FPGA-BASED STEREO-VISION SYSTEM

MIMICKING HUMAN BINOCULAR VISION

By

GVARAMI LABARTKAVA

Bachelor of Computing, School of Mathematics and Computer Science,

Free University of Tbilisi, Georgia, 2017

A thesis

presented to Ryerson University

in partial fulfillment of the

requirements for the degree of

Master of Applied Science

in the Program of

Electrical and Computer Engineering

Toronto, Ontario, Canada, 2019

AUTHOR'S DECLARATION

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

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

I further authorize Ryerson University to reproduce this thesis 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 thesis may be made electronically available to the public.

FPGA-based Stereo-Vision System mimicking Human Binocular Vision

Master of Applied Science, 2019

GVARAMI LABARTKAVA

Electrical and Computer Engineering

Ryerson University

ABSTRACT

Human vision is a complex system which involves processing frames and retrieving information in a real-time with optimization of the memory, energy and computational resources usage. It can be widely utilized in many real-world applications from security systems to space missions. The research investigates fundamental principles of human vision and accordingly develops a FPGA-based video processing system with binocular vision, capable of high performance and real-time tracking of moving objects in 3D space.

The undertaken research and implementation consist of:

1. Analysis of concepts and methods of human vision system;
2. Development stereo and peripheral vision prototype of a system-on-programmable chip (SoPC) for multi-object motion detection and tracking;
3. Verification, test run and analysis of the experimental results gained on the prototype and associated with the performance constraints;

The implemented system proposes a platform for real-time applications which are limited in current approaches.

Acknowledgment

I would like to express my deepest appreciation to my supervisor Dr. Lev Kirischian for giving me direction, ideas and discussions for solving challenges in the research, sharing with me his great experience, and introducing me to new conceptual and effective methodologies for observation and analysis of the research. This research wouldn't have been possible without his contributions, supervising and support. It should be emphasized that his discussion was not only helpful for this research but broadened my mind and helped me better understand and critically analyze processes, generally, in everyday life.

I would also like to thank Dr. Vadim Geurkov for his constructive advice and support during the research and my master's studies. His lectures, suggestions, and comments about the research were extremely valuable and helped me to solve various problems and improve the design.

Lastly, I would like to thank the Electrical and Computer Engineering Department at Ryerson University and the Embedded Reconfigurable Systems Laboratory (ERSL) for providing me with laboratory equipment and supportive services, which helped me successfully conduct the research.

Table of Contents

List of Tables

List of Figures

List of Abbreviations

Abbreviation	Definition
AMBA	Microcontroller Bus Architecture
AXI	Advanced eXtensible Interface
BRAM	Block Ram
CC	Clock Cycle
FOV	Field of Views
FPGA	Field Programmable Gate Arrays
FPS	Frames per second
FT	Foot
HDL	Hardware Description Language
HDMI	High-Definition Multimedia Interface
HSL	Hue, Saturation, Lightness
HSYNC	Horizontal Synchronization signal
ILA	Integrated Logic Analyzer
I/O	Input/Output
IP	Intellectual Property
KF	Kalman Filter
KLT	Kanade–Lucas–Tomasi (tracker)
c	Mean Absolute Error
MF	Mean Filter
MIO	Multiplexed Input/Output
MS	millisecond
OF	Optical Flow
OpenCV	Open source Computer Vision
PARTS	Programmable and Reconfigurable Tool Set
PCB	Printed Circuit Board
PL	Programmable Logic
PS	Processing System
RAM	Random-Access Memory
RANSAC	Random Sample Consensus
RGB	Red, Green, Blue

RMSE	Root Mean Square Error
SIFT	Scale-Invariant Feature Transform
SoPC	System-on-Programmable Chip
SURF	Speeded Up Robust Features
SVM	Support Vector Machine
UART	Universal Asynchronous Receiver/Transmitter
VSYNC	Vertical Synchronization signal

Chapter 1

1. Introduction

The principles of human vision are widely used nowadays in industrial and research computer vision applications. Standard technologies of video systems store and process entire frames, whereas human vision only focuses on a small part of the whole Field-of-View (FOV), which optimizes power, timing and memory usage. The research aims to utilize the human “attention” technique, and implement and design of two principal concepts of human binocular vision:

1) Stereo vision for 3D-pose estimation
2) Peripheral vision for a wide field of view

with the ability of real-time multi-object motion detection and tracking in 3D space with a wider field of view.

The main challenges of the research involve: analysis of the principles of human vision; the explanation of the organization of human eyes which are able to track objects in 3D space; the process of data acquisition with the capability of memory and resources optimization and extracting features for the further processing for the custom applications.

It should be emphasized that proposed design utilizes FPGA, multiple video-sensors and embedded ARM for achieving stereo and peripheral vision goals and keep high frame rate for the processing which lets further industrial or research development use for real-time applications with limited resources. Moreover, it proposes custom actions and features using the trajectory of the objects.

In this chapter, the motivation of the research and reasons for the usage of the proposed platform is discussed, and the objectives of the research and implementation are determined. In addition, it highlights the parts of the original contribution and proposes the detailed organization of the rest of the work.

1.1 Motivation

There are many real-world applications, where principles of human vision are used for different purposes. Vision applications involve 3D scanners, medical imaging, space mission, security systems, photography and etc.

The main motivation of the research is to solve the limitations of current approaches in video and vision systems.

Human vision integrated two main base vision components:

1. Stereo vision, which is used for determining the position in 3D space
2. Peripheral vision: which makes it possible to capture and perceive objects in a wide field of view (> 150-degree angle) frames

Standard ways to achieve the peripheral vision and panoramic view includes special wide-angle cameras with a smaller focal length than standard ones or image stitching techniques which involve detecting overlaps between differently exposed frames captured by single/multiple cameras and calibration, blending and merging them into a single frame. There are various frameworks and algorithms, which proposes computer vision implementations such as OpenCV, Dlib, GIST1, SIFT-based image-stitching and others.
Traditional TV/Cinema video applications continuously store and process entire frames and use a significant amount of memory and computational resources, while using concepts of human vision can optimize frame processing for real-time applications.

The main limitations of traditional approaches and methods are:

1. **Timing**: The software frameworks are designed for ARM architecture and are limited due to timing and resource constraints. In addition, the algorithms don't include hardware parallelism and represent sequential logic or require processing time, which limits their usage in real-time applications.
2. **Memory**: Large resource overhead associated with big video memory modules, which is necessary due to acquisition, recording and processing entire video frames in most of the methods. However, human vision focuses only on the object-of-interest and can see the rest in quite low resolution.
3. **Power**: Current approaches use significant power consumption, which can be minimized by using human vision principles. In contrast to existing technologies, human vision is

energy optimized because of usage “attention” concept for video capturing and processing approximately 5-10% of the entire Field-of-View (FOV)

4. **Complexity**: Software and Hardware designs are quite complex due to the utilization of principles and concepts of TV/Cinema video-acquisition and processing (e.g. entire frame-by-frame). The complexity can be optimized if only a small portion of FOV associated with a moving object is captured and processed.

1.2 Objectives

The main objective of the research is to find efficient mechanisms (algorithms and methods) for detecting all moving objects in FOV, selecting object(s) of interest and tracking them in 3D space in real-time, minimizing hardware processing and memory resources.

Aspects for the research, development, implementation and verification involve:

1. Investigate existing techniques of vision systems and analyze the pros and cons of the current approaches and identify main principles of the human vision for further utilization in the system
2. Create algorithms and methods for real-time object selection, location in 3D space and tracking
3. Develop a platform and design a prototype with real-time motion detection and tracking in the 3D space, with minimization of the memory, processing time and computational resources according to the created algorithms and methods
4. Design a flexible and automated testing environment for verifying modules of the architecture and in order to validate visual and 3D coordinate results of tracked objects
5. Verify the test results, determine performance metrics, and identify possible features and limitations for further development.

1.3 Original contributions and methodology

Original contributions of the research involve:

a. Analysis of existing approaches and methods, which includes the research of background information and related works in human vision application and identifying the main principles by observing the visual data processing
b. Creation of the system architecture to solve the problem, which includes

i. Defining efficient algorithms and methods for object detection, location and tracking
ii. Design and implementation of the system, which uses the observed principles from a human vision for optimizing computational resources

c. Development of the experimental set-up, implementation of the developed architecture and performing the set of experiments, which includes a description of the set-up, methodologies how the tests are performed, experimenting with different approaches for achieving acceptable performance and resource constraints
d. Verification of the setup and finding the pros and cons of the proposed method and architecture, which involved analysis of results, and discussion for future development

It should be mentioned that research solves several challenges related to the development of human vision applications:

- The main challenge of the design is performance and resource constraints. Correspondingly, the implementation includes optimization of memory, time and computational resources, which makes it possible to be used in the application where other approaches fail to solve problems in their specifications.
- It should be emphasized that in the design, the custom board with multi-purpose video-sensors, FPGA device and embedded ARM are combined. Separation of the design for the hybrid approach with specific constraints and keeping integrity for verification, identifying the areas which part should be taken in each area are an essential part of the development.
- Another challenging part of the research includes noise filtration from the video frames and the results estimated by 3D pose calculation. Multiple approaches are used and analyzed in the research for achieving the desired results.
- Testing vision application is always challenging as most of the verification procedures include confirmation by humans. The research proposes several layers of verification, description of methods and creation of the test environment.
- As the final design and implementation are considered as a platform for future development, it includes a customization interface for flexible and easy usage in future applications.

1.4 Thesis organization

The rest of the thesis is organized as follows:

In Chapter 2, background information and related works are described which are utilized for identifying the main concepts of the human vision for further implementation and design, as well as, discussing the main limitation and problems of the current approaches. Firstly, background information is discussed, which includes:

1. Human vision and eye anatomy
2. Binocular vision
3. Detection techniques
4. Tracking algorithms
5. Image stitching
6. Noise reduction, smoothing results and fitting curves

Related works section involves the specific implementations of the algorithms which support the research and decisions made for choosing the right methods. In addition, other FPGA based vision systems analyzed for reusing methodology or identifying the limitations.

Chapter 3 covers the methodology, explains the techniques utilized in the design, and proposes the architecture. Main parts included in this chapter involve:

1. Human vision observations used in the architecture
2. Object representation technique
3. Image stitching method
4. Video-sensor operations for 3D processing
5. A solution to the noisy results
6. Architecture for stereo and semi-panoramic vision components
7. Final combined architecture for developed binocular vision system

Chapter 4 describes in details the implementation of the proposed architecture and methodology, and consists of:

1. The experimental setup, which presents utilized technology and sensors, and describes hardware and software specifications
2. Description for each module and process of the architecture, which involve:
    a. Functional specification
    b. Principle of operation, which explains how the modules operate

c. Interface as input and output signals with a specification of the type and mode Symbol as an illustration for the block diagram
d. Process diagram for clarification the operation, where it is needed

Chapter 5 covers the results and evaluation of the research. It involves:

1. Process describing test data and environment
2. Measurements for verification of results
3. Performance metrics and properties of the designed system

Chapter 6 is the last chapter, which summarizes the research, describes the usage of the final design and implementation and proposes future work for additional features and possible solutions for research limitations.

.

Chapter 2

2. Background Information and Related Works

This chapter involves explanation and descriptions of background information and related works of human vision, popular frameworks and algorithms highlighting the achievements and limitations.

2.1 Background Information

2.1.1 Human vision

A human vision involves a complex process of capturing, storing and processing an image. Exploring the anatomy of an eye and configuration of the vision is crucial for the research as the purpose is to mimic the basic principles of the human vision. It should be noted that modern video-sensors resemble the structure of the eye and use the main methods of physics of it. The eye is the main unit tool of human for capturing images. The general structure of it is displayed in Figure 2.1 [1]. The main components involve [2] [3] [4]:

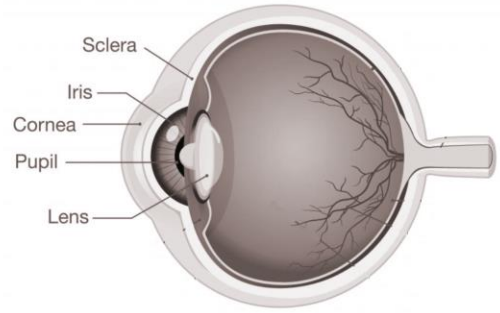


Figure 2.1: Eye anatomy

- The **cornea** is a transparent spherical structure which utilizes projecting entered light energy [2][3].
- The **pupil** which is a central part of the iris and appeared as a black hole because of absorption of light [2] [5].
- The **lens** which is one of the essential parts of the eye and is used for focusing light into the retina [2] [3].

- **Iris** is a circular unit, which controls the bandwidth of incoming light by changing the pupil size [5].
- **Aqueous humor**, which is a provider of important nutrition for pressure maintenance [2].
- **Vitreous humor** is a helper part inside the eyeball which is used as a gel for filling space between retina and lens[2].
- **Rods and cones**, which are photoreceptors for scotopic vision[2].
- **Retina** plays an essential role and is responsible for converting the light into a neural signal form[3].
- The **sclera**, which is another helper for supporting essential protection[6].

Firstly, the incoming light goes inside the cornea. The iris controls the amount of light by altering the pupil. It should be noted that the shape of the lens can be changed for optimizing the power depending on how long distances object is detected and identified [2]. Once the light reaches the end of the first layer of the eye, it is projected on the retina. At the final stage, the retina converts light in neural signal form and sends to the option nerves [7]. It should be emphasized that the image is inverted, and the same geometry and optical image formation is used in digital video-sensors [1] [2] [7].

The process of eye operation was briefly overviewed above. However, it only includes a description of a single eye operation. The importance of analysis of human vision is two eyes operation. The two important components of the operation of binocular vision are stereo vision and peripheral vision, which are described in detail in the next subchapters and are considered as main principles of the research for creating the design and implementation[8][9][10].

The most animals, which have two eyes and the relatively larger intersection of the Field of Views (FOV) from each eye, are considered to have binocular vision. The main advantages of binocular vision are the following [7] [8][9][10]:

1) Being able to detect objects 3D position and most importantly the distance
2) have a wider FOV which helps keep tracking objects in a wider range
3) Can operate if in emergency case one of the eyes doesn’t work or are damaged

Stereo vision

Stereo vision or stereopsis vision is referred to as a system, which is able to estimate the depth and 3D position from the processing of images obtained from 2 eyes.

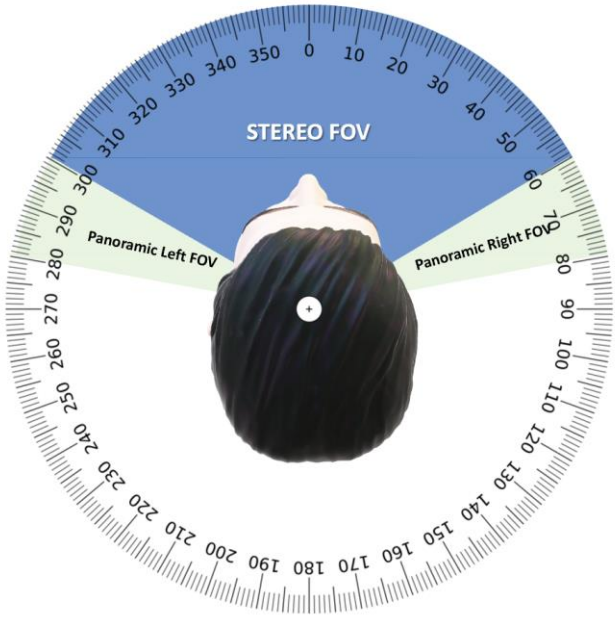


Figure 2.2: Human vision system fields

As it can be seen from Figure 2.2, the stereo horizontal visual field in case of human is ~100-120 degrees.

Once the object is presented in the stereo visual field of view, the distance is retrieved from the triangulation [11].

Before describing a geometrical way for calculating the distance, it's important to define terminology[12][13][14]:

- A **baseline** is called the distance between two eyes.
- An **epipolar plane** is called the plane which goes through the center of projection
- **Conjugate pair** is another term used for point/object which can be seen from both eyes.
- A **disparity** is a difference between two conjugate pairs
- A **disparity map** is a table which maps disparity values to distance values, usually presented as a look-up table or implemented by functions in computer vision applications.

Geometry of triangulation

The geometry of triangulation defines the way to calculate the distance of a point/object seen from two or more video-sensors [15] [16]. As a simple standard case, as shown in Figure 2.3:

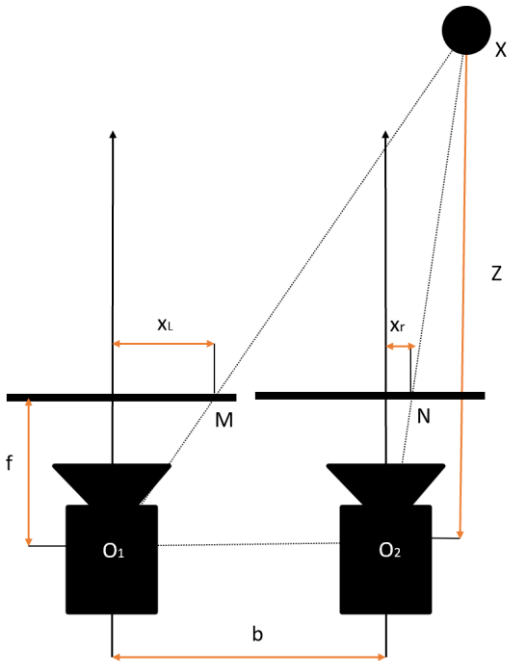


Figure 2.3: Triangulation of video-sensors

Since the triangles M, N, X and O_1, O_2, X are similar, the depth can be determined by the equation, where x_L and x_r are distance from the object on projected plane from left and right video-sensors accordingly, *d* is a disparity, *b* is a baseline value, f is a focal length and *Z* is a real distance [15] [16]:

$$d = |x_L - x_r|$$

$$\frac{d}{b} = \frac{f}{Z}$$

Because the baseline *b* and focal length *f* are known and predetermined values and x_L and x_r known measured values, the only unknown variable is *Z* distance, which can be derived as follows [15]:

$$Z = f\frac{b}{d}$$

It should be emphasized that this is the geometrical way calculation, however as the disparity is inversely proportional to the depth, the predetermined lookup table can be defined for

optimization and better accuracy purposes. The distance determination can be achieved by mapping disparity to distance values as well as taking into account: width, previous distance, error prediction and other parameters.
This method replicates the human binocular vision; However, planes of each video-sensor can have different angles in case of some animals' vision and implementations require more complex calculations, additional verification, rectification, stitching and other processing, which result in higher requirements for resources utilization [13].

It should be emphasized that the stereo vision triangulation approach raises several problems and challenges such as calibration and object/point matching known as the correspondence problem [13]. Since video-sensors are separated in independent units, external check and rectification are required to have the same image results in terms of brightness, saturation and other image properties in order to minimize the error of object matching[17].

The solutions for image matching or correspondence problem vary to the digital implementations and are still in research in human vision. There are various effective solutions. However, the main limitations are a large number of processing resources and complex methods. Traditional implementations vary to the complexity and use different techniques[13] [14][17][18]:

- **Pixel matching** is one of the naive solutions and involves matching the array of pixels by a certain threshold
- **Centroid matching**, which involves detection object/movements, filtering the noise and verifying results by comparing and matching centroids
- **Edge detection** involves pre-processing of the initialized images, identifying edges and matching them for the desired area of an object, the limitations and challenges of this approach include high noise rate which is normally caused from the complicated backgrounds/objects [19]
- **Scale-invariant feature transform** is one of the powerful algorithms which works for different scale, angle images, and even different video-sensor units, however, requires a relatively high amount of time for processing [20].
- **Object classification** is one of the high computational resource utilization algorithms, but it's popular due to high accuracy. It firstly identifies the object, classifies to different shapes and matches them [21].

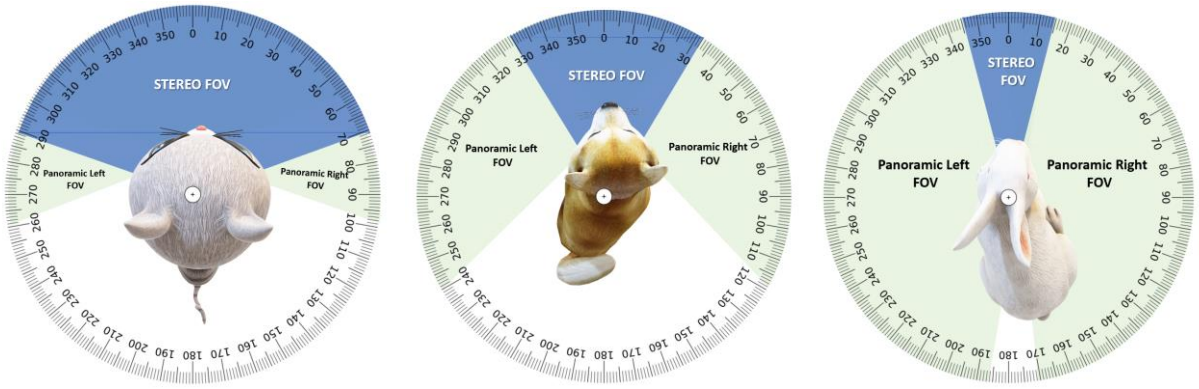
- **Graph cuts** and **Ground truth**, which involve segmentation of images and labeling the pixels [22]

Characteristics of Simple Stereo Vision in Monocular Vision

It should be mentioned that a single eye/video-sensor can be utilized with or without other eyes for distance calculation. If the object is classified to the certain type (for example a human face, a cat, a car and other known objects), the approximate distance can be determined based on the experience of the dimensions of the particular objects [23]. On the other hand, if the object is entered to the stereo vision field area and the distance is already identified for the lazy processing or the case where it can be still observed from non-stereopsis vision area, changing the distance can be determined based on the dimension [24][25].

Peripheral Vision

Another essential approach of binocular vision in addition to the stereo-optical field of view is a peripheral vision. It should be mentioned that the field of view of stereo and peripheral area varies for different animal vision systems [11] [26]. Examples are illustrated in Figure 2.4.



a) Cat Vision b) Dog Vision c) Rabbit Vision

Figure 2.4: Vision system fields in different animals

As can be seen, the whole field of view reaches 140 to 180 degrees for a human vision, while in the case of a rabbit, it is ~340 degrees. However, the rabbit stereo vision field is relatively shorter, and accordingly, the calibration and matching of images involve different processes as well. In illustration 10, other examples and variation of a dog [27] and a cat can be seen [11][28].

The properties are mainly derived from the lifestyle of the animal and according to development from the evolution [6][29][30].

Two modes of operation of peripheral vision and the field are identified:

1. **Static mode** of vision, which is the case when eyes are fixed, and as it was aforementioned, a human can reach ~200 degrees of the whole field of view in this mode.
2. The **dynamic mode** includes moving of the eyes or, more advanced, a head. It should be noted that even eyes can move, it can be adjusted only with the same angle in normal behavior. This property is kept as to avoid more complex calculation and calibrations for processing of images from different eyes. By the movement of the eye, human vision can reach 270 degrees of field of view and 360 degrees by the movement of the head. It should be noted that when the object is presented in the closer distance eyes can be adjusted to different angles, which are considered as adaptation and is described in a later chapter.

In order to achieve the same approach in computer vision, there are possible hardware and software current solutions:

1. Single camera-based solutions involve manufacturing higher (120>) field of view video-sensors [31]. However, this solution isn't cost-efficient in terms of price and power usage and normally have a lower frame rate due to higher processing.
2. Another approach is utilizing multi video-sensors with different angles or rotating the single video-sensor by motors. The main limitation of these approaches is high processing time and resources for rectifying and stitching images as they are captured from different angles.

The role of the nose should be emphasized in calibration. It is used as a trusted object which normally can be seen from both of the eyes, but it is filtered from final result image of the vision/ However, it isn't simply excluded, but it is utilized for matching and merging images for retrieving a single combined image with a wider field of view. That's the main reason that animals which have a bigger nose like eagles, have better calibration and precision for determining targets with higher distances and are more efficiently integrating images from different eyes[29].

Attention

Another important observation from human vision is the property of attention. Attention is a term used for the object/area focused by vision. It should be noted that humans just focus only on a single object/area, and computational resources are reduced for background processing [32][33][34].

The main features and observations from the human vision from the attention involve [33]:

- Always at a time, only one area or object is prioritized by the human vision; it is a subpart of the whole image and has a maximum and minimum range of dimensions.
- The attention area is always seen and captured in higher resolution while the background area is in lower resolution. The main motivation of this approach is minimizing resources and getting higher accuracy for 3D estimation.
- It should be emphasized that the background area is still presented with lower resolution and isn't excluded. The motivation comes from processing the image for changing the area of attention in the future if another object is prioritized.
- The attention can be defined by different criteria of priorities, mostly it takes the higher speed of movement but can be derived by shape, distance and other factors.

The information and observations presented in this section are explained in more details in the implementation and design section and described how they are integrated into the research development.

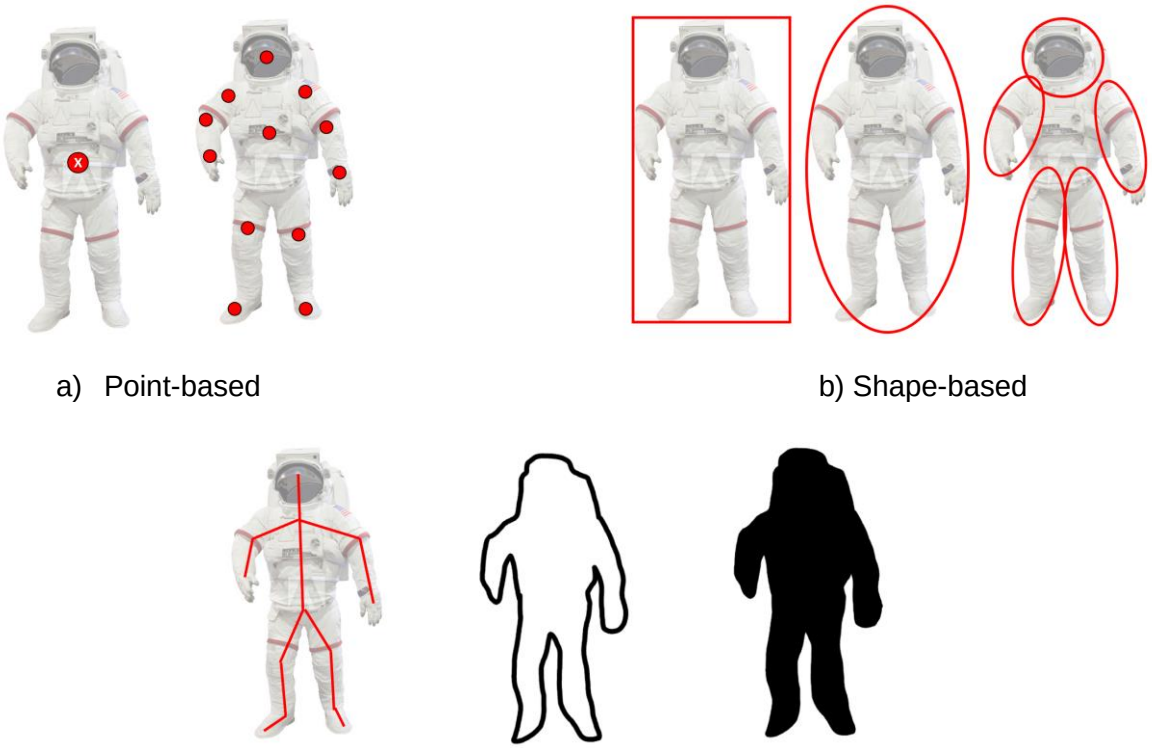
2.1.2 Detection and Tracking

Object detection and tracking are essential parts of vision applications. This section overviews the different types of detection and tracking algorithms and discusses the usability of the research purposes in human vision.

It should be noted that some application separates the process of object detection and tracking since the detection a specific object (for example, face) can take a long time, but tracking object with already known properties can take relatively fewer steps.

Since the tracking involves the process of determining the changed position through the time from the previous instances of the object, algorithms depend on the representation of the object

by detection. There are numerous ways of representation which are illustrated in Figure 2.5.[35]



a) Point-based b) Shape-based

c) Model-based d) Contour-based e) Cluster-based

Figure 2.5: Object detection representations

The main approaches include:

1. The **point-based** approach involves identifying the single or multiple points which can be utilized in tracking or further processing. The point should be defined depending on the importance. The simplest way is a centroid of the object. More complex methods are multiple object-specific points, for example, landmarks of a face or a body of a human. The advantage of the approach is the usability in tracking since tracking can save time by avoiding whole image processing and estimate result by identifying transformations of points[35].
2. The **shape-based** approach involves fitting the predefined shape as single or multiple rectangles or circles to the desired object. This approach is widely used in initial stages of various detection and tracking algorithms as firstly it identifies the area where an object is presented and skipping rest information and avoids processing [35].

3. **Model-based** is a complex approach which is a mixture of detecting points and connections between them. This approach has higher precision and is flexible for further application processing since it already involves extracted features from the object; however, it requires higher computational resources than previous techniques[36].
4. **Contour-based** detection is used for tracking objects by external edges. Once the object is identified, the tracker searches the same contours with acceptable modification of the object and changing the position according to it [37].
5. **Cluster-based** is a popular approach for motion tracking, the performance varies depending on the application and cluster definitions, however in motion analysis where the silhouette can be easily defined this approach is useful [35] [38].

It should be mentioned that these approaches don't exclude each other and can be combined for achieving desired results.

The types of detection vary as well depending on the desired shape of the object instance [35]:

Point detectors are class of detectors which searches specific points with the same feature for the desired object. This class is suitable for images which have high resolution and texture since the normal features of points are limited and need to be expressed accurately enough to be detectable with this method. There are various algorithms which specialize in the detection of a certain feature. As an example, Harris Corner Detector is a well-known approach which detects corners in the image which can be presented as points and tracked for further processing. A more complex algorithm is Scale-invariant feature transform (SIFT) which involves detecting general key points for features which are most likely preserved in the next frame. This approach has a high accuracy even the images are captured from different video-sensors; however, it involves lots of steps of processing which aren't suitable for application which requires shorter cycle time of processing [35].

Segment detector is a class of detectors which firstly segments images with certain criteria, identifies clusters, and use the previously localized segments for tracking. One of the most popular approaches is mean Mean-shift algorithm, which involves defining a kernel function and iterate it through the images to the highest density while convergence occurs. The definition of means relies on the description of the kernel function, which can be varied to the applications.

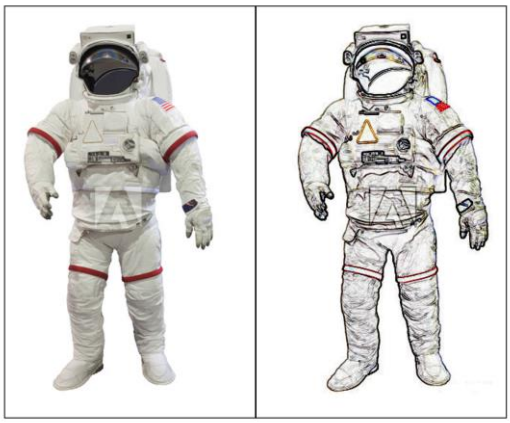
The advantage of usage means shifts for clustering is independent of the image values and shapes [35].

The **Classifying** method includes training the classifier based on the features of the object. The process requires having a ready database or dynamically improve the algorithm by giving the indication of success or failure to the detector. The widely used classifiers are regression models, Support Vector Machine (SVM), neural networks. The algorithms are mostly used for complex applications such as face detectors, shape detectors, and others and require lots of resources for storing data and computations as well are complex in terms of implementations [35][39].

Once points or areas are presented by detection it's important to define which features of the group of pixels are utilized for tracking. There are various features which can be combined in these classes [35]:

1. **Color Values**
   This color value is the basic features of the pixels and is suitable for algorithms where objects are represented in point instances of data and depend on the color value of the pixels. While there are two traditional approaches for representing the color as HSV (Hue, Saturation, Value) and RGB (Red, Green, Blue), there isn't a clear advantage each of it for general usage of tracking and depends on the application. This approach is based on the assumptions that the colors of the objects are preserved while frames are changing. However, the main limitations are considered as a lack of information on features such as shape, texture and etc.
2. **Contours**
   Contour or edge-based approach is one of the widely used features. The idea is that edges of the object are preserved or slightly change during tracking and even preserved by changing the illumination. The most popular detector is a Canny edge detector which identifies the contours and edges of in the image. It also involves smoothing the image and noise reduction with a Gaussian filter [40]. The main steps of the algorithm include gradient calculation and applying a threshold to identify potential edges. Another widely used approach is Sobel operation, which involves detecting edges by applying convolution function kernels[41]. The example of the filtered image is presented in Figure 2.6.

$$Sobel\ filer = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix}$$



a) Original image b) Sobel filtered image

Figure 2.6: Sobel filter on the example image

3. **Brightness constancy**

   Brightness is another widely used feature, the corresponding algorithms based on the idea that the brightness of the object is preserved. One of the well-known algorithms is Optical Flow (OF), which determines the distribution of movement retrieved from the brightness [42]. While there are different implementations of OF, one of the most popular is Fleet and Weiss approach which proposes calculation of gradients depending on the velocities of the brightness. This feature and related algorithms are suitable for motion-based tracking applications [42].

4. **Patterns and Distributions**

   Patterns and distribution from images as textures are another feature which is used for tracking objects. This approach can be considered as an extension of color-based approach, however, considering mainly the color value. The process involves identifying the connection between the values of the ordered color values. The implementation for detection and tracking uses histograms calculation. One of the simple examples is Co-occurrence matrix, which is a statistical technique for finding the density of pixel values for detecting occurrences within the acceptable distance of different (following) images [43].

Tracking algorithms involve defining the trajectory of the objects and varies to the representation of objects by the detection in the form shown in Figure 2.6 and using the described above [35].

They can be classified into three main groups [35]:

Point-based tracking

These class of tracking algorithms depend on the point representation of the instances and taking into consideration the absolute color value or brightness of the pixel [44]. The tracking requires object detection for each frame to retrieve the same processing of detection to map the identified points. The well-known approaches are Kalman Filter, MGE tracker by Salari and Sethi and others.

Pre-defined Shape-based tracking

Shape-based tracking involves having a predefined template as a shape which is searched in the next frame with the same properties. The feature properties can be a combination of textures, edges, color values or other additional characteristics of the model. The shapes are normally rectangle or eclipse. This class is suitable for motion tracking as the motion can be surrounded with the desired shape and can be simply mapped to the next image with just position modification or rotational and affine transformations. Well-known algorithms are Mean-shift, which was discussed above, KLT (Kanade–Lucas–Tomasi feature tracker) which includes finding a disparity vector which minimizes the difference between function-specific measurements of certain areas from different images [44] [45].

Model-based tracking

A model-based approach is one of the complex techniques which traditionally based on the representation of objects by contours, multiple points, textures or another form [36]. The idea is to integrate features inside the tracking for higher precision. This naive method usually involves pre-processing such as brute-forcing the whole images for fitting gradients to the certain model or shape, which is one of the most expensive operations in terms of image processing time and computational resources. The possible implementations are Hausdorff algorithm which is based on the calculations of distances of two subsets in a defined metric space [46]; State-space models by Isard and Blake which tracks the contour by maximization of a posteriori probability [35].

2.1.3 Image stitching

Image stitching is a process of merging images which are sharing the same parts as overlaps. It is widely used in the medical area, virtual game industry, photography and other vision applications [47].

In the general approach, images can be captured from a different location, a different angle, with different illumination, resolution or even from different video-sensors.

While the algorithms vary, the standard process of image stitching contains the following milestones [48] [49] [50]:

1) Keypoint detection, which involves identifying the features from the images for the verification and detection of overlaps
2) Matching images, the process where features are extracted by keypoint detection, compared, and intersections are recognized.
3) Blending is a process of mapping images to the final combined image with a chosen projection layout. This step also involves adjusting images and calibration for having the same appearance.

Keypoint detection

Detecting keypoints or features from the images is the initial step of image stitching. It should be noted that the key points should be extracted the way to be invariant of possible transformations for the application [48]. As an example, if the images are rotated the key points should be rotation invariant. The features or key points can be a single point, lines, contour, area from the image or in any other form. The simple approach is corner detection, which was widely used in the past, however, nowadays most popular and widely verified approach are Scale-Invariant Feature Transform (SIFT) and Speeded Up Robust Features (SURF). The popularity of these approaches is based on the idea that they are scale, rotate, translation, illumination, and blur invariant. It should be mentioned that they are also widely used in modern 3D reconstruction, object recognition and other machine vision algorithms [51][52].

SIFT

SIFT-based keypoint localization is based on a set of SIFT defined specific features [53]. The Operation of the algorithm is extracting local features from the images based on the intersect points appearance. It should be noted that the process involves reducing errors in illumination change, filtering noise, and keep invariant with a particular level of the change of viewpoint. Another advantage is that the process minimizes the probability of mismatch and false-positive results [51] [53][54].

The principle of operation of SIFT involves the following steps[53] [54]:

Gaussian filter and Scale-space extrema

Firstly, the Gaussian filter is applied to the image using convolution. Secondly, the difference between two scales padded by a constant factor is computed[55], as shown in the following equation [51]:

$$D(x,y,\sigma) = \big(G(x,y,k\sigma) - G(x,y,\sigma)\big) * I(x,y) = L(x,y,k\sigma) - L(x,y,\sigma)$$

The pyramid is created for each octave by finding the difference of Gaussian, as shown in Figure 2.7 [56].

This purpose of this procedure is to identify the orientation and scale-invariant points. Finally, after differential calculation, local extremes are searched and considered as a potential feature for further processing [53][57].

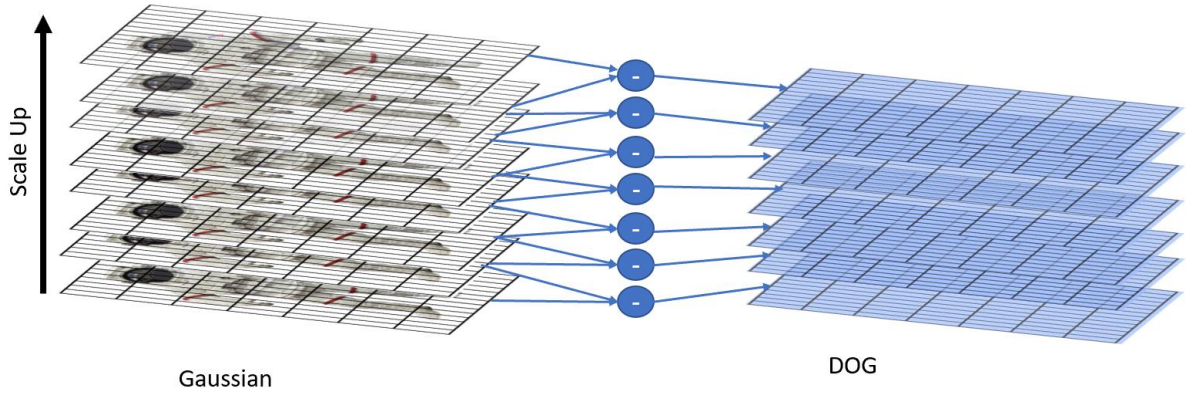


Figure 2.7: The pyramid of DOG

Key Point Localization

Once potential keypoints are identified, they are filtered for verification purposes. For stability, several processing steps are used as threshold filtering, removing edges due to higher sensitivity from Gaussian differences and etc [53] [56].

Key Point Descriptor

The area of localized keypoints (ex: 16x16) are separated into internal blocks and gradients are created, which results in vector as descriptors for each keypoint[53] .

The vector of invariant keypoint is retrieved from the final step and processed in the next steps of the image stitching algorithm [53][58].

SURF

While SURF is a similar approach to SIFT, it requires less time for finding features. The principle of operation involves[51] [59]:

Interest Point Detection

This step involves conversion image to integral form. The integral form represents the sum values of the intensities from the images with the operation defined as follows:

$$I_{\Sigma}(X) = \sum_{i=0}^{i \le x} \sum_{j=0}^{j \le y} I(i,j)$$

The integral form of an image is convoluted with a box filter, which itself consists of Laplacian of Gaussian, which is the second-order derivative of Gaussian. In the end, the points are similarly sampled but upward direction and identified with non-maxima suppression [51] [59].

Interest Point Description

In this step, initially, Haar wavelet filters as matrix M_1 and M_2 are applied to obtain the responses in a vertical and horizontal direction [60].

$$M_1 = \begin{bmatrix} +1 & -1 \end{bmatrix} \qquad M_2 = \begin{bmatrix} +1 \\ -1 \end{bmatrix}$$

The responses are normalized and converted to unit vectors in order to keep invariant to contrast and illumination. The circle window with radius *π/3* is slid through the result and dominant orientation is derived from the wavelet responses weighted by the Gaussian filter[51][59].

To sum up, both approaches are effective for feature extraction. While they have some advantages of invariance of certain properties of an image, both are acceptably invariant to noise, scale, rotation and contrast. However, the limitation of using these approaches involves higher processing time and computational resources if the full set of features are utilized. Moreover, the main limitation is considered in industrial applications that they are restricted due to patent [51] [52] [59].

Features matching

After detecting features from both images, the next step to match them to identify overlapped part and map objects to each other. The naive approach is to try every possible value as brute-forcing by matching and choose the result which minimizes the total error. This approach always guarantees the most accurate result; however, it's the time consuming and doesn't work with typical vision constraints if certain parts aren't optimized by some indications. The most popular approach is Random sample consensus (RANSAC), which is the non-deterministic iterative method which can be used for bijection that maps set of features to another set of the image data [61]. On each iteration, it tries random approaches by choosing a set of features to be matched, verifying them, and calculates the sum of error, which is corrected every next run. The accuracy of the result is proportional to the number of iterations [48][54].

Blending Images

Blending is a process of adjusting images together in order to minimize the intensity difference between intersected areas and calibrate the image to have the same appearance as a combined merged result. The possible differences between images can be caused due to[54]:

- Radial distortion
- Viewpoint differences of video-sensors
- Light change due to position differences of video-sensors
- Errors as mismatching features
  and other possible errors

The final image results are mapped to the projection layer, which can have different forms. The most popular traditional projections for panoramic images are [62]:

Cylindrical projection, which maps 3D stitched images space to 2D space by stretching original horizontal lines and directly associating vertical lines. The analogy can be seen as covering the object with a cylinder and obtain the result which is projected to the cylinder surface. As a result, horizontal lines appear stretched while the vertical view is preserved. The example is illustrated in Figure 2.8. Equirectangular projection can be considered as a subcase of cylindrical projection where 360° horizontal by 180° vertical field of view is presented. Similarly, horizontal lines appear curved while vertical lines remain straight [62][63].

Figure 2.8: Cylindrical Projection

Hammer projection is an equal-area projection, which stretches stitched image horizontally. The final image has distortion on the outer part of the image, but the central area is preserved[64]. The example is displayed in Figure 2.9.

Figure 2.9: Hammer Projection

Stereographic projection is another mapping technique, where the points are mapped in the relation of the view from a certain point and give a spherical view on the 2D plane [64]. The example is illustrated in Figure 2.10.

Figure 2.10: Stereographic Projection

It should be noted that while above algorithms and steps of processing have general-purpose usage, human vision involves certain advantages which can be utilized for choosing the algorithm not to overcomplicate the problem and save computational resources.
The main features and observations from the human vision, which are taken into consideration while mimicking it as the digital system involve[28]:

1. In human vision, eyes as video-sensors have same base properties as dimensions, resolution and other parameters. However, captured images can still be different because of position differences and possible errors and still need calibration and noise filtration processes.
2. Video-sensors are presented on the same linear baseline with the same angle with a known distance from each other.
3. As it was mentioned before, the nose plays an important role in human vision. It can be seen from both of eyes but isn't presented in the final stitched image. It is used as a well-known object with known distance, shape and used as an essential key point of feature for merging images.

These properties of human vision give advantages which can be used in implementation and design to save computational resources as follows:

1. Video-sensors in the implementation can be presented the same way with the same linear baseline in order to save time for searching viewpoint sensitive invariant features.
2. Since the video-sensors are the same and frames have the same property and aren't rotated the images which should be matched and have overlaps in pre-defined range, it isn't required to search keypoint invariant to rotation, scale and other features.

2.1.4 Noise reduction, Smoothing and Fitting curves

Noise reduction is one of the essential parts in the digital image processing applications since noise is typically presented on digital images due to capturing errors, transmission, possible memory corruptions or other reasons. Moreover, it plays another important role in smoothing the results derived from image processing for higher accuracy and having a better experience from the user side[65] [66].

Digital Image noise filtration

Filtration is a traditional method which is widely used in computer vision for noise reduction. The idea is to filter the errors without affecting the essential information contained inside the image. The filters vary to the applications, implementation and assumptions [66].
The basic approach is convolutional filters, which assumes that the noise is presented to the individual pixels and the pixels have a common relation to the neighborhood values.
The naive implementation is defining the average matrix which is known as Mean Filter (MF) and applies convolution through the 2D image. The 2x2 form of the matrix is described as following:

$$MF = \begin{bmatrix} \frac{1}{4} & \frac{1}{4} \\ \\ \frac{1}{4} & \frac{1}{4} \end{bmatrix}$$

This approach successfully fixes the basic noise however it loses too much information and strongly depends on the neighborhood values. The better idea is to have weight values of the neighbors, as the farther neighbors are less likely to affect the pixel than the closest one. The

most popular approach of this class of filters is the Gaussian filter[40]. The Gaussian equation describes the probability density function of a normalized distribution:

$$p_G(z) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(z-\mu)^2}{2\sigma^2}}$$:

the 5x5 matrix form representation is as follows:

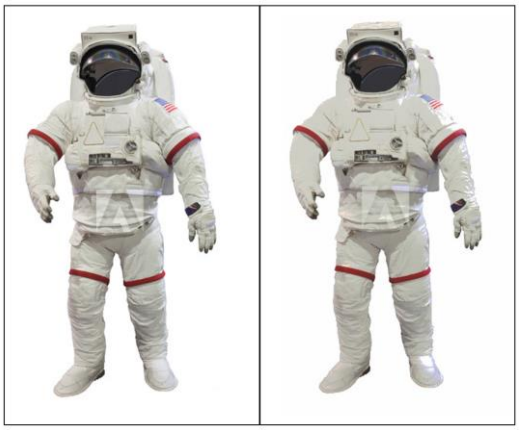
$$GM = \frac{1}{273}\begin{bmatrix} 1 & 4 & 7 & 4 & 1 \\ 4 & 16 & 26 & 16 & 4 \\ 7 & 26 & 41 & 26 & 7 \\ 4 & 16 & 26 & 16 & 4 \\ 1 & 4 & 7 & 4 & 1 \end{bmatrix}$$

Another approach is a median filter a non-linear filter which replaces each pixel value by a median of the neighborhood pixel values. This approach is widely used in the applications where images can be corrupted by the defect and or canceling the pixel values. Another advantage of the algorithm is that it preserves the edges[67]. The filter example is illustrated in Figure 2.11.

a) Original image b) Median filtered image

Figure 2.11: Median filter on the image example

Another non-linear, edge-preserving filter is a bilateral filter which resembles a Gaussian filter and utilizes it by computation weighted average of the intensity values and setting to the center pixel [40] [67]. The filtered image can be seen in Figure 2.12.



a) Original image b) Bilateral filtered image

Figure 2.12: Bilateral filter on the image example

It should be mentioned that there are other approaches which utilize more complicated algorithms such as deep learning, classification the regions and etc. which remove noise for not only single-pixel, but affected the region of the image, however, computation resources of such methods are quite large, and implementation involve complex steps which aren't suitable for research requirements [66].

Fitting the curve

The image is not only the data which needs noise reduction in machine vision applications. The results which are derived from the image processing such as positions, angles or other values, also need to be smoothed due to stability as well as better user experience. It should be noted that data is presented as a digital signal or an array of values and don't require as much time constraints as an image since the single value is retrieved per frame instead of per pixel.

The basic approaches involve the same technique as was described before for noise filtration for images such as an average of weighted average with gain values. They give a reasonable result for smoothed data however the precision is based on the length of the filer, which is proportional to the latency value, they aren't suitable due to longer response time in real-time application[68].

Another naive approach is threshold filtering. The idea is to define a certain threshold and filter values if the difference is high enough. In addition, it counts the number of continuously filtered pixels, and if the count reaches more than another threshold value should be changed the position. This technique is illustrated in Figure 2.13, where blue points are original data and read

points represent smoothed results, the period of the frame measurement is taken 33.3 milliseconds (ms) as for standard 30 frames per second (fps) video sensor operation.

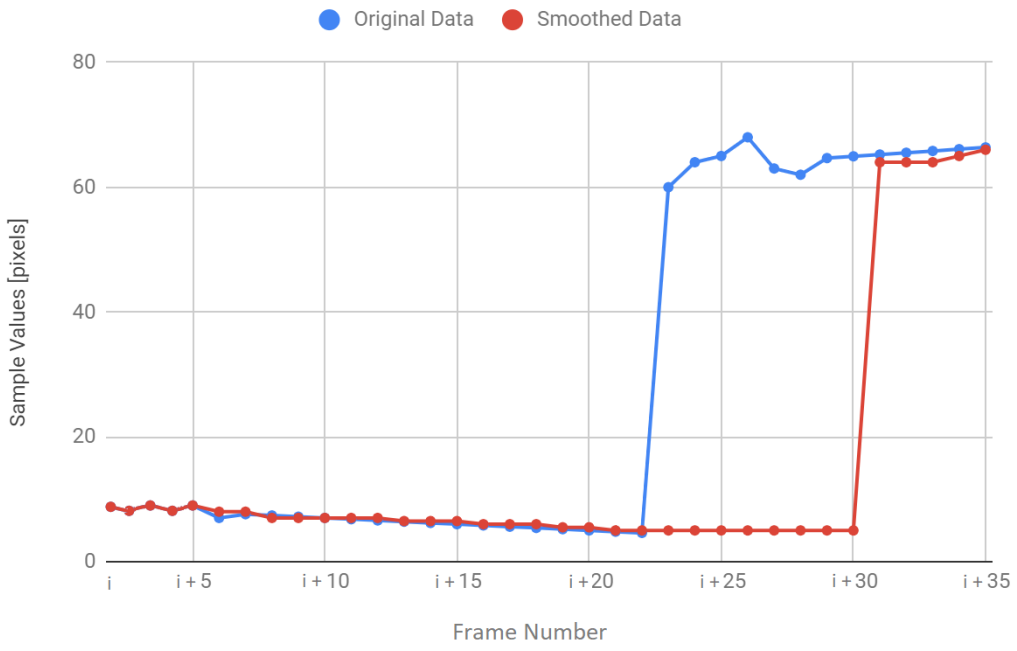


Figure 2.13: Threshold filtering

The main limitation is the long latency as it can be seen from the example. Moreover, it doesn't exclude small jittering, which is below the threshold.

One of the best practices and techniques is fitting the curve, which is a process of finding a mathematical function from the data which minimizes the distance between the measured points. This approach is widely used and gives higher accuracy results. Mathematical functions can vary; thus, the most popular approach is polynomial functions. The idea is to define a polynomial with a previously determined factor and change coefficients based on the income data[69]. It's important to define proper power value of polynomial in order to avoid overfitting which example is shown in Figure 2.14, where red points are real data, the green curve is an optimal smoothing result, and the blue curve represents the overfitted curve [70]. Similarly, to the previous graph, the period of the frame measurement is 33.3 ms.

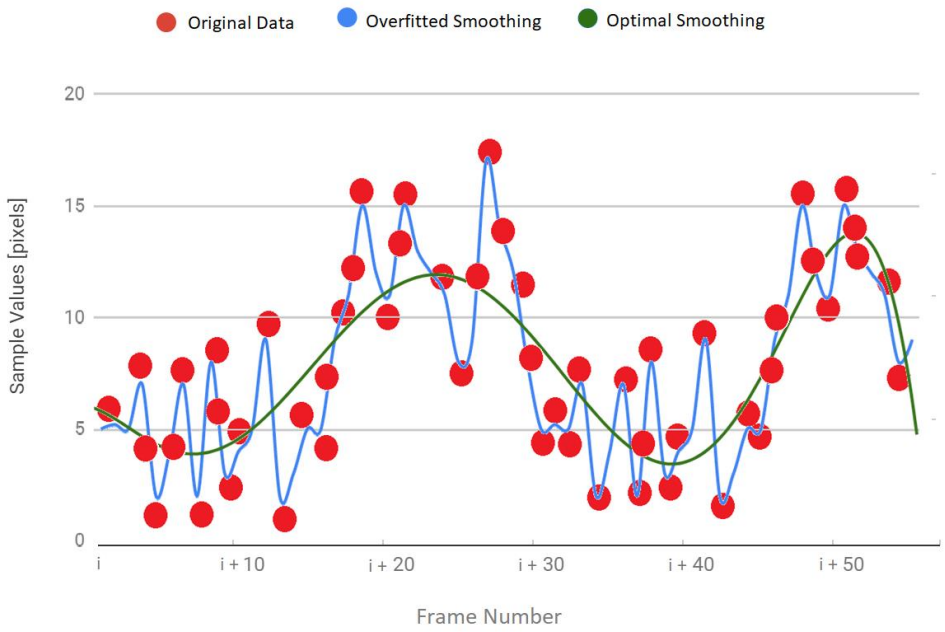


Figure 2.14: Overfitting example

Kalman Filter

One of the popular solutions from best practices is the Kalman filter, which is a linear quadratic estimation. It is widely used in real-time tracking application since have a practical structure and is optimal in calculation with smaller latency and cycle time. It operates by minimizing the mean square error of the estimated parameters[71]–[73].

The principles of operation of Kalman filtering is shown in Figure 2.15:

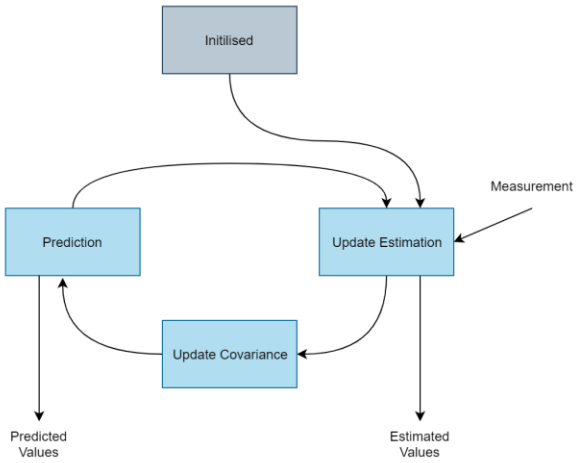


Figure 2.15: Principles of operation of KF

The input of the Kalman filter is measurement values in addition to a noise matrix which can be defined statically as an initial step or changed in runtime. The inputs are presented as:
Process noise matrix, which describes the uncertainty about the process of the variable.
Measurement noise matrix, which describes how much error measurement can have.
The matrix values depend on the model and values being measured and smoothed.
As an output, the filter receives a smoothed estimated value with a covariance matrix which describes the confidence level of estimation of the smoothed parameter [71]–[73].

The definition of each parameter is described below in Table 2.1 [71]–[73]:

Table 2.1: Definition of Kalman Filter Parameters

$\mathbf{F}_k$	State model matrix
$\mathbf{H}_k$	observation model
$\mathbf{Q}_k$	Process Noise matrix
$\mathbf{R}_k$	Measurement noise matrix
$\mathbf{B}_k$	Control-input model
$\mathbf{P}_k$	Post error covariance matrix

The process changes time to time and defines parameters by the assumption that it can be derived from the previous state as following [71]–[73]:

$$x_k = Fx_{k-1} + Bu_{k-1} + w_{k-1}$$

The prediction step involves the update of the state and error covariance:

$$\hat{\mathbf{x}}_{k|k-1} = \mathbf{F}_k\hat{\mathbf{x}}_{k-1|k-1} + \mathbf{B}_k\mathbf{u}_k$$

$$\mathbf{P}_{k|k-1} = \mathbf{F}_k\mathbf{P}_{k-1|k-1}\mathbf{F}_k^{\mathsf{T}} + \mathbf{Q}_k$$

After the prediction, measurement, Kalman gain, and error covariance are updated, and the state is estimated as follows [71]–[73]:

Measurement residual:

$$\tilde{\mathbf{y}}_{k|k} = \mathbf{z}_k - \mathbf{H}_k \hat{\mathbf{x}}_{k|k}$$

Gain update:

$$\mathbf{K}_k = \mathbf{P}_{k|k-1} \mathbf{H}_k^{\mathsf{T}} \mathbf{S}_k^{-1}$$

Covariance update:

$$\mathbf{P}_{k|k} = (\mathbf{I} - \mathbf{K}_k \mathbf{H}_k) \, \mathbf{P}_{k|k-1}$$

(Post) State re-estimation:

$$\hat{\mathbf{x}}_{k|k} = \hat{\mathbf{x}}_{k|k-1} + \mathbf{K}_k \tilde{\mathbf{y}}_k$$

The process is repeated to the same cycle with prediction and estimation.

It should be noted that the Kalman filter is the optimal linear filter and is based on the assumptions that

1) The model should be defined precisely which replicates the real system process
2) The noise which is reduced is independent, and the process and measurement noise are at least roughly known.

Which means that the algorithm accuracy depends on how the parameters and model are defined [71]–[73].

2.2 Related Works

Various vision system frameworks and designs are developed in recent years. In this section, some of the implementations of the main components of human vision are described, overviewed, and highlighted advantages and disadvantages.

FPGA-based Panoramic Vision works

The first research “High-Speed Simultaneous Image Distortion Correction Transformations for a Multicamera Cylindrical Panorama Real-time Video System Using FPGA” by Yuan Xu and Qinghai Zhou proposes a simultaneous technique of barrel correction, perspective transform and projection to the cylindrical space using multiple video-sensors [74]. The experimental setup is developed on the Spratan6 FPGA utilizing DDR3 memory. It proposes architecture with minimized latency and high performance. The setup involves multi-camera distributed as a pentagon with the additional video-sensor on top, which is illustrated in Figure 2.16 [74].

Figure 2.16: Proposed multi-video-sensor setup of the related research

Since the video-sensors are presented in different angles and don't share the same baseline, it requires more complex transformations of points for matching the intersected region. The main milestone of the algorithm involves:

1. Calibration and adjustment video-sensor parameters
2. Transformation coordination which involves Barrel correction and cylindrical reprojection of each image
3. Identification of the overlapped area
4. Blending the final image

While the algorithm process consists of typical image stitching steps, additionally it involves specific approach for coordination of transformations and the setup, which speed up the approach by statically setting the cameras with predefined distances and angles, which are symmetric for each lower camera as well. It should be mentioned that implementation also utilizes software part in SoPC as well, which involves:

1. Sensor configuration
2. Perspective transform matrix initialization
3. Image adjustments
4. Display setup
5. User interface

The hybrid design is useful for implementing the parts which don't require saving computational resources for each cycle of operation and debugging/testing purposes [74].

Another application which worth mentioning is "Image Blending in a High Frame Rate FPGA-based Multi-Camera System" by Popovic and Seyid[75]. The research proposes the blending technique using FPGA. The main technique of operation is based on Gaussian blending

methods. The test setup includes a large number of cameras, as shown in Figure 2.17: 104 and 15 video-sensor are utilized in the setup[75].

Figure 2.17: Setup of cameras in related research

Orthographic plane projection through the center of the spherical distribution of multicamera setup is used. The implementation involves two types of Gaussian blending: a standard one and restricted Gaussian technique, which restricts blending image to the region where pixels aren't close enough to the center of the field of view. The implementation is done by using XILINX Virtex 5 FPGA with the designed board for setting up video-sensors because of the limitation of input-output pins of the development board. The high-level architecture is presented in Figure 2.18 [75].

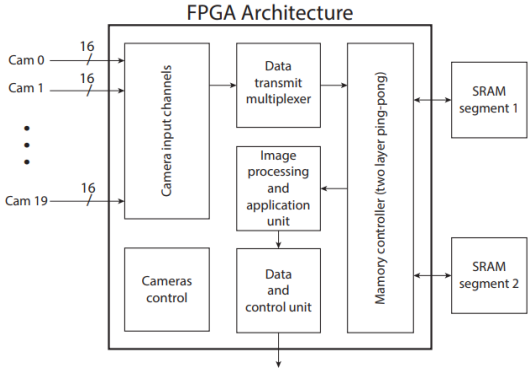


Figure 2.18: FPGA Architecture of the related research

As it can be observed it utilizes two external SRAM memory, which stores an array of twenty following frames.

FPGA-based Stereo Vision works

The previously described researches proposed only a stitching component for a panoramic field of the view of human vision. It's important to review the stereo vision component from binocular vision.

The research "Hardware implementation of an SAD based stereo vision algorithm" by Ambrosh and Humenberger proposes the hardware design and implementation of the base algorithm for stereo vision [76]. The proposed implementation covers only a core algorithm based on the area matching instead of features. It defines the range of disparity from 0 to 100 and for all possible values calculates the matching score in parallel and choosing the best score value. The algorithm is applied for each line of the image frame. The implementation depends on the assumption that the maximum disparity is 100 and it doesn't involve any position estimation, triangulation or object detection from the images. As a result, it maps each line of the original image to resulted disparity map. The design is developed on the Altera Quartus II FPGA and the architecture is displayed in Figure 2.19 [76].

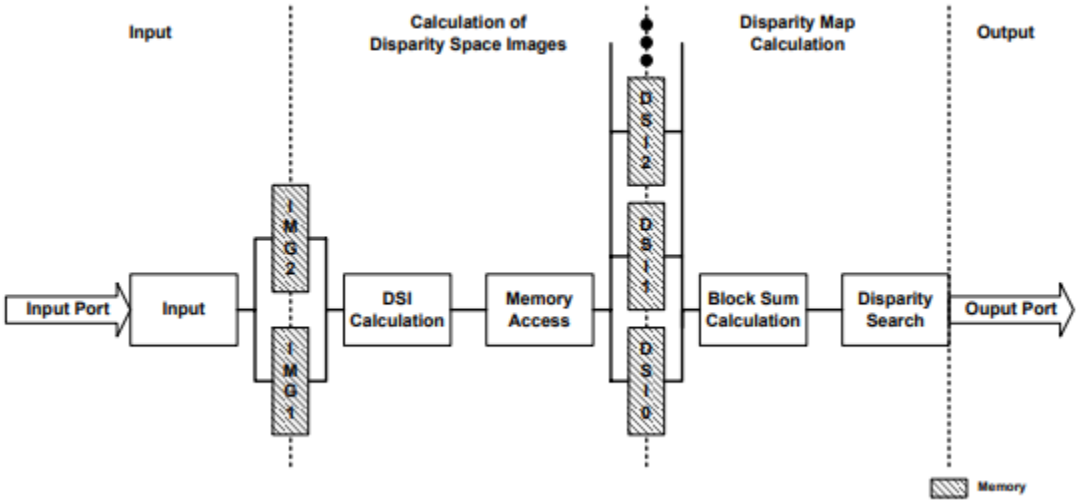


Figure 2.19: FPGA High-level design of the Stereo Vision related research

As a result, the algorithm doesn't require lots of computation, the total time takes only 2.4 ms and can operate 425 fps, however, it's just a simple core part of stereo vision and doesn't fully cover the all main components [76].

Another design and implementation on FPGA are “FPGA Design and Implementation of a Real-Time Stereo Vision System“ by Jin and Cho which propose a fully pipelined system for real-time application[77]. It is based on the utilization of the density of disparity and census transformation. The architecture involves rectification module, Hamming distance measurement, tournament selection, subpixel estimation, and error detection, which results calculated disparity image. The implementation is developed on Virtex-4 FPGA using two VSS-8350CL video-sensors. The advantage of the approach is based on pixel processing rather than sharing the same results for each line, which was described in the previous research. However, it uses and window side for each region where the disparity is estimated. Other advantages of the design are pipelining technique and modularity of the architecture, which makes the design able to operate faster and be flexible for improvement and further modifications. The research focuses only to disparity value calculation and doesn’t involve 3D trajectory estimation or object detection. [77]

More complete research example which involves most of the parts of stereo vision is “Real-Time Stereo Vision System using Semi-Global Matching Disparity Estimation: Architecture and FPGA-Implementation” by Banz and Hesselbarth, which proposes stereo vision system [78]. It covers the process of minimizing the noise of images, disparity calculation with rectification, and visualization with rendering module. The stereo vision implementation is based on the epipolar geometrical setup of two video-sensors. The hardware architecture utilizes MI-bus which connects noise filters, memory units, control of the video-sensor, disparity estimation and rendering, and is illustrated in Figure 2.20. The median filter with a 3x3 matrix is used for post-processing step for noise reduction. [78]

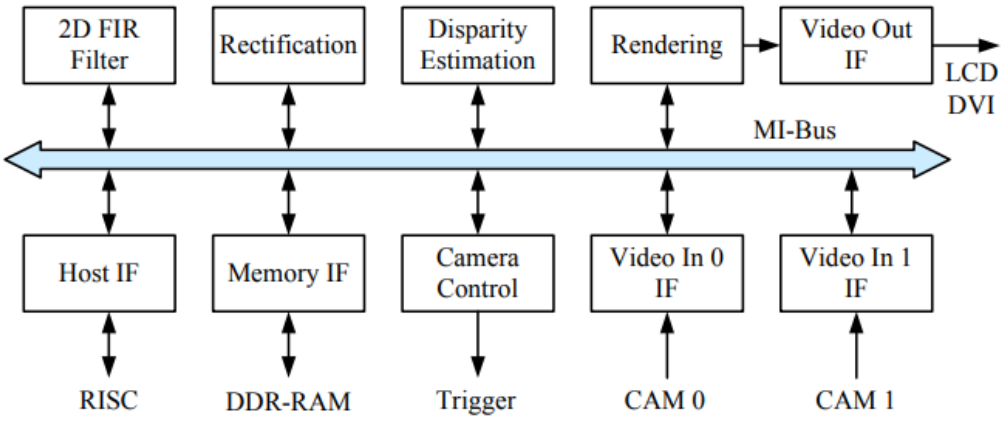


Figure 2.20: Hardware Architecture of the Stereo Vision related research

The stereo vision is based on feature matching key algorithm and is done by parallelizing semi-global matching with a combination of rank transforms with a 9x9 matrix in the pre-processing step. The design uses Virtex 5 FPGA and outputs result through VGA module. It should be mentioned that this approach is scalable and adjustable by the requirement of the latency and processing time [78].

The more complex approach which uses the extended concept of a reconfigurable computer instead of single FPGA is “Real-Time Stereo Vision on the PARTS Reconfigurable Computer” by Woodfill and Herzen [79]. The main advantage of the implementation is the Programmable and Reconfigurable Tool Set (PARTS) engine, which increases the efficiency of real-time video applications. The design covers the implementation of the census stereo disparity algorithm which maps two images to the dense depth map image pixel by pixel. Despite the powerful engine, the algorithm only specializes in a subpart of stereo vision and proposes implementation of disparity calculation with only acceptable shift range of the matrix. The matrix is given to 5x5 and is scanned by the neighborhood around a certain pixel, and the best score results are obtained. The dataflow of the algorithm is represented in Figure 2.21 for the PARTS engine. [79]

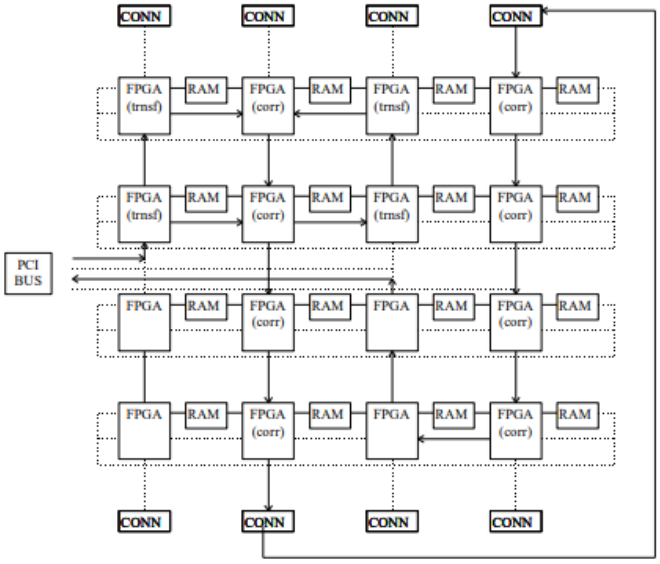


Figure 2.21: The dataflow of the algorithm for the PARTS engine

The result implementation reaches 42 to 225 frames per second depending to the image resolution. It should be mentioned that the approach is overcomplicated and doesn’t utilize all

the advantages of the described engine, thus it doesn't show the main necessity for the implementation and a clear dominance to the single FPGA [79].

Machine Vision Software Platforms

Besides the research and designs on the reconfigurable platform, it's important to review software analogies and libraries.

One of the popular machine vision toolkits is DLIB, which provides C++ based implementation of various machine vision problems [80]. The toolkit contains portable code and can be run on different OS including Windows, Linux, Mac OS and any POSIX system.

It provides a wide range of domains of the algorithms for machine vision, including:

- Image Processing domain which integrates the implementation of feature extraction with SURF, HOG, FHOG, general detection, frontal face detection, pose estimation, face recognition and other complex algorithms.
- Machine Learning domain which involves implementation of classification and regression problem solving with Support Vector Machines, Deep Learning, Multilayer Perceptron, Clustering algorithms such as Whispers, Newman and other linear or kernel k-means clustering
- Numerical algorithms module which involves fast operational matrix implementation, non-linear optimization algorithms, min cut/max flow problems solutions.

It also includes additional modules such as threading, networking, GUI and other utility parts [80].

Another widely known software implementation is an Open source computer vision (OpenCV) library, which is widely used in industrial and research purposes to solve real-world vision problems [81]. The initial idea of creating the library was to develop open source vision infrastructure with optimized code. It covers the vision areas with the implementation of deep learning frameworks such as TensorFlow, Caffe and etc. As well includes gesture recognition, object identification, motion tracking, augmented reality algorithms and others. It is commonly used as another alternative to Dlib. The newer version also includes GPU optimizations with OpenCL and Cuda based interface [81].

2.3 Summary

To sum up, various implementation and design of human visions on the reconfigurable platform was reviewed. Despite high performance and accurate results of the hardware implementations, they were focused on a specific feature of the vision, utilizing most resources on it and don't cover complete approach and all basic components of human vision. Software implementations are wide and contain reusable methods of various areas of human vision however they aren't useful to human vision systems because of a general limitation of the computer systems using programmable procedures rather than reconfigurable systems. Differently, to software platforms, reconfigurable computing system paradigm provides real-time adaptations, massive data processing, multitasks parallel independent operations, power minimization, computation and memory usage by direct management and mapping. Because of that, software implementations aren't useful for high-performance real-time systems which could mimic the human vision system [82].

Chapter 3

3. Methodology and Architecture

This chapter involves the methodology and design architecture of the binocular vision system which replicates human vision. Firstly, the observations are highlighted which are utilized in methodology and realized in implementation and design. In the methodology, principles and techniques with the specification of the algorithms are described. The motivation for using the chosen technology is explained. The binocular vision is separated into two parts as the stereo and peripheral vision. Finally, the combined architecture and design is proposed and discussed.

3.1 Observations

Since the aim of the research is to create the analog of the human vision system, it's essential to highlight the main principles of human vision, which are utilized later in the implementation. Two main concepts of the binocular vision as it was discussed in background information are:

1) A stereo vision which enables the human vision to the calculated depth of the object in 3D space
2) A peripheral vision which supports a wide field of view vision

Accordingly, the implementation is divided into two parts, but the main concepts of vision are shared, which involves[28][33] [83]:

1. One of the fundamental properties of human vision is attention. Human vision only focuses on a single window or area or object from the whole view. Instead of seeing the complete image in the high resolution, only the area which is inside the attention is processed in higher resolution while the surrounded part is blurred. The main motivation of having this property in human vision, is to save the time, computations and memory for the processing the whole image and only specify the part which is in high interest[33].
2. The criteria for setting a particular area of the frame to attention window or changing it, can vary to the priority of interest, conditions and history of previous objects. However, one of the main factors is the speed of motion. The objects which appear to move faster normally takes human visual attention, and after observing the features and properties, it is kept or rejected and moved to another object. Similarly, if the object stops moving

after a certain time and condition, it's getting out of interest and new objects, which are moving or prioritized, are taking human attention[83].

3. The attention window is always centered on the interested area, which highlights the importance of finding the centroid properly. Moreover, it doesn't move and avoids jitter by a small threshold distance.
4. The vision is based on the continuous frames, and the object movement is ranged based on the speed, this allows optimize the approach for the tracking as the changing whole frame or unexpectedly showing the object in other location can be considered as noise or abnormal operation.
5. Two almost identical eyes are located on the same baseline and normally share the same angle, which allows saving computational resources for image merging and not to waste time for searching features which are scale, rotation invariant and highly sensitive to viewpoint change.
6. The field of view can be extracted by moving the eye or head in order to cover the whole area. This approach is useful for human; however, in industrial applications, the same results can be achieved by using more than 2 video-sensors since the cost of the sensor is relatively cheap than using the motor. Moreover, multiple video-sensors can share the same techniques of processing algorithms and don't require external setup. Same resource utilization can be achieved by setting inactive and active video-sensors and simulate movement by shifting and adjusting combinations.

Prioritizing criteria for defining attention

Objects, which are coming to the attention area and processed in high resolution by human vision, can be prioritized or filtered based on the properties[83]:

1. **Shape and dimensions**: in real-life application, it's important to define what kind of shape and dimensions are taken into consideration. The real dimensions of the object can be easily retrieved by scaling 2D dimension with the distance factor retrieved by stereo vision. For instance, if the prioritized object is a human, the dimension and the range of it can be predetermined and differentiated from other shapes such as car or spacecraft. Another reason for using shape and dimension-based filtration is a reduction of the noise. For example, if a single pixel of a small area of the image is corrupted or an object is a small mosquito which isn't in the interest of application, they can be easily removed by verifying dimensions. The examples can be seen in figure 3.1 [84][85].

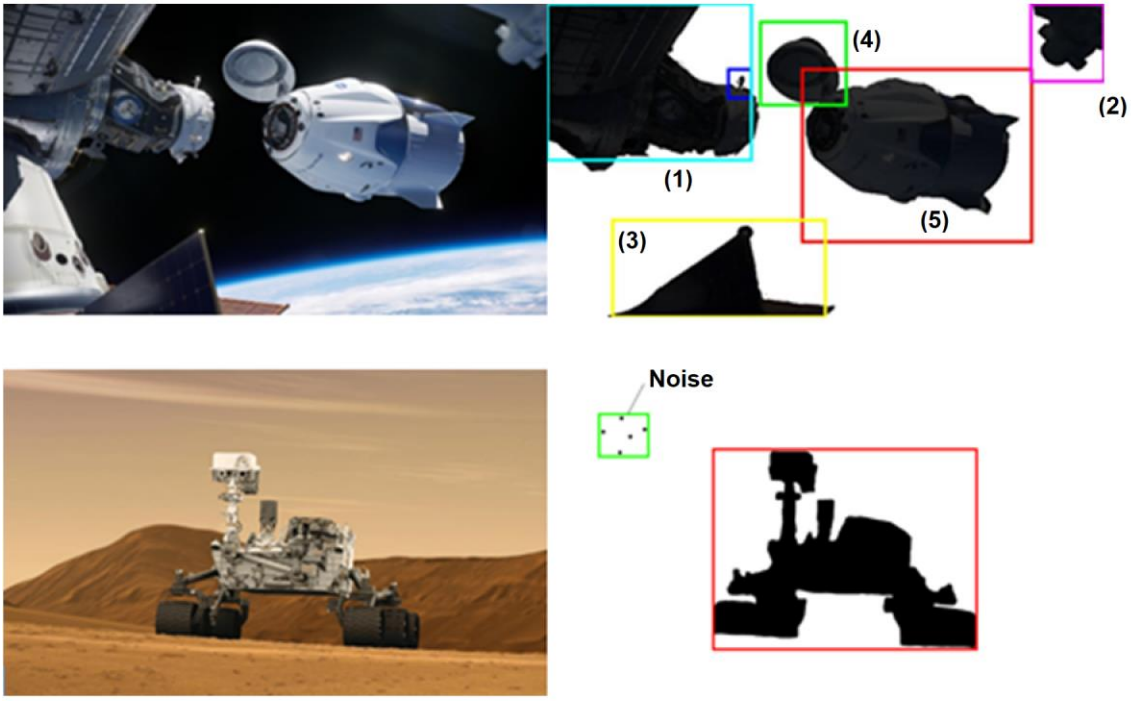


Figure 3.1: Examples of prioritizing objects [84][85]

2. **Distance** is another important parameter for prioritizing objects. The interest of area can be defined only for objects which are closer than certain distance and, accordingly, far objects are filtered. Oppositely, farther objects such as (4) (5) in figure 3.1 can be in the area of interest, and the closer object (3) can be excluded.
3. It's important to retrieve the **speed** of movement of the object. Normally slower objects take less attention than faster-moving objects but depend on the situation a specific range of speed can be defined and utilized.
4. The **direction** of the movement is another important factor of human attention. Cyclic moving objects (ex: trees) can take less attention than the object, which is coming forward to the human, which is normally due to a possible fear factor. The direction of movement is also important to find objects which share the same vector of movement and can be combined to a single object.
5. **Predicted location** is another criterion for prioritizing. The predicted location can be retrieved from the known vector of the object and history of a movement. If the object is coming to the "restricted" area defined by the vision application, it can be taken into

attention before entering for pre-processing or possible reactions. It should be mentioned that this feature can be utilized as security and defense applications.

3.2 Methodology

In this section, main utilized algorithms and methods are described, which involves object detection, object representation, tracking, techniques of memory and resource minimization, multi-object extended approach, disparity calculation, and other features.

Object detection

The principle of object detection is based on the movement of an object. Every two following frames are kept in the memory for each cycle of operation. The difference is calculated for each pixel value, which is filtered with a certain threshold. The example result of movement detection and conversion to the binary form is shown in Figure 3.2.

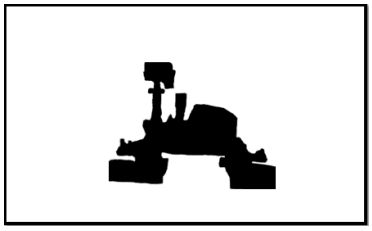


Figure 3.2: Movement detection instance example

This is a standard procedure for movement detection. The procedure allows detecting moving objects based on the assumption that video-sensors are continuously operating. If a new object appears, it won't be missed because of showing the difference to the previously retrieved frame. Moreover, the threshold value filters the possible pixel noise caused by the video-sensor capture or transmission of the digital image.

Presentation of the object instances

The object is presented in two properties:

1. A **rectangular area** of the object which is surrounded through the object
2. A single point which is the actual **centroid** of the object inside.

The rectangular area of the object is not only the dimensions of the moving area but as well the window of "attention", which is called **blob**. This means that the area is captured, processed, and kept in high resolution and outside are in low resolution, as displayed in Figure 3.3.

(Note: the term blob will be referred to the attention as the area which is kept in higher resolution in following part of the thesis).

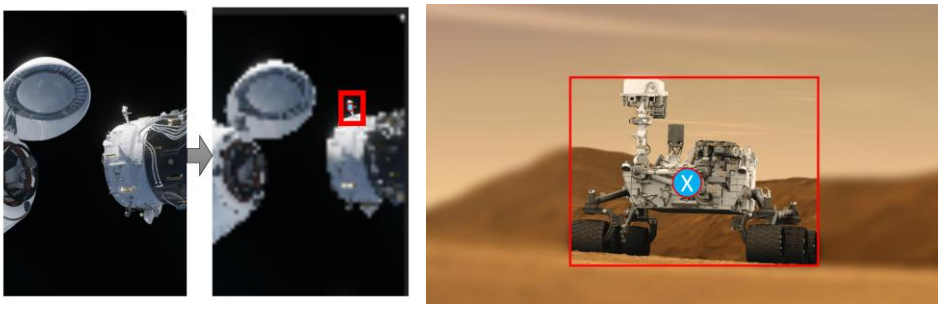


Figure 3.3: Blob representation example

Two types of memory for image storage are defined:

1. **Background storage**, which is for 4x4 compression of the frame
2. **Blob storage**, which is high definition area of the focused object

The blob is not always kept in the highest possible definition, and it also depends on the distance, size and dimensions of the object. The idea is to define the resolution adaptation, which is proportional to the dimensions and properties to it. If the object is far, the highest resolution is taking place, and if it's relatively closer and bigger enough and doesn't fit into the predefined size of memory for the blob, it is compressed proportionally. The idea is illustrated in Figure 3.4. To sum up, the size of each memory unit type is already fixed, and dynamic resolution is utilized for blobs.

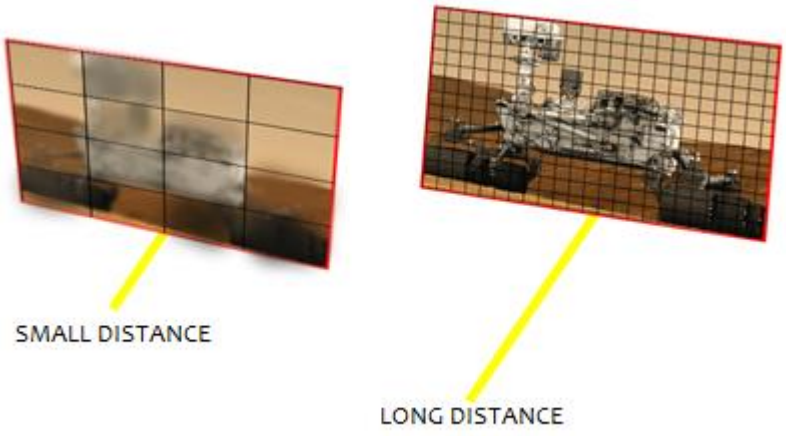


Figure 3.4: Blob resolution adaptability

It should be noted that if the dimensions of the object are small and far enough that doesn't require whole memory defined for the blob, the extra surrounding is kept in higher resolutions. The main idea and concept of this technique come from the human vision, and the motivation is to have constant accuracy regardless of the distance. It should be emphasized that the centroid need to be defined in higher accuracy for a farther object than closer objects in order to calculate the distance with acceptable error. Figure 3.5 shows the error function for the distance calculation. The function shows the correlation between an error of the distance (caused by a single-pixel error in centroid calculation) and the distance. The x-axis shows the disparity values, and the y-axis presents the error values. As it can be observed smaller getting the disparity, which means that the object is farther, the single-pixel error can cause a huge error in distance calculation. The reason for it is the inverse proportional relation of the distance and disparity by the definition of triangulation formula.

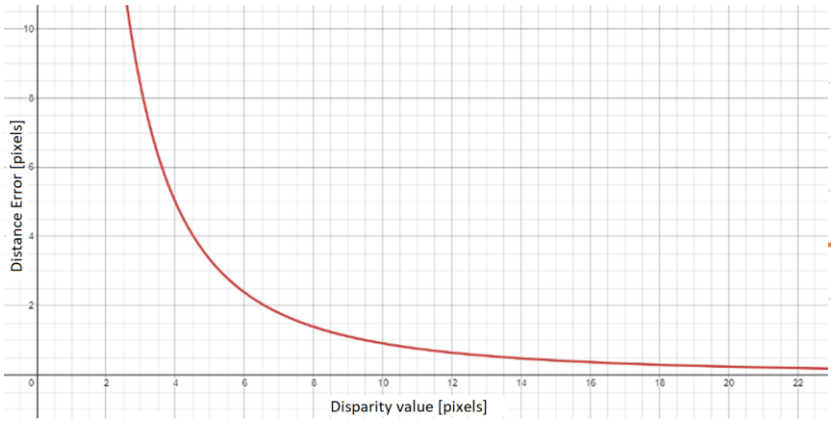


Figure 3.5: Distance Error function according to the single-pixel error of a disparity

It should be noted that once an object window is determined, the result is verified from different video-sensors. Verification is performed by two parameters:

- Dimensions of the blob: the size of the width and height of the blob should be within the acceptable threshold distance value in order to be considered as the same object determined from different video-sensors
- If the object was tracked before and the same blob has moved, the vector of the movement should be the same within acceptable error from the different video-sensors.

Another important note which should be emphasized is that processing and keeping blob in the higher definition is performed for the next frame after verifying blobs, in order not to waste time reprocessing of the same frame and keep high cycle time.

It should be mentioned that if the object stops moving, a blob is kept active until the certain number of frames after which it becomes the part of the background. However, the last centroids in 3D space are kept for up to previous 8 objects, and as they get active again, it's being tracked and processed in the same high-resolution blob. The reason for keeping the blob, in case of a certain time of inactivity, is to avoid losing the attention to the slow-moving objects.

Centroid calculation

The actual centroid of the object doesn't always match the center of the blob because:

- Taking the center of the window instead of the actual centroid of the object leads to lower accuracy of disparity calculation. The possible noise of movement or different views of video-sensors can result in slightly different shapes of motions and blob sizes which can be accepted by blob verification step, but the error must be excluded in the centroid part
- The actual centroid depends on the object shape. For instance, if the most part of object shape is distributed in the left side of a blob, the center should be adjusted to it
- If the object is smaller enough that fits the area of the blob size the surrounded part is also kept inside the blob, and the actual center can be different
- As it was mentioned before smaller movements of the object doesn't change the center of the blob while the actual center of the object is altered

Calculation horizontal centroid in enough for distance estimation since it leads to horizontal disparity and the video-sensors are horizontally separated to each other and vertically share the same line. However, with the same technique, the vertical centroid is determined for further processing, Y-coordinate estimation, and better visualization.

As it can be observed from human vision, even the object is moving within a certain relatively smaller distance the blob isn't moving. The same technique is applied to the implementation. Inside the blob rectangle, 4 times smaller (2 times smaller width and 2 times smaller height) imaginary concentered rectangle is defined. The position of the blob only changes if the centroid crosses this imaginary rectangular boundary in order to avoid smaller jitters in blob positioning.

This method is illustrated in Figure 3.6, where the green rectangle represents an imaginary area of centroid movement without changing the blob position.

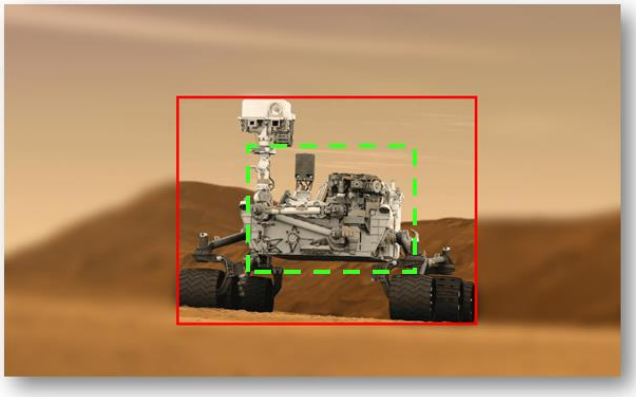


Figure 3.6: Internal sub boundary of centroid movement

Multi-object tracking

As it was mentioned and observed the attention of the human vision is always concentrating on a single area or objects. The developed system can be extended with the concept for multi-object detection and tracking and utilize the same methodology. The detection process is described below:

1. All the objects which are detected previously are presented in an additional virtual border, which is a bigger concentered window. It is padded with a predefined value through the borders to the blob window (similarly as it was made to the inner disparity movement area). The outer rectangle reserves the space to the object not to be mixed to another object and considered as critical area when tracking multi-objects. This approach is illustrated in Figure 3.7, where the blue rectangle represents the outline virtual boundary. Moreover, this method prevents detection the part of the previously tracked object as a separate object.

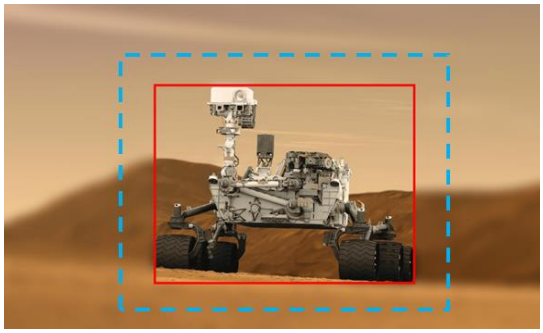


Figure 3.7: External boundary of object tracking

2. All the predefined outer window areas are skipped during the detection procedure, and if the new object is presented in the remaining part of the image, it's detected, and the new blob is defined.
3. If objects are crossing each other, the object which goes behind, which is determined by the calculation from stereo vision, is kept in memory. It is reserved for several frames, and only the visible frontal object is presented unless they separate each other.

Video-sensor operation

Human binocular vision requires two video-sensors. However, the mechanism of moving the head and eyes gives an extra field of vision. In the design and implementation, more than two video-sensors are utilized. Basic setup involves the operation of 3 video-sensors which are separately utilized for the stereo and panoramic vision. The reason for choosing more than 2 standard video-sensors instead of additional mechanism for movement or wide-angle cameras is that it a cost-efficient approach. The standard camera which supports 60-90 degrees field of view is relatively cheaper than cameras with a higher field of view, they use fewer resources and require lower power usage. Having an additional motor for rotation similarly to human vision would make the design more expensive and overcomplicate the approach as would require external control and additional power. Despite the fact multi-cameras are utilized, they can be activated in combination by the adaptability, which saves the computational resources.

The video-sensors share the same baseline and angle, similar to the human vision, which results in easier processing and feature extraction for stereo vision and stitching images. As well

they are the share the same configuration which provides the same initial resolution, dimensions, format, and other properties of captured images.

As it was discussed, the images, captured by video-sensors, requires pre-processing for noise reduction and better quality. The pre-processing is done inside the camera sensor unit and involves the following processes:

1. Black and white pixel correction, which supports fixing stuck pixels
2. Automatic white balance control, which removes unrealistic color casts
3. Lens correction, which avoids distortion
4. Contrast center auto adjustment
5. Color banding, which is in the process for avoiding inaccurate color presentation
6. Color interpolation
7. Adaptive Gain Control with the Least Mean Squares, which prevents artifacts
8. Automatic Brightness Control, which adjusts brightness automatically

The methodologies described above were the common components for stereo and peripheral vision. In this section specific methodologies and principles for stereo and peripheral vision are described.

Stereo vision specific methodologies

Video-sensor Adaptation

While three video-sensors are utilized for tracking object, only two of them are in the active state. The selection of active cameras depends on the location of the object. The approach is illustrated in Figure 3.8.

Figure 3.8: Video-sensors adaptation examples

It should be noted that not only horizontal location is considered, if the object is farther, 2 furthermore cameras are utilized in order to have higher accuracy for distance calculation. This technique replicated the human vision feature that eyes are adjusted by the distance of the object. In terms of implementation, it provides the dynamic adaptation of the cameras as there is no necessity to process all of them for stereo vision. The provided technique can save power, memory and computational resources.

Verification

As it was mentioned before, objects are verified from both video-sensors to avoid mismatching them and prevent possible noise, which can be specific to a single video-sensor. The verification is done by the following procedure:

1. Single-camera verification: for a small sample of frames the object dimensions and 2D location should keep the same with an acceptable change as verification that camera functions correctly and consider the area as an object.
2. The dimensions of the object from different video-sensors should be similar to each other. The small difference is acceptable due to the fact that images aren't captured from the same location
3. If an object was previously analyzed and tracked, the 2D vector from each video-sensor should match.

Once it is verified and the centroid is calculated by the procedure described above, the next step is to find the distance of the object.

Distance determination

The idea of distance calculation is based on the standard triangulation illustrated in Figure 3.9, however, some of the parts are modified and optimized.

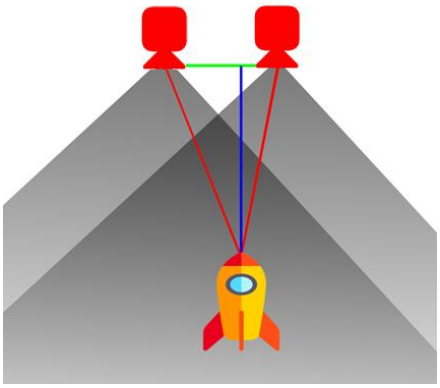


Figure 3.9: Standard triangulation method

Let's assume that the object captured is the same and the centroids are calculated correctly and are c_1 and c_2 from video-sensor 1 and 2 accordingly.
The disparity value can be calculated as follows:

$$disparity = abs(c_1 - c_2)$$

Since there is inverse proportional relation between disparity and distance with the constant factors based on the focal length and baseline (assume that despite the adaptation of video-sensors, the distance is fixed). Instead of specifying the parameters, the range as an example [0 to 1024] can be defined and result value can be simply rescaled as follows:

$$distance = 1024/(disparity + 1)$$

The reason for adding unit 1 to the disparity, is that it can be zero for very far distances.
This is a simpler way of calculation; however, this approach can be improved by replacing it with a lookup table since:

1. Division is always computationally expensive operation while lookup table gives result in a single (or double depends on the size) clock cycle.
2. Depend on the application lookup table can be redefined and customized. As well it doesn't require to resynthesize and reroute the whole design and can be simply modified by changing lookup table values.
3. If the specific more complex formula is utilized, the lookup table can simply be filled with values derived from the function in compile time.

Table 3.1 illustrates one of the simple lookup table which was used testing in the lab, the range is relatively smaller due to the smaller distance and customized to 10 steps, measured in foot (ft), and have the similar function to the division with a smaller range and smaller sensitivity changes based on the object length. Figure 3.10 illustrates the functional form of the table.

Table 3.1: Example LUT of distance determination by disparity value

Disparity range [pixels]	Distance [ft]
1-18	10
19-54	9
55-108	8

Table 3.1 (continued)

109 – 180	7
181 – 270	6
271 – 378	5
379 – 504	4
505 – 648	3
649 – 810	2
811-1024	1

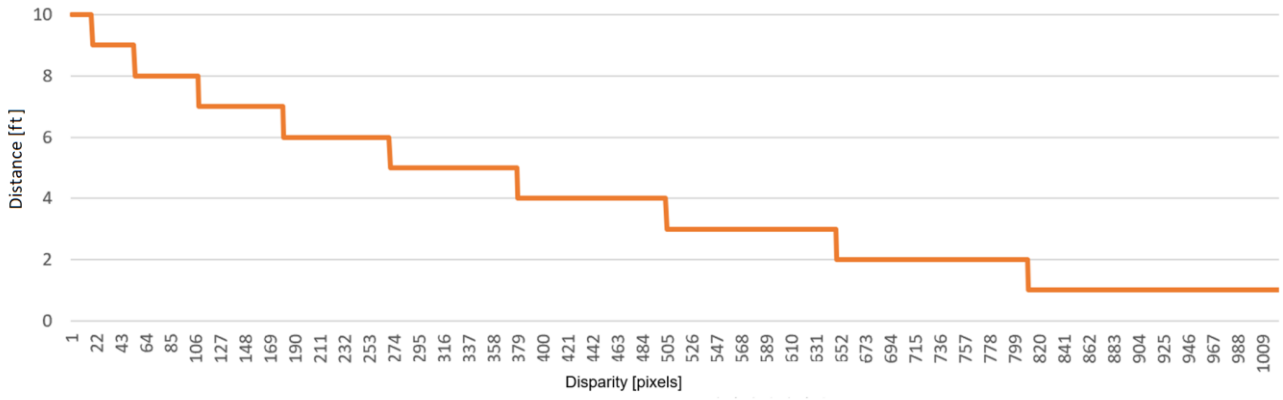


Figure 3.10: Functional representation of the example LUT of distance determination

Smoothing Results

It's essential to maintain a stable result in order to:

1. Remove noisy data and minimize errors
2. Exclude small jitters or sharp movement to keep smoother results for human interface

Since smoothing results are performed in the post-processing stage, which is executed per frame instead of each pixel, it can utilize whole time spend on a single frame processing. The longer time gives the possibility to use high accuracy and a more complicated algorithm than threshold filtering or averaging.

Linear quadratic estimation filter is utilized for achieving desired results. Figure 3.11 illustrates the diagram of the entire smoothing procedure.

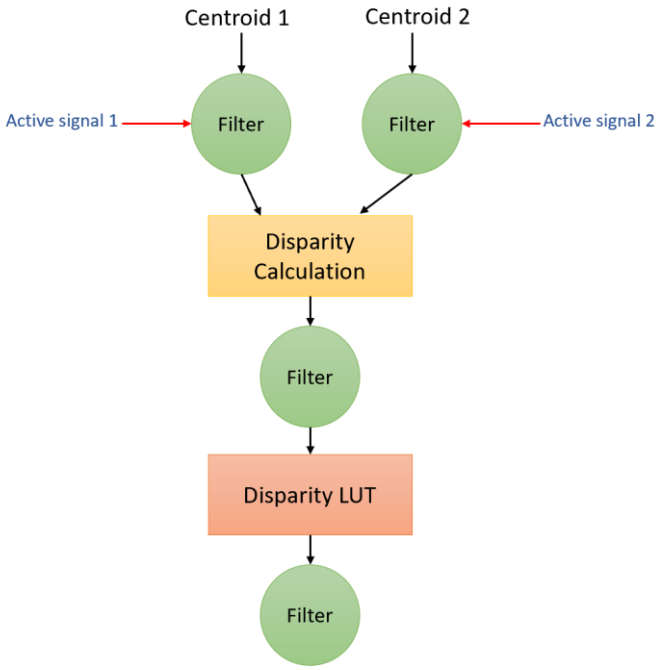


Figure 3.11: Smoothing filter operational layers

As it can be observed, firstly, the centroids are smoothed separately and controlled by an active signal, which is the indication if the objects were moved and the data is new or not. Active signals are important because if the last data was a noise and the object stops moving, the rest of the time noise data will get a higher advantage, which is prevented. After smoothed values disparity is calculated, which is smoothed with another layer of Kalman filter as well. Finally, the distance value is retrieved from the disparity lookup table and smoothed. Despite the fact that Kalman filter is used in every stage of smoothing, the model of each one and the parameters for measurement and covariance noise differ to each other, because of different sensitivity and probability to the error.

Representation of results

As the distance is calculated and centroid provides horizontal and vertical location it's important to represent results in a feasible way for custom application and user easy representation.

Mapping the trajectory of the object movement

The custom 3D grid is defined which can be used for possible applications. The 3D position is mapped to the grid for better representation. The trajectory as the history of previous points makes it possible to determine the vector of movement by sampling last 8-16 points and predict the future location of the object. The mapping trajectory is illustrated in Figure 3.12.

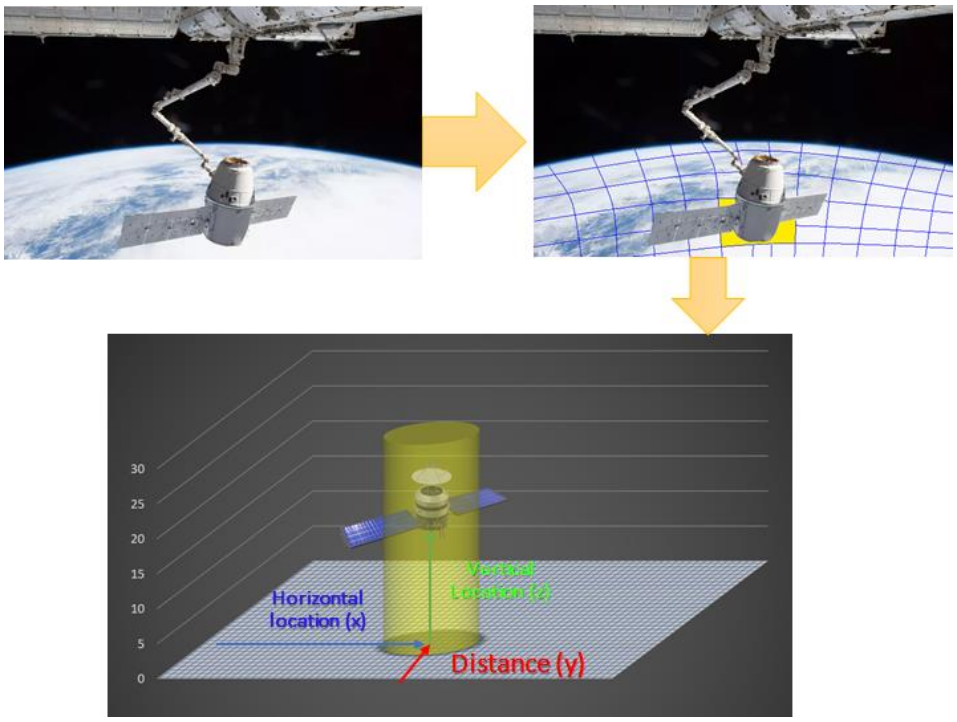


Figure 3.12: Mapping trajectory on the 3D grid

Detection if the object has passed the border of pre-determined area-of-interest and action respective to this event

The grid gives a general platform which can be used in various areas such as security systems, defense, space mission and others. On top of that, the implementation also supports defining the imaginary borders and alarm-based events. The events can be varied by the logic if an object enters the area, leaves the area or prediction of it or a combination of these approaches. Figure 3.13 illustrates the proposed custom borders for the object where borders are defined in red color and the blob is highlighted in yellow color.

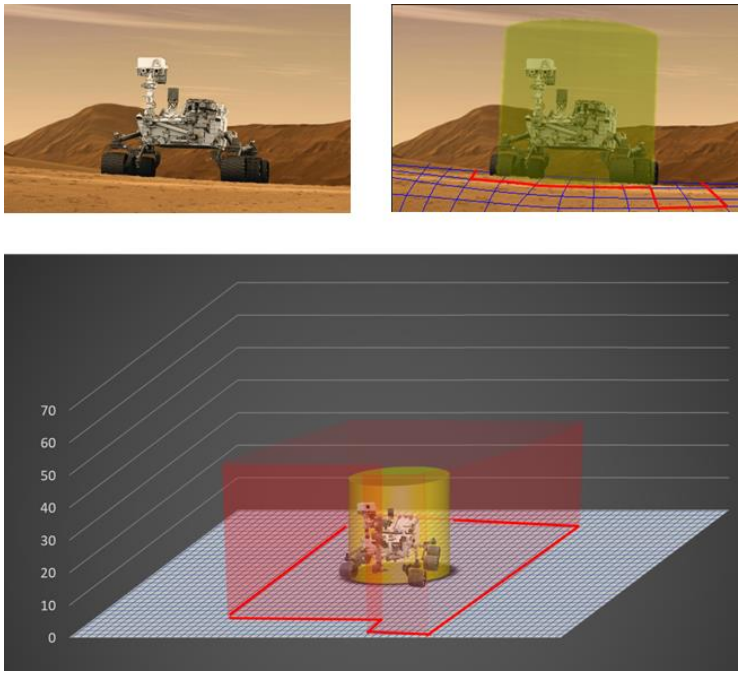


Figure 3.13: Custom Borders on the grid for event-based alarms

Peripheral vision specific methodologies

The main aim of the part is to obtain a wide-angle view capture of an area with saving computational resources to utilize in monocular vision object tracking. Since the standard video-sensors are limited in the angle of the field of view which is typically 55-90 degrees, multiple video-sensors are utilized to achieve a wider field of view. The algorithms and implementations, which were discussed in related works section, involve complex operations for keypoint matching and perspective projection. The approach, developed in the thesis implementation, is based on a system which utilizes 3 video-sensors, which aims to create a merged image by stitching images with simple rectification and use the techniques of distance calculation with single monocular vision.

The video-sensors are attached on the flat surface. The distance between each camera is ~10 cm which is similar to the distance of human eyes, which ranges 58 mm to 68 mm. The longer distance is chosen because of the purpose to be able to use the same setup for stereo vision and determine distance with higher accuracy. Figure 3.14 illustrates the base idea of the proposed technique.

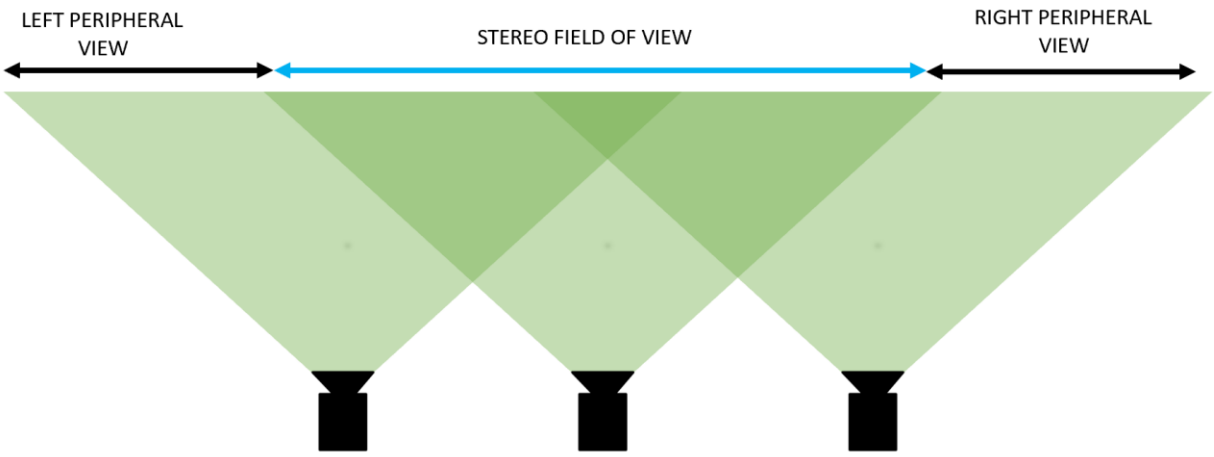


Figure 3.14: Distribution and setup of video-sensors

The whole image, which is seen from all video-sensors can be divided into three parts as it is illustrated in Figure 3.15. As it can be seen, the images from left and right video-sensors are cropped and stitched to the original center camera image, however since the video-sensors have different viewpoints, the rectification and blending are required to get smooth continues image result. In the implementation, frames of side video-sensors are cropped with predetermined offset and rectified to the middle frame because the effective stitching algorithms are computationally intensive, and the main point is to use for monocular distance determination and tracking in a larger field of view. Despite the fact that visual results aren't effective, the challenges and purpose of having wider FOV for tracking 3D objects in peripheral view and avoid computationally expensive operations are achieved.

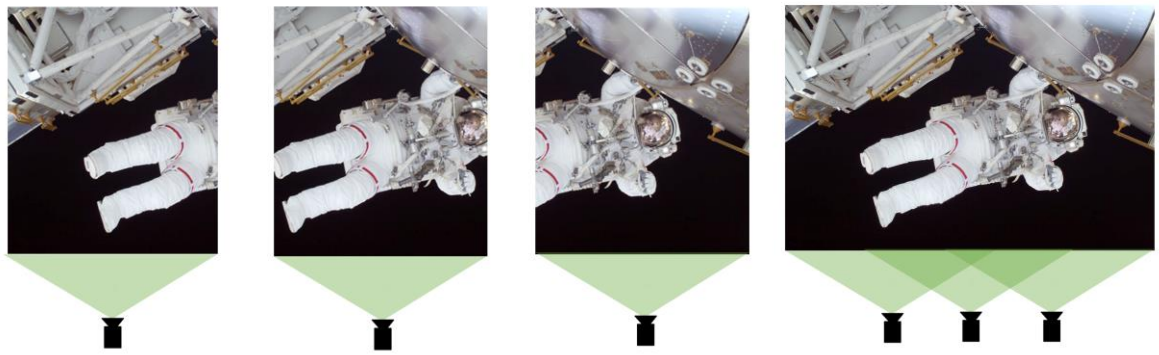


Figure 3.15: Overlapping example of the images retrieved from video-sensors

Once the object leaves the stereo field of view, it is still tracked from the peripheral part and the distance is calculated in proportional relation to the length of the width as it is illustrated in Figure 3.16.

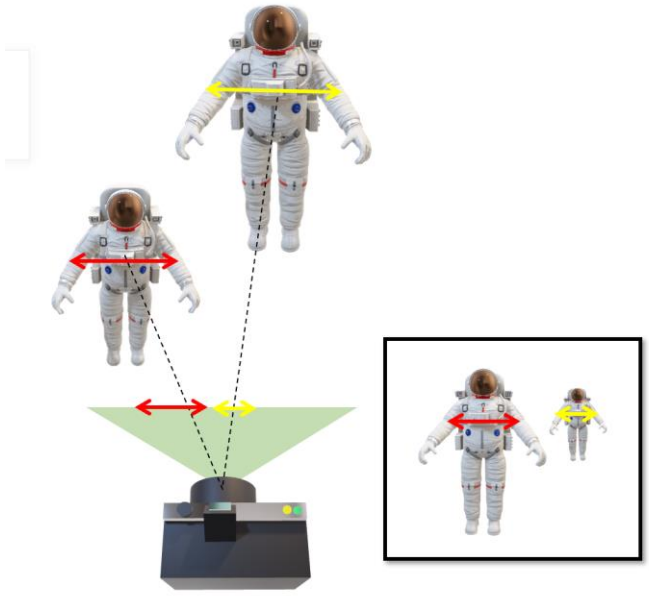


Figure 3.16: Monocular distance calculation principle

The proposed method saves time for the processing frame to find the key points and features to be matched, which are the most time consuming and computationally expensive operations.

3.3 Architecture

The architecture of the developed system consists of two components:

1. Stereo vision, which operates based on the overlapped field of views of video-sensors and calculates 3D coordinates using the triangulation technique
2. Semi-Panoramic vision, which is used for peripheral vision. It merges frames obtained from video-sensors to a single stitched image.

Finally, the combined architecture is proposed, which integrates these two components and, additionally, determines the distance from peripheral vision using monocular tracking.

Stereo Vision Architecture

The stereo vision system architecture is illustrated in Figure 3.17 and consists of nine different type of modules.

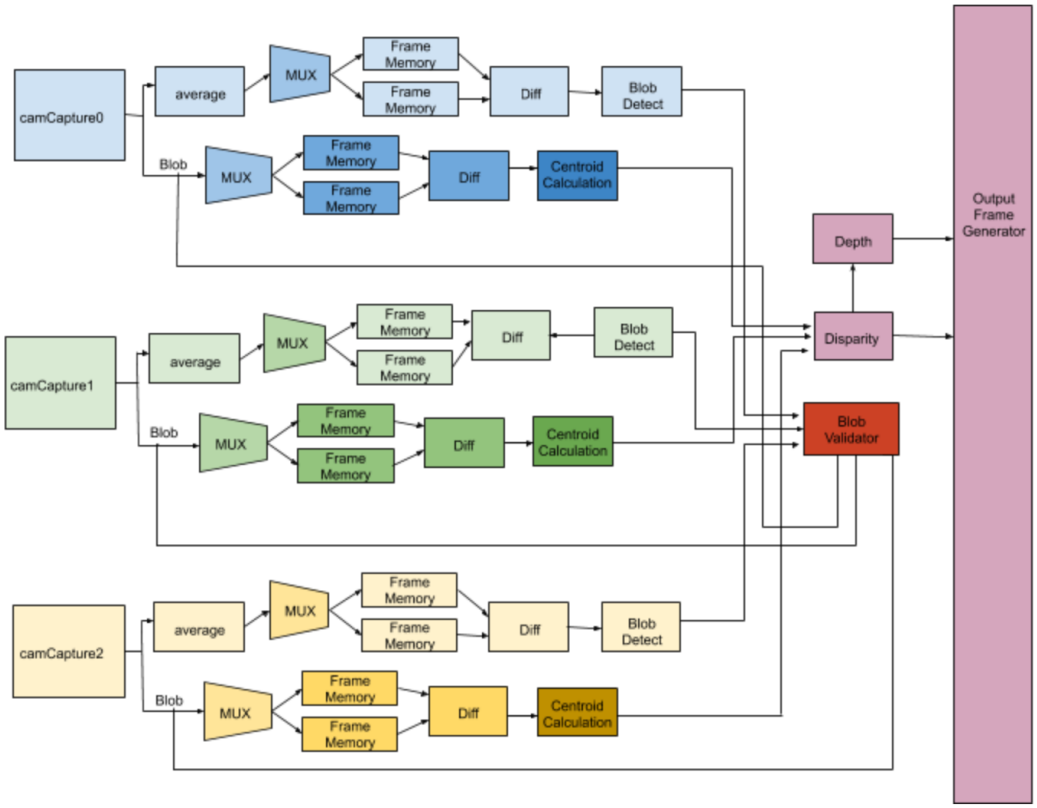


Figure 3.17: Stereo Vision Architecture

The principles of operation are the following:

It takes three camera images as input and processes them in parallel. Firstly, each image is averaged by 4x4 (16 pixels in group: 4 rows x 4 columns) in order to achieve higher speed for background processing. After averaging part. It should be noted that the two following frames are kept in the memory, and the multiplexer controls and navigates memory part for each new frame.

After saving the frame, the difference between the two frames are generated and the movement area of the object is detected. The blob is determined, which represents a window where the movement was detected. Sizes of detected blobs from all video-sensors are compared, validated and verified by blob validator module. If the validation is successful, new blob coordinates and dimensions are taken for each video-sensor.

The blobs are stored, and motions are detected with the same technique, but in higher resolution without averaging because more accuracy is needed for further processing.
For each acquired blob, the centroid is calculated, which follows the disparity determination. According to disparity, the distance is obtained by LUT. Finally, the output frame displays: averaged video-sensor outputs, detected blobs in higher resolution, the depth, the disparity and other custom values.
In addition, the input controls configure the mode of the output frame generator, which make it possible to show different representations of the results.

Panoramic Vision Architecture

Panoramic vision architecture, which is shown in Figure 3.18, consists of four components. Initially, it takes three video-sensor images and stores them into corresponding frame memory. It should be noted that the middle video-sensor is preserved originally, while side video-sensors are cropped. After the acquisition of the data, the inputs are rectified, merged and sent to the output generator module.
It should be mentioned that the camera capture module is the same as in stereo vision architecture and output frame generator shares the same components, these overlaps are shared in combined design.

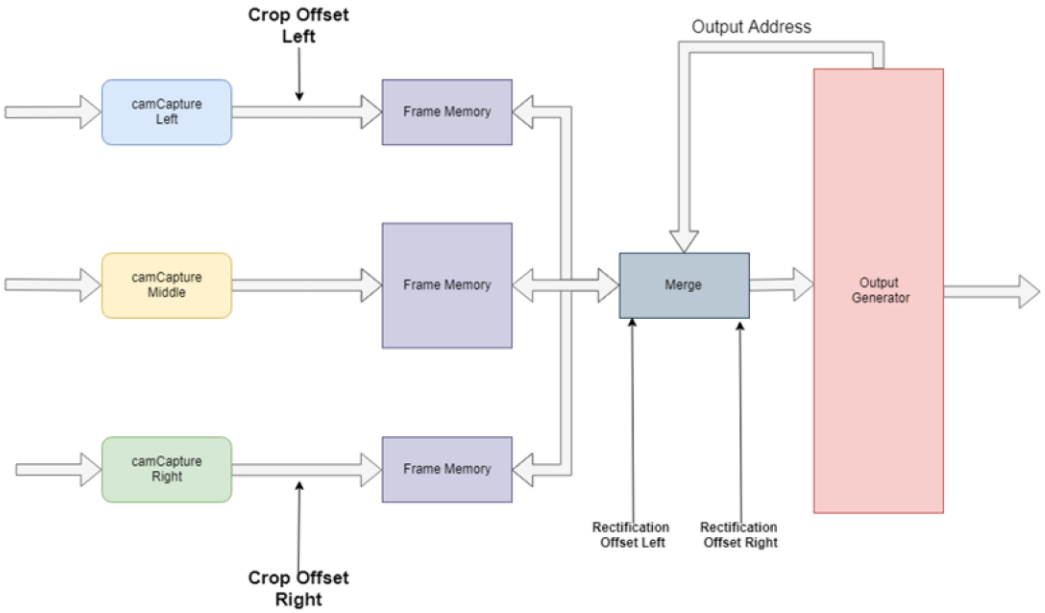


Figure 3.18: Peripheral Vision Architecture

Integrated Final High-level Architecture

The final architecture combines stereo and peripheral (semi-panoramic) vision components. As it was mentioned, the camera capture module and output generations with small internal components are shared. The principle of operation is the following: if the object is viewed from at least two video-sensors, it is analyzed through stereo vision and once it leaves the stereopsis field of view, the distance is calculated and tracked based on the width and height of the object. The combined design diagram is illustrated in Figure 3.19.

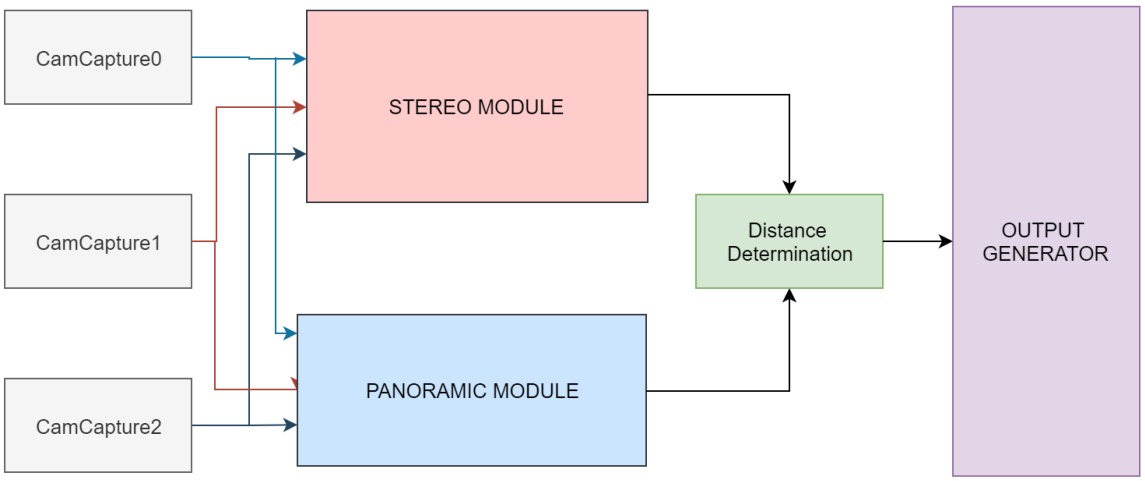


Figure 3.19: Integrated Final High-level Architecture

Chapter 4

4.1 Implementation and Design

4.1 Experimental Setup

The design and implementation are developed in a hybrid way, which means that it utilizes FPGA hardware design with the addition of software, executed in the ARM. The software is implemented on procedural and imperative C language and only used for parts where time limitation constraints are lower and don't affect processing cycle time. The usage of software includes:

1. Initialization stage, where video-sensor registers are initialized, and HDMI is set up.
2. Post-processing part, where the results are processed per frame instead of the pixel. For instance, smoothing or verification of 3D vectors.
3. Debugging and testing purposes such as setting and modifying values and logging important data
4. Customization purposes such as graphical user interface

Hardware design is done on VHDL using Xilinx ISE/Vivado Design Suite computer-added design (CAD) tools.

Zedboard

Zedboard is a board which is architected for evaluation and development purposes. It is a part of Xilinx Zynq-7000 family with integrates hardware programmable FPGA in the combination of the ARM processor [86]. It's widely used in development because it provides analytics and control tools, hardware acceleration, set of design and CAD tools. Moreover, it is designed to achieve the lowest system power. With 85000 Programmable logic cells, it contains the ARM Cortex-A9 which is a 32-bit processor with the ARMv7-A architecture with 4 cache-coherent cores. The block diagram of the Zedboard is illustrated in Figure 4.1 [86].

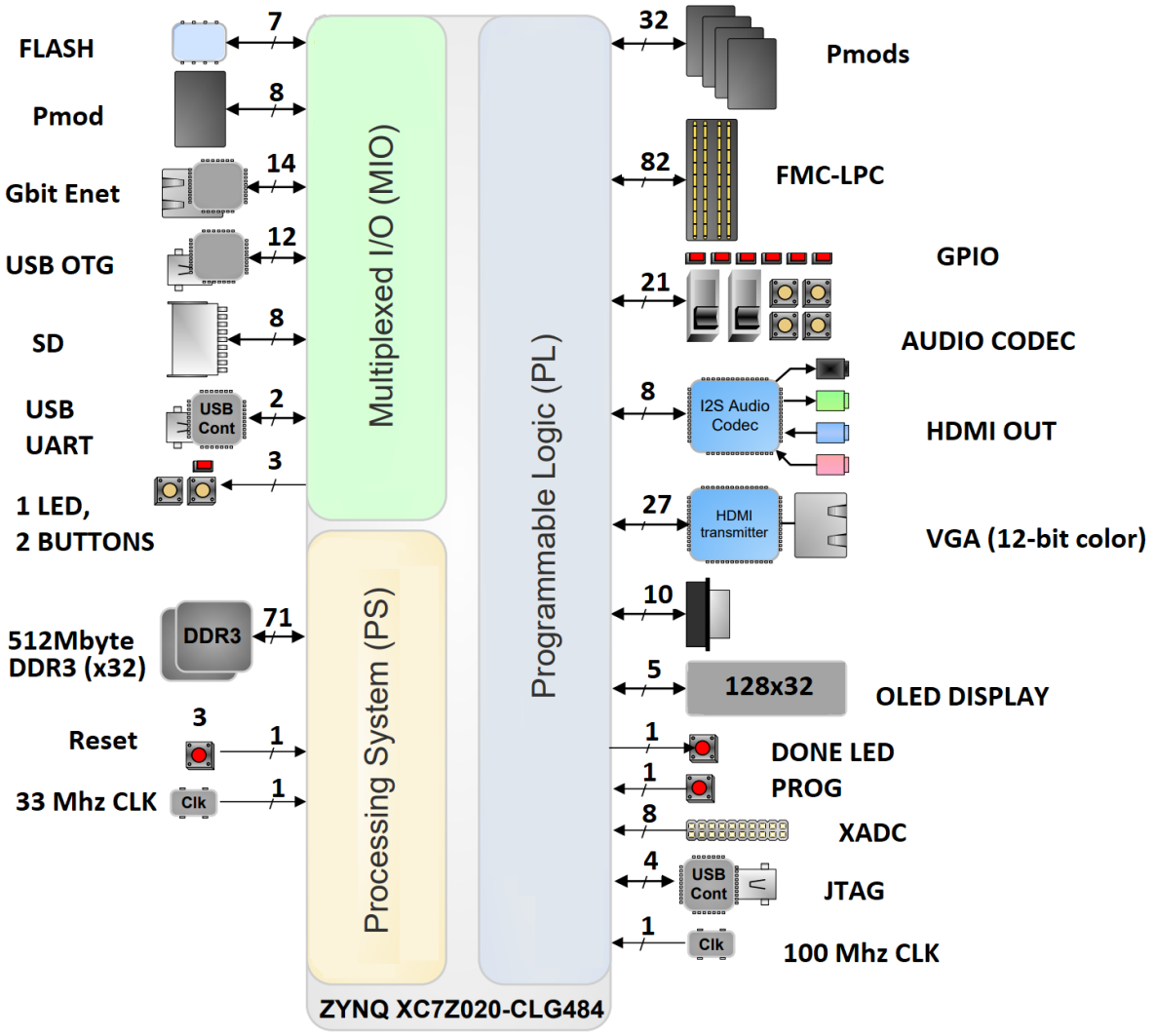


Figure 4.1: The block diagram of the Zedboard [86]

As it can be observed it contains HDMI 225Mhz transmitter with a digital video interface, which is compatible to HDMI 1.3 version with support of 1080p60 16-bit, YCbCr, 4:2:2 mode values. 512 MB Double Data Rate 3 (DDR3) Synchronous Dynamic Random-Access Memory which enables to utilize it for saving frames for processing, however since it's external on the board and the design is optimized for memory usage BRAMs (4 KB) are utilized for faster operation [86]. Zedboard supports 140 BRAMs with the ability to be configurable and convert as a true dual-port memory unit. From the I/O pins, the switches and buttons are utilized for debugging purposes and FMC is used for connection to the printed circuit board (PCB) which connects to video-sensors [86].

Video-sensors

Special purpose OVM7690 video-sensors are utilized in the proposed solution. The video-sensor is 2.5x2.9x2.5 mm cube which combines a wide variety of functions inside a single chip[87]. Another advantage of using a camera is that it can be directly soldered to the PCB instead of using external mount device. The resolution of the video-sensor is 640 x480 pixels which are suitable for research constraints. It supports F/4.0 lens with a 64-degree field of view. It uses relatively low power as requires 2.6-4 *v* analog power supply. the temperature operation, in order to get a stable image, is 0 to 50-degree Celsius which is acceptable for testing setup. It supports different output formats YUV 442, RGB 565, CCIR656 and raw RGB data, which is used in the implementation. Input clock frequency ranges from 6-27 Mhz. The frame rate operates in 2 modes:

- VGA: 30 fps
- QVGA: 60 fps

To summarize, the video-sensors is low cost, ultra-low power usage solution with the acceptable performance and suitable for research purposes. Moreover, it should be noted that it integrates standard basic image processing inside the chip such as exposure, gamma, color saturation and reduction of the noise functions. The video-sensor and functional block diagram are represented in Figure 4.2 [87].

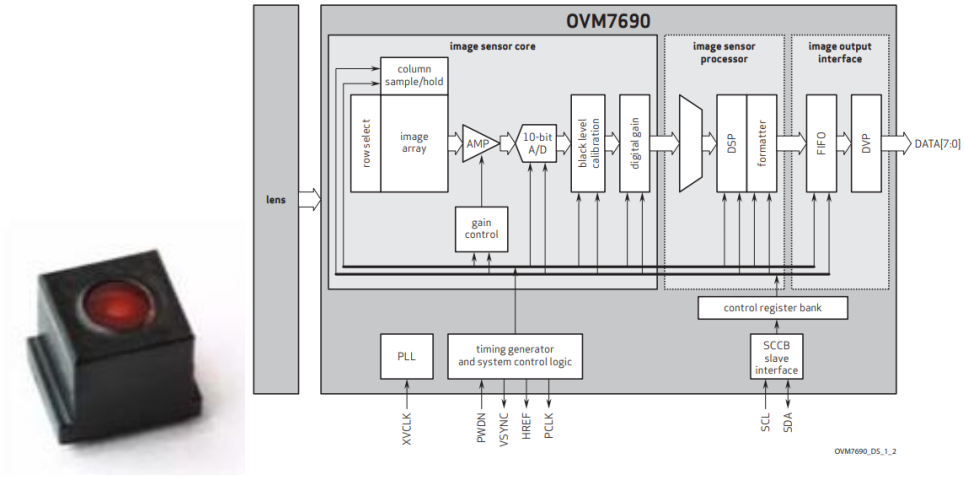


Figure 4.2: The video-sensor and functional block diagram of it [87]

The video-sensors is configured in the initialization stage through the ARM by setting values to the registers, the main parts of configuration involve [87]:

- The output format is settled to Bayer pattern RAW RGB mode. The patter follows a combination of red, green and blue values with twice as many green pixels as blue or red in order to replicate the principle of the physiology of the human eye. The pattern is illustrated in Figure 4.3 [88].

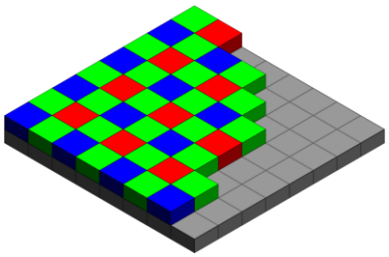


Figure 4.3: Bayer pattern

- VSYNC, HREF, and PCLK are selected and used for output state
- Pre-processing AGC and AEC algorithm are enabled with unlimited step size
- Automatic gain control is enabled for automatic adjustment of the image frame
- Exposure control is activated
- Auto mode is selected for contrast center adjustment
- Other noise reduction image processing functions are selected and enabled including color interpolation, black and white pixel correction, lens correction, auto white balance and etc.

Setup

3 video-sensors are mounted to the custom printed circuit board (PCB) with the addition of

1. 6-Bit Bidirectional Level shifter for translation voltage domain
2. Adjustable and Fixed Low-Dropout Voltage Regulator to maintain a stable voltage level
3. The clock generator generates 25 MHz clocks for video-sensors
4. 2-bit Bus Transceiver with support of dynamic Level-Shifting and Translation
5. FMC module for I/O

The layout of the PCB is displayed in Figure 4.4. The distance between closer cameras are 10 centimeters and mimics human vision parameters.

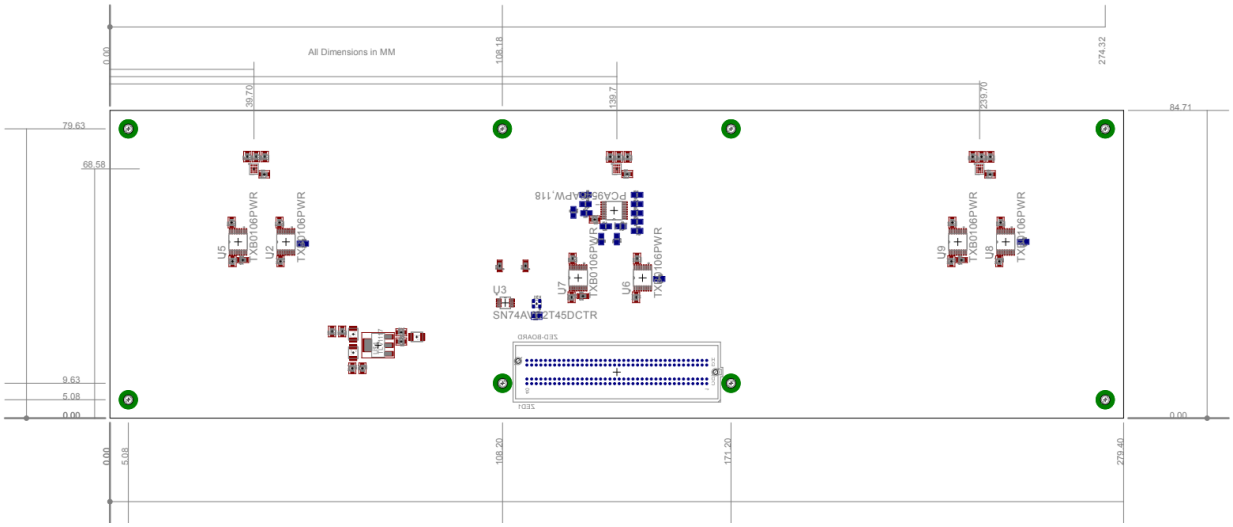


Figure 4.4: The layout of the PCB

The custom board is mounted to the Zedboard through FMC module.

The experimental setup is displayed in Figure 4.5.

Figure 4.5: Experimental Setup

4.2 Implementation and Design

In this section technical description of the implementation and design is presented, which is separated into 2 parts: Stereo vision and peripheral vision. The implementation shares the same base of submodules and same setup with modifications and addition to the specific vision component. In the end, the combined final implementation and design is proposed and described.

Modules of Stereo-Vision

Top-Level Module

Top-Level Module of stereo vision is the biggest module which integrates all of the submodules of the architecture, which is illustrated in Figure 3.17.

Functions:

Top-Level Module Performs the following functions:

1. Receives frames from 3 video-sensors
2. Averages frames from video-sensors and saves in memory
3. Find the difference between 2 frames (for Movement detection)
4. Detect the blob area
5. Finds centroids in the blob
6. Calculates Disparity and Depth of the Object
7. Generates output Frame and Displays Result

Structure:

Top-Level Module includes 9 different types of submodules.

It should be mentioned that the following submodules are used for each video-sensor frame processing:

1. CamCapture Module which takes the image frame from video-sensor decodes and performs function 1
2. Average Module, which performs function 2
3. Diff Module, which performs Function 3
4. Blob detect Module, which performs Function 4
5. Centroid Calculation Module, which performs Function 5

The Following Submodules are shared among the video-sensors:

6. Blob Validator, which validates and verifies detected blobs
7. Disparity Calculation, which performs function 6 (one part)
8. Depth Calculation, which performs function 6 (depth part)
9. Output Frame Generator, which performs function 7

The symbol and interface of the module are shown in Figure 4.6 and Table 4.1 respectively.

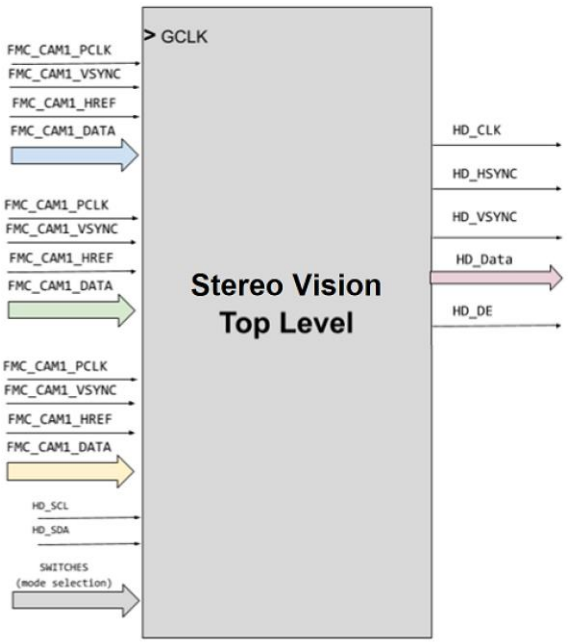


Figure 4.6: Symbol of Stereo Vision Top Level

Table 4.1: Interface of the stereo-vision top-level

Name	Type	Mode	Description
GCLK	std_logic	IN	Global Clock
FMC_CAM0_PCLK	std_logic	IN	input clock of camera 0
FMC_CAM0_DATA	std_logic_vector(9 downto 0)	IN	frame pixel of camera 0
FMC_CAM0_HREF	std_logic	IN	start of a line of camera 0
FMC_CAM0_VSYNC	std_logic	IN	start of a frame of camera 0
FMC_CAM1_PCLK	std_logic	IN	input clock of camera 1
FMC_CAM1_DATA	std_logic_vector(9 downto 0)	IN	frame pixel of camera 1
FMC_CAM1_HREF	std_logic	IN	start of a line of camera 1
FMC_CAM1_VSYNC	std_logic	IN	start of a frame of camera 1

Table 4.1 (continued)

FMC_CAM2_PCLK	std_logic	IN	input clock of camera 2
FMC_CAM2_DATA	std_logic_vector(9 downto 0)	IN	frame pixel of camera 2
FMC_CAM2_HREF	std_logic	IN	start of a line of camera 2
FMC_CAM2_VSYNC	std_logic	IN	start of a frame of camera 2
RST	std_logic	IN	reset
SWITCHES	std_logic_vector(6 downto 0)	IN	Controlling testing modes and changing crop and rectification adjustment
HD_CLK	std_logic	OUT	Output clock for HDMI
HD_DATA	std_logic_vector(17 downto 0)	OUT	Data output for HDMI
HD_HREF	std_logic	OUT	start of a line of the output frame
HD_VSYNC	std_logic	OUT	start of an output frame
HD_DE	std_logic	OUT	Data Enable signal for output

Camera Capture Module

Functional Description

Camera Capture Module converts raw input taken from the video-sensor to a more usable format. Figure 4.7 presents the raw input [87], as it can be observed only the active rectangle

from the arrays are real video-sensor frame while other parts are dummy data which is filtered with the module and generated as an output.

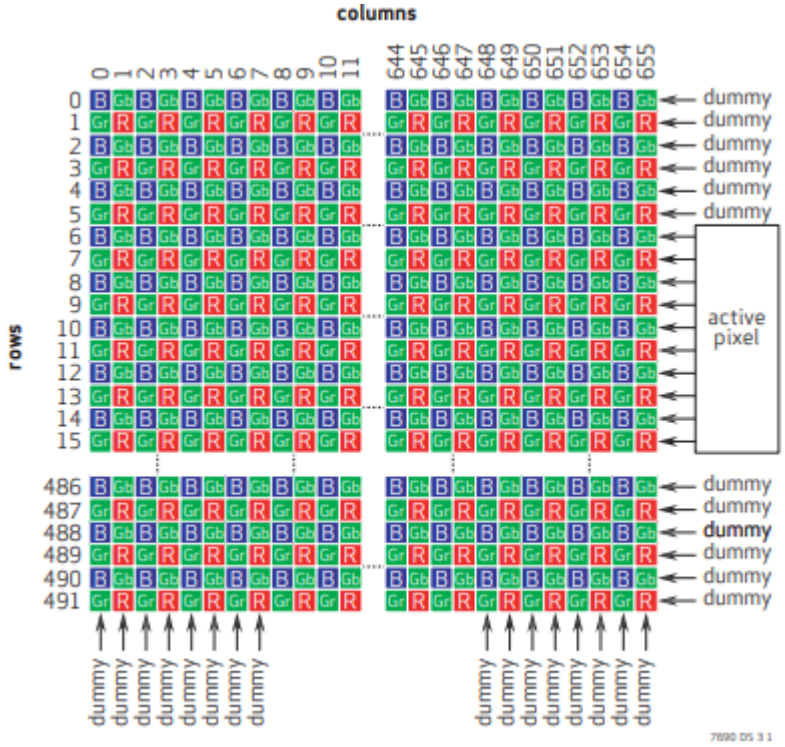


Figure 4.7: Structure of an image frame [87]

The original image arrays consist of 656 columns and 492 rows which are 322752 pixels in total. Only 307200 are the part of the active area of the pixels: 640 columns and 480 rows. Basically, the other area is used for calibration of black level and interpolation.

Principles of operation

The module is implemented with a state machine, which generates new horizontal and vertical synchronization signals and filters the dummy data from the pixels array. Moreover, the input data are 10 bits from where the most significant 8 bits are taken and processed.

In should be noted that the video-sensor input clock is used for processing the input data while the output signal is synchronized through the global clock.

Interface and Symbol

Figure 4.8 displays the symbol of the module and Table 4.2 describes the interface.

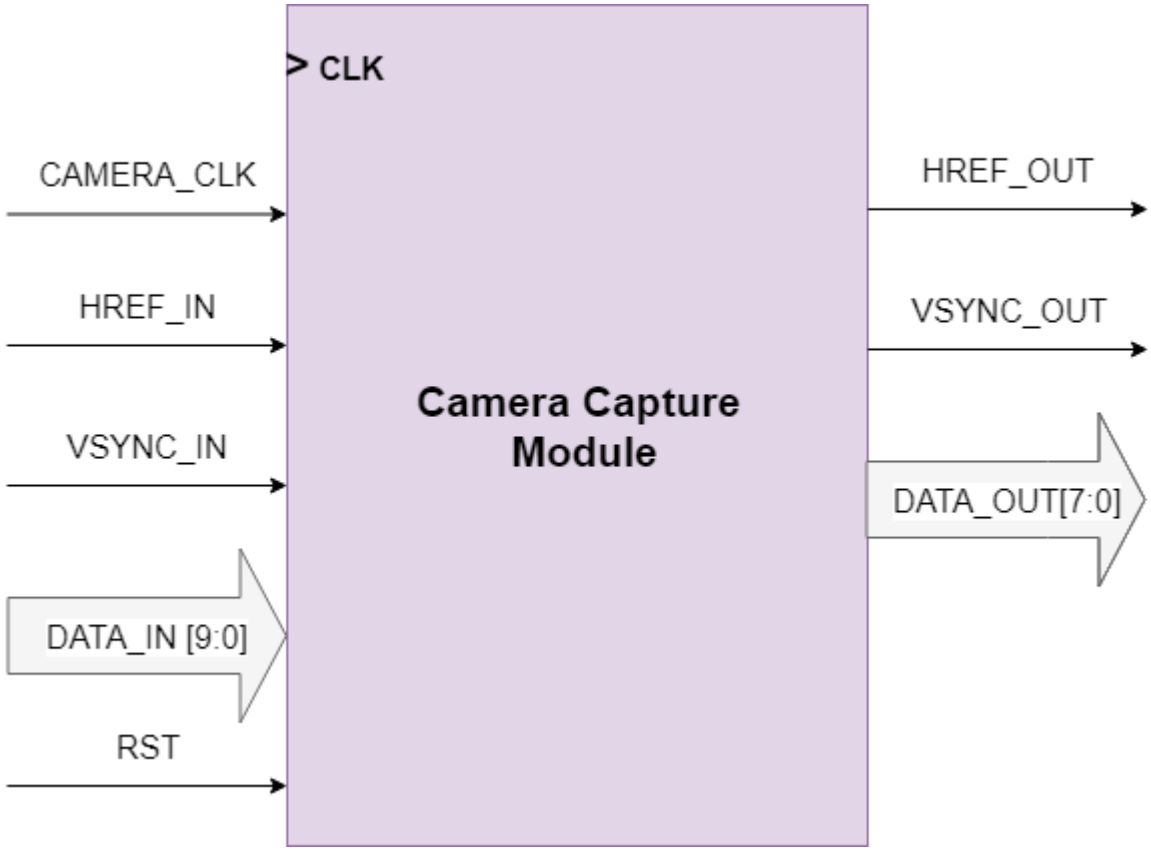


Figure 4.8: Symbol of the Camera Capture Module

Table 4.2: Interface of the Camera Capture module

Name	Type	Mode	Description
CLK	std_logic	IN	input clock of the camera
DATA_IN	std_logic_vector(9 downto 0)	IN	frame pixel of the camera
HREF_IN	std_logic	IN	the start of a line of camera
VSYNC_IN	std_logic	IN	start of a frame of camera 0
RST	std_logic	IN	reset

Table 4.2 (continued)

DATA_OUT	std_logic_vector(7 downto 0)	OUT	Data output
HREF_OUT	std_logic	OUT	start of a line of an output
VSYNC_OUT	std_logic	OUT	start of the output

Average module

Functional Description

The average module takes the input raw RGB image and converts to lower resolution image by averaging the region 4x4. The motivation of averaging is to compress and minimize the image for further processing as a background and identifying the potential moving objects and blobs.

Principles of operation

The image input is taken as the Bayer pattern stream. In order to achieve average for each color two extra lines are reserved as buffers which store previous values for each column. The average is done and on each step by addition and shift operation (division by 2) instead of as a stream instead of adding and dividing by 4 which saves space for each pixel. The calculation is presented for each color channel respectively:

$$R_{avg} = (((R_1 + R_2)/2 + R_3)/2)/2$$
$$G_{avg} = (((G_1 + G_2)/2 + G_3)/2)/2$$
$$B_{avg} = (((B_1 + B_2)/2 + B_3)/2)/2$$

Interface and Symbol

The symbol is illustrated in Figure 4.9 and the interface in Table 4.3, as it can be seen new HREF_OUT and VSYNCH_OUT are used as horizontal Pixel clock and the start of the new line as respectively and the clock is kept as global CLK.

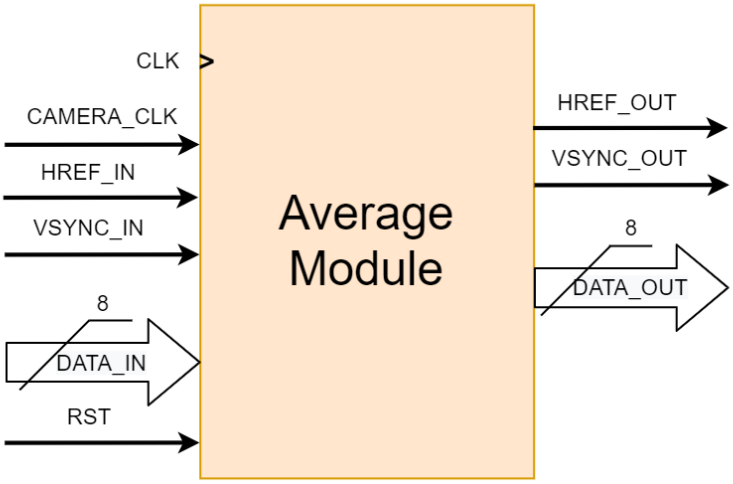


Figure 4.9: Symbol of Average Module

Table 4.3: Interface of Average Module

Name	Type	Mode	Description
CLK	std_logic	IN	input clock of the camera
DATA_IN	std_logic_vector(7 downto 0)	IN	frame pixel of the camera
HREF_IN	std_logic	IN	the start of a line of camera
VSYNC_IN	std_logic	IN	start of a frame of the camera
RST	std_logic	IN	reset
DATA_OUT	std_logic_vector(7 downto 0)	OUT	Data output
HREF_OUT	std_logic	OUT	start of a line of an output
VSYNC_OUT	std_logic	OUT	start of output

Diff module

Functional Description

In order to detect movement, it is essential to calculate the difference between the 2 following frames. The difference is calculated for background compressed image frames for further detection of a blob and for the high definition blob frames for more accurate calculation of centroids. The movement is detected by the threshold for each color channel pixel value. The absolute value of subtraction of pixel values is calculated, compared to the threshold and mapped to a binary output. The module supports threshold scaling mode which enables varying threshold value proportional to the value of the pixels as the noise can be varied to the intensity of the pixel.

Principles of operation

The module takes input from both images as 8-bit pixel values and is controlled by enable signal for disabling calculations on the inactive area of the image. In order to use module flexibly for testing or applying the same logic for different noisy images, the module takes the threshold value as input instead of constantly setting it. The threshold can be dynamically calculated and it slightly differs for blob and background area because of different possible errors and sensitivity importance. Since the output is mapped to binary result the *data_out* is single *std_logic* type and the *busy* signal is used for control and indication of the validity of output.

Interface and Symbol

The interface and symbol are illustrated in Figure 4.10 and Table 4.4 respectively.

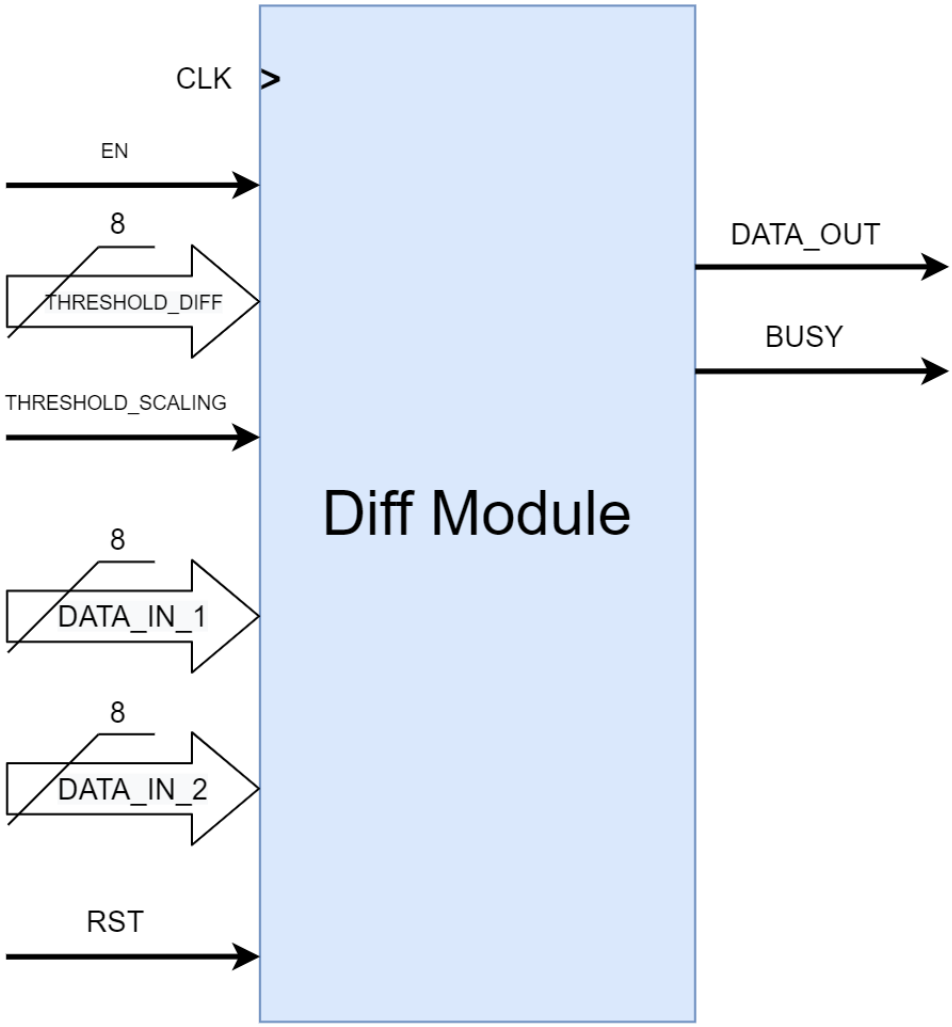


Figure 4.10: Symbol of Diff Module

Table 4.4: Interface of the Diff Module

Name	Type	Mode	Description
CLK	std_logic	IN	input clock of the camera
EN	std_logic	IN	Enable

Table 4.4 (continued)

THRESHOLD_DIFF	std_logic_vector(7 downto 0)	IN	Threshold value 0-255
THRESHOLD_SCALING	std_logic	IN	Threshold dynamic scaling mode enable
DATA_IN_1	std_logic_vector(7 downto 0)	IN	pixel value from frame 1
DATA_IN_2	std_logic_vector(7 downto 0)	IN	pixel value from frame 2 which is following frame
RST	std_logic	IN	reset
DATA_OUT	std_logic	OUT	Binary result output
BUSY	std_logic	OUT	Busy control signal

Blob detection module

Functional Description

This module detects blob on provided movement frame. *Enable*, *HREF* and *VSYNCH* signals are used for control purposes, which are generated for the background frame and aren't the same as the original image. Width of the data pixel value is 1 as an indication of the movement instead of the 8-bit pixel value. The output is left and right margin values which range from 0 to 640 and the top and bottom value which ranged from 0 to 480. The busy signal is used as an indicator if output data is valid or not.

Principles of operation

The purpose of the module is to find the smallest rectangle where the moving object can fit, in order to achieve it, output values are calculated as follows:

- Left value: the extremum minimum column from minimum values for each row where movement is detectable
- Right value: the extremum maximum column from maximum movement values for each row
- Top value: the extremum minimum column value from minimum movement values for each column
- Bottom value: the extremum maximum column value from maximum movement values for each column

Interface and Symbol

Figure 4.11 and Table 4.5 displays symbol and interface respectively.

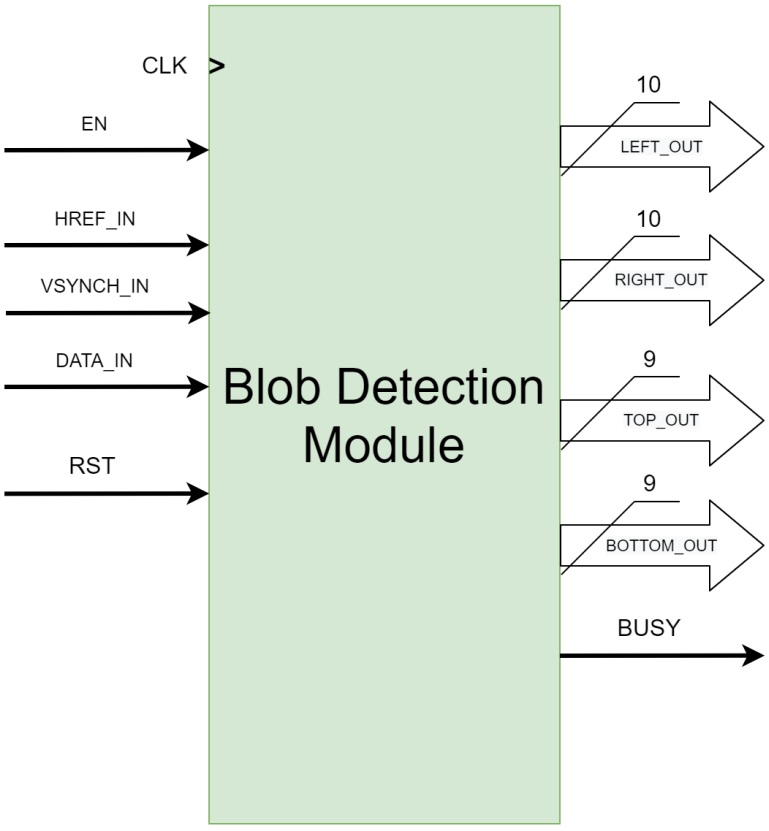


Figure 4.11: Symbol of Blob Detection Module

Table 4.5: Interface of the Blob Detection Module

Name	Type	Mode	Description
CLK	std_logic	IN	input clock of the camera
EN	std_logic	IN	Enable
DATA_IN	std_logic_vector(7 downto 0)	IN	Movement indication for pixel
HREF_IN	std_logic	IN	the start of a line of a frame
VSYNC_IN	std_logic	IN	start of frame
RST	std_logic	IN	reset
LEFT_OUT	std_logic_vector(9 downto 0)	OUT	Left margin value
RIGHT_OUT	std_logic_vector(9 downto 0)	OUT	Right margin value
TOP_OUT	std_logic_vector(8 downto 0)	OUT	Top margin value
BOTTOM_OUT	std_logic_vector(8 downto 0)	OUT	Bottom margin value
BUSY	std_logic	OUT	Busy control signal

Centroid Calculation module

Functional Description

This module calculates the centroids from the given high definition blob image, adds it a padding of left and top value to horizontal and vertical value accordingly and outputs results.

Principles of operation

A horizontal centroid is calculated by finding the average of determined movement range. The same method is used for determination vertical centroid. It should be noted that horizontal centroid is enough for disparity and distance calculation however it is calculated for further possible processing purposes such as additional verification. The centroid calculation algorithm is described in the methodology section in details.

Interface and Symbol

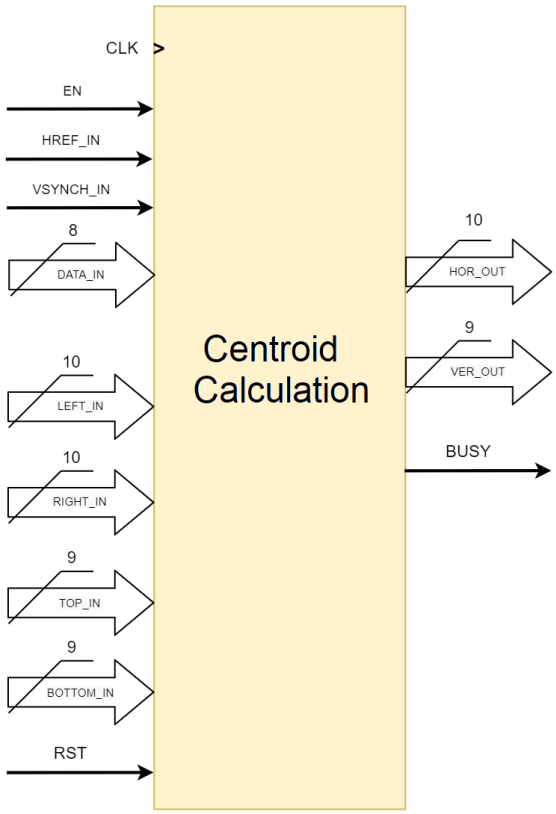


Figure 4.12: Symbol of Centroid Calculation Module

Table 4.6: Interface of the Centroid Calculation Module

Name	Type	Mode	Description
CLK	std_logic	IN	input clock of the camera
EN	std_logic	IN	Enable
DATA_IN	std_logic	IN	Pixel value of blob
HREF_IN	std_logic	IN	the start of a line of a frame
VSYNC_IN	std_logic	IN	start of frame
RST	std_logic	IN	reset
LEFT_IN	std_logic_vector(9 downto 0)	IN	BLOB Left margin value
RIGHT_IN	std_logic_vector(9 downto 0)	IN	BLOB Right margin value
TOP_IN	std_logic_vector(8 downto 0)	IN	BLOB Top margin value
BOTTOM_IN	std_logic_vector(8 downto 0)	IN	BLOB Bottom margin value
HOR_OUT	std_logic_vector(9 downto 0)	OUT	Horizontal value of the centroid
VER_OUT	std_logic_vector(8 downto 0)	OUT	Vertical value of the centroid
BUSY	std_logic	OUT	Busy control signal

Blob validation module

Functional Description

The function of the module is to validate the blobs in order to verify that the same area or objects are tracked from the different video-sensors.

Principles of operation

The input values are coordinates provided from three video-sensors. Firstly, the dimensions are compared with an acceptable error value and, secondly, if the blob is previously tracked which is indicated by comparing previous locations, the vectors of movement is calculated and compared to each other. If the new values pass the verification steps, they are outputted, or the last validated values are kept as an output. It should be emphasized that validation signals control if the object is seen and detected to the video-sensor and the verification is chosen accordingly to it.

Interface and Symbol

Figure 4.13 illustrates the symbol and description of each input and output values are shown in Table 4.7.

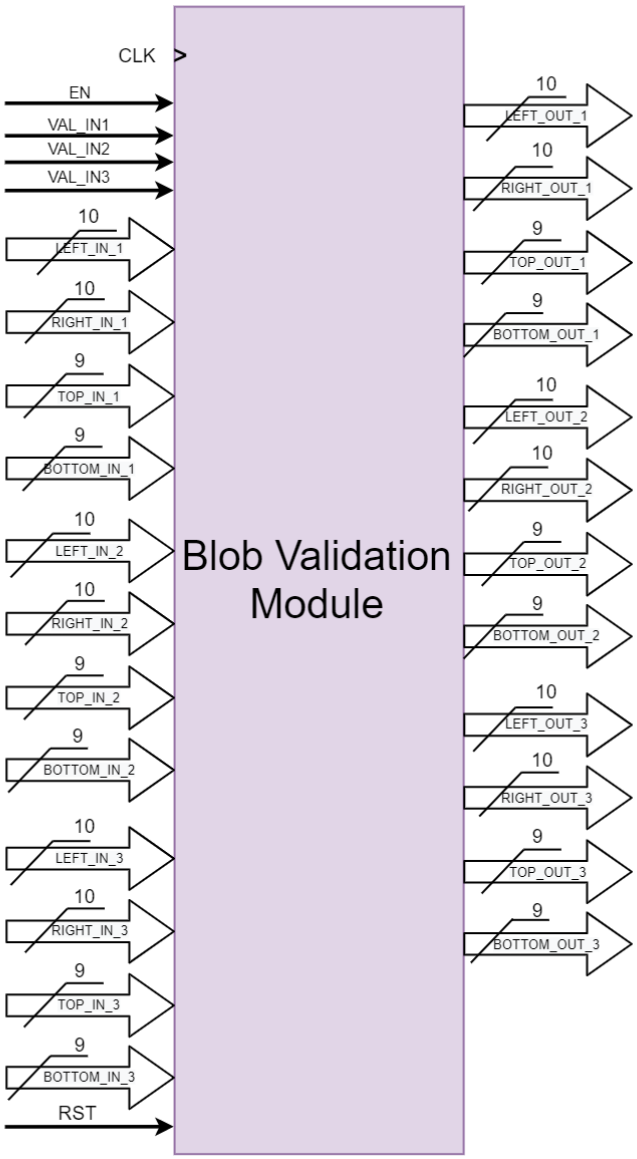


Figure 4.13: Symbol of Blob Validator Module

Table 4.7: Interface of the Blob Validator Module

Name	Type	Mode	Description
CLK	std_logic	IN	input clock of the camera
EN	std_logic	IN	Enable
VAL1	std_logic	IN	Valid result indicator from camera 1
VAL2	std_logic	IN	Valid result indicator from camera 2

Table 4.7 (continued)

VAL3	std_logic	IN	Valid result indicator from camera 3
RST	std_logic	IN	reset
LEFT_IN_1	std_logic_vector(9 downto 0)	IN	BLOB Left margin value from camera 1
RIGHT_IN_1	std_logic_vector(9 downto 0)	IN	BLOB Right margin value from camera 1
TOP_IN_1	std_logic_vector(8 downto 0)	IN	BLOB Top margin value from camera 1
BOTTOM_IN_1	std_logic_vector(8 downto 0)	IN	BLOB Bottom margin value from camera 1
LEFT_IN_2	std_logic_vector(9 downto 0)	IN	BLOB Left margin value from camera 2
RIGHT_IN_2	std_logic_vector(9 downto 0)	IN	BLOB Right margin value from camera 2
TOP_IN_2	std_logic_vector(8 downto 0)	IN	BLOB Top margin value from camera 2
BOTTOM_IN_2	std_logic_vector(8 downto 0)	IN	BLOB Bottom margin value from camera 2
LEFT_IN_3	std_logic_vector(9 downto 0)	IN	BLOB Left margin value from camera 3
RIGHT_IN_3	std_logic_vector(9 downto 0)	IN	BLOB Right margin value from camera 3
TOP_IN_3	std_logic_vector(8 downto 0)	IN	BLOB Top margin value from camera 3
BOTTOM_IN_3	std_logic_vector(8 downto 0)	IN	BLOB Bottom margin value from camera 3
LEFT_OUT_1	std_logic_vector(9 downto 0)	OUT	BLOB Left margin output value from camera 1
RIGHT_OUT_1	std_logic_vector(9 downto 0)	OUT	BLOB Right margin output value from camera 1
TOP_OUT_1	std_logic_vector(8 downto 0)	OUT	BLOB Top margin output value from camera 1
BOTTOM_OUT_1	std_logic_vector(8 downto 0)	OUT	BLOB Bottom margin output value from camera 1
LEFT_OUT_2	std_logic_vector(9 downto 0)	OUT	BLOB Left margin output value from camera 2
RIGHT_OUT_2	std_logic_vector(9 downto 0)	OUT	BLOB Right margin output value from camera 2
TOP_OUT_2	std_logic_vector(8 downto 0)	OUT	BLOB Top margin output value from camera 2
BOTTOM_OUT_2	std_logic_vector(8 downto 0)	OUT	BLOB Bottom margin output value from camera 2
LEFT_OUT_3	std_logic_vector(9 downto 0)	OUT	BLOB Left margin output value from camera 3
RIGHT_OUT_3	std_logic_vector(9 downto 0)	OUT	BLOB Right margin output value from camera 3
TOP_OUT_3	std_logic_vector(8 downto 0)	OUT	BLOB Top margin output value from camera 3
BOTTOM_OUT_3	std_logic_vector(8 downto 0)	OUT	BLOB Bottom margin output value from camera 3

Disparity Calculation module

Functional Description

This module calculates disparity from the horizontal centroid values of the video-sensor

Principles of operation

CEL_CAMERA controls the validity of the video-sensors as well selection of it according to the position and adaptation of video-sensor operation. The disparity is calculated as follows:

$$disparity = abs(centroid_i - centroid_j)$$

Firstly, the centroids are subtracted and then the absolute value is taken.

Interface and Symbol

The interface is described in Table 4.8 and the symbol is illustrated in Figure 4.14.

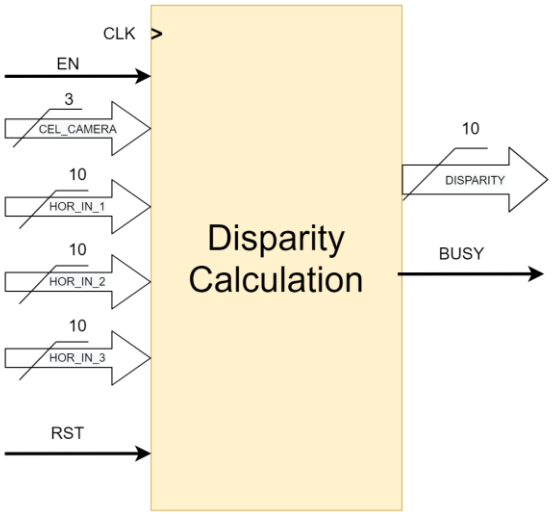


Figure 4.14: Symbol of Disparity Calculation Module

Table 4.8: Interface of the Disparity Calculation Module

Name	Type	Mode	Description
CLK	std_logic	IN	input clock of the camera
EN	std_logic	IN	Enable
CEL_CAMERA	std_logic_vector(2 downto 0)	IN	Selection of the camera valid results

Table 4.8 (continued)

HOR_IN_1	std_logic_vector(9 downto 0)	IN	Horizontal value of the centroid from camera 1
HOR_IN_2	std_logic_vector(9 downto 0)	IN	Horizontal value of the centroid from camera 2
HOR_IN_3	std_logic_vector(9 downto 0)	IN	Horizontal value of the centroid from camera 3
RST	std_logic	IN	reset
DISPARITY	std_logic_vector(9 downto 0)	OUT	Calculated disparity result
BUSY	std_logic	OUT	Busy control signal

Depth Determination module

Functional Description

This module calculates the distance according to the disparity. The *cel* value indicates a combination of from which video-sensors were centroid and disparity are obtained in order to know the base distance between video-sensors.

Principles of operation

Since the relation between disparity and distance is inverse proportional and it requires intensive computation, it's implemented with lookup tables. This approach makes it possible to define values based on the application and specification as well as it is useful for debugging and testing purposes. Two types of look-up table which maps 10-bit input value to 10-bit output value are predefined and filled. Two types are chosen because of different baseline on the video-sensors (closer or and further cameras). Single BRAM (4 kilobytes in size) is utilized for the solution.

Interface and Symbol

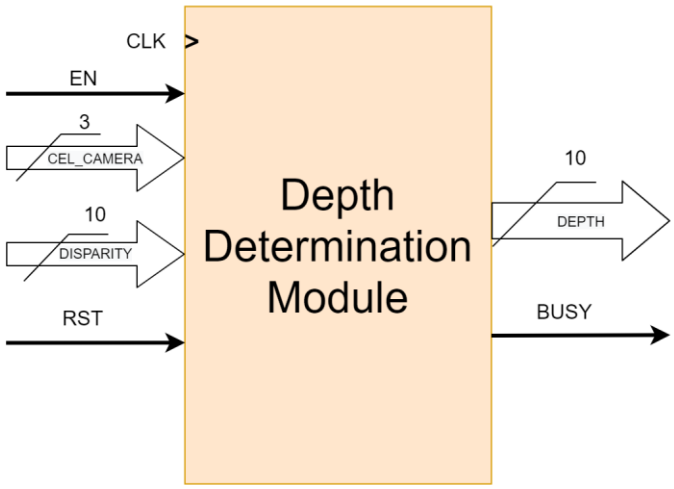


Figure 4.15: Symbol of Depth Determination Module

Table 4.9: Interface of the Depth Determination Module

Name	Type	Mode	Description
CLK	std_logic	IN	input clock of the camera
EN	std_logic	IN	Enable
CEL_CAMERA	std_logic_vector(2 downto 0)	IN	Selection of the camera valid results
DISPARITY	std_logic_vector(9 downto 0)	IN	Disparity value
RST	std_logic	IN	reset
DEPTH	std_logic_vector(9 downto 0)	OUT	Depth result value
BUSY	std_logic	OUT	Busy control signal

Output Frame Generator module

Functional Description

This module takes the input images from video-sensors with the 3D coordinate value and generates output HDMI frames for visualization results. It should be noted that since the visualized image can vary by the configuration and can combine blob, background, movement frame or original image from different video-sensors, the module outputs the address being displayed and the custom pixel RGB value is provided as input and determined externally.

Principles of operation

The module takes RGB input value, custom configuration retrieved from ARM and generates YCbCr 1080x720 HDMI frame. For the conversion from RGB to YCbCr internal IP is utilized provided by Xilinx.

Along with a camera image which can be represented with or without blob, movement and from different video-sensors and scaling, the trajectory grid is drawn, and application-specific methods are applied depending on the configuration.

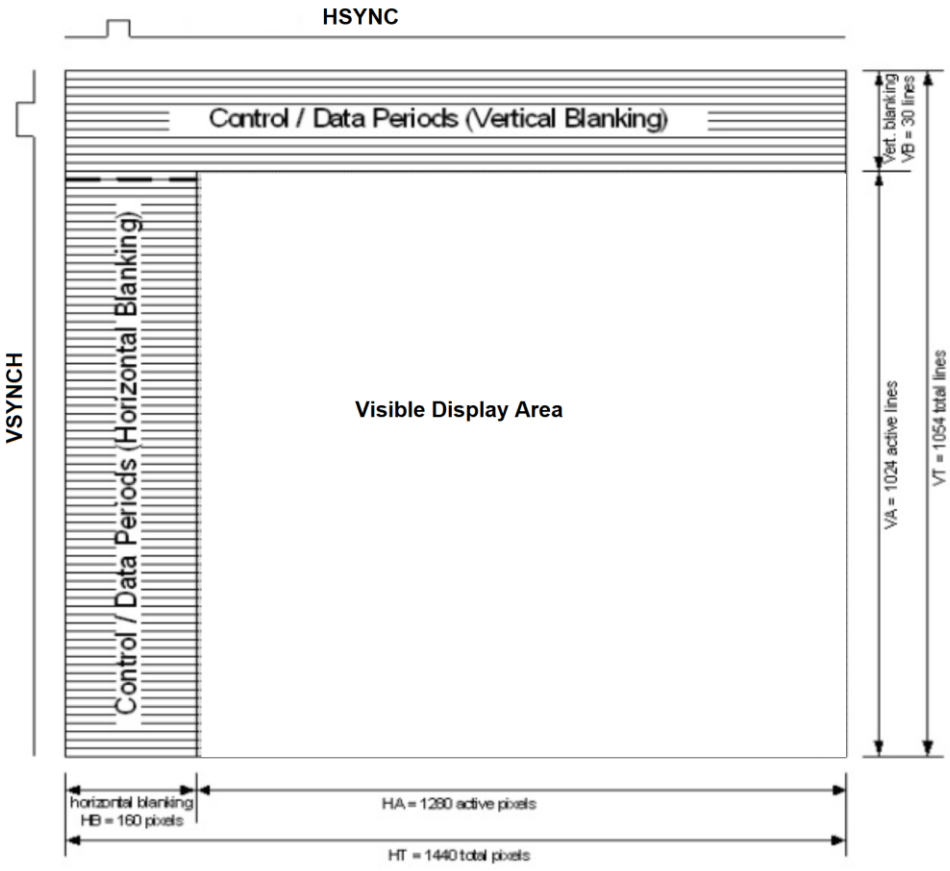


Figure 4.16: HDMI frame structure

The modules generate full HDMI frames filled with blanking and display pixels which are illustrated in Figure 4.16 [89].

Interface and Symbol

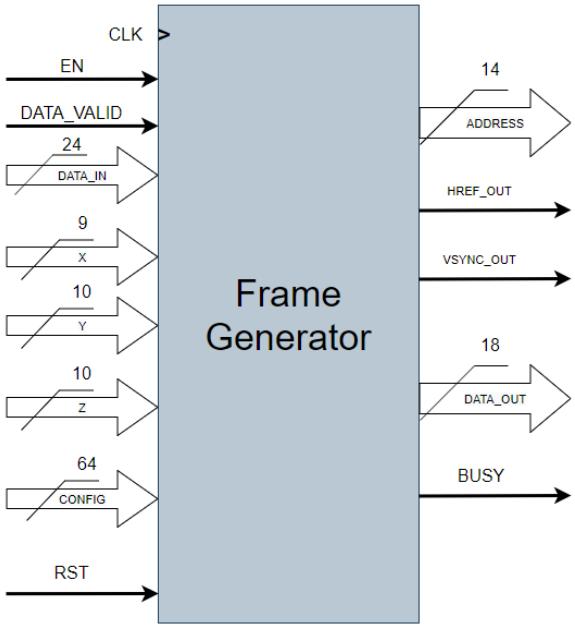


Figure 4.17: Symbol of Frame Generator Module

Table 4.10: Interface of the Frame Generator Module

Name	Type	Mode	Description
CLK	std_logic	IN	input clock of the camera
EN	std_logic	IN	Enable
DATA_IN	std_logic_vector(23 downto 0)	IN	frame pixel
DATA_VALID	std_logic	IN	If the data is valid
X	std_logic_vector(8 downto 0)	IN	X coordinate of object
Y	std_logic_vector(9 downto 0)	IN	Y coordinate of object

Table 4.10 (continued)

Z	std_logic_vector(9 downto 0)	IN	Z coordinate of object
CONFIG	std_logic_vector(63 downto 0)	IN	Custom configuration and data
RST	std_logic	IN	reset
ADDRESS	std_logic_vector(13 downto 0)	OUT	Requested Address of the output
DATA_OUT	std_logic_vector(17 downto 0)	OUT	Data output
HREF_OUT	std_logic	OUT	start of a line of an output
VSYNC_OUT	std_logic	OUT	start of the output
Data Enable	std_logic	OUT	Data Enable signal for output

Modules of Peripheral Vision

Functional Description

Peripheral Vision Component Performs the following functions:

1. Receives frames from 3 video-sensors
2. Saves each color channel to the corresponding BRAM
    a. Keeps the middle video-sensor as the original input
    b. Crops side camera inputs
3. Rectifies frames and merges them
4. Generates output Frame and Displays Result

The symbol of the peripheral vision component shares the same input and output signals as stereo-vision and it is displayed in Figure 4.18.

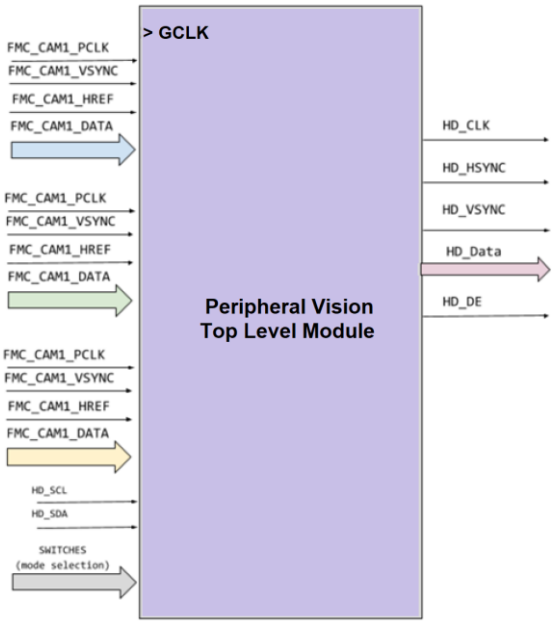


Figure 4.18: Symbol of Peripheral Vision Component

Camera Capture Module

Camera Capture Module is the same as defined for the stereo vision module.

Data Acquisition

Principles of operation

After converting inputs through a camera capture module, data is saved in BRAMs. It should be emphasized that the pixel array of input is arranged according to the Bayer pattern. Each color is saved separately to the corresponding BRAM. It should be noted that while blue and red color is directly saved, green color data is averaged as it is proposed in the Bayer pattern. It should be noted that since the side video-sensor inputs are cropped only half of the data are saved in the memory. In addition, the cropping is performed with a custom offset which is adjustable through the switches of the top-level module input.

In total 115 BRAMs are utilized for saving the given frame.
The calculations are shown below.

The total number per color per camera frame is calculated as:

$$(480 * 640)/4 \ Bytes = 76800 \ Bytes$$

Accordingly, total BRAM number per camera per color is:

$$76800 \ Bytes = \lceil \ 76800/(4 * 1024) \rceil \ BRAM = 19 \ BRAM$$

Total number of BRAMs utilized per camera is:

$$19 * 3 \ BRAMs/Camera = 57 \ BRAMs/Camera$$

Since the side cameras half data are cropped the number of BRAMs are reduced as:

$$\lceil 57/2 \rceil \ \ BRAMs/Side \ Cameras = 29 \ BRAMs/Side \ Cameras$$

As a sum, the total number of BRAM for all camera images is:

$$57 + 2 * 29 = 115 \ BRAMs \ [TOTAL]$$

Block Diagram of Components

The frame data acquisition process operates according to the Bayer pattern. It should be noted that this is a part of the top-level module as a separate process for each video-sensor instead of the module. The diagram of data acquisition is shown in Figure 4.19.

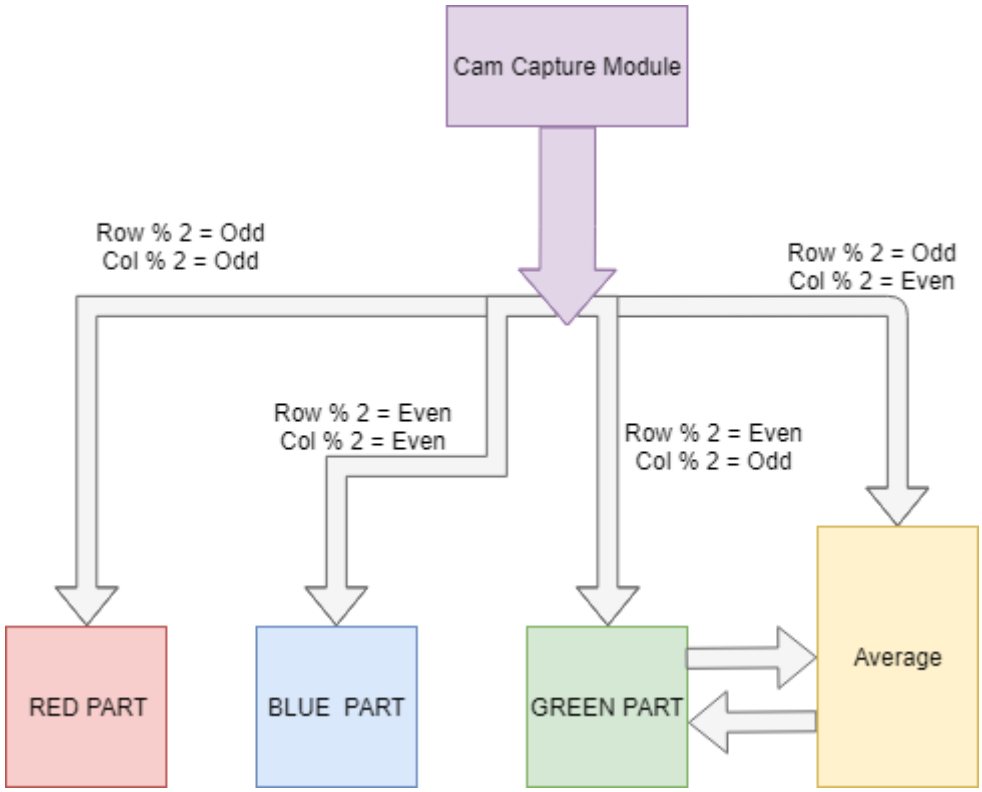


Figure 4.19: Frame Data Acquisition Merging

Principles of operation

The merging of the frames is done by mapping output addressed to the corresponding addresses of the appropriate BRAMs. It should be noted that the rectification offset of each side video-sensor is taking place in the calculation of the BRAM address as well.

The frames are cropped horizontally with a maximum of the side cameras rectification number which is adjustable through the switches of the top-level module.

The generated output frame is in HD resolution (1080x720), where each pixel represents 24 bits RGB value (8 bits for each color). Since the saved data frames resolutions are smaller each pixel is 4 times repeated (2 times per row, 2 times per column). Accordingly, the address range from (1080x720) is mapped to (720x480). The columns 0-180 is used for left-side camera cropped frame. Columns 180-540 are used for middle camera frame without modification. Columns 540-720 are similarly used for right camera cropped frame.

Block Diagram of Components

Figure 4.20 illustrates the process of mapping the output address to the corresponding BRAM and merging the frames. It should be noted that similarly to data acquisition this is represented as a process instead of a separate module.

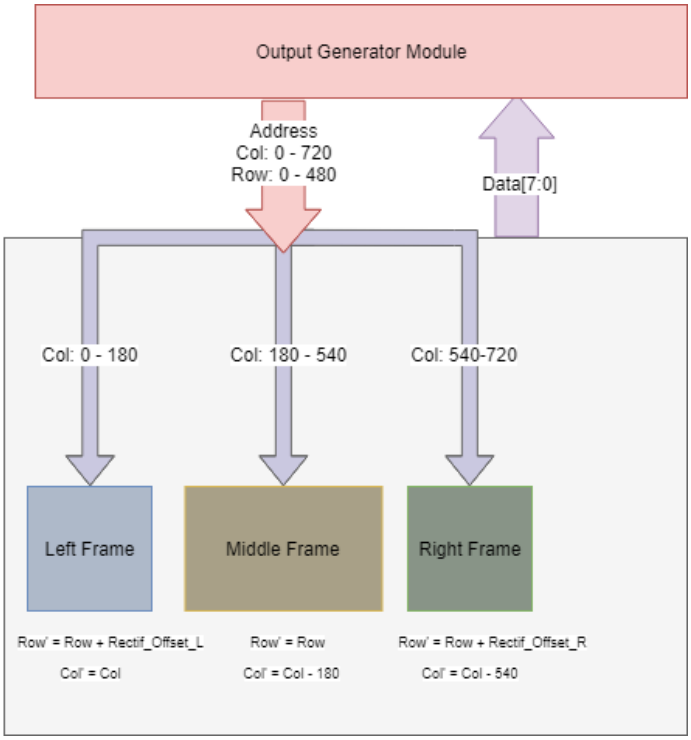


Figure 4.20: Merging Block Diagram

Output Frame Generator Module

Functional Description

The output frame generator module is similar to the stereo component output generator module and takes the inputs of the merged frame as 24 bits pixel (RGB) and generates1080x720 resolution frame for displaying to a monitor via HDMI port.

Principles of operation

The module requests addresses of RGB data as an input to the proper frame. It takes 24 bits data with validated the *data valid* signal and converts to 18 bits pixel data frame as a final result. Every time a new address is requested, the module waits for data valid signal and after data is registered properly, the new address as a request is generated. Moreover, it should be noted that the module enables to add any custom visual data for testing or clarification purposes.

Interface and Symbol

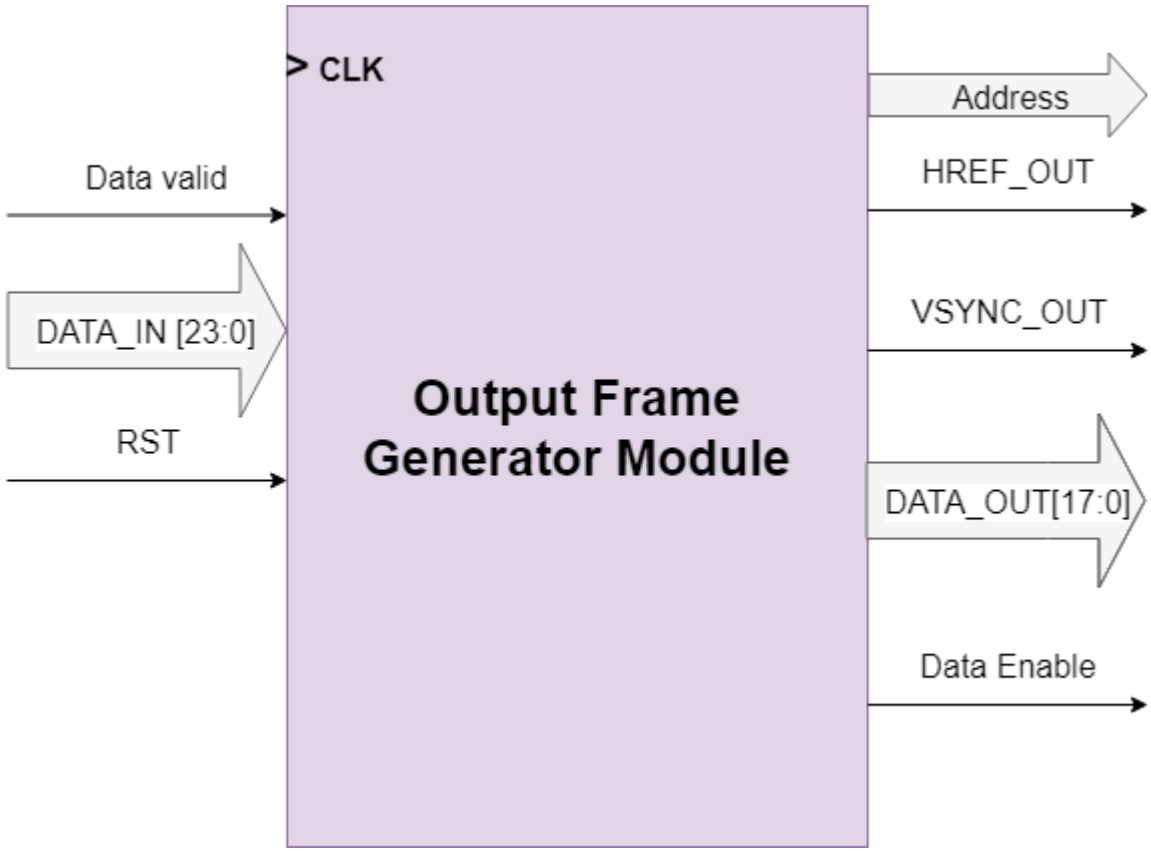


Figure 4.21: Symbol of Output Frame Generator Module

Table 4.11: Interface of the Frame Generator Module

Name	Type	Mode	Description
CLK	std_logic	IN	input clock of the camera
DATA_IN	std_logic_vector(23 downto 0)	IN	frame pixel
DATA_Valid	std_logic	IN	If the data is valid
RST	std_logic	IN	reset
ADDRESS	std_logic_vector(13 downto 0)	OUT	Requested Address of the output
DATA_OUT	std_logic_vector(23 downto 0)	OUT	Data output
HREF_OUT	std_logic	OUT	start of a line of an output
VSYNC_OUT	std_logic	OUT	start of the output
Data Enable	std_logic	OUT	Data Enable signal for output

Block Design and IPs

It should be emphasized that most of the modules, processes and signal operations are implemented with minimizing utilization of external IP (intellectual property) cores, however, clock generator IP for generating 74.25 MHz clocks of HDMI frames is used and illustrated in Figure 4.22.

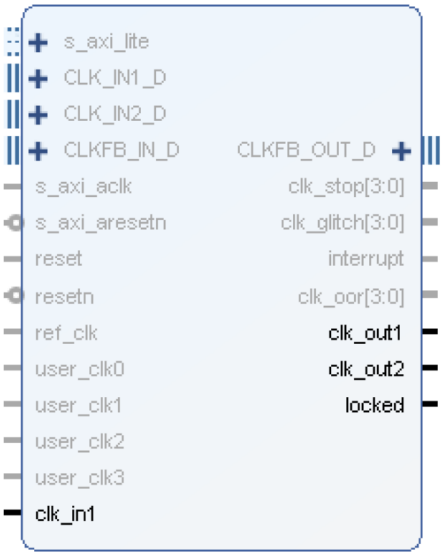


Figure 4.22: 74.25 MHz Clock generator

In addition, block rams are utilized for storing background and blob image frames with the latency of a single clock cycle. The IP is illustrated in Figure 4.23.

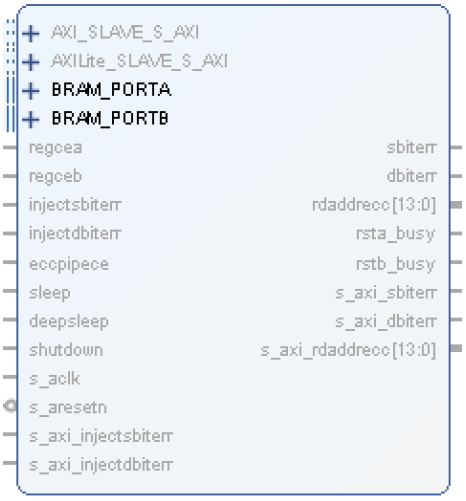


Figure 4.23: BRAM IP core

The design integrates ARM and FPGA components. ARM is utilized for the processing stages which aren't limited in time, aren't processed per pixel and are relatively harder to implement inside FPGA. They are mainly for customization purposes and user interface interaction. The communication between ARM and FPGA is done through Advanced eXtensible Interface (AXI) which is part of Arm Advanced Microcontroller Bus Architecture (AMBA) and based on master-slave communication. Figure 4.24 illustrates a high-level block design.

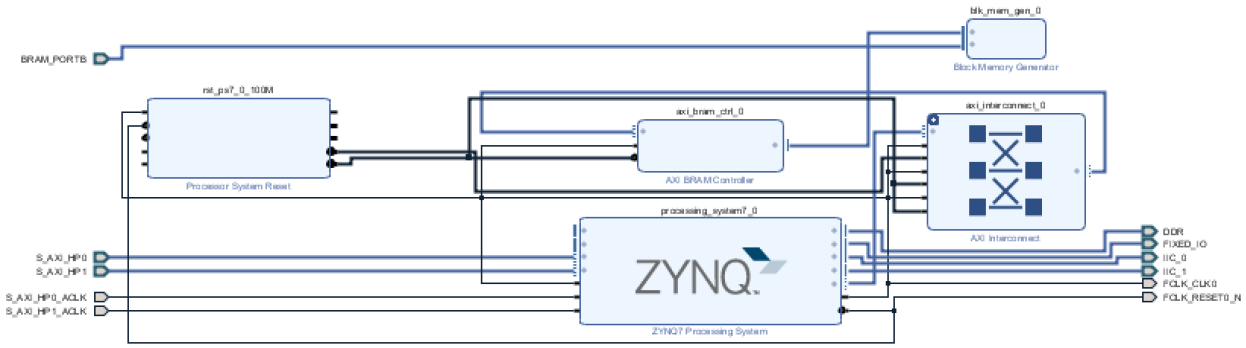


Figure 4.24: High-level Block Design

Software implementation

As it was mentioned ARM part is utilized for the processes which aren't limited in time and are utilized for each frame, such as user interaction, debugging such as logging through Universal Asynchronous Receiver/Transmitter (UART) and testing purposes.

The operation and implementation in software part involve:

- User interaction such as defining custom actions and events to the grid for application-specific requirements and usage
- Smoothing the parameters. As it was illustrated in Figure 3.11 before, the Kalman filter is utilized for smoothing results and fitting in the curve. Four Kalman filters are applied with different parameters for measurement and covariance noise to horizontal centroids, disparity and depth.

The pseudocode is described below as a lightweight implementation of the smoothing filter.

```
x_temp_estimated_i = x_estimated_prev_i; // save temporal measurement
P_temp_i = P_prev_i + Q_i; // save temporal  prev P matrix
x_estimated_i = x_temp_estimated_i + K_i * (z_measured_i - x_temp_estimated_i); //calculation  the Kalman gain
P_i = (1- K_i) * P_temp_i; //correction the value
P_prev_i = P_i; //update  prev values
x_estimated_prev_i = x_estimated_i;
K_i= P_temp_i * (1.0f/(P_temp_i + R_i));
result = x_estimated_i; /// obtain result
```

Chapter 5

5. Results and Evaluation

In this chapter, implementation and design are evaluated in different ways which are discussed in details and the results as reports of resource utilization and visual representation are presented.

5.1 Testing Methodology

Testing machine vision applications are always challenging since it requires verification output from human perception. In order to maximize test coverage and automate verification, several ways of verification are proposed:

Virtual Camera

Virtualization camera means replacing *camCapture* module by mock module with the same interface. The virtual camera module outputs the generated moving simple shapes such as rectangle, circle and others with custom padding with disparity value. Figure 5.1 illustrates the virtual camera example output, where the virtual camera generates rhombus shape as an output which is highlighted in green as blob rectangle and centroid is displayed in blue. In the third image, horizontal disparity value and matched frames are visualized. The virtual camera approach is used for verifying object detection, tracking and disparity calculation and accuracy measurement since the defined trajectory is predetermined and can be identify expected and actual results and compares them.

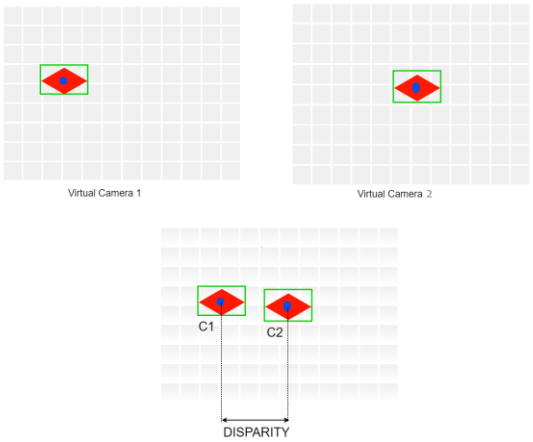


Figure 5.1: Video-sensors test configuration

Camera chip sensor was selected to provide test configuration for visual verification of video-sensors and testing proper operation of camera module. It is achieved by test configuration of the video sensor registers, which enable to overlay color bar (displayed in Figure 5.2) to the input frame with a certain transparency.

Figure 5.2: Overlay Color Bar

Integrated Logic Analyzer

In order to verify the design module in runtime Integrated Logic Analyzer (ILA) [90], which is an IP core supported from Xilinx, is utilized. ILA is a module which can be used for monitoring the signal of each module in the design. It involved various triggers such as Boolean logic, edge transition and etc. It should be noted that it uses the same clock constraints as used in the design since it operates runtime.

The procedure of using ILA involved defining it into design and capture manually or automatically by triggering signals, monitoring and comparing them.

Test dataset

The test data involves values which are retrieved during the processing of frames such as centroids, blob coordinated and dimensions, disparity and distance. They are compared to the prepared test data in order to test the functionality of the module in automation mode. Test data consist of two categories:

1. Manually selected data which includes simple, complex and edge cases of the values
2. Recorded data which is data retrieved from test movement of a person for realistic operation cases

Verifying design modules

The modules are verified through different methods which involve ILA, recording results and compare actual and expected values to test data and visual verification. However, since the design consists of multiple modules in order to maximize the performance of finding faults or identifying errors, the faster approach is proposed, which is similar to a binary search / dichotomy method. The algorithm of testing is the following:

1. Apply initial testing data from a virtual camera and verify results, by comparison, the expected value and actual values captured by ILA or recorded by saving in external memory or logging through UART
2. If there are errors which aren't acceptable, divide the design into two parts and apply the output to the middle layer of the design
    a. If the part indicated is error-free apply the same strategy to another part of the design
    b. If there is error indicated repeat the step 2 and divide design by two layers again

    Note: moving the start and endpoint of testable design can vary (input or output) depending on the modules involved and test data available to the input

The technique is useful for testing and finding the problem in design effective way in logarithmic complexity.

5.2 Results

One of the challenges of the research was to remove environmental video noise in order to calculate 3D vectors in the highest accuracy. While the initial image processing and excluding and thresholding operation on pixels minimize errors, as it was described multi Kalman filter is applied for noise reduction in the final step.

Figure 5.3 and 5.4 illustrate the reduction noise of horizontal centroid based on the recorded real test data. The red curve shows the smoothed result while the raw data is presented in blue points. The period of the frame is 33.3 milliseconds (ms) as the selected video sensors operate 30 frames per second (fps). The same measurement is used for the graphs shown in this section.

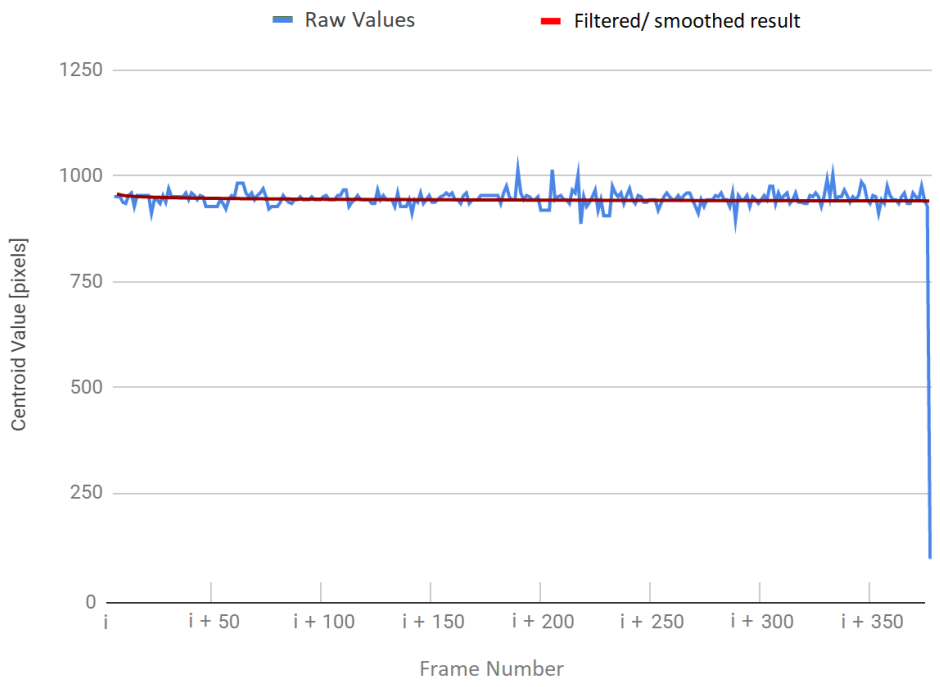


Figure 5.3: Reduction of the centroid noise based on test dataset 1

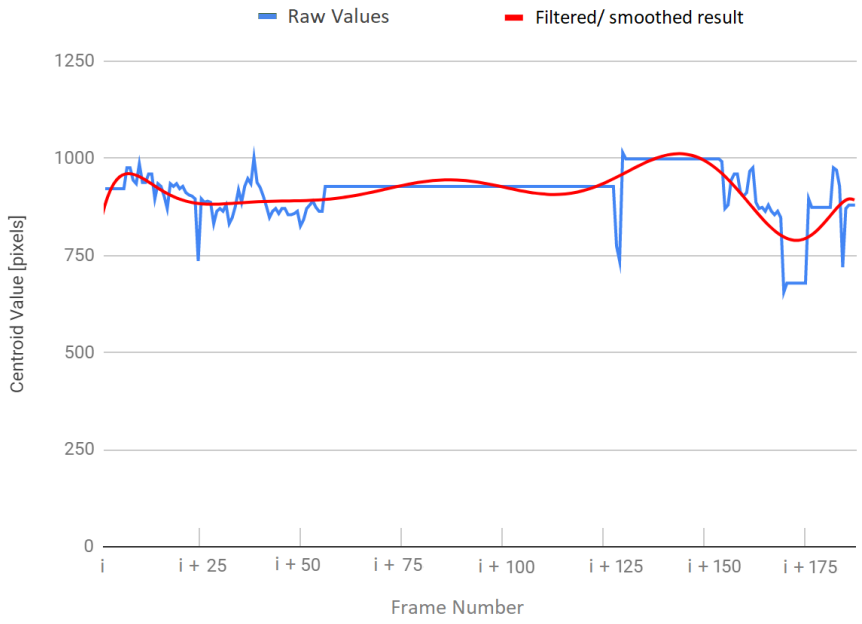


Figure 5.4: Reduction of the centroid noise based on test dataset 2

KF is applied with different covariance and noise matrixes based on the model for the disparity and distance which are illustrated in Figure 5.5 and 5.6 respectively. Note that compared initial

data is without using KF in any layer, while the final smoothed result shows the final results used in all: centroid, disparity and distance pre-smoothing stages. The red curve shows the smoothed result and raw disparity is shown in green, raw distance is presented in purple color.

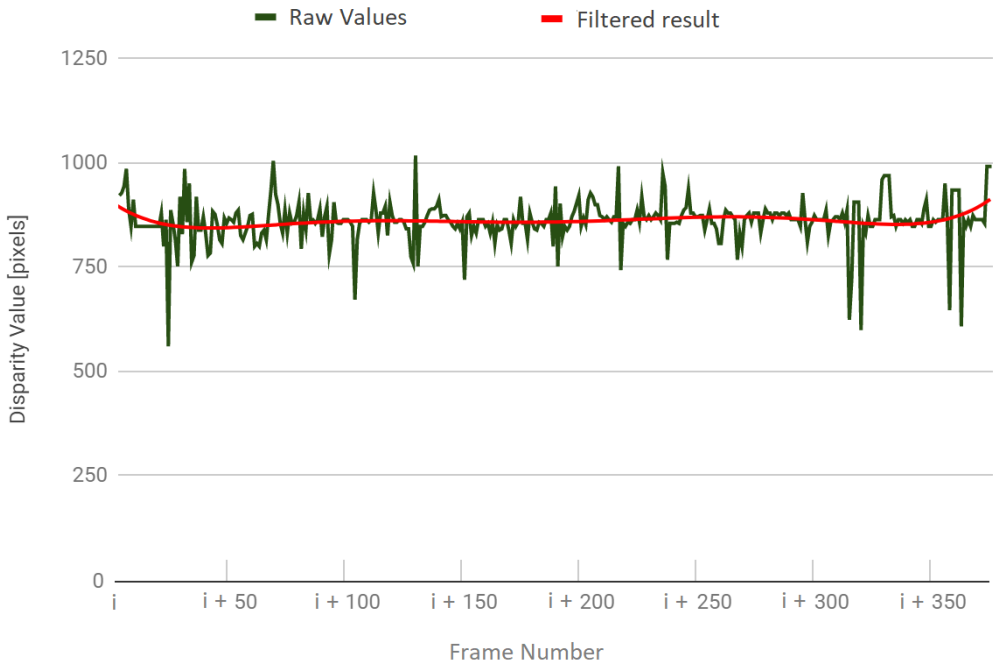


Figure 5.5: Smoothing graph of the disparity

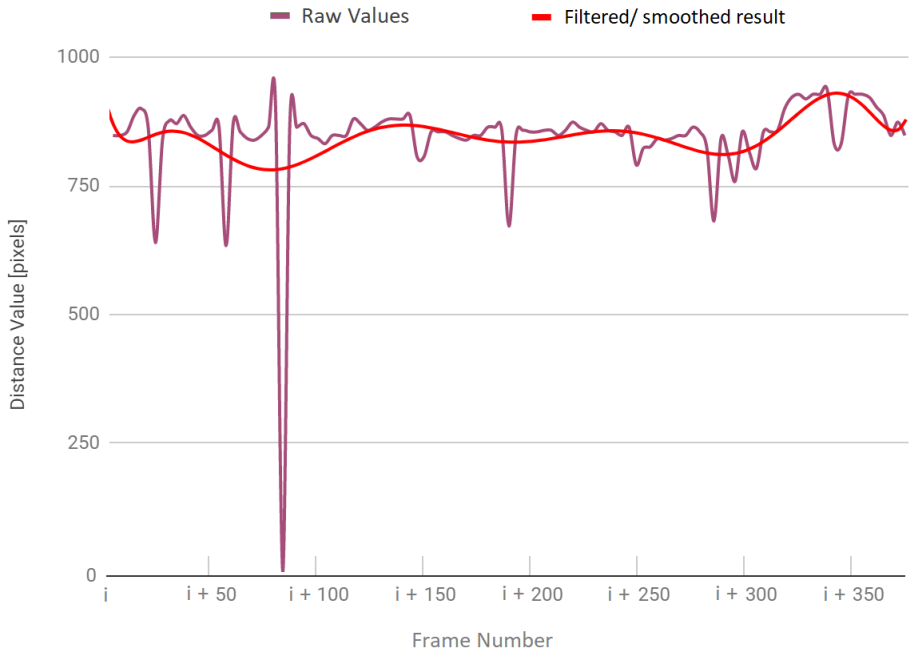


Figure 5.6: Smoothing graph of the distance

Performance metrics and properties

Performance table of the design includes measurement error as Root Mean Square Error (RMSE) and Mean Absolute Error (MAE) of 3D vectors by comparing actual values to real test expected data, the test data is generated from the virtual camera involving object movements by in different direction with 1-10 pixels movement range per frame. Properties of system operation and performance metrics are presented in Table 5.1 and 5.2, respectively.

Table 5.1: Properties of system operation

Property	Value
Number of video-sensors	3
Frames per second operation of video-sensors	30
Input video-sensor frame resolution	640x480
Utilization of Block RAMs (BRAMs)	138
Total On-Chip power	2.327 W

Table 5.2: Performance metrics

Metric	Value
RMSE	0.383 [pixels]
MAE	0.172 [pixels]

RMSE and MAE are chosen because of common usage for measuring accuracy in vision/video applications. The input variables (vector coordinates) are measured in pixel unit precision. The actual values of the estimations, as it can be seen from the table, are less than 1, which give effective and competitive results compared to the standard complex machine vision application based on software implementations (which have high precision but are limited in power, memory and timing resources).

Visual results

Figure 5.7 represents the frames taken from 2 different video sensors at the same time. The yellow rectangle represents the centroid calculated for the detected moving object. The red rectangle shows the blob area, which can be seen is presented in high resolution for higher precision results while the background is averaged and blurry for optimization resources.

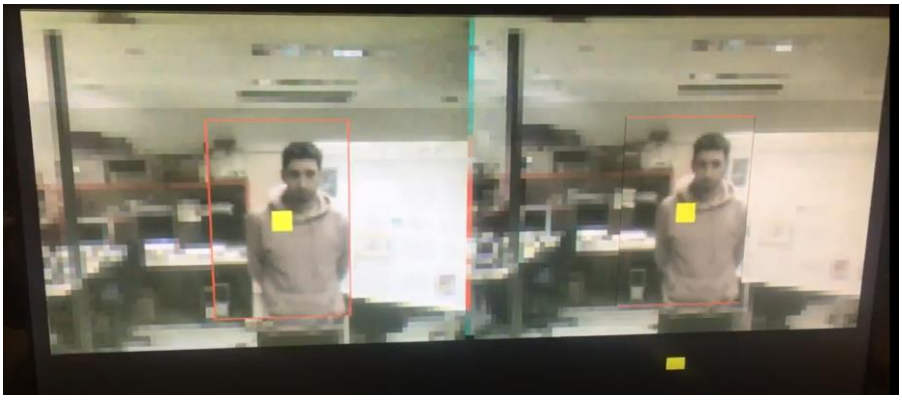
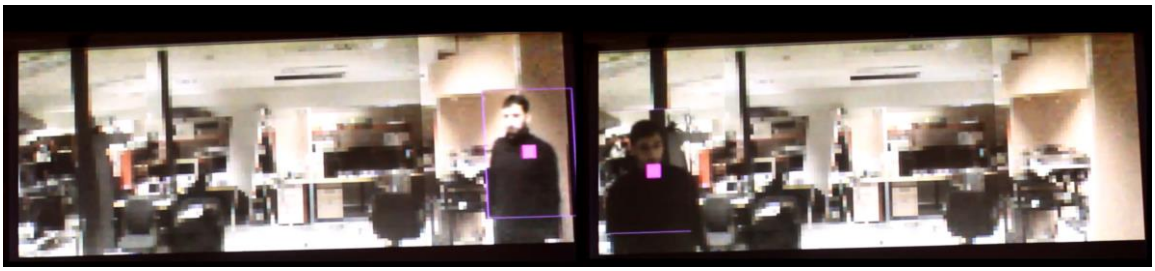
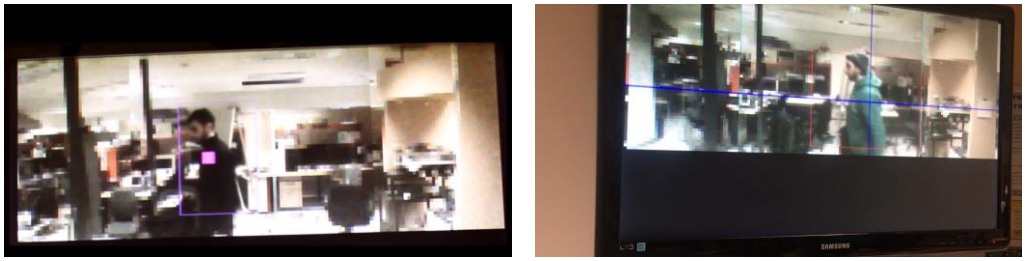


Figure 5.7: Visual images of single object detection with a blob

Figure 5.8 displays the results of stitched images for the semi-panoramic field of view. As it can be seen the merged final image continues and covers ~120-degree fields of view (a, b, c). Figure 5.8 (d) shows the calculation results of centroid which is represented as the intersection of blue lines as horizontal and vertical centroids, respectively.



(a) (b)



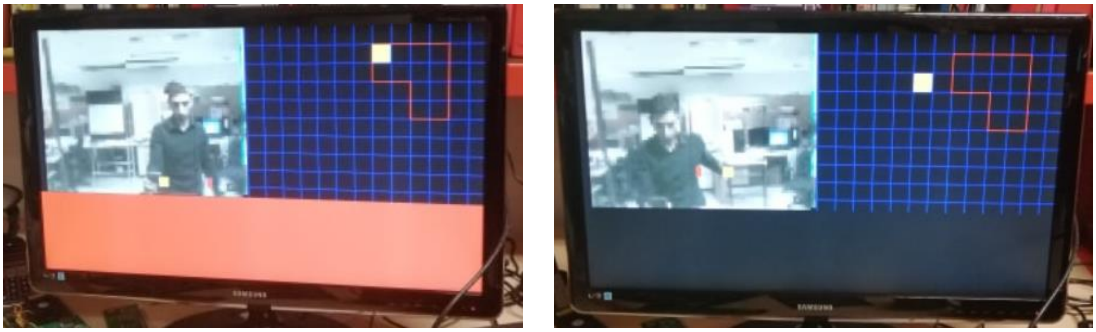
(c) (d)

Figure 5.8: Pictures of the semi-panoramic field of view

Multi objects are similarly processed and tracked in their separate high definition blob areas, represented as different color windows which are shown in Figure 5.9.

Figure 5.9: Visual images of Multi-object tracking

The customization of the designed and implemented system as platform allows developing further applications such as defining the restricted area in 3D and arise events when objects entered inside it. Figure 5.10 represent such alarm-based application, the x-axis shows horizontal centroid while the y-axis represents the distance from the cameras on the pre-defined grids, as it can be seen while the object is inside the restricted location (a) alarm is notifying with flashing light.



(a) (b)

Figure 5.10: Pictures of the custom application

Comparison to existing systems

The concept of attention from the human vision, which is developed in the proposed system, optimizes solution for saving computation resources which give competitive and optimized results. As it was discussed in related works section, there are numerous FPGA-based designs and implementations for the stereo vision systems. Table 5.3 shows the comparison of the properties and resource utilization of the existing approaches.

Table 5.3: Comparison to existing systems

Developed System	Frame size [px]	Disparity range [px]	Moving object detection and tracking	Number of 4KB RAM Blocks	FPS
a Real-Time Stereo Vision System by S. Jin [77]	640 × 480	64	No	322	30/60 +
Stereo Vision with Semi-Global Matching Disparity Estimation by C. Banz [78]	640 × 480	128	No	823	30
Stereo Vision on the PARTS by J. Woodfill [79]	320 x 240	24	No	N/A	42
System developed in the work	640 × 480	480	Yes	138	30/60 +

As it can be observed, other approaches are limited in disparity range as they standardly calculate the disparity map for the whole image instead of the certain object and estimate each possible disparity value per line/pixel which limits the possible variation for disparity. Moreover, they use just two cameras which give a lower field of view and less precision for the disparity for farther object.

The major part of resources for the stereo vision of the related existing approaches are spend on rectification, which is avoided in the system developed in the R&D work by utilizing the idea of detection and tracking moving objects and representing them in the blobs, which already reflects rectification in the coordinates.

The concept of attention to moving objects and using information from the previously processed frames accelerates processing and minimize memory resources comparing to existing systems.

The proposed system operates at 30 FPS and depends on the camera constraints. However, if the video sensor can provide higher FPS, the design can be adjusted to higher operational frequency proportionally keeping the same amount of resources and functionality.

Moreover, the main advantage of the proposed system is real-time moving objects detection and tracking. This feature allows to prioritize them based on the properties and develope further custom applications. Unlikely to other approaches which have limited functionality for moving objects detection and tracking, the proposed approach allows to determine and track multiple objects in real-time.

Summarizing the above, the developed system in this R&D work proposed, implemented and verified the concept of attention to moving object(s) optimizing the resources, power consumption and cost of real-time object tracking video systems.

Chapter 6

6. Conclusion and Future work

6.1 Conclusion

In this research human vision concepts were analyzed and, based on the main principles, FPGA-based embedded vision system was created, implemented and tested. The system utilizes 3 special purpose video-sensors, FPGA platform and embedded hard processing core ARM Cortex-A9. This hybrid architecture makes it able to process image frames and mimic human vision in high performance and minimize the computational and memory resources as well as define GUI for application-specific usage.

The implementation and proposed design involved:

1. Analyses of two main components of human binocular vision: a stereo vision, which calculates the 3D vectors of moving objects and peripheral vision for a wider field of view and monocular distance calculation.
2. Identifying main limitations of traditional approaches and implementations and developing algorithms and methods to solve them
3. Utilization the idea of the blob, which replicates the human attention concept, minimizes resources and optimizes the solution for high processing time and latency reduction
4. Experimental setup with hybrid architecture consisting of the FPGA and embedded ARM with addition of the custom board and multipurpose video chip sensors.
5. Creation the system architecture and solve the problem with determined methodology from human vision and solve the challenges such as smoothing and noise reduction technique which allows convert non-stable raw output values to use in real time vision applications
6. Framework with custom control and configuration for application-specific purpose usage as event-based alarms and notifications

Moreover, the developed system is verified and tested through various steps in defined testing infrastructure with a virtual camera, recorded test datasets, integrated logic analyzer and other methods.

6.2 Future work

The developed system can be improved by adding more features for more flexible and wider area usability or by improving performance. There are various algorithms for merging images, noise reduction, distance determination, which can result in better performance or a more accurate solution. The partial reconfiguration of Zedboard can be utilized for adaptation and flexibility of design. As well the system can be developed on another board with larger memory space or other capability features which can be utilized to speed up the solution.
Additional features which replicate concepts of human vision or can be used in industrial and educational applications can be developed inside of FPGA or in ARM:

- Movement recognition, which can record the history of object movement and analyze if the new object matches the previously known pattern
- Object Detector which can be expanded by addition with more features such as complex classification as face detectors, body shape detectors and etc.
- Landmark detection, which involves locating landmark points around the facial or any general-purpose component, which is nowadays widely used in object recognition and face detection-based application.
- Object Recognition, which supports matching objects with previously known models, which, as an example, can be implemented with algorithms of popular machine learning classifiers
- Text recognition for identifying the characters data and analyzing them

Proposed design with separation of modules and ARM support for customization gives the system flexibility to be expanded easily for further development.

Appendix

Waveforms from Integrated Logic Analyzer

Capture of the functional waveforms for main results (final 3D coordinated and the values of the map) is performed runtime using Integrated Logic Analyzer (ILA) from Xilinx [90] as one of the suggested methodologies for testing. The description and type definition of each signal is presented in the table A.1. The results are compared to the recorded data or with pre-determined values in case of virtual camera. The Figures A.1-A.4 shows sample waveforms for performed verification.

Table A.1: Description of the input signals of the Integrated Logic Analyzer

Name	Description	Type
CLK	Input Clock	std_logic
HOR_CENTROID_2	Horizontal centroid coordinate of the blob from active camera 1	std_logic_vector(13 downto 0)
HOR_CENTROID_1	Horizontal centroid coordinate of the blob from active camera 2	std_logic_vector(13 downto 0)
ROW_OUT	Row number on the trajectory map	std_logic_vector(5 downto 0)
COL_OUT	Column number on the trajectory map	std_logic_vector(5 downto 0)
RIGHT_BLOB_COORD	Movement detected right horizontal coordinate (current) in blob search	std_logic_vector(13 downto 0)
LEFT_BLOB_COORD	Movement detected left horizontal coordinate (current) in blob search	std_logic_vector(13 downto 0)

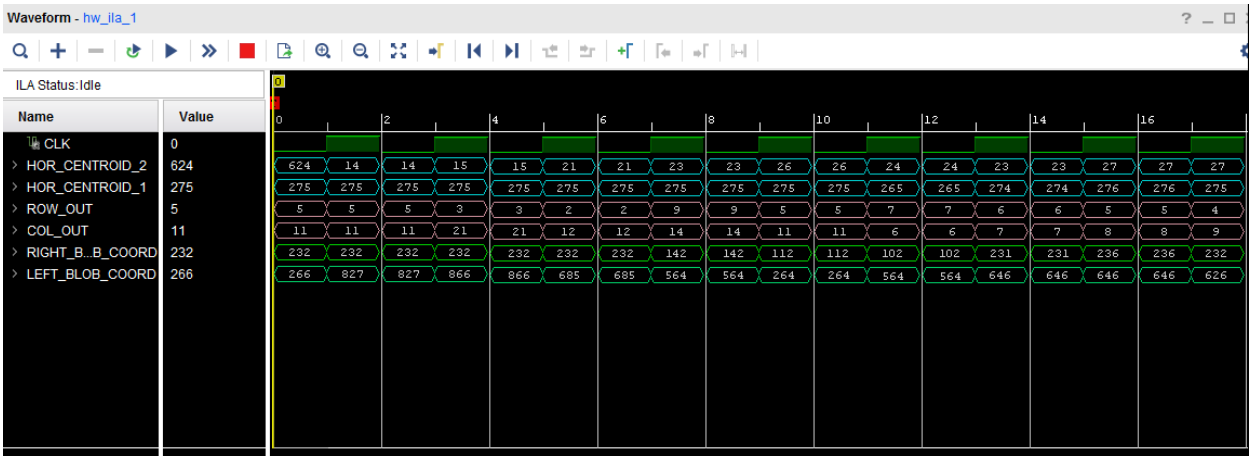


Figure A.1: Waveform for recorded data 1

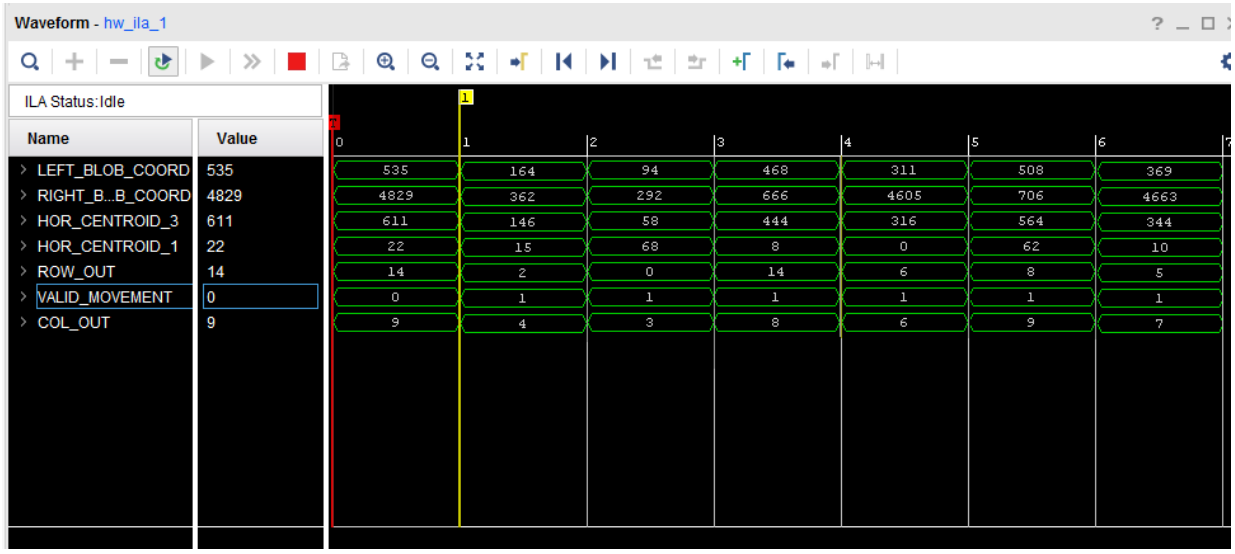


Figure A.2: Waveform for recorded data 2

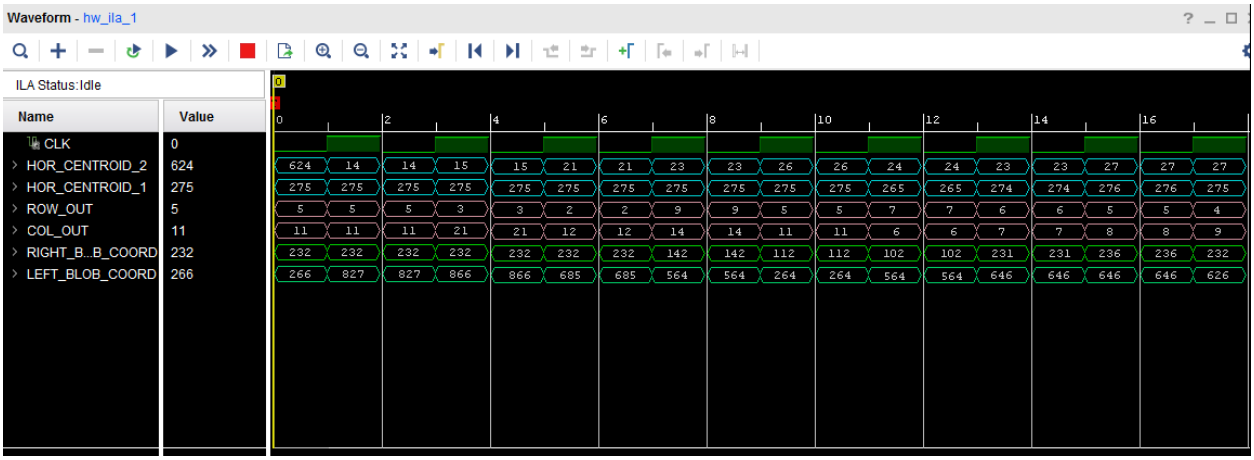


Figure A.3: Waveform for virtual camera output 1

ILA in Figure A.4 is configured for 32 window data depth and 12 window samples for detailed processing.

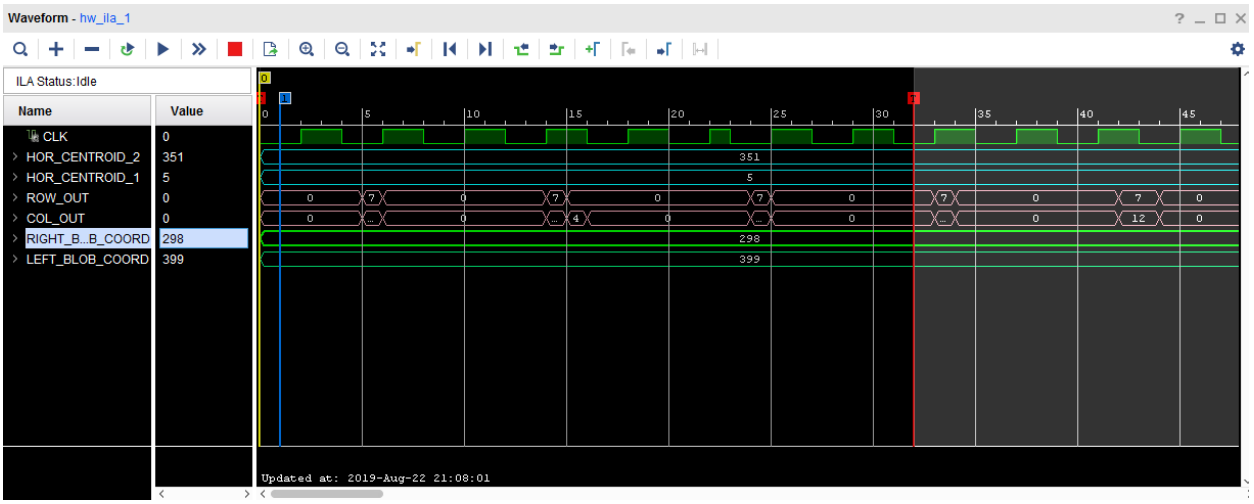


Figure A.4: Waveform for virtual camera output 2

References

[1] T. Newman, “An introduction to eyes and how they work,” *Medical News Today*, 2018. [Online]. Available: https://www.medicalnewstoday.com/articles/320608.php#targetText=Lens%3A%20Once%20light%20has%20traveled,flexible%2C%20making%20focusing%20more%20difficult. [Accessed: 21-Jun-2019].

[2] C. W. Oyster, “Components of the Eye,” in *The Human Eye: Structure and Function*, Sinauer Associates, 1999, pp. 323–701.

[3] C. E. Willoughby, D. Ponzin, S. Ferrari, A. Lobo, K. Landau, and Y. Omidi, “Anatomy and physiology of the human eye: Effects of mucopolysaccharidoses disease on structure and function - a review,” *Clinical and Experimental Ophthalmology*, vol. 38, no. 1. pp. 2–11, 2010.

[4] E. Jackson, “Function and structure of the eye,” *Am. J. Ophthalmol.*, vol. 24, no. 3, pp. 277–281, 1941.

[5] S. Bonaque-González, A. Amigo, and C. Rodríguez-Luna, “Recommendations for post-adaption care of an ocular prosthesis: A review,” *Contact Lens and Anterior Eye*, vol. 38, no. 6. pp. 397–401, 2015.

[6] H. Kobayashi and S. Kohshima, “Unique morphology of the human eye and its adaptive meaning: Comparative studies on external morphology of the primate eye,” *J. Hum. Evol.*, vol. 40, no. 5, pp. 419–435, 2001.

[7] S. Barbero and S. Marcos, “Analysis of the optical field at the human retina from wavefront aberration data,” in *Optics InfoBase Conference Papers*, 2008, vol. 25, no. 9, p. 2280.

[8] M. Fahle, “Why two eyes?,” *Naturwissenschaften*, vol. 74, no. 8, pp. 383–385, 1987.

[9] R. K. Jones and D. N. Lee, “Why two eyes are better than one: The two views of binocular vision,” *J. Exp. Psychol. Hum. Percept. Perform.*, vol. 7, no. 1, pp. 30–40, 1981.

[10] O. Sacks, “Stereo Sue: why two eyes are better than one,” *New Yorker*, vol. 82, no. 18, pp. 64–73, 2006.

[11] R. Dzierwa, “Field of view,” *Appliance*, vol. 54, no. 3, p. 32, 1997.

[12] J. Garding, J. Porrill, J. E. W. Mayhew, and J. P. Frisby, “Stereopsis, vertical disparity and relief transformations,” *Vision Res.*, vol. 35, no. 5, pp. 703–722, 1995.

[13] N. Lazaros, G. C. Sirakoulis, and A. Gasteratos, “Review of stereo vision algorithms: From software to hardware,” *Int. J. Optomechatronics*, vol. 2, no. 4, pp. 435–462, 2008.

[14] J. Wang and M. Fischler, “Visual Similarity, Judgmental Certainty and Stereo Correspondence,” *Proc. ARPA Image Underst. Work.*, pp. 1237–1248, 1998.

[15] S. Negahdaripour, “Epipolar geometry of opti-acoustic stereo imaging,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 29, no. 10, pp. 1776–1788, 2007.

[16] Y. Li, Y. Ruichek, and C. Cappelle, “3D triangulation based extrinsic calibration between a stereo vision system and a LIDAR,” in *IEEE Conference on Intelligent Transportation Systems, Proceedings, ITSC*, 2011, pp. 797–802.

[17] Y. Cui, F. Zhou, Y. Wang, L. Liu, and H. Gao, “Precise calibration of binocular vision system used for vision measurement,” *Opt. Express*, vol. 22, no. 8, p. 9134, 2014.

[18] S. Nefti-Meziani, U. Manzoor, S. Davis, and S. K. Pupala, “3D perception from binocular vision for a low cost humanoid robot NAO,” *Rob. Auton. Syst.*, vol. 68, pp. 129–139, 2015.

[19] G. Klein and D. Murray, “Full-3D edge tracking with a particle filter,” in *BMVC 2006 - Proceedings of the British Machine Vision Conference 2006*, 2006, pp. 1119–1128.

[20] H. Zhou, Y. Yuan, and C. Shi, “Object tracking using SIFT features and mean shift,” *Comput. Vis. Image Underst.*, vol. 113, no. 3, pp. 345–352, 2009.

[21] O. Javed and M. Shah, “Tracking and object classification for automated surveillance,” in Proc. European Conf. Computer Vision, vol. 4, 2002, pp. 343–357
[22] J. Malcolm, Y. Rathi, and A. Tannenbaum, “Multi-Object Tracking Through Clutter Using Graph-Cuts,” *IEEE International Conference on Computer Vision*. pp. 1–5, 2007.
[23] A. Saxena, S. Jamie, and A. Y. Ng, “Depth estimation using monocular and stereo cues,” in *IJCAI International Joint Conference on Artificial Intelligence*, 2007, pp. 2197–2203.
[24] A. H. Holway and E. G. Boring, “Determinants of Apparent Visual Size with Distance Variant,” *Am. J. Psychol.*, vol. 54, no. 1, p. 21, 1941.
[25] I. Ulrich and I. Nourbakhsh, “Appearance-Based Obstacle Detection with Monocular Color Vision,” *Artif. Intell.*, no. August, pp. 866–871, 2000.
[26] J. Smythies, “A note on the concept of the visual field in neurology, psychology, and visual neuroscience,” *Perception*, vol. 25, no. 3, pp. 369–371, 1996.
[27] P. E. Miller and C. J. Murphy, “Vision in dogs.,” *Journal of the American Veterinary Medical Association*, vol. 207, no. 12. pp. 1623–1634, 1995.
[28] I. P. Howard and B. J. Rogers, “Binocular Vision and Stereopsis,” in *Oxford University Press*, New York: Oxford University Press, 1995, pp. 106–746.
[29] H. Strasburger, I. Rentschler, and M. Jüttner, “Peripheral vision and pattern recognition: A review,” *J. Vis.*, vol. 11, no. 5, 2011.
[30] J. Pettigrew, “Evolution of Binocular Vision,” Optom. Visual neuroscience, vol. 2, no. 12, p. 366, 1925.
[31] Z. Zhang, H. Rebecq, C. Forster, and D. Scaramuzza, “Benefit of large field-of-view cameras for visual odometry,” in *Proceedings - IEEE International Conference on Robotics and Automation*, 2016, pp. 801–808.
[32] P. J. Burt, “Attention mechanisms for vision in a dynamic world,” in *Proceedings - International Conference on Pattern Recognition*, 1988, pp. 977–987.
[33] Y. Sun, R. Fisher, F. Wang, and H. M. Gomes, “A computer vision model for visual-object-based attention and eye movements,” *Comput. Vis. Image Underst.*, vol. 112, no. 2, pp. 126–142, 2008.
[34] Y. Yeshurun and M. Carrasco, “Attention improves or impairs visual performance by enhancing spatial resolution,” *Nature*, vol. 396, no. 6706, pp. 72–75, 1998.
[35] A. Yilmaz, O. Javed, and M. Shah, “Object tracking: A survey,” *ACM Computing Surveys*, vol. 38, no. 4. 2006.
[36] D. M. Gavrila and L. S. Davis, “3-D model-based tracking of humans in action: a multi-view approach,” in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 1996, pp. 73–80.
[37] M. Yokoyama and T. Poggio, “A contour-based moving object detection and tracking,” in *Proceedings - 2nd Joint IEEE International Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance, VS-PETS*, 2005, pp. 271–276.
[38] H. Medeiros, J. Park, and A. C. Kak, “Distributed object tracking using a cluster-based Kalman filter in wireless camera networks,” *IEEE J. Sel. Top. Signal Process.*, vol. 2, no. 4, pp. 448–463, 2008.
[39] A. Vedaldi, V. Gulshan, M. Varma, and A. Zisserman, “Multiple Kernels for object detection,” in *Proceedings of the IEEE International Conference on Computer Vision*, 2009.
[40] G. Deng and L. W. Cahill, “Adaptive Gaussian filter for noise reduction and edge detection,” in *IEEE Nuclear Science Symposium & Medical Imaging Conference*, 1994, no. pt 3, pp. 1615–1619.
[41] O. Vincent and O. Folorunso, “A Descriptive Algorithm for Sobel Image Edge Detection,” in *Proceedings of the 2009 InSITE Conference*, 2009, pp. 98–106.
[42] J. J. Yu, A. W. Harley, and K. G. Derpanis, "Back to basics: Unsupervised learning of optical flow via brightness constancy and motion smoothness," arXiv pre-print,

arXiv:1608.05842, Sept. 2016.
[43] V. Takala and M. Pietikainen, “Multi-object tracking using color, texture and motion,” in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2007, pp. 1–7.
[44] D. S. Guru and R. Dinesh, “Non-parametric adaptive region of support useful for corner detection: A novel approach,” *Pattern Recognit.*, vol. 37, no. 1, pp. 165–168, 2004.
[45] T. Schoenemann and D. Cremers, “Globally optimal shape-based tracking in real-time,” in *26th IEEE Conference on Computer Vision and Pattern Recognition, CVPR*, 2008, pp. 1–6.
[46] S. C. Park, S. H. Lim, B. K. Sin, and S. W. Lee, “Tracking non-rigid objects using probabilistic Hausdorff distance matching,” *Pattern Recognit.*, vol. 38, no. 12, pp. 2373–2384, 2005.
[47] C.-Y. Chen and R. Klette, “Image Stitching: Comparisons and New Techniques,” Lecture Notes in Computer Science, vol. 1689, pp. 615- 622, 1999.
[48] R. Szeliski, “Image alignment and stitching: A tutorial,” *Foundations and Trends in Computer Graphics and Vision*, vol. 2, no. 1. pp. 1–10, 2006.
[49] E. Adel, M. Elmogy, and H. Elbakry, “Image Stitching based on Feature Extraction Techniques: A Survey,” *Int. J. Comput. Appl.*, vol. 99, no. 6, pp. 1–8, 2014.
[50] A. Mills and G. Dudek, “Image stitching with dynamic elements,” *Image Vis. Comput.*, vol. 27, no. 10, pp. 1593–1602, 2009.
[51] D. Mistry and A. Banerjee, “Comparison of Feature Detection and Matching Approaches: SIFT and SURF,” *GRD Journals- Glob. Res. Dev. J. Eng.*, vol. 2, no. March, pp. 7–13, 2017.
[52] J. Velasco *et al.*, “Nutrient and particulate inputs into the Mar Menor lagoon (Se Spain) from an intensive agricultural watershed,” *Water. Air. Soil Pollut.*, vol. 176, no. 1–4, pp. 37–56, 2006.
[53] T. Lindeberg, “Scale Invariant Feature Transform,” *Scholarpedia*, vol. 7, no. 5, p. 10491, 2012.
[54] M. Brown and D. G. Lowe, “Automatic panoramic image stitching using invariant features,” in *International Journal of Computer Vision*, 2007, vol. 74, no. 1, pp. 59–73.
[55] A. Bundy and L. Wallen, “Difference of Gaussians,” in *Catalogue of Artificial Intelligence Tools*, 1984, pp. 30–30.
[56] Abidrahmank, “Introduction to SIFT (Scale-Invariant Feature Transform),” *OpenCV*, 2013. [Online]. Available: https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_feature2d/py_sift_intro/py_sift_intro.html. [Accessed: 03-Jun-2019].
[57] E. N. Mortensen, H. Deng, and L. Shapiro, “A SIFT descriptor with global context,” in *Proceedings - 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, CVPR 2005*, 2005, vol. I, pp. 184–190.
[58] Y. Li, Y. Wang, W. Huang, and Z. Zhang, “Automatic image stitching using SIFT,” in *ICALIP 2008 - 2008 International Conference on Audio, Language and Image Processing, Proceedings*, 2008, pp. 568–571.
[59] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool, “Speeded-Up Robust Features (SURF),” *Comput. Vis. Image Underst.*, vol. 110, no. 3, pp. 346–359, 2008.
[60] M. Kokare, P. K. Biswas, and B. N. Chatterji, “Texture image retrieval using rotated wavelet filters,” *Pattern Recognit. Lett.*, vol. 28, no. 10, pp. 1240–1249, 2007.
[61] R. Raguram, O. Chum, M. Pollefeys, J. Matas, and J. M. Frahm, “USAC: A universal framework for random sample consensus,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 8, pp. 2022–2038, 2013.
[62] M. Trapp and J. Dollner, “A generalization approach for 3D viewing deformations of single-center projections,” in *GRAPP 2008 - Proceedings of the 3rd International*

Conference on Computer Graphics Theory and Applications, 2008, pp. 163–170.
[63] S. Peleg and M. Ben-Ezra, “Stereo panorama with a single camera,” *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, vol. 1, pp. 395–401, 1999.
[64] M. Kennedy and S. Kopp, "Understanding Map Projections," ESRI (Environmental Systems Research Institute) press, 2000.
[65] A. Swain, “Noise filtering in Digital Image Processing,” 2018. [Online]. Available: https://medium.com/image-vision/noise-filtering-in-digital-image-processing-d12b5266847c. [Accessed: 21-Apr-2019].
[66] R. Ahmadi, J. Kangarani Farahani, F. Sotudeh, A. Zhaleh, and S. Garshasbi, “Survey of image denoising techniques,” *Life Sci. J.*, vol. 10, no. 1, pp. 753–755, 2013.
[67] B. Weiss, “Fast median and bilateral filtering,” in *ACM SIGGRAPH 2006 Papers, SIGGRAPH ’06*, 2006, pp. 519–526.
[68] P. Milanfar, “Symmetrizing smoothing filters,” *SIAM J. Imaging Sci.*, vol. 6, no. 1, pp. 263–284, 2013.
[69] W. S. Cleveland and C. Loader, “Smoothing by Local Regression: Principles and Methods,” in *Statistical Theory and Computational Aspects of Smoothing*, 1996, pp. 10–49.
[70] D. M. Hawkins, “The Problem of Overfitting,” *Journal of Chemical Information and Computer Sciences*, vol. 44, no. 1. pp. 1–12, 2004.
[71] A. V. Balakrishnan, “The kalman filter,” *Math. Intell.*, vol. 1, no. 2, pp. 90–92, 1978.
[72] J. Andrade-Cetto and A. Sanfeliu, “A The Kalman Filter,” in *Springer Tracts in Advanced Robotics*, vol. 23, 2006, pp. 119–125.
[73] G. Welch and G. Bishop, “An Introduction to the Kalman Filter,” *University of North Carolina at Chapel Hill, Chapel Hill, NC*, 2006. [Online]. Available: https://www.cs.unc.edu/~welch/media/pdf/kalman_intro.pdf. [Accessed: 21-Apr-2019].
[74] Y. Xu, Q. Zhou, L. Gong, M. Zhu, X. Ding, and R. K. F. Teng, “High-speed simultaneous image distortion correction transformations for a multicamera cylindrical panorama real-Time video system using FPGA,” *IEEE Trans. Circuits Syst. Video Technol.*, vol. 24, no. 6, pp. 1061–1069, 2014.
[75] V. Popovic *et al.*, “Image blending in a high frame rate FPGA-based multi-camera system,” *J. Signal Process. Syst.*, vol. 76, no. 2, pp. 169–184, 2014.
[76] K. Ambrosch, M. Humenberger, W. Kubinger, and A. Steininger, “Hardware implementation of an SAD based stereo vision algorithm,” in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2007, no. June 2007, pp. 1–6.
[77] S. Jin *et al.*, “FPGA design and implementation of a real-time stereo vision system,” *IEEE Trans. Circuits Syst. Video Technol.*, vol. 20, no. 1, pp. 15–26, 2010.
[78] C. Banz, S. Hesselbarth, H. Flatt, H. Blume, and P. Pirsch, “Real-time stereo vision system using semi-global matching disparity estimation: Architecture and FPGA-implementation,” in *Proceedings - 2010 International Conference on Embedded Computer Systems: Architectures, Modeling and Simulation, IC-SAMOS 2010*, 2010, pp. 93–101.
[79] J. Woodfill and B. Von Herzen, “Real-time stereo vision on the PARTS reconfigurable computer,” in *IEEE Symposium on FPGAs for Custom Computing Machines, Proceedings*, 1997, no. April, pp. 201–210.
[80] “Dlib C++ Library,” *10 March 2019*, 2019. [Online]. Available: http://dlib.net/. [Accessed: 12-May-2018].
[81] O. Bornet, “OpenCV Library Wiki,” *OpenCV*, 2008. [Online]. Available: https://opencv.org/. [Accessed: 12-May-2018].
[82] L. Kirischian, “Virtualization of Reconfigurable Computing System Architecture,” in *Reconfigurable Computing Systems Engineering*, CRC Press, 2018, pp. 259–308.

[83] H. Shinoda, M. M. Hayhoe, and A. Shrivastava, “What controls attention in natural environments?,” *Vision Res.*, vol. 41, no. 25–26, pp. 3535–3545, 2001.
[84] NASA/JPL-Caltech, “Mars Rover Curiosity in Artist’s Concept [Image],” 2011. [Online]. Available: https://mars.nasa.gov/resources/3512/mars-rover-curiosity-in-artists-concept-wide/. [Accessed: 21-May-2018].
[85] SpaceX, “NASA set date for first SPACEX crew dragon test flight [Image],” 2018. [Online]. Available: https://www.theverge.com/2019/2/22/18236771/nasa-spacex-dragon-commercial-crew-dm-1-test-flight. [Accessed: 20-Jun-2018].
[86] Digilent, “ZedBoard Hardware User’s Guide,” 2012. [Online]. Available: http://www.zedboard.org/sites/default/files/documentations/ZedBoard_HW_UG_v2_2.pdf. [Accessed: 10-Jun-2019].
[87] O. Technologies, “OVM7690 CameraChip Datasheet,” 2008. [Online]. Available: http://aitendo3.sakura.ne.jp/aitendo_data/product_img/camera/OV7690/ov7690_full.pdf. [Accessed: 05-Jun-2019].
[88] B. E. Bayer, “Color Imaging Array,” *U. S. Pat. 3971 065*, 1976.
[89] D. Kaneria, “Display: HDMI,” 2014. [Online]. Available: https://www.slideshare.net/DhavalKaneria/hdmi-35332718. [Accessed: 20-Jun-2018].
[90] Xilinx, “Integrated Logic Analyzer LogiCORE IP, PG172 (v6.1),” 2016. [Online]. Available: https://www.xilinx.com/support/documentation/ip_documentation/ila/v6_2/pg172-ila.pdf. [Accessed: 05-Jul-2019].