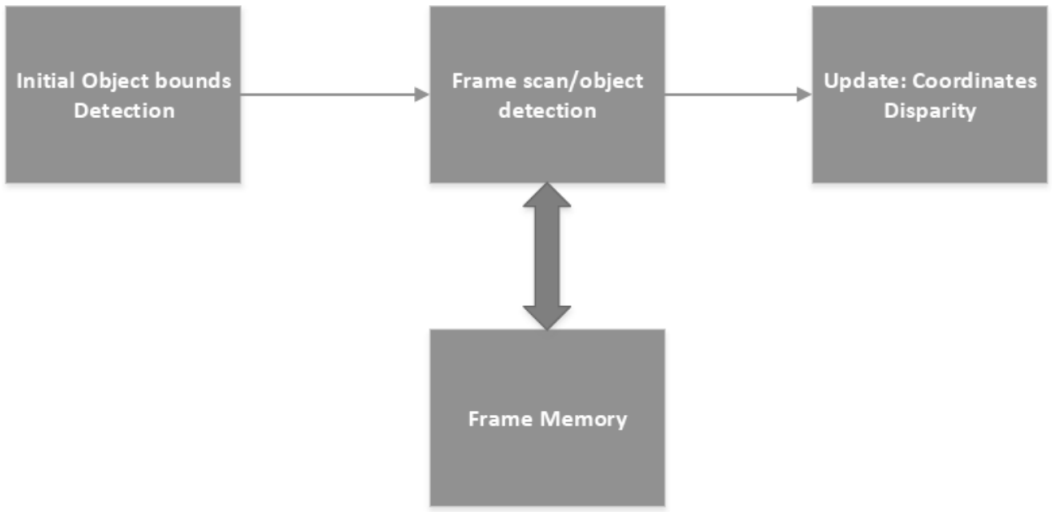


*Figure 11. Design High level Operational Diagram.*

## 3.7 Process Diagram

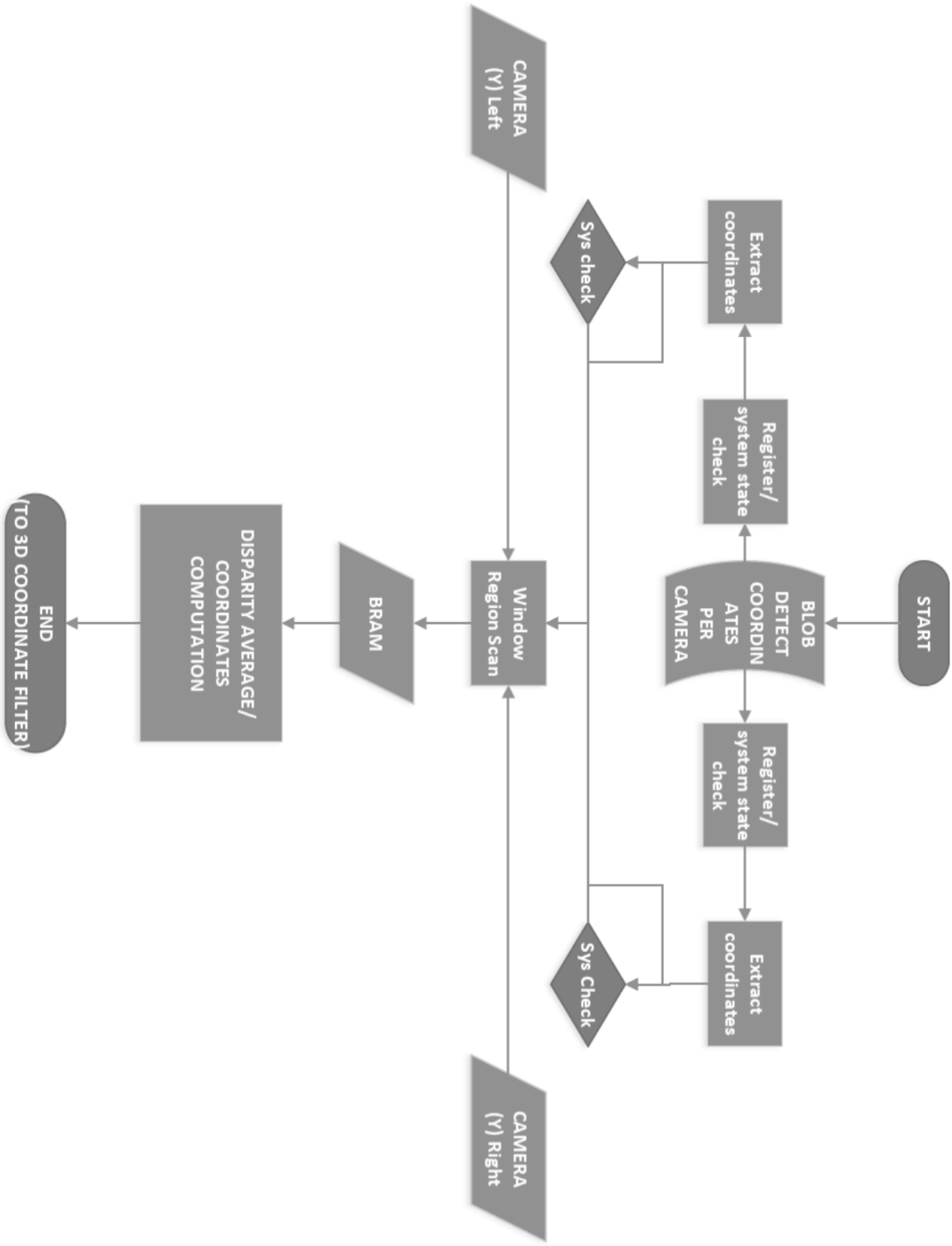


*Figure 12. System General Process Diagram. The functional program is depicted, with both left & right data streams. Input coordinates of are corrected for prior compression, then the object scan is perfomed within the region of interest window. When the possible object is found then edge detection and disparity computation is performed.*

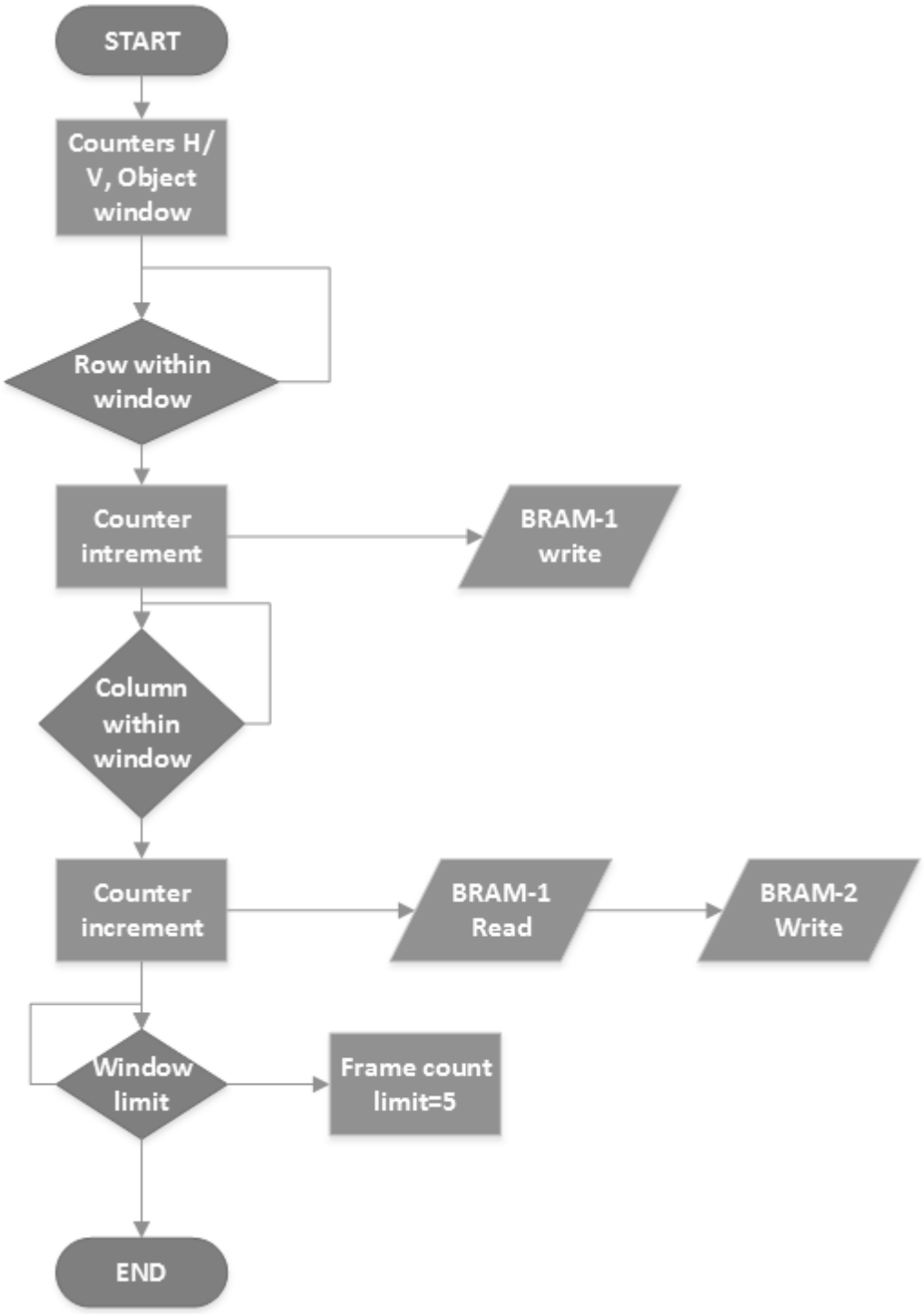


*Figure 13. Row/Line Operations. Each row pixel is check if its within the region of interest. When a frame is read in to memory it is subsequently copied for future comparison to determine frame disparity.*

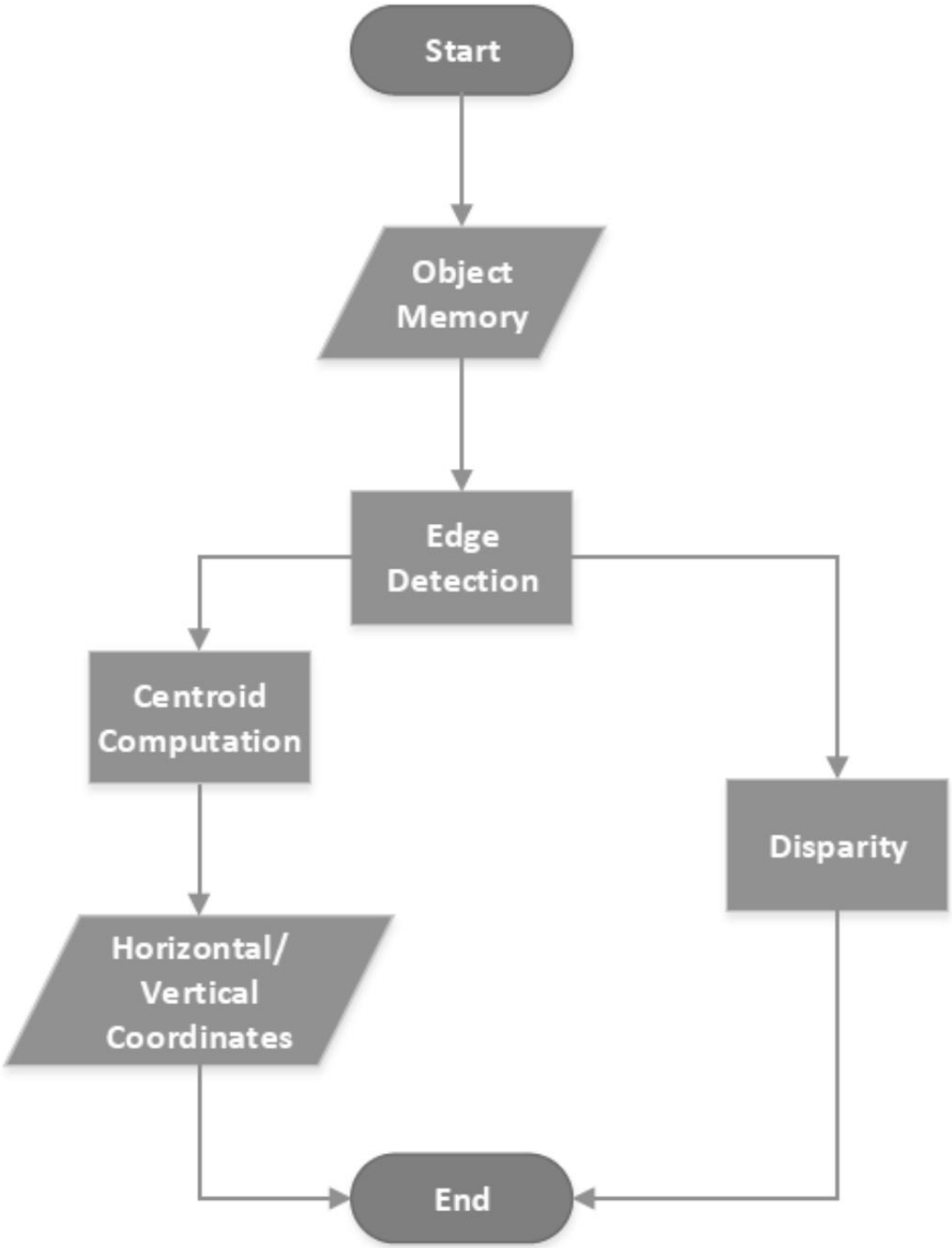


*Figure 14. Frame Disparity & Co-ordinate updates. Suspected Object memory is search via a divide and conquer method to find the pixel address of the edges of the object. These are used to compute the object centre coordinates & disparity.*

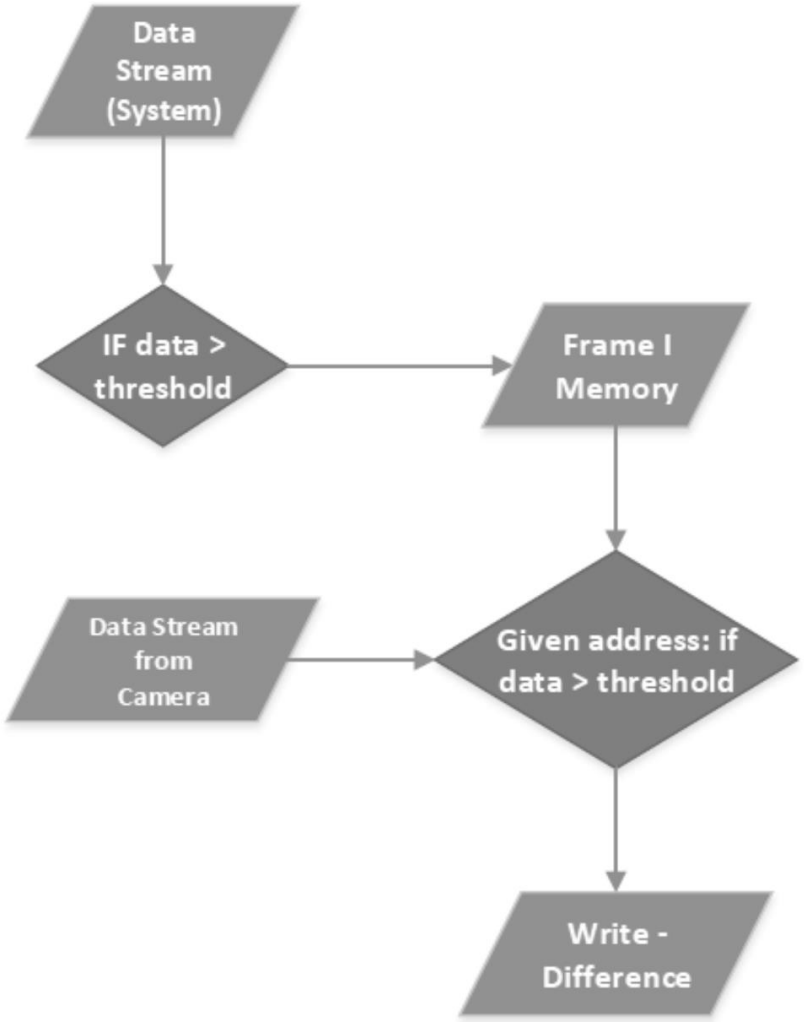


*Figure 15. Frame/Memory Operations. The detected pixel data stored from one frame is compared to the next frame, pixel changes indicate the edges of the suspected object.*

## 3.8 Model Timing Diagram

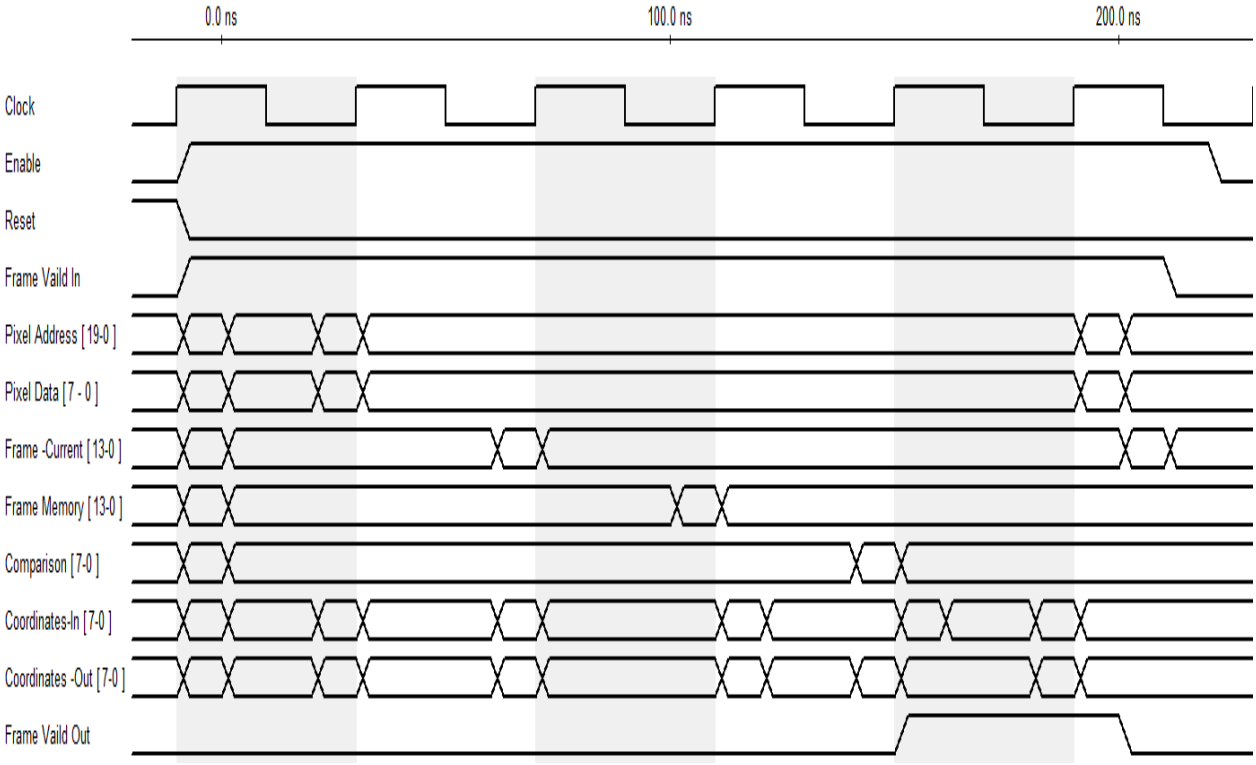


*Figure 16. Timing Model of system. This general presentation of system timing shows the critical transitions that are needed for operation, from enabling the system, synchronization of inputs to frame/pixel comparison and coordinate outputs.*

# 4 Implementation & Test verification

This section of the report presents how the design developed from the project specification, the VHDL coding effort, hardware integration and testing methods used to produce the project design and its results. The main FPGA hardware platform used was the Digilent Zybo: with Xilinx, Zynq FPGA was used to implement the designed solution for the high resolution stereo system (first used was the Xilinx Spartan3E for initial test development, but the system was migrated to the Zybo Zynq-7010 for greater memory and future project development needs). The overall goal for implementation is to integrate the project module into the M.A.R.S. platform. A key indicator of its performance is the functioning of the module on its own, supplied by generated stimuli and the monitoring of critical signals from it to conform functionality. Since the generated signals emulate actual operating conditions then it serves as a key milestone in confirming operational performance.

## 4.1 Module Description

The module for the high resolution object tracking stereo system was programmed in VHDL (using Xilinx ISE version 14.7), consisting of an entity containing the input for both cameras, clocks, frame valid signal, object blob detection module signals (including object dimension) and the output signals providing the updated object coordinates (two sets of horizontal & vertical), frame valid and average pixel displacement value. The behavioural architecture describes the internal processing of the module. BRAM (block logic reconfigurable RAM) components are instantiated within the architecture to access its memory functions (ex. pixel data storage and retrieval). The code is organized into internal signal registers and process statements. (This design was chosen to make code organization and readability more efficient). The processes operate concurrently to read in old object blob coordinate data, then to gather new camera data from the known coordinates of the object within the frame area and finally to compute the average pixel displacement of the object from both left & right frames and to provide new object coordinates.

## 4.2 Functional Overview

The module can fit into an existing stereo vision system [10] where luminescence data from both cameras are filtered by the averaging block (reduce memory requirement by 64 times), the difference of current and subsequent frames is computed. Then the object blob detection block determines the pixels which are part of the object. Now the high resolution object tracking module will use the object blob detection module data with updated camera data to provide refined coordinate and depth information. The 3D

coordinate & filter block takes the object coordinate data and provides object centroid and depth data that is transferred to a pic-microcontroller/PC interface [10]. It now can accept the more accurate coordinate data of the high resolution object tracking module with improved depth data (figure 17).

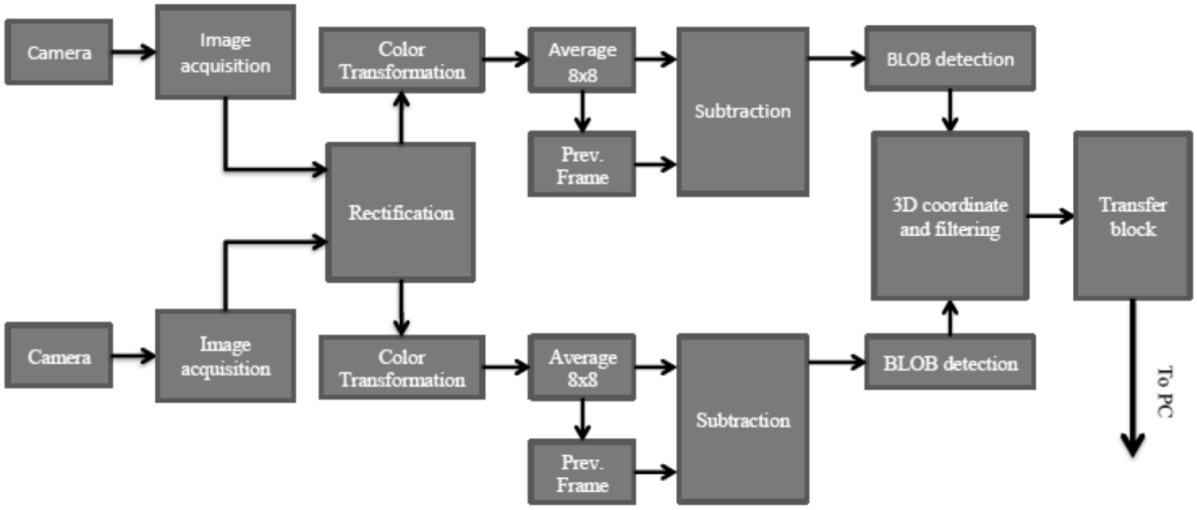


*Figure 17. Stereo System Design Overview, M.A.R.S. platform* [10].

The high resolution object tracking module as mentioned earlier breaks down its task into three main parts. This is performed for both the left and right camera streams of data. First to obtain near current object coordinates from the object detector module. Second to gather new object pixel data at full resolution given the current coordinates within each frame. And finally to compute the average pixel displacement of the object and its new coordinates. Given the capability of the High Resolution object tracking module, it therefore supersedes the 3D coordinate and filtering block as well. This requires the instantiation of the appropriate data transfer components as required to transfer the more accurate coordinate and disparity data to the vision system output (figure 10,17).

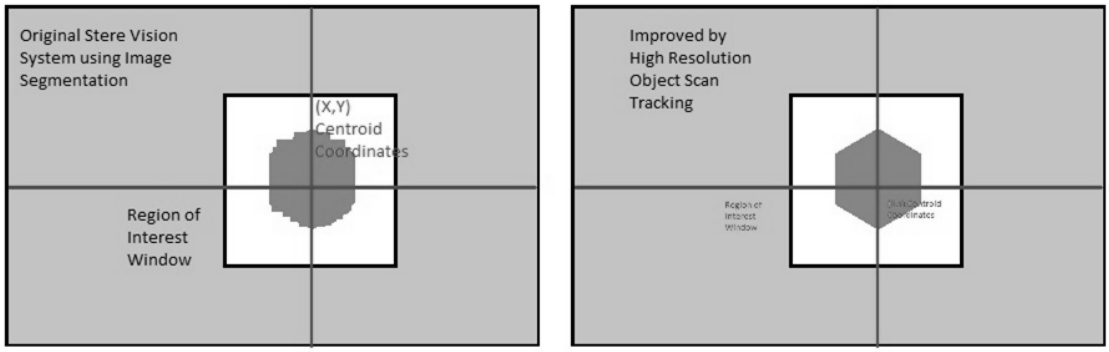


*Figure 18. Region of Interest Window modelled by VHDL process. From the left an original system resolution, on the right object enhanced by the high resolution object scan tracking module.*

All this is done using concurrent processes. The first labelled "init" reads in the pair horizontal & vertical coordinate from the external object blob detection module (both right & left instances).
Direct dimensional data from the Blob Detect module is also used since direct computation from the resultant coordinates can produce non relative data (the object may be at the left/bottom end of the screen like (e.g. coordinates 600,400) where half of that value may not be the original dimension of the object). For integration into the MARS platform, the performance of two shifts (1 left then 3 left again) are needed because the coordinate was produced by dividing by 2 so to get the width a multiplication by two is needed. Also since the original coordinates were based on pixel averaging by 8x8 blocks (80 x 60 pixels); therefore, for each plane coordinate now must be further shifted by 3 (multiplication by 8) so we can have the object horizontal & vertical dimensions respectively at full resolution (e.g. 640x480) (figure 2,17,18). The main process utilizes counters to determine the pixel frame active area (VGA horizontal/vertical scan cycle) and the bounds set by the current object coordinates. (Pixel data is designed as a continuous stream, where its processing must be performed per pixel, per row for every line till the complete frame has been scanned). Then within this region of possible object interest, the pixel data is written into memory (two BRAMs, each with 8-bit data, 14bits addressing). This forms a window where the object is possibly located (the window is size slightly larger than the suspect object by ~25% to account for this uncertainty by ensuring the object edges are within the region of interest). To confirm that the object is tracked over successive frames, an additional counter is run so that every five frames a check is performed on the object edge address locations. This ensures that the object track algorithm is keeping up with the object as it changes location in space. The object data from the video stream (representing the video frame) is first checked to confirm active data at the pixel address location, then confirmed data is copied to the BRAM's (one for each left & right video stream). The stored data is then

check against the next frame of new video stream data. The system is based on detecting movement; if data at a memory address location and the corresponding address in the video stream have changed (e.g. high to low) then object displacement has been detected. It is then copied to another set of BRAMs (16Kb x 1bit data) forming the detected object, composed of pixel at memory addresses where only displacement has occurred. With the second set of memory holding displaced pixels of interest (the possible object) its memory addresses are then examined (horizontal and vertical address data is assigned where the lower seven bits & the upper seven bits respectively), comparing the pixel location to reference boundaries to define the coordinates of the object's top, bottom, left & right boundaries (the boundary is updated with the pixel location when its within the region of interest). Implementation involved an iterative process where syntax checking, VHDL module integration & system debugging took precedence. The focus of the effort was on ensuring the storage and correct passing of BRAM memory data, back into key processes for object dimensioning, while insuring the primary behaviour of the main code is expressed in code correctly; thus leading to the proper flow of processing from input to output. One of the key code section was in the main process, that described the pixel row & object confirmation. Since implementation is being performed on an FPGA, beyond the syntax check, synthesis, map & routing, program file generation processes are required get the design ready.

## 4.3 Testing Methods

Initial testing of the module functionality is performed via a combination of test-bench simulation using Xilinx ISE: ISim and Chipscope (logic simulation and near runtime internal FPGA analysis tools respectively). The test-bench file used by ISim, instantiates the main module as the unit under test "UUT" providing input stimuli relevant for device operation. The test-bench file is designed to provide input signal defaults (ex. clock), while the critical signals of the cameras, its pixel addressing, valid frame, clock and the pair of horizontal & vertical coordinate signals are actively simulated (figure 19-20). The inputs, key internal signals and output signals are monitored to confirm correction function. In this case, the "Davg" average disparity value and the pair of updated object horizontal & vertical coordinates. To emulate the, Blob detect signals, coordinate values are feed from the pattern generator. They represent a box, measuring at maximum 100x100 (variable down to 50+) pixels for testing (the size is arbitrary and is used to represent a significant object on screen, that isn't too small yet does not fill the frame area so memory is not overloaded). The test-bench method of testing is generic to VHDL tools. A specific and powerful method is provided by Xilinx ISE, Chipscope Pro. The Chipscope is a testing/analysis tool that allows for the virtual implementation of inputs, outputs and internal signals with data logging capability. This allows for virtual logic analysis of the FGPA design by directly running the design on the FPGA, not

test-bench file is required. Given the memory limitations, complexity of the emulation of the system and of just this project module complexity and memory requirements, implementing the VIO would maximize the recourses of the FPGA, along with the IP cores for the integrated control (ICON) & integrated logic analyzers (ILA) functions. (Inclusion of the virtual input/output (VIO) IP core does provide virtual access to I/O & FPGA signal, included I/O emulation. This is a versatile feature, yet it adds further complexity to incorporate it in the project design module; therefore, it was not implement in favor of direct ILA implementation and measurements). This is evident since emulation of camera pixel data of a moving object is required. In the case of the project the ICON & ILA cores are instantiated in to the design.
The ILA is used to capture the outputs (coordinates, frame valid signals) and inputs (generated object coordinates, colour, dimension, video synchronization data). (ICON supplies the control signal).
The trigger signal for the chipscope is when the enable signal goes high. (A more versatile method would have been to use VIO, but would require more coding effort to interface). Chipscope is used to capture and analysis the performance of module outputs, input and critical internal signals of the design under test via the near real time waveform record. ISim is much quicker to run given the many cycles to check just one frame on its waveform viewer, its mainly used for quick logic checks and analysis at small time scales (pico-milliseconds).

## 4.4 Design Implementation & Considerations

The method for implementing the objective of the project was developed to check the object position from existing data, comparing it to new high resolution stereo camera information and provide improved object coordinates & average object pixel displacement beyond that provided by the previous system using averaging. The project method checks the continuous stream of pixels against the coordinate limits. This method uses existing data obtained by the system while creating an improved data set used by the system as a whole. Since the project module will be part of the system, it will allow use area resources just for memory (each BRAM is sized to use 16Kb, a Xilinx BRAM is at maximum 32Kb and 4 BRAM's are used) in addition to the rest of the system requirements for logic and memory. The memory considerations are that image frames for both camera streams are held in RAM (averaging, past frame, subtraction block operations), while addition RAM is need for the project module and its memory needs (current frame & detected object). Design complexity is greatest in the main process where the boundary for the region of interest is determined (finding the top, bottom, left and right edges of the suspected object). Originally since the project module had many input/output ports using many bus lines (each camera data stream & pixel addressing requires 20bits) caused IO block utilization to rise to 73 out of 100 IOB. As this could not be implemented on its own (nor with a combined system) to "top level" file was created where only a clock and reset pins are visible to the Zynq and interconnection between the design

module and the top module provides functionality for testing. Another consideration for design is a stereo object check, where if the object is detected by one camera but not the other then its an artifact or the apparent object is too close for both cameras to see; therefore, distance cannot be determined. This may require a complete design review before incorporation as it would need to check both camera streams. Also a key consideration is to check the object itself for artifacts, pixel at a row that extend well beyond the confines of the main object. This can be caused a complex object (a person) that moves their arm quickly, their body is tracked, but the centroid where the arm extended has shifted. The coding is in VHDL so it could be run on any FPGA, except for the module which uses memory IP blocks from Xilinx. An alternative hardware platform is the use of an embedded SoC (system on chip) microprocessor or its emulation on a capable FPGA system (Xilinx with ARM processing core) via the Microblaze soft (virtual) processor. This allows for greater programming flexibility by using a higher level programming language (embedded/system C/C). Recoding of the instruction is best performed by following the previous functional instructions as described in section 4.2. Note that an actual embedded microprocessor/microcontroller operates using sequential instruction words, processed per clock cycle. Given the embedded platform this can take numerous clock cycles in a similar fashion to general computer processors; therefore, video frame rate requirement could not be met. The use of an FPGA allows for the direct logic implementation, significantly reducing operating cycles well within project requirements. The use of logic reconfiguration, both temporally and spatially would allow more efficient use of FPGA resources of logic & memory to perform all program functions. Meaning that given the FPGA's ability to only activate logic hardware that is needed, so other non active logic areas (spatial) are free to use for additional programs. This is also true in using logic resources over time (temporal) as completed programs free up resources over a period of clock cycles to the next program. For example, programs for image rectification, object tracking and range computation would share the FPGA resources as required to complete the programs.

## 4.5 System Integration & Test

To implement the design, the new project module must be integrated into the system design. As referred to in previous sections, the project module is placed between the object blob detection and 3D coordinate/filtering modules, across both camera streams (luminescence data is extracted from the colour transformation block). It also has direct access to both camera feeds to obtain new data. This approach utilizes the object blob detection & its previous modules (figure 17) while providing new camera data as full resolution (figure 19). The system was originally synthesised and run on the Xilinx Virtex-4 FPGA.

The hardware requirement is flexible to other Xilinx FPGA's. Near future implementation can use the Zynq development board, which can provide logic reconfiguration of resources of time & FPGA area will be the most optimal. This would allow each module to utilize FPGA logic resources as needed, then reallocate them to other modules as required to complete system functions. This is critical in controlling and utilizing memory allocation of the system for pixel/frame data.

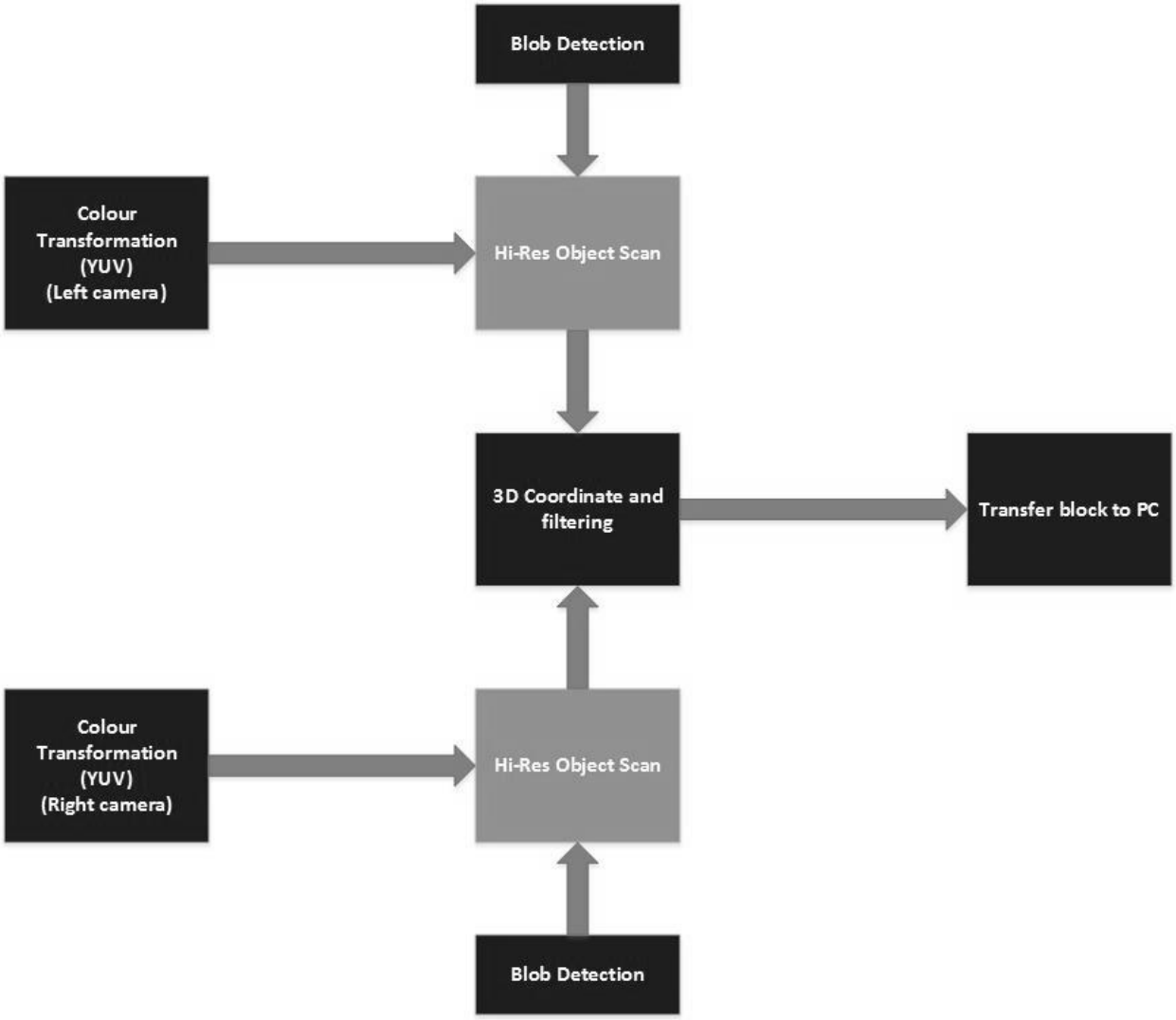


*Figure 19. System Integration of Project Module: Hi-Resolution Object Scanner (the lighter blocks show the project module, darker blocks are part of the existing M.A.R.S. system).*

Hardware selection and implementation was based on available equipment for test purposes. The existing main stereo vision system uses the Virtex-4 and may upgrade to utilize the Zybo for near future designs. The initial design used the Spartan3E as a reference, yet implementation was not possible due to lack of supported memory structures (Xilinx). Therefore, further testing & programming was performed on the

Zybo (Zynq Z-7010). This required the use of the Zynq set of constraints to specify the clock pin location. The testing of the design used both ISim and Chipscope via its IP cores. While ISim uses a pre-set test-bench and simulates the design logic over a period of time, Chipscope provides near real time signal recording. ISim was first used to confirm the basic operations of the program, its counters and to confirm that data for being transferred within the stages of the program. Chipscope proved more effective given its real time operation and signal probing capabilities via additional debug signals. For testing purposes the system is incorporated into a top level test HDL file (figure 20). This is the approach is used for the project module as well. The project module DUT (Design Under Test), along with a test pattern generator (defaults to VGA-640x480, 8bit colour pixels) and chipscope ILA & ICON, IP cores are instantiated into this top level file, where the test pattern generator connects to the DUT (using the green colour signal since it's doubly represented in the Bayer pattern) and DUT entity & critical internal signals are logged by the Chipscope ILA. Only a clock and reset pins are available at the top level, while the DUT I/O's are within this larger architecture (note: the Zynq does not have enough I/O pins for over 70 pins) (figureCH3). To confirm correct operations from emulation required that most bus signals (cameras, pixel address, memory address, boundary values…) had to be scaled to the test generators output requirements, mainly 10-bit coordinate and pixel addresses (unlike the 7-bit bus needed for the M.A.R.S.) which effects the size of internal buses and interconnecting signals. Full hardware integration requires a simple reduction of bus sizes for interoperability.

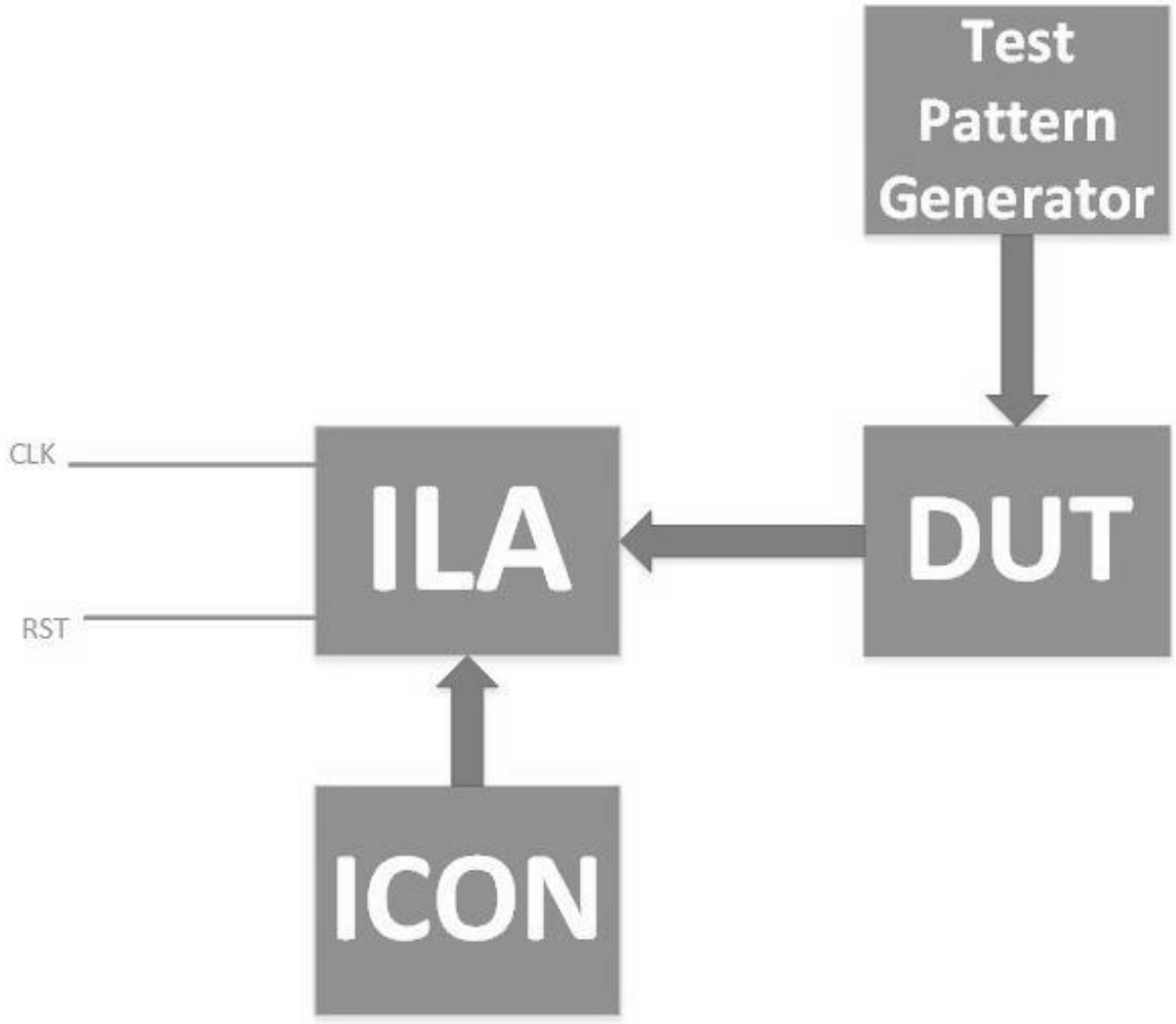


*Figure 20. Top Level Test System. To emulated the actual system, a pattern generator is used to provide video data for object tracking to the project module (Design Under Test). The outputs & critical internal signals are sent to the ILA for further analysis. (The ILA & ICON are virtual analysis instruments provided by the Xilinx Chipscope program: ILA: Integrated Logic Analyser, ICON, Integrated Control modules).*

Data captured from Chipscope presents the simulated camera data stream, the internal memory processing and object boundary determination per clock cycle and frame period. This is presented in the following figures (figure 5-10) show input and output clocked transitions, control signal states and internal memory address/data signals. The figures can be divided into two sets (figure 21-23, 24-26) presenting two test runs respectively. The object tracking test results can be confirmed by observing the waveform outputs of the resultant coordinates (left & right) and the disparity measures value. Given the debugging signal set up, the internal signals/buses for the object edge detection (left, right, top, bottom) of both camera streams (left=1, right=2) has also been used to double check the outputs (figure 21-26). Both sets show the input/output coordinates (both left and right camera streams are shown), disparity, edge detectors (top,

bottom, left, right), control strobe signals (flag1-3, frame-valid, write-enable, mem-on), the horizontal & vertical counters (produced from the pattern generator), the input object dimension and half its value used for internal calculation and both sets of the memory addresses/data (the main memory uses “transaddr” while the suspected object memory is controlled by “objaddr”. The “H” & “V” are the internal object search addresses used to divide the task of finding the objects edges allow its horizontal and vertical addresses. Also the compare bus signal is used to check the function of the initial object detection by comparing memory to stream frame data.

## 4.6 Verification

From the test results the disparity value is computed correctly, the output coordinates of the suspect object are not fully meeting requirements. Monitoring of internal signals are revealed to show edge detection algorithm is not converging from the window of interest fast enough (per frame) and is begin reset by the new position data. This is further discussed in the next chapter in comparative analysis.

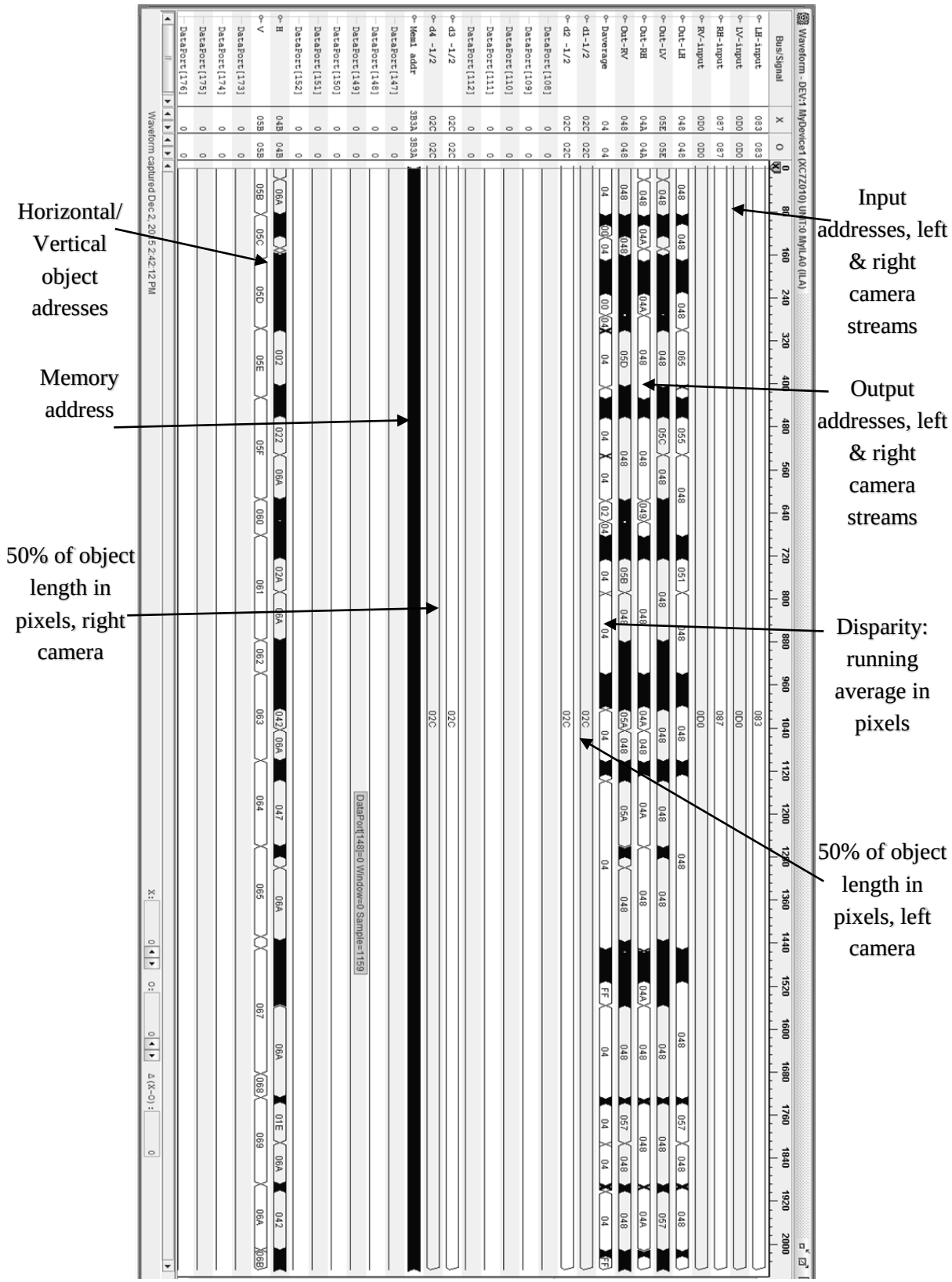


*Figure 21. Chipscope Data Capture/Operations. Input Green signal and memory addressing are shown at the lower part of the image ("obj data"-1/2, "mem addr#") with input and output coordinates show near the centre.*

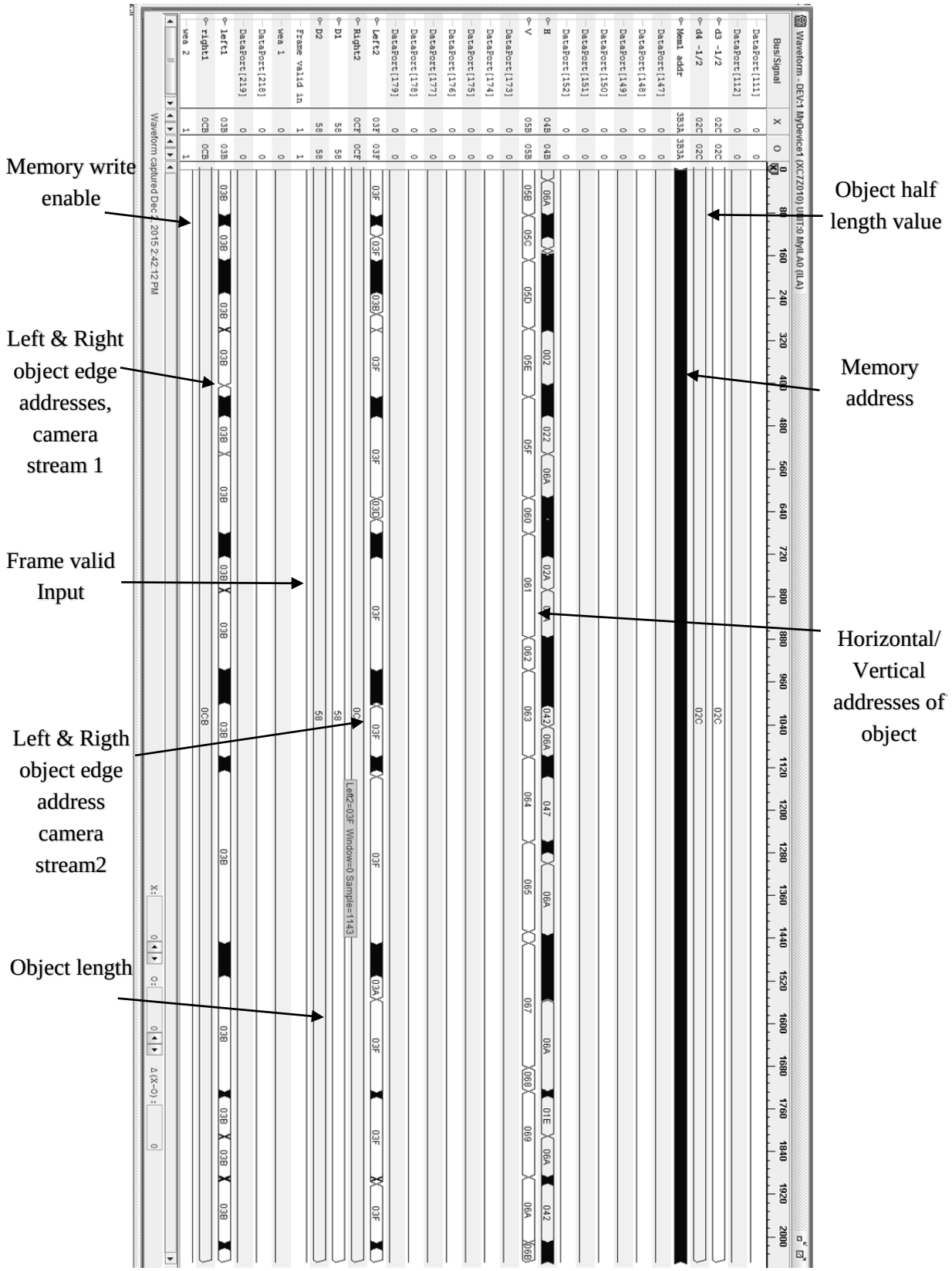


*Figure 22. Additional Cycles, enable signals shown (top section). The lower section shows the computed half object dimension value.*

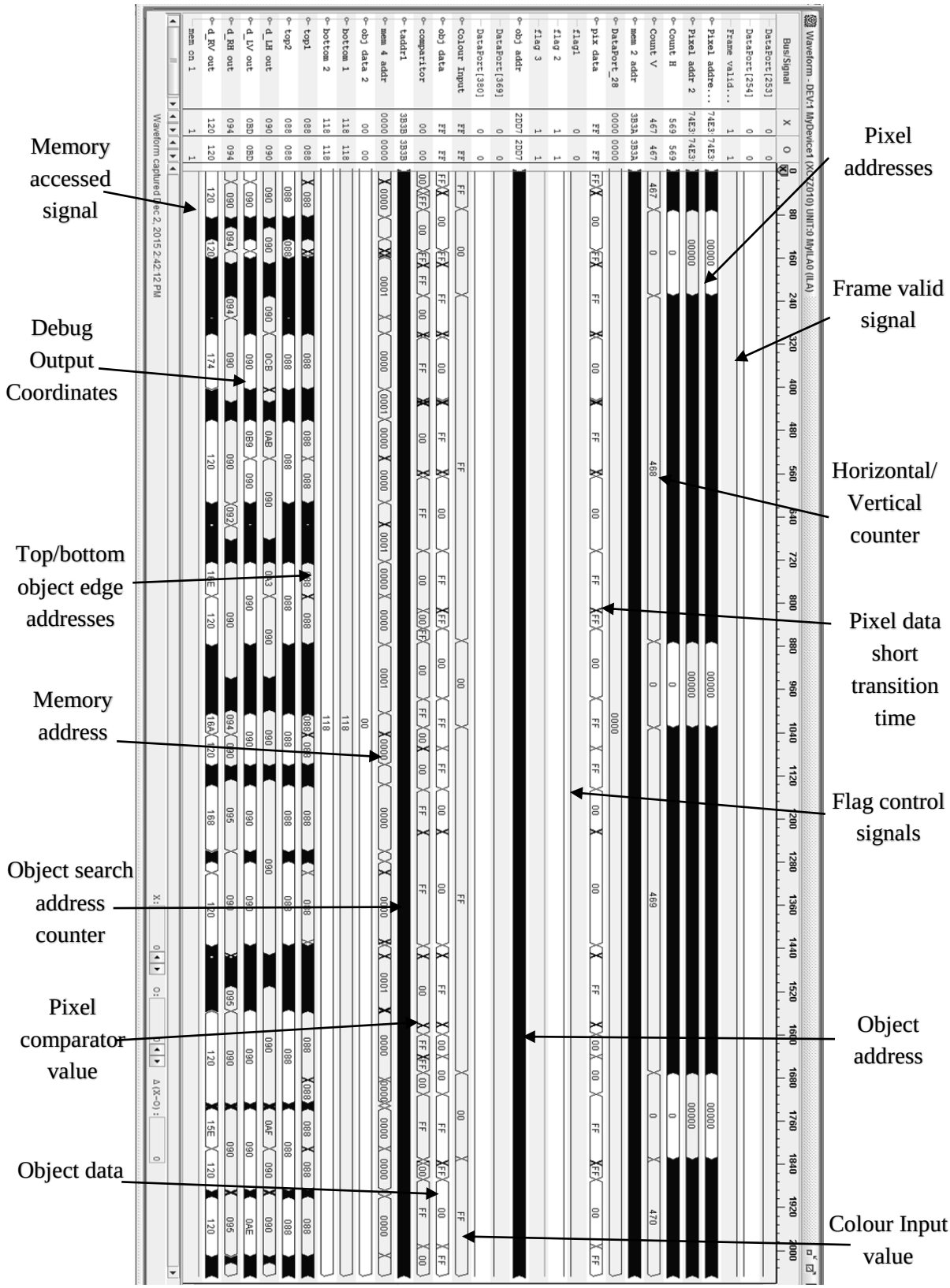


*Figure 23. Coordinate updates, with data input (green colour signal) from pattern generator. And the output edge dimensions are shown near the bottom.*

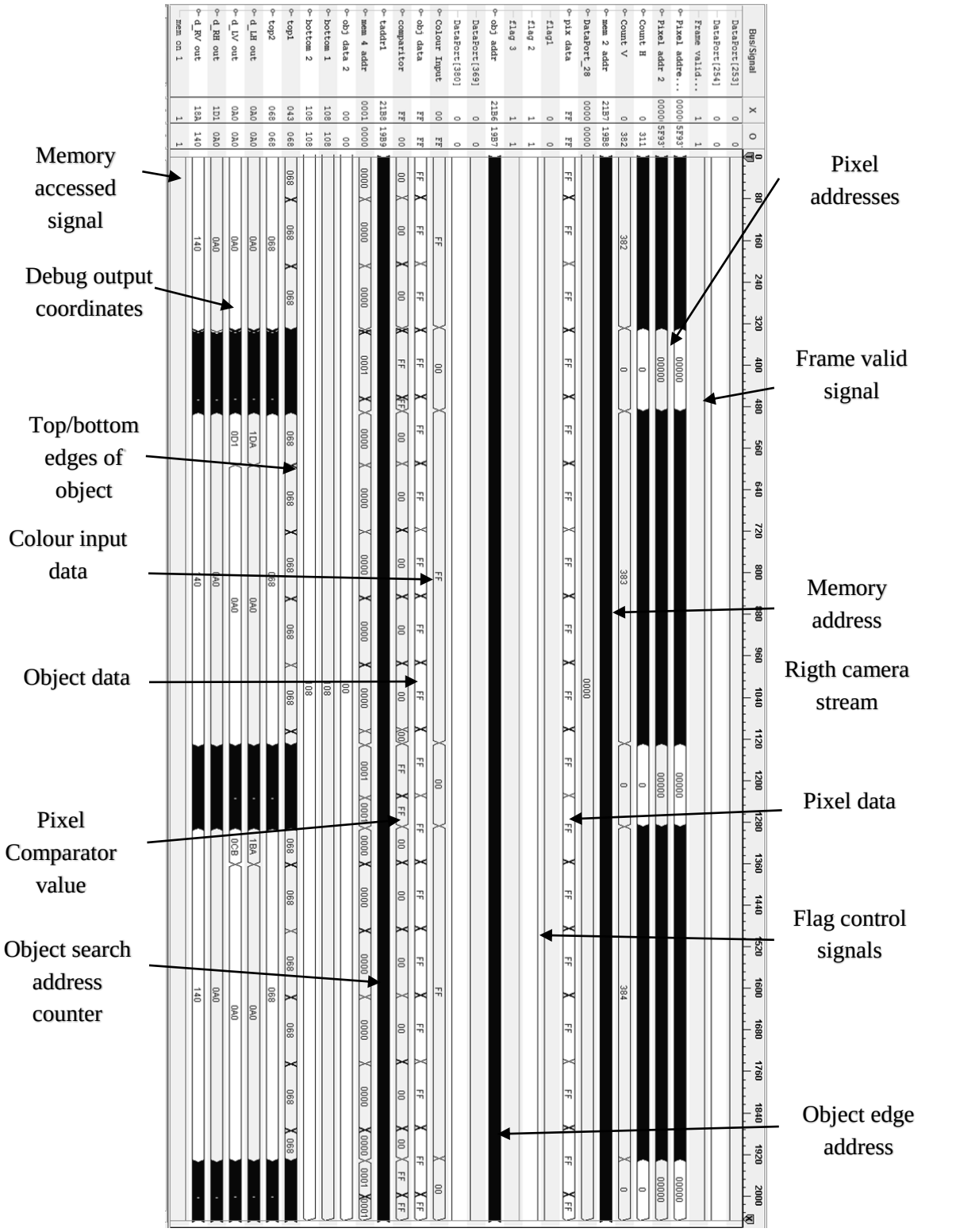


*Figure 24. Pixel address and the horizontal/vertical counter transitions are shown at the top. Then main memory address & object data, followed by control flag signals. The generated pixel data is at the centre, followed by additional object & comparator data showing where edge transitions have occurred. The top & bottom edge address is presented and then the output dimensions of the object.*

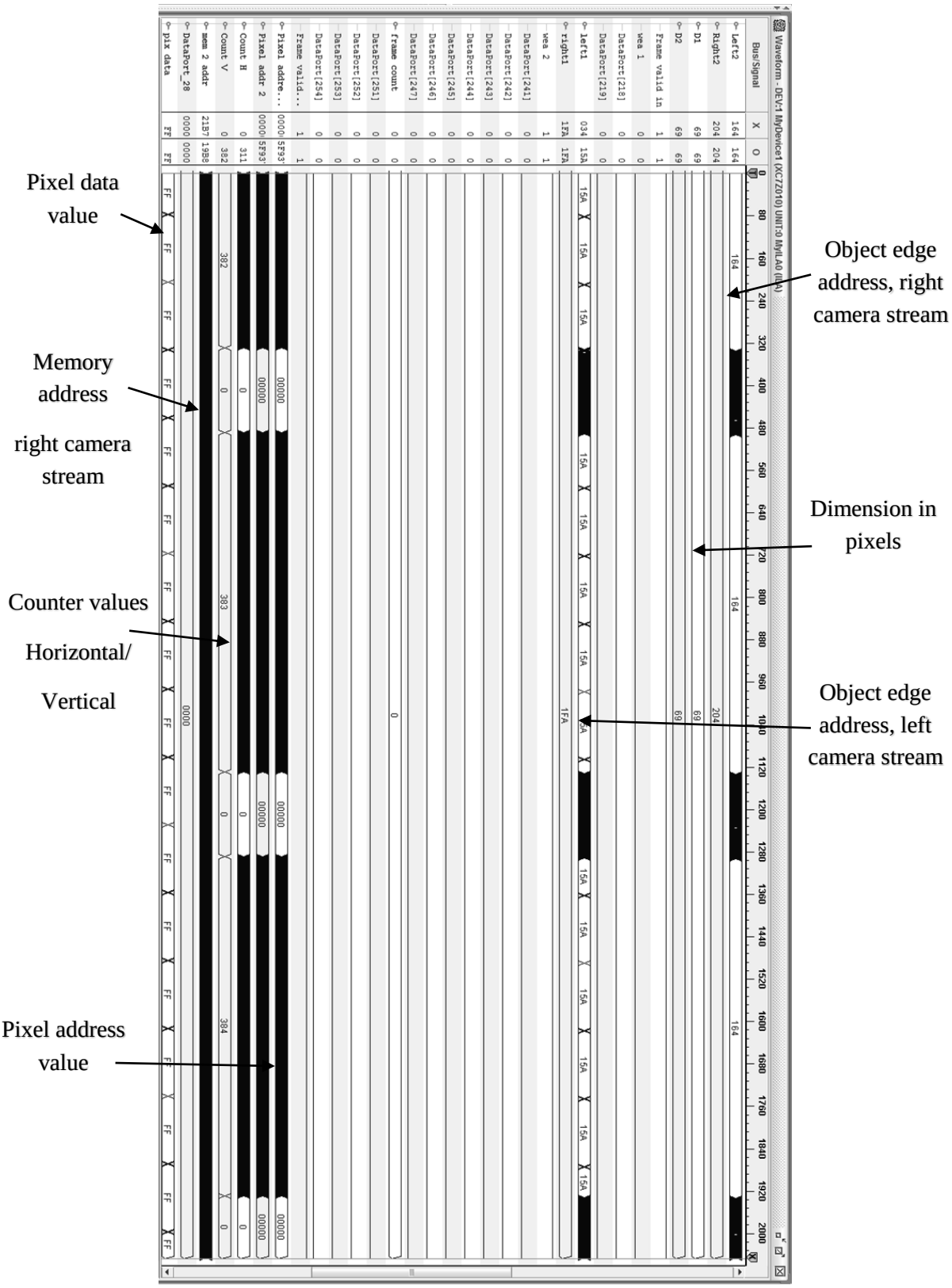


*Figure 25. From the top (timing data) to bottom, shows the left & right object edge addresses. Then the write-enable, frame count & frame valid signals, then the main pixel address counter, main memory address and pixel data.*

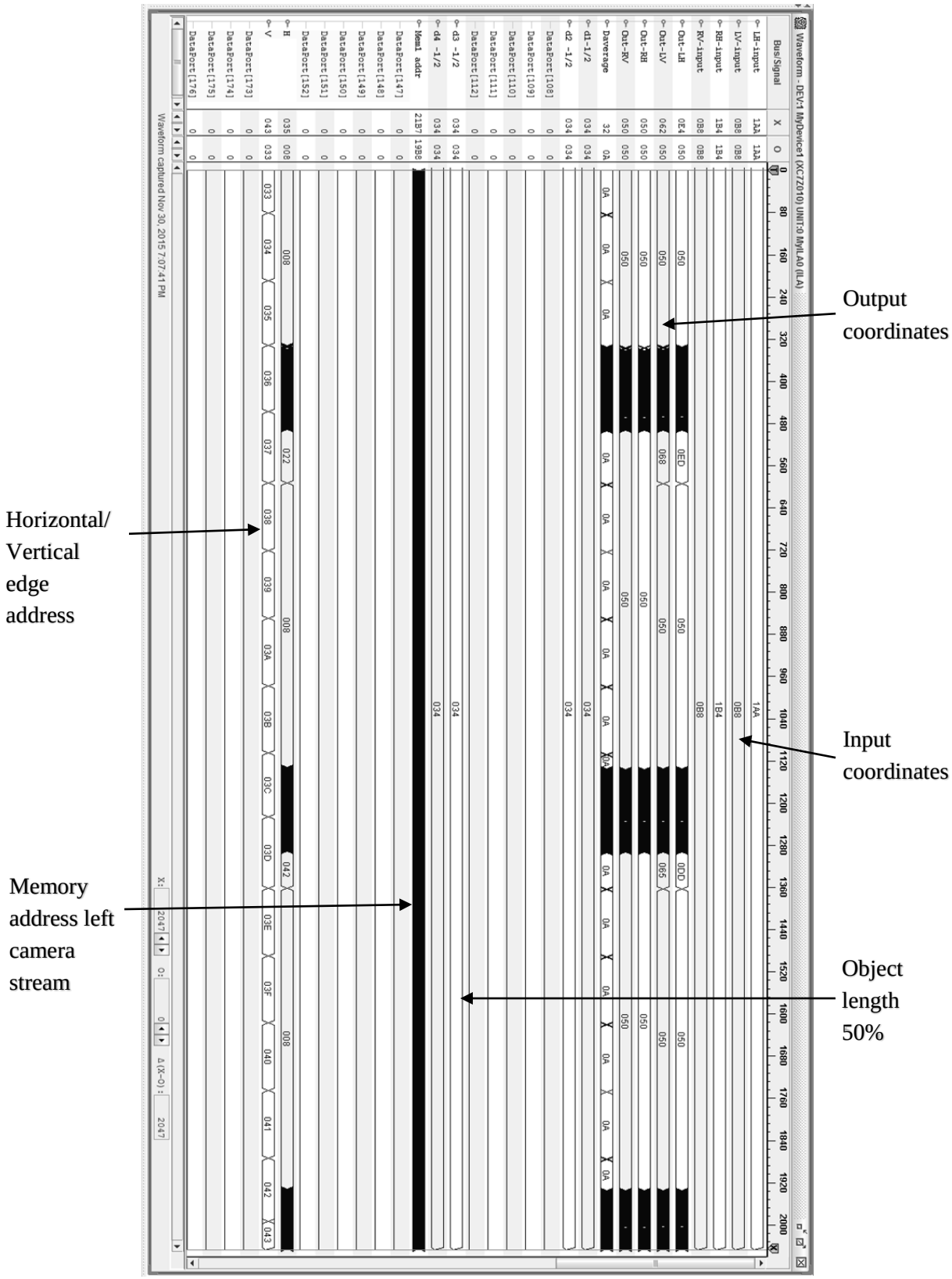


*Figure 26. In & output coordinate appear from the top where timing information is shown, next is the disparity, half dimension values, then the horizontal & vertical object address locations are shown near the bottom of the page.*

# 5 Comparative analysis

The project's implementing a High resolution object tracking HDL module to improve upon the existing M.A.R.S. system [10] produced an interesting set of results owning to the difference in platform used (the Zynq XC7Z010CLG400 vs Virtex-4 XC4VLX160). The first step was verification of the module operation. The project's High resolution object tracking module was emulated on both the Zynq and Virtex-4, for comparative analysis. As previously stated, a pattern generator was used to provide simulated object data, while the resultant coordinates and displacement of the detected and tracked objected were recorded. The analysis also reports on logic resources utilized and the degree of accuracy of the object tracking module on the respective Xilinx platforms. Due to hardware access availability and the ongoing needs of the, Embedded Reconfigurable System Lab (ERSL) a Virtex-4 could not be utilized directly, so modelling (simulation of hardware resources) was used to obtain the Virtex-4 resource utilization values. As a further demonstration of concept, a MATLAB model of disparity and rectification is presented to show the concept directly in a high level model [4], [5], [8], [10]. The first of the stereo vision process is rectification of the stereo pair of images, the process by which corresponding points on the images (figure 27) are found [3], [6], [8]. Then the disparity from the rectified images are computed [3], [6], [8] (figure 27-28). In the case of this report the rectified image frames of the object of interest are sent to the high resolution object tracking algorithm where based on the known coordinates the object window [8] will search reconfirming the object's coordinates and disparity in the form of displacement from left and right video streams. This information along with the camera focal length, the distance between them and the coordinates within each respective frame are used in the equation (3), solving for the distance from the cameras to the object where f is the focal distance, c is the distance between cameras and a (a=c/2, c=2a) is distance from the central optical axis with Z being the distance to the object [4], [6], [8].

Equation 3: $Z = \frac{-2af}{X_L - X_R}$, since $\frac{X_L}{f} = \frac{x-a}{f}, \frac{X_R}{f} = \frac{x+a}{f}$

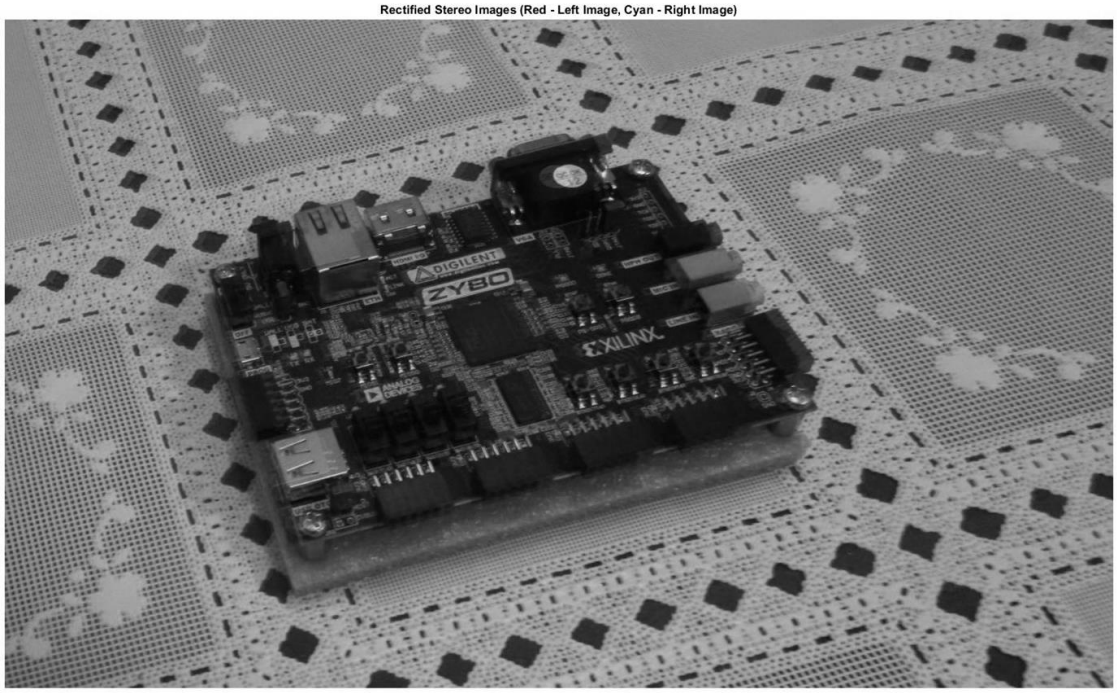


*Figure 27. Rectified Stereo Composite Image with anaglyph.*

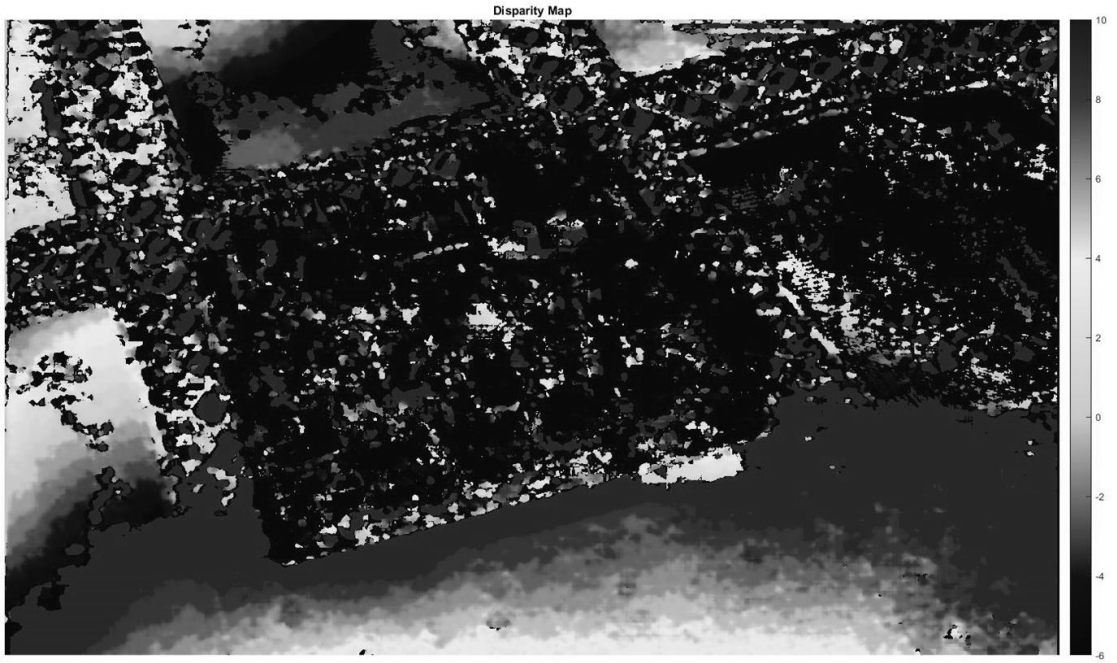


*Figure 28. Disparity Map of rectified image, with map scale.*

For example, a stereo vision camera set up (figure 4, 29), where each camera has a focal length of 3.7mm, the distance between then is 30mm (both cameras are horizontally parallel); therefore, if XL=0.05 and XR=0.1mm (on camera frame area) then the object is 222.0 cm distant. For a broader range of systems, increasing the distance between the stereo set of camera (disparity between left and right images) results in object detection at much greater ranges from 10m to kilometres; the cameras would also require its optics to be adjusted, both in focal length and lens size.

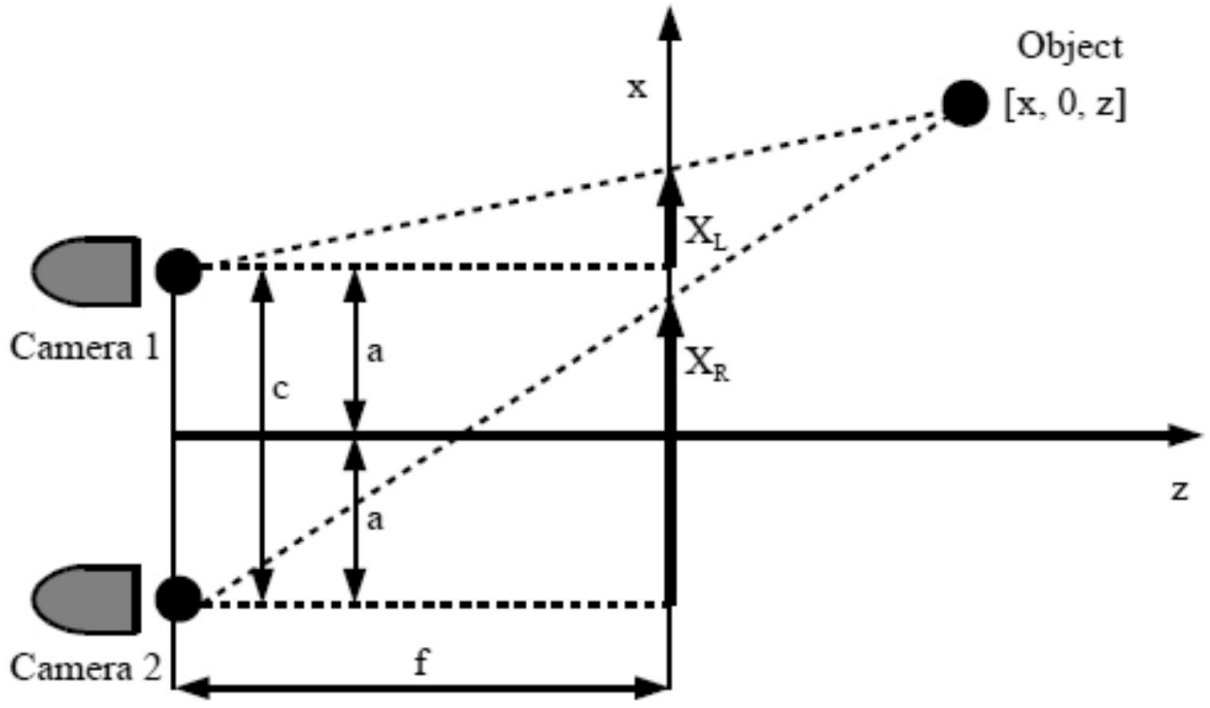


*Figure 29. Stereo vision Configuration. Object estimation of the Z axis using rectified cameras. The Y axis is zero to present the Z axis in 2D. Camera distance is c=2a, camera focal length is f and XL & XR is the x-coordinate displacement of the cameras* [8].

## 5.1 Resource Utilization

The following is a report directly from the Xilinx ISE for the Zynq project design, primarily presenting key logic resource utilization from the synthesis report (table 3,4). This is extracted from the full Xilinx

hardware synthesis report. Additional power resource measurements were also taken, showing voltage levels and power usage of the Virtex-4 (figure 31) and the Zynq (figure 32). The measure of resources also includes the monetary cost of the FPGA platform, where the entire system can be loaded and run on a low end FPGA at a cost of $100 USD (Xilinx Arty: Artix-7) or the current platform used in this project the Xilinx Zybo at $180 USD [11]. In comparison a high end fully featured FPGA can upwards of serval thousands of dollars [11]. Memory & logic usage on the Xilinx Zynq system are listed in the below (table 3,4). Also note that Xilinx BRAM's (logic elements organized into memory) are 36kb each, they were configured to operate as duel port RAM of 18kb wide [11]. This data must also take into consideration the original base system as implemented on the Zynq platform as well as the complete amount of logic resources utilized by the complete project system (base + project module), table 5.

| **Logic Resource** | **Amount** | **Power** | **Usage** |
|---|---|---|---|
| **BRAM** | 4 (7 of 6000 slices) | Supply power | 0.129 W |
| **Adder/Subtractors** | 50 | Dynamic | 0.027 W |
| **Registers** | 70 | Quiescent | 0.102 W |
| **Latches** | 31 | **Thermal Properties** | |
| **Comparators** | 35 | Effective TJA | 5.5 C/W |
| **Multiplexers** | 32 | Max Ambient | 84.3C |
| **LUT** | 284: 17600 logic slices | Junction | 25.7C |

Table 3. Zynq Resource Utilization.

The base system by its self also used 703 registers and 8 BRAMs (RAM36E) [11], while using 504 LUTs as logic. The actual sizes of the BRAMs used from the resource analysis (figure 33) listed the 16Kb size configuration that used 8 BRAMs. While the 8Kb configuration used 2 BRAMs. That accounts for 10 BRAMs of totaling 24Kb, which is still less than the maximum 36Kb size limit [11]. The total BRAMs available is 60 BRAMs (each 36kb), so more memory can be utilized. The key is in limiting additional usage to what is needed, since memory is taking from the general hardware logic used by the program.

| Logic Resource | Amount | Power Type | Usage |
|---|---|---|---|
| **BRAM** | 4 | Supply power | 1.076 W |
| **Adder/Subtractors** | 42 | Dynamic | 0.000 (sim) W |
| **Registers** | 121 | Quiescent | 7.076 W |
| **Latches** | 14 | **Thermal Properties** | |
| **Comparators** | 44 | Effective TJA | 5.9 C/W |
| **Multiplexers** | N/A | Max Ambient | 78.7C |
| **LUT** | 474 | Junction | 56.3C |

Table 4. Virtex-4 (Simulated) Resource Utilization.

| Logic Resource | Amount | Power Type | Usage |
|---|---|---|---|
| **Registers** | 703 /35200 | Supply Power | 0.1 W |
| **LUT** | 562/17600 | Dynamic | 0.0 (sim) W |
| **BRAM** | 4 of 60 (macro units) | Quiescent | 0.1 W |
| **LUT as BRAM** | 26/6000 | **Thermal Properties** | |
| **Adders/Subtractors** | 46 | Effective TJA | 5.5 C/W |
| **Comparators** | 30 | Max Ambient | 84.5 |
| **Multiplexers** | 106 | Junction | 25.5 |

*Table 5. Zybo Complete System (Base+Project) model resource utilization.*

| Test_Vis_Sys Project Status (11/16/2015 - 16:12:01) | | | |
|---|---|---|---|
| **Project File:** | HiResObjScan.xise | **Parser Errors:** | No Errors |
| **Module Name:** | Test_Vis_Sys | **Implementation State:** | Programming File Generated |
| **Target Device:** | xc7z010-1clg400 | • **Errors:** | No Errors |
| **Product Version:** | ISE 14.7 | • **Warnings:** | 223 Warnings (219 new) |
| **Design Goal:** | Balanced | • **Routing Results:** | All Signals Completely Routed |
| **Design Strategy:** | Xilinx Default (unlocked) | • **Timing Constraints:** | All Constraints Met |
| **Environment:** | System Settings | • **Final Timing Score:** | 0 (Timing Report) |

[+] Device Utilization Summary

[-] Performance Summary

| | | | |
|---|---|---|---|
| **Final Timing Score:** | 0 (Setup: 0, Hold: 0, Component Switching Limit: 0) | **Pinout Data:** | Pinout Report |
| **Routing Results:** | All Signals Completely Routed | **Clock Data:** | Clock Report |
| **Timing Constraints:** | All Constraints Met | | |

[-] Detailed Reports

| Report Name | Status | Generated | Errors | Warnings | Infos |
|---|---|---|---|---|---|
| Synthesis Report | Current | Mon Nov 16 16:08:58 2015 | 0 | 216 Warnings (216 new) | 161 Infos (161 new) |
| Translation Report | Current | Mon Nov 16 16:09:17 2015 | 0 | 0 | 0 |
| Map Report | Current | Mon Nov 16 16:10:13 2015 | 0 | 3 Warnings (3 new) | 6 Infos (5 new) |
| Place and Route Report | Current | Mon Nov 16 16:10:43 2015 | 0 | 1 Warning (0 new) | 1 Info (0 new) |
| Power Report | | | | | |
| Post-PAR Static Timing Report | Current | Mon Nov 16 16:11:02 2015 | 0 | 0 | 3 Infos (0 new) |
| Bitgen Report | Current | Mon Nov 16 16:11:51 2015 | 0 | 3 Warnings (0 new) | 1 Info (0 new) |

[+] Secondary Reports

*Figure 30. Emulated Full FPGA Report Summary Zynq platform.*

*Figure 31. Virtex-4 Simulated Power Resources.*

| Device | |
|---|---|
| Family | Zynq-7000 |
| Part | xc7z010 |
| Package | clg400 |
| Temp Grade | Commercial |
| Process | Typical |
| Speed Grade | -1 |
| Environment | |
| Ambient Temp (C) | 25.0 |
| Use custom TJA? | No |
| Custom TJA (C/W) | NA |
| Airflow (LFM) | 250 |
| Heat Sink | Medium Profile |
| Custom TSA (C/W) | NA |
| Board Selection | Medium (10"x10") |
| # of Board Layers | 8 to 11 |
| Custom TJB (C/W) | NA |
| Board Temperature (C) | NA |

| On-Chip | Power (W) | Used | Available | Utilization (%) |
|---|---|---|---|---|
| Clocks | 0.002 | 5 | --- | --- |
| Logic | 0.000 | 939 | 17600 | 5 |
| Signals | 0.000 | 2389 | --- | --- |
| BRAMs | 0.024 | 1 | 60 | 1 |
| IOs | 0.000 | 1 | 230 | 0 |
| Leakage | 0.102 | | | |
| Total | 0.129 | | | |

| Thermal Properties | Effective TJA (C/W) | Max Ambient (C) | Junction Temp (C) |
|---|---|---|---|
| | 5.5 | 84.3 | 25.7 |

| Supply Summary | Voltage | Total Current (A) | Dynamic Current (A) | Quiescent Current (A) |
|---|---|---|---|---|
| Vccint | 1.000 | 0.031 | 0.025 | 0.006 |
| Vccaux | 1.800 | 0.006 | 0.000 | 0.006 |
| Vcco33 | 3.300 | 0.001 | 0.000 | 0.001 |
| Vccbram | 1.000 | 0.003 | 0.002 | 0.001 |
| Vccpint | 1.000 | 0.020 | 0.000 | 0.020 |
| Vccpaux | 1.800 | 0.013 | 0.000 | 0.013 |
| Vcco_ddr | 1.500 | 0.002 | 0.000 | 0.002 |
| Vccadc | 1.710 | 0.020 | 0.000 | 0.020 |

| Supply Power (W) | Total | Dynamic | Quiescent |
|---|---|---|---|
| | 0.129 | 0.027 | 0.102 |

**The Power Analysis is up to date.**

(*) Place mouse over the asterisk for more detailed BRAM utilization.

Table View

*Figure 32. Zynq Power Metering of Resources.*

| Device | |
|---|---|
| Family | Zynq-7000 |
| Part | xc7z010 |
| Package | clg400 |
| Temp Grade | Commercial |
| Process | Typical |
| Speed Grade | -1 |

| Environment | |
|---|---|
| Ambient Temp (C) | 25.0 |
| Use custom TJA? | No |
| Custom TJA (C/W) | NA |
| Airflow (LFM) | 250 |
| Heat Sink | Medium Profile |
| Custom TSA (C/W) | NA |
| Board Selection | Medium (10"x10") |
| # of Board Layers | 8 to 11 |
| Custom TJB (C/W) | NA |
| Board Temperature (C) | NA |

| Characterization | |
|---|---|
| Production | v1.0.2012-07-11 |

| On-Chip | Power (W) | Used | Available | Utilization (%) |
|---|---|---|---|---|
| Clocks | 0.000 | 6 | --- | --- |
| Logic | 0.000 | 567 | 17600 | 3 |
| Signals | 0.000 | 1036 | --- | --- |
| BRAMs | 0.000 | * | * | * |
| IOs | 0.000 | 100 | 230 | 43 |
| Leakage | 0.100 | | | |
| Total | 0.100 | | | |

| Thermal Properties | Effective TJA (C/W) | Max Ambient (C) | Junction Temp (C) |
|---|---|---|---|
| | 5.5 | 84.5 | 25.5 |

| Supply Summary | | Total | Dynamic | Quiescent |
|---|---|---|---|---|
| Source | Voltage | Current (A) | Current (A) | Current (A) |
| Vccint | 1.000 | 0.005 | 0.000 | 0.005 |
| Vccaux | 1.800 | 0.006 | 0.000 | 0.006 |
| Vcco 18 | 1.800 | 0.001 | 0.000 | 0.001 |
| Vccbram | 1.000 | 0.000 | 0.000 | 0.000 |
| Vccpint | 1.000 | 0.020 | 0.000 | 0.020 |
| Vccpaux | 1.800 | 0.013 | 0.000 | 0.013 |
| Vcco_ddr | 1.500 | 0.002 | 0.000 | 0.002 |
| Vccadc | 1.710 | 0.020 | 0.000 | 0.020 |

| Supply Power (W) | Total | Dynamic | Quiescent |
|---|---|---|---|
| | 0.100 | 0.000 | 0.100 |

*Figure 33. Power simulation of unified system (base + project). Note the asterisk, for used BRAM: 8k BRAM=2, 16k BRAM =8.*

## 5.2 Object Tracking Performance

The determination of performance of the system and its modules depends on its accuracy of the design (high resolution object scanner) module to track an emulated object's coordinates as it changes location within the frame, over multiple frames. Also comparison of the accuracy of the determined disparity of both left & right frames is the key performance factor (figure 4,18,34). Actual system integration and operation with the project's module is the final real performance measurement. The same design was emulated on both the Zynq and Virtex-4 FPGA (with the Zynq serving as the base model for operation). With the project module that updates the system resolution with updated higher resolution image data, without compression; performance is improved by 64 times. (Since the original system segmented each image in 8x8 blocks, each block is 80x60 pixels. As a consequence, the smallest object can only be approximated to only 10 x 7.5 pixels in size). This provides a clear advantage for per pixel operations on disparity and coordinate computation. As stated in the previous chapter verification section 4.6, the disparity is correctly computed, the object coordinate value updates were not within frame valid timing; this applies to both cameras streams respectively. Therefore, this design has not met one of the sub requirement of the project and the original system. The M.A.R.S platform has a working 2D object tracking system which compensates for this. Monitoring of the internal signals revealed that the edge detection algorithm is not converging from the window of interest within frame processing times.
The edge detector vectors (left, right, top, bottom) for both camera data streams (left=1, right=2) appear not to converge on the object dimensions but hold its original window of interest values (object plus a margin of 28 pixels). The object tracking system relies on confirming the pixel are apart of the object of interest with the search window. With operations at a 1D level, pixel change (the indication of movement) was found by subtracting frames and checking per pixel. Higher level tools like MATLAB operate on a frame in 2D, allowing for more sophisticated filters, such as Sobel, Kalman, meanshift, KTL, CAMshift etc., to check a group of pixels from the region of interest to determine the object [3]. Though more efficient, the additional complexity and interfacing to the M.A.R.S. was found to be too cumbersome given 1D video data streams.

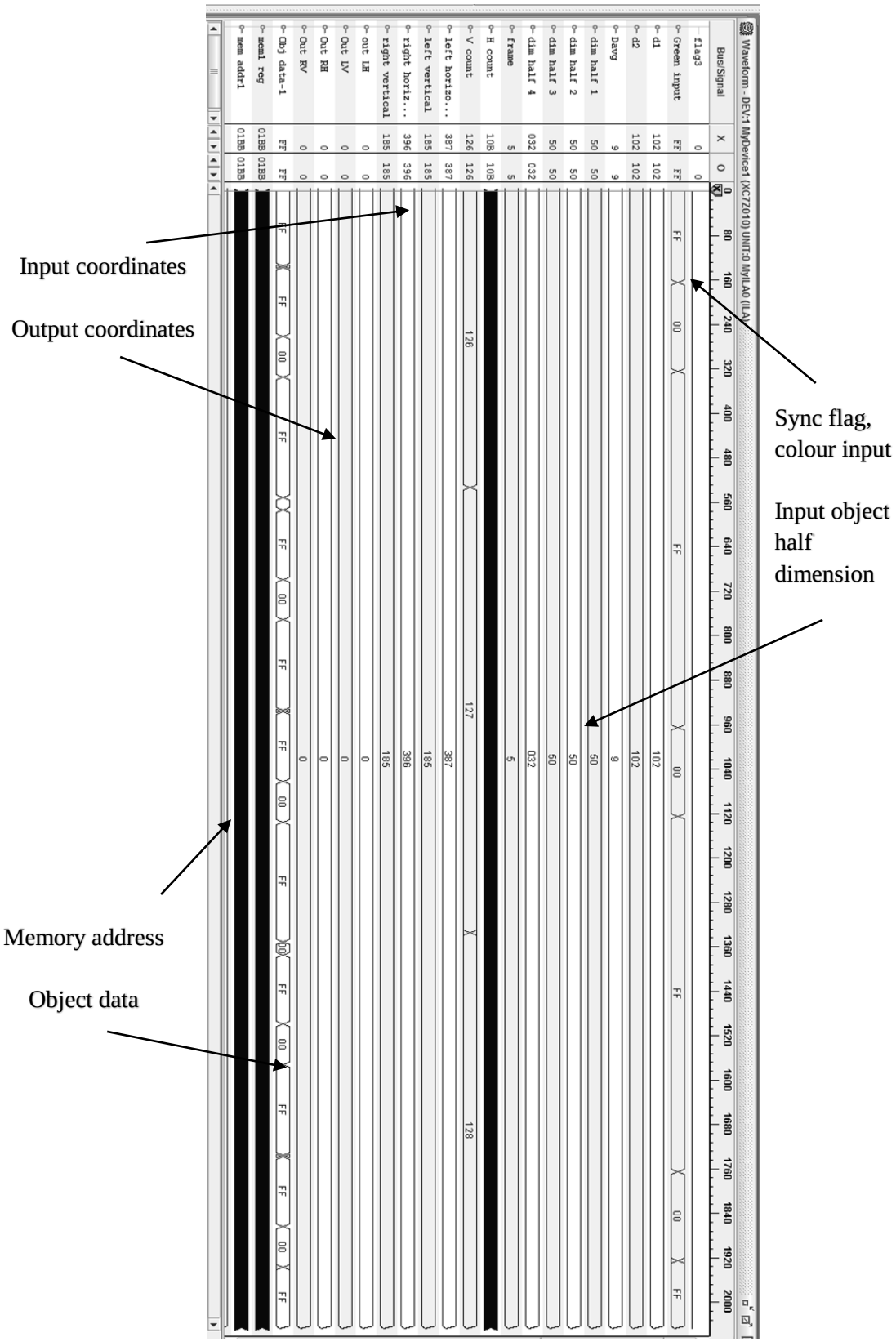


*Figure 34. Input/output Coordinate, disparity tracking (Davg). The dimensions are listed at d1/2, while the input coordinates are left/right horizontal/vertical and the output coordinates are names in the out - LH, LV, RH, RV convention.*

# 6 Summary

## 6.1 Conclusion

The high resolution stereo vision project objective was to improve object tracking & disparity accuracy by focusing the window of interest on the suspect object area at known coordinates and recheck at camera pixel resolution for a moving object. And from this data compute more accurately the distance to the object. This was performed while utilizing FPGA memory as needed without degrading image quality through pixel block averaging as in M.A.R.S., to save memory. The complete system, project and M.A.R.S., base system uses a similar amount given logic optimization and additional space still available. The current state of the art in research was examined to find related and similar approaches in computer vision and image processing. Development of the solution method focused on isolating the one method to use to complete the project objective. Implementation focused on making the 2D and 3D operations function on a per frame basis and minimize memory use a required by focusing the search on a region of interest within the frame. This involved a multi HDL level approach to efficiently use logic resources and IOBs given the multiple bus inputs and outputs needed for system interfacing. The project HDL model with test pattern generator used logic and memory resources as need, only four BRAMs were used in the FPGA platform.

## 6.2 Future Work

The recommendations for future work on this project, lay in its application and implementation towards further projects in multi object tracking and robotic systems for vehicle navigation. The current design focuses on one object within the region of interest, but for real applications multi-object tracking is necessary. Note that further testing and calibration of the 2D tracking in this project is still required. This is fully functional in the original M.A.R.S. project, so for completeness having the integrated system (either added to or as a stand alone) operate is a goal to strive for. For robotic applications the stereo vision system will need to interface with a real time control system that will use stereo vision to find or to track objects of interest and proceed given the program objectives. For example, a robotic probe must position itself close to a target of interest, a UAV could recognize landmarks or the landing site visually for landing and takeoff operations, a robot arm can be guided autonomously by an artificial neural network to manipulate objects in its environment. Also this can be applied to human machine interfaces in the form of augmented reality interfacing, where a projection (image or holograph) is used to represent the interface. The operator's actions in 3D space are interpreted by the system to execute the commanded operation (like using a virtual console or control column).

# References


[1] G. R. Curry, *Pocket Radar Guide - Key Radar Facts, Equations, and Data*. SciTech Publishing, 2010.

[2] S. Karp and L. B. Stotts, “Fundamentals of Electro-Optic Systems Design : Communications, Lidar, and Imaging.” Cambridge University Press, West Nyack, 2012.

[3] W. Kong, D. Zhou, Y. Zhang, D. Zhang, X. Wang, B. Zhao, C. Yan, L. Shen, and J. Zhang, “A ground-based optical system for autonomous landing of a fixed wing UAV,” *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on*. pp. 4797–4804, 2014.

[4] A. Din, B. Bona, J. Morrissette, M. Hussain, M. Violante, and M. F. Naseem, “Embedded Low Power Controller for Autonomous Landing of UAV Using Artificial Neural Network,” *2012 10th Int. Conf. Front. Inf. Technol.*, pp. 196–203, 2012.

[5] L.-W. Zheng, Y.-H. Chang, Z.-Z. Li, Z. Li-Wei, C. Yuan-Hsiang, and L. Zhen-Zhong, “A study of 3D feature tracking and localization using a stereo vision system,” in *Computer Symposium (ICS), 2010 International*, 2010, pp. 402–407.

[6] J. Kovacevic, S. Juric-Kavelj, I. Petrovic, J. Kovačević, and I. Petrović, “An Improved CamShift Algorithm Using Stereo Vision For Object Tracking,” *MIPRO, 2011 Proc. 34th Int. Conv.*, pp. 707–710, 2011.

[7] S. Schraml, A. N. Belbachir, N. Milosevic, and P. Schön, “Dynamic stereo vision system for real-time tracking,” *Circuits Syst. (ISCAS), Proc. 2010 IEEE Int. Symp.*, pp. 1409–1412, 2010.

[8] M. Puheim, M. Bundzel, and L. Madarasz, “Forward control of robotic arm using the information from stereo-vision tracking system,” in *Computational Intelligence and Informatics (CINTI), 2013 IEEE 14th International Symposium on*, 2013, pp. 57–62.

[9] Q. Baig, O. Aycard, T. D. Vu, and T. Fraichard, “Fusion between laser and stereo vision data for moving objects tracking in intersection like scenario,” *Intelligent Vehicles Symposium (IV), 2011 IEEE*. pp. 362–367, 2011.

[10] R. Raducu, “Soc for real -time object tracking in 3D space,” Ryerson University, 2014.

[11] Xilinx, “7 Series FPGAs Memory Resources,” *Xilinx Inc.*, 2014. [Online]. Available: http://www.xilinx.com/support/documentation/user_guides/ug473_7Series_Memory_Resources.pdf.

[12] A. W. Ruan, Y. Wang, K. Shi, Z. J. Zhu, Q. Wu, X. Han and Y. B. Liao, "SOC HW/SW co-verification technology for application of FPGA test and diagnosis," in Computational Problem-Solving (ICCP), 2011 International Conference on, 2011, pp. 1-6.

[13] B. J. Tomas, Yingtao Jiang and Mei Yang, "SoC scan-chain verification utilizing FPGA-based emulation platform and SCE-MI interface," in System-on-Chip Conference (SOCC), 2014 27th IEEE International, 2014, pp. 398-403.

[14] Zhenni Li, Jingjiao Li, Liang Li, Yue Zhao and Chaoqun Rong, "A SoC design and implementation of dynamic image edge detection based on the LEON3 open source processor," in Natural Computation (ICNC), 2013 Ninth International Conference on, 2013, pp. 1263-1267.

[15] Kuen-Jong Lee, Tong-Yu Hsieh, Ching-Yao Chang, Yu-Ting Hong and Wen-Cheng Huang, "On-Chip SOC Test Platform Design Based on IEEE 1500 Standard," Very Large Scale Integration (VLSI) Systems, IEEE Transactions on, vol. 18, pp. 1134-1139, 2010.

[16] Zhenni Li, Jingjiao Li, Yue Zhao, Chaoqun Rong and Ji Ma, "A SoC design and implementation of H.264 video encoding system based on FPGA," in Intelligent Human-Machine Systems and Cybernetics (IHMSC), 2014 Sixth International Conference on, 2014, pp. 321-324.

[17] S. Ben Othman, A. K. Ben Salem, H. Abdelkrim and S. Ben Saoud, "MPSoC design approach of FPGA-based controller for induction motor drive," in Industrial Technology (ICIT), 2012 IEEE International Conference on, 2012, pp. 134-139.

[18] B. Caglar, E. Caglav and I. Cadirci, "Combined microcontroller and FPGA control of flyback inverter using a system-on-chip device," in Power Electronics and Motion Control Conference and Exposition (PEMC), 2014 16th International, 2014, pp. 990-995.

[19] A. W. Ruan, Y. B. Liao, P. Li, W. Li and W. C. Li, "An automatic test approach for field programmable gate array (FPGA)," in Integrated Circuits, ISIC '09. Proceedings of the 2009 12th International Symposium on, 2009, pp. 474-477.

[20] Liao Yongbo, Ruan Aiwu, Wang Yu, Xiang Chuanyin, Wang Lin, Huang Haocheng and Zhu Jianhua, "Interconnect resources testing and faults diagnosis in field programmable gate arrays," in Electronic Measurement & Instruments (ICEMI), 2011 10th International Conference on, 2011, pp. 185-189.

[21] F. Moosmann and C. Stiller, "Joint self-localization and tracking of generic objects in 3D range data," in Robotics and Automation (ICRA), 2013 IEEE International Conference on, 2013, pp. 1146-1152.

[22] A. Azim and O. Aycard, "Detection, classification and tracking of moving objects in a 3D environment," in Intelligent Vehicles Symposium (IV), 2012 IEEE, 2012, pp. 802-807.

[23] A. Azim and O. Aycard, "Layer-based supervised classification of moving objects in outdoor dynamic environment using 3D laser scanner," in Intelligent Vehicles Symposium Proceedings, 2014 IEEE, 2014, pp. 1408-1414.

[24] J. Witt and U. Weltin, "Robust stereo visual odometry using iterative closest multiple lines," in Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on, 2013, pp. 4164-4171.

[25] Denglu Wu, Yingkui Du, Baojie Fan, Jiandong Tian and Yandong Tang, "A real-time vision system for telerobotic manipulator with large time delay," in Information Science and Engineering (ISISE), 2012 International Symposium on, 2012, pp. 460-464.

[26] Junda Zhu, Y. F. Zheng and R. E. Ewing, "Measuring object speed using stereo tracking," in Robotics and Automation (ICRA), 2011 IEEE International Conference on, 2011, pp. 5949-5954.

[27] R. Deepu, B. Honnaraju and S. Murali, "Automatic generation of second image from an image for stereovision," in Advances in Computing, Communications and Informatics (ICACCI, 2014 International Conference on, 2014, pp. 920-927.

[28] He Yuqing, Chen Shengfu, Gu Feng, Han Jianda and Dong Huiying, "UKF based moving target localizing and tracking method in 3D environments," in Control Conference (CCC), 2012 31st Chinese, 2012, pp. 5088-5093.

[29] S. A. Rodriguez F, V. Fremont and P. Bonnifait, "An experiment of a 3D real-time robust visual odometry for intelligent vehicles," in Intelligent Transportation Systems, 2009. ITSC '09. 12th International IEEE Conference on, 2009, pp. 1-6.

[30] A. Boyali and M. Kavakli, "3D and 6 DOF user input platform for computer vision applications and virtual reality," in Innovations in Intelligent Systems and Applications (INISTA), 2011 International Symposium on, 2011, pp. 258-263.

[31] R. Deepu, B. Honnaraju and S. Murali, "Automatic generation of second image from an image for stereovision," in Advances in Computing, Communications and Informatics (ICACCI, 2014 International Conference on, 2014, pp. 920-927.

## Definitions

| | |
|---|---|
| Blob | An averaged representation of an object, having insufficient pixel detail to approximate the true edges of the object. |
| Centroid | The geometric centre of a plane figure, the arithmetic mean position of all points in the shape. |
| Disparity | The difference between left and right camera points of view of the object. |
| Feature | The selection of a part or property of an object of interest for further image processing. |
| Field of View | The extent of observable surrounding that is visible to a given angle. |
| Focal Length | The distance between the camera outer lens to its inner lenses and light sensor. |
| F.P.G.A. | Field Programmable Gate Array: integrated circuit designed to allow designer to reprogram the circuit logic using hardware description language. |
| I.O.B. | FPGA circuit logic recourses used as input/output connections. |
| I.P. | Virtual integrated circuit proprietary design that can be programed and operate on range of FPGA hardware platform. |
| Rectification | The image processing process of matching the horizontal rows of two similar images. |
| Stereo Vision | The computer vision method of duplicating human vision by using a pair of digital cameras to produce a unified image with depth information. |

## Index