International Journal of Scientific Research in Science and Technology
Print ISSN: 2395-6011 | Online ISSN: 2395-602X (www.ijsrst.com)
doi : https://doi.org/10.32628/IJSRST523103196

Object Detection Using Adaptive Block Partition and RCNN Algorithm

R. Ajay Krishnaraju[1], J. Poovarasan[1] , S. Santhies Kumar[1], S. Surya[1], P. Tamilselvan[2]
[1]Students, [2]Professor
Department of ECE, Rajiv College of Engineering and Technology, Puducherry, India

ARTICLEINFO

Article History:

Accepted: 05 June 2023
Published: 24 June 2023

Publication Issue
Volume 10, Issue 3
May-June-2023

Page Number
1009-1023

ABSTRACT

Advancements in image and video processing are growing over the years for industrial robots, autonomous vehicles, cryptography, surveillance, medical imaging and computer-human interaction applications. One of the major challenges in real-time image and video processing is the execution of complex functions and high computational tasks. To overcome this issue, a hardware acceleration of different filter algorithms for both image and video processing is implemented on Xilinx Zynq®-7000 System on-Chip (SoC) device consists of Dual-core Cortex™-A9 processors which provides computing ability to perform with the help of software libraries using Vivado® High-Level Synthesis (HLS).

The acceleration of object detection algorithms include Sobel-Feldman filter, posterize and threshold filter algorithms implemented with 1920 x 1080 image resolutions for real-time object detection. The implementation results exhibit effective resource utilization such as 45.6% of logic cells, 51% of Look-up tables (LUTs), 29.47% of Flipflops, 15% of Block RAMs and 23.63% of DSP slices under 100 MHz frequency on comparing with previous works.

There are a few reasons why tracking is preferable over detecting objects in each frame. Tracking facilitates in identifying the identity of various items across frames when there are several objects. Object detection may fail in some instances, but tracking may still be achievable which takes into account the location and appearance of the object in the previous frame. The key hurdles in real-time image and video processing applications are object tracking and motion detection. Some tracking algorithms are extremely fast because they perform a local search rather than a global search. Tracking algorithms such as meanshift, Regional Neural Network probabilistic data association, particle filter, nearest neighbor, Kalman filter and interactive multiple model (IMM) are available to estimate and predict the state of a system.

Keywords: IMM, System on-Chip, High-Level Synthesis

I. INTRODUCTION

The introduction of high speed computational hardware platforms deployed for image processing applications are increasing in recent years. It is no surprising that image processing units are already included in cellphones and cameras, but the demand of image processing algorithms remains a barrier to bringing a wide variety of theoretical advances into the reality of real-time implementation. On the other hand, the large number of resources available on FPGAs, along with the freedom it provides for testing and developing Application-Specific Integrated Circuits (ASICs), has made FPGA a best platform for implementing image processing algorithm in real time and autonomous vehicles. FPGAs are well-suited in providing trade-off between parallelism and flexibility when compared with other hardware platforms shown in Figure 1.1. The brief survey of hardware architectures reveals that, despite considerable advances in designing new algorithms or enhancing current ones, only a small amount of attention is paid to the realization.

1.1 FPGA ARCHITECTURE

FPGAs are composed of a collection of programmable logic blocks (PLBs) embedded in a programmable interconnect. Basic computational and storage aspects are provided by the programmable logic block, which can be used in electronic designs. The logic blocks in FPGA architecture are made up of a few logic cells that include look-up tables (LUTs), multiplexers, D-flip flops and various combinations of memory and logic blocks are found in recent FPGAs. To connect these logic blocks, routing channels and I/O 4 blocks are required [1]. In the programmable routing architecture, pre-fabricated switches and programmable wires are positioned in vertical and lateral routing channels. It allows logic and I/O blocks to communicate with one another. Around the FPGA chip, the programmable routing network connects I/O blocks. The functional components and routing framework are connected to peripherals using configurable I/O pads. Logic circuits surround the I/O pads, forming I/O cells that take up a lot of space on the chip. The simplest form of FPGA Architecture is shown in Figure 1.1

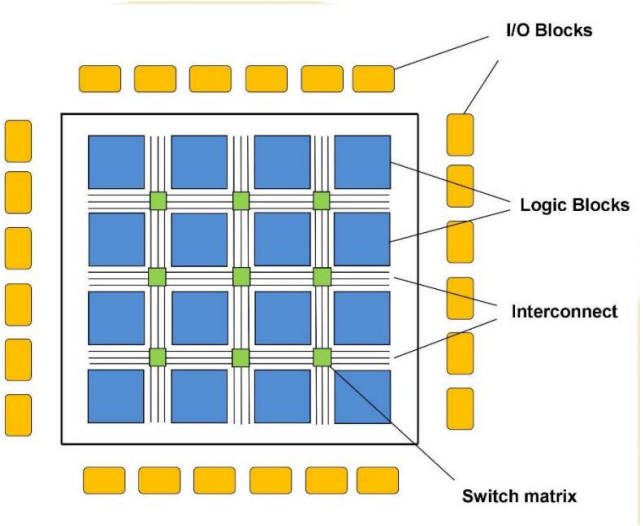

Figure 1.1 Generic FPGA Architecture

FPGAs are reprogrammable platforms that enable the reuse of hardware components and software libraries. Xilinx creates SoCs that combine the software programmability of a processing unit with the device fully programmable of an FPGA. They offer a variety of boards to their potential consumers who want SoC platforms for design, which are grouped into three categories: cost-optimized, mid-range, and high-end. Devices in the cost-optimized category include the Artix® and Zynq-7000 series. These boards offer developers a low-cost way to construct programs that do not take substantial software processing. As a result, these devices are available with either single-core or dual-core ARM Cortex-A9 processors.

1.1.1. Xilinx Zynq-7000 SoC

Zynq-7000 APSoCs are exclusive and typical from all other Xilinx FPGA families. It is built with a dual-core ARM Cortex-A9 Processing System (PS), Advanced Microcontroller Bus Architecture (AMBA) Interconnects and a variety of peripheral devices

including a USB JTAG interface, Quad SPI flash memory, UART, CAN and Ethernet as well as Xilinx Programmable Logic (PL) of Artix 7-series [2]. Figure 1.4 shows the schematic view of Xilinx XC7Z020-1CLG484C SoC device. Figure 1.5 gives the overview structure of FPGA hardware which is chosen as primary hardware for proposed design to meet prerequisites. The significant features of Zynq-7000 SoC are listed below [3].

Memory

• Support 32 data width
• IIC - 1 KB EEPROM
• 16MB Quad SPI Flash
• DDR3 Component Memory 1GB

Configuration

• USB JTAG configuration port (Digilent)
• 16MB Quad SPI Flash

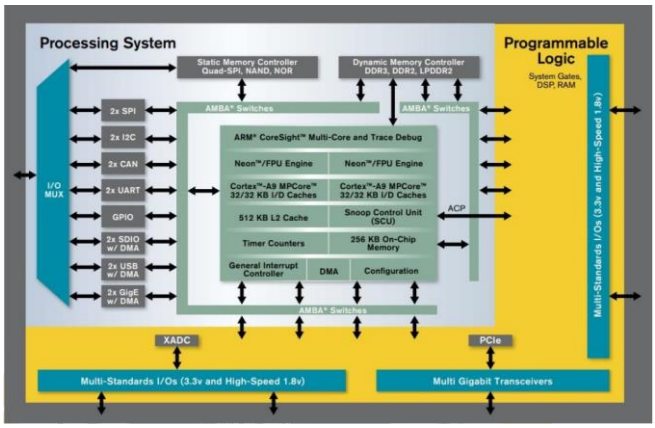

Figure 1.2. Schematic View of PL and PS Portions of ZYNQ XC7Z020- 1CLG484C [3]

The programmable logic section comprises of CLBs, LUTs, BRAMs, DSP slices and FFs and so on. LUTs can be configured as a single 6-input LUT (64-bit ROMs) with a single output or as two 5-input LUTs (32-bit ROMs) with distinct outputs but shared addresses or logic inputs. Each LUT output can be registered in a flip-flop if desired. A slice is formed by four such LUTs and their eight flip-flops, as well as multiplexers and arithmetic carry logic, and two slices comprise a customizable logic block (CLB). Four of the eight flip-flops per slice (one flip-flop each LUT) can be configured as latches if desired. The block diagram of Zynq SoC is shown in Figure 1.6.

Zynq-7000 SoC series of devices enables designers to target both cost-sensitive and high-performance applications from a single platform using industry-standard tools. While all devices in the Zynq-7000 family have the identical PS, the PL and I/O resources may vary. As a result, the Zynq-7000 SoCs can execute plethora of applications such as networking, cryptography, wireless networks, video and surveillance, tracking and detection, medical imaging, autonomous systems and industrial applications. The resources available in Zynq-7000 SoC are listed in Table 1.3.

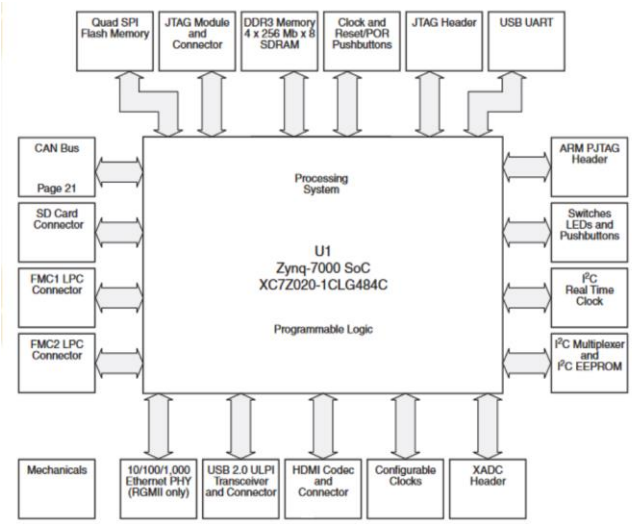

Figure 1.3. Overview of ZYNQ ZC7Z020 SoC Block Diagram [2]

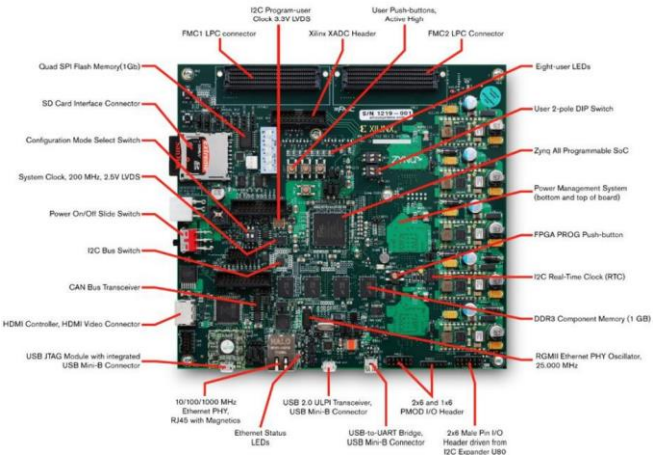

Figure 1.4. Block Description of ZYNQ ZC7Z020 SoC [3]

1.2 XILINX VIVADO

Vivado Design Suite, created by Xilinx, is used for HDL design synthesis and analysis [4]. Vivado is an IDE that allows users to create low-level hardware designs for Xilinx FPGAs. This suite includes a plethora of Xilinx-developed intellectual property (IP) that may be included into designs to minimise development time. Users can also create their own HDL-based IP for application modification with Vivado. The hardware designs can be developed as a set of HDL files that are linked together, or by utilising the built-in block diagram GUI, which allows users to drop in IP blocks and manually connect signal in Vivado. When a design is finished, Vivado can output a bitstream file that can be used to configure the FPGA.

Before simulation or synthesis, the tool provides design validation, which allows the user to ensure that the developed hardware design is correctly configured and free of major design flaws. Users can build testbenches for their designs to emulate the functionality of their applications. When a simulation is done, a testbench is an HDL-based framework that wraps around the hardware design and provides it with a sequence of inputs that will be executed and outputted to the user. Running simulations within Vivado is a useful tool for 10 users to assess the correctness and functionality of their designs prior to synthesis. The complete design flow of Vivado HLS is illustrated in Figure 1.7.

Simulation is merely a technique for functional testing of a design; it does not ensure that the design will pass synthesis. The most significant feature that Vivado offers is synthesis. The synthesis process will convert the user's design, which may be in the form of HDL code or a schematic, into a netlist. This step is crucial since the netlist is the component in charge of mapping and connecting logic gates and FFs throughout the fabric. In general, synthesis is the process of converting a software design into the hardware components required to physically represent the application. When the netlist is aimed at FPGA hardware, it ensures that when an output signal is generated, it can transmit the data to the input of the next component in the time required to transport the data physically. This concept is known as setup and hold slack in static timing analysis, and it is defined as the difference between the data required time and the data arrival time.

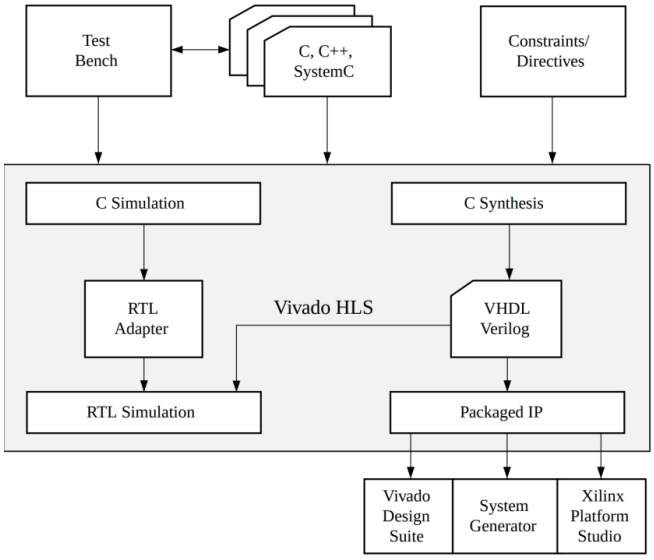

Figure 1.5 VIVADO HLS Design Flow [4]

To transition from one state to the next, each custom IP produced for this project implements an FSM that is reliant on the system clock and the defined state variable. This architecture enables a function to be divided into numerous states, each of which requires one clock cycle. This design enables a function to be divided into numerous states, each of which requires one clock cycle to perform. This has the advantage of allowing a timing issue to be tracked back to a specific state within an IP block when a static timing analysis report is generated. Once the source of the timing error has been identified, it can be rectified by providing it with additional states to complete its execution.

The Vivado 2020.1 SDK tool was used for this research work to build high-level software designs that operate on the FPGA processors and interface with the hardware design in the FPGA fabric. These software designs are in charge of retrieving parameter and frame data from the FPGA's I/O ports and writing

it to BRAM. The SDK included a graphical user interface (GUI) for developing applications directly on the MicroBlaze® soft-processor found in ZC702 FPGAs and the dual-core ARM Cortex-A9 CPU. It differs from the standard Eclipse IDE in that it can import Vivado-generated hardware designs, create and configure Board Support Packages (BSPs), support single-processor and multi-processor development for FPGA-based software applications, and includes off-the-shelf software reference designs that can be used to test the applications hardware and software functionality.

1.3 NEED FOR HARDWARE ACCELERATION

Hardware acceleration refers to the utilization of hardware resources to accomplish certain activities more quickly than software execution on platforms, such as FPGAs and general processing units (GPUs). High performance, lower power consumption, lower latency, improved parallelism and bandwidth, and better utilization of space and 12 functional components available on an integrated circuit are all positive aspects of hardware acceleration.

FPGAs are often considered as first option towards true hardware acceleration since they have a reconfigurable fabric that can express a software programme as logic gates. The trade-off between flexibility, performance, and power consumption is constantly examined when considering hardware platforms to accelerate domain-specific applications. FPGAs, on the other hand, fall somewhere in between the two and provide a good balance between these three measures [5].

1.4 MOTIVATION AND PROBLEM STATEMENTS

FPGA based hardware acceleration for image and video processing techniques provide high performance and parallelism. The major concern in implementation process is that effective utilization of hardware resources such as BRAMs, DSP slices, LUTs, FFs and PLBs. The challenging task in real-time image processing is tracking of multiple objects. For object tracking algorithms, accuracy and speed are considered as the primary parameters for evaluation and validation. CNN-based tracking algorithms are not time efficient, and feature extraction involves a multi-layer network to perform the operation. These characteristics prompted the researchers to choose FPGAs to implement tracking and prediction.

1.5 ORGANIZATION OF THE THESIS

Chapter 1 discusses FPGA architecture and basic concepts of hardware acceleration and the literature on the variants of FPGA acceleration for different image and video processing applications such as object detection, tracking and motion detection followed by the problem statements and objectives of the research work.

A comprehensive literature survey about existing works about hardware acceleration and implementation of different image processing algorithms and architectures are explained in chapter 2 with the motivation to carry out the research objectives.

In chapter 4, multiple object tracking and motion detection is performed. The proposed MDKF is explained with system model, various ablation studies using standard datasets and numerical analysis along with FPGA implementation. Chapter 5 briefs the properties and analysis of proposed MDKF algorithm for an aircraft application with path tracking based on linear measurements using updated state estimations. Based on these estimations, Kalman gain equations are derived. The results are compared with conventional Kalman filter.

YOLOv4, a deep learning algorithm implemented on FPGA platform is demonstrated in chapter 6. The hardware acceleration of deep learning algorithm for realtime object detection is proposed. It also includes datasets, evaluation parameters, hardware modules and comparison with other existing implementations. Chapter 7 outlines the concluding remarks with future perspectives.

II. RELATED WORK

Hardware acceleration gained progression with the evolution of FPGAs and GPUs. Particularly, FPGAs are mostly suited for acceleration when compared with GPUs because of its cost efficiency, reconfigurable ability and flexibility. In this chapter, the detailed literature review of FPGA based acceleration for image processing techniques including deep learning algorithms are discussed.

2.1 FPGA BASED ACCELERATION OF ALGORITHMS FOR OBJECT DETECTION

Numerous studies are being conducted with the intention to develop specialized hardware accelerators to perform a specific function in computer vision applications. In this subsection, various acceleration of edge detection filter algorithms and the implementation techniques are explained.

J. C. Mora et al. presented cost-efficient video processing development system with the implementation of some spatial filters such as color filtering, Sobel and posterize for 320 x 240 resolutions on Zedboard embedded platform [6]. J. K. Kong Wong et al. demonstrated hardware acceleration on Altera Cyclone IV FPGA for video processing with reduction in computational time and memory bandwidth requirements is described in literature [7]. The hardware implementation shown in Figure 2.1 could minimise both the computational load on a general-purpose computer and the calculation time required. The above mentioned hardware implementations make use of soft-core processors, while others make use of custom-built processing and executional units. The implemented hardware architecture does not take expandability or deployment for usage with other systems into account [7].

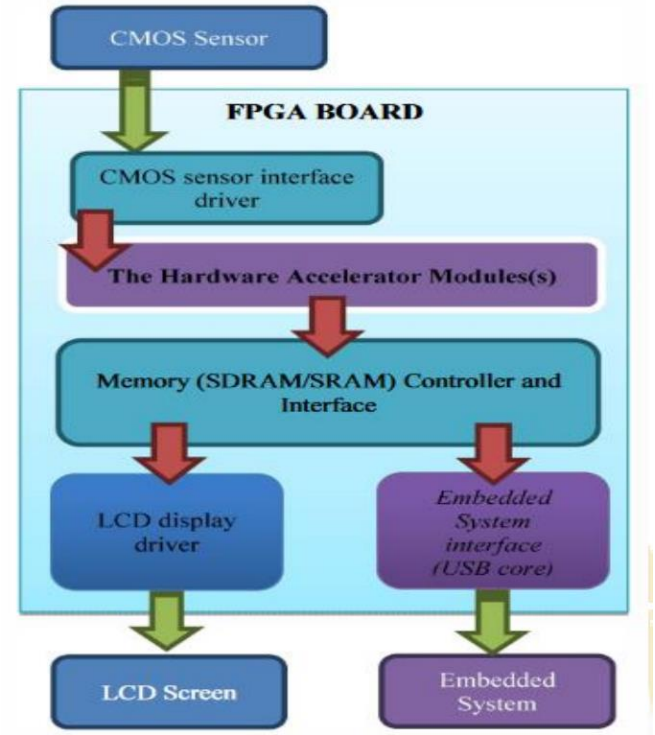

Figure 2.1 Hardware Accelerator Proposed in [7]

The significant design aspect in hardware acceleration is platform based design (PBD) in which each platform is a layer in the design flow that abstracts the underlying and resulting design-flow processes [8]. The dedicated hardware architecture of designed with PBD which explored the design requirements implemented on Virtex-5 FPGA for real-time video and image processing system [9]. J. K. Pandey et al. proposed PBD based hardware accelerated architecture on Xilinx ML-507 platform capturing 640 x 480 resolutions of realtime video frames at 60 frames per second (fps) with effective resource utilization [10].

J. Rettkowski et al. explained different types of design architectures using Histogram of Oriented Gradients (HOG) based on OpenCV method for processing resolution of 1920 x 1080 pixel resolutions which is achieved at 39.6 fps implemented on Zynq®- 7000 SoC [11]. The proposed acceleration in the literature [11] outperformed the software approach in terms of performance and flexibility and integrated OpenCV functions implemented on ARM processors. Several

types of hardware-accelerated systems for machine vision applications based on FPGA implementations are described briefly in [13-17].

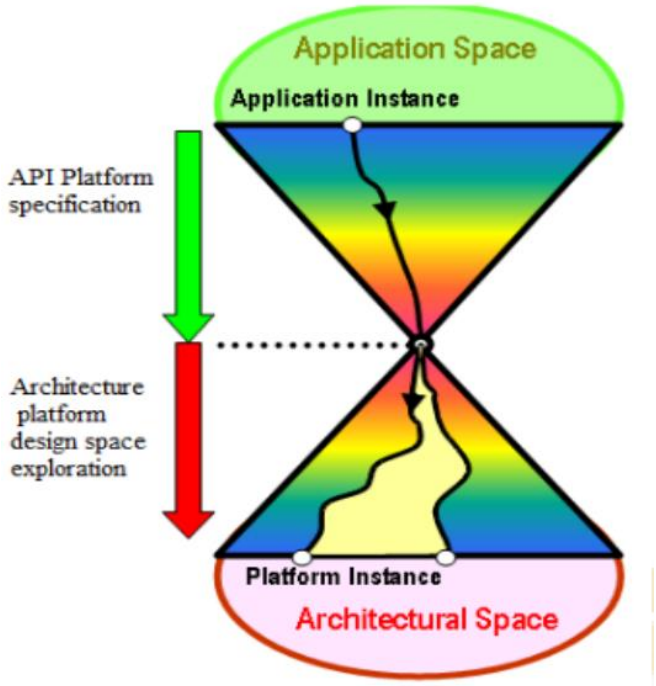

Figure 2.2 Platform-Based Design Approach [8]

FPGAs are able to perform multi-threading processes that help them to execute numerous applications, including automotive industries [1]. The majority of research investigations have shown the advancement of image processing and computer vision in driver-assistance systems (DA). Claus et al. developed DA systems for various driving scenarios in order to improve security using multi-processor SoC (MPSoC) architecture following dynamic partial reconfiguration (DPR) [18] and also explored "Autovision" framework for hardware accelerated engines shown in Figure 2.3. Numerous validations are concentrated on advanced DA systems for hardware implementation of lane detection in real-time environment.

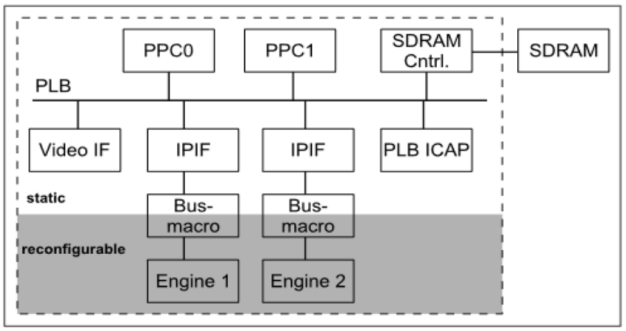

Figure 2.3 FPGA Based Reconfigurable Auto-Vision System [18].

Using TB-FMCH-3GSDI2A device for 1920 x 1080 resolution images, Hough Transform is implemented using Vivado HLS tool at 100 MHz with 130 fps and the frame rate of 60 fps is achieved for 480 × 270 resolution on Xilinx Zynq-7000 platform with reduction in latency [19].

FPGAs are suitable for high computational operations such as cluster analysis, image recognition and computer vision applications [20]. Edge detection is a widely adapted method in image segmentation and surveillance systems. The hardware implementation for video streams (4096 x 2160) on Xilinx devices is examined by M. Kowalczyk et al. [21]. Rapid prototyping methods are used to create FPGA-based edge detection architectures for HD video streaming [22]. The study and analysis about edge detection techniques for image and video processing algorithms implemented on FPGAs are discussed and compared with various methods [23].

2.1 ACCELERATION OF RCNN ALGORITHMS ON HARDWARE

FPGA, being a reconfigurable architecture, may solve a variety of system performance issues, and designers can adapt the resources according to the demand and target applications [24]. Such features have prompted the researchers to construct tracking and detection systems using FPGA devices. To solve tracking error in gesture detection, an enhanced tracking algorithm using EKF based on multiple feature extraction are

presented [25]. The convolutional Siamese network (SiamFC) based on adaptive Kalman filter with is designed to increase tracking performance in challenging scenarios at 43 fps [26]. Nevertheless, advances in deep learning approaches for object trackers are emerging, the computational process is highly complex, and data annotations for bounding box prediction will affect overall system performance. The tracking problems for occluded images are addressed and EKF is employed to address these issues [27]. For unmanned aerial vehicles (UAVs), C. G. Prevost et al. presented state estimates and trajectory prediction using EKF [28].

The different types of Kalman filter-based tracking methods for multiple objects are discussed in [29-33]. The most difficult issue in multiple object tracking is identifying and resolving occlusion obstructions developing various types of Kalman filters. Zhou et al. explained a robust deep learning method for multi-object tracking through occlusions [36]. This multi-instance tracking method is based on the assumption of updating the model with samples from prior frames to minimize update errors and ensuring the proposed tracker classifies new samples [37]. The tracking performance of deep neural networks (DNN)- based tracking algorithms are improved over the years [38–41]. To improve the performance of feature extraction networks, Yuan et al. suggested a self-supervised deep correlation tracker approach for manually labelled images [42].

FPGAs deliver real-time solutions for deep learning applications. and also provide improving flexibility by parallel computations to implement learning-based algorithms for object detection. Dong et al. suggested an object detection technique based on deep Q-learning and a hyperparameter optimization algorithm [43]. However, these CNN-based tracking algorithms demand more computational time and feature extraction requires multi-layered networks. As a result of these factors, many trackers are unable to reach the needed tracking speed and precision. These tracking methods can be employed on embedded platforms to overcome the issues highlighted earlier.

By developing an iterative tracking algorithm referred to as multi Track BeforeDetect (MF-TBD) on FPGA, Zhang et al. demonstrated that hardware implementations performed better than software simulations in terms of speed and performance [53]. Iqbal et al. integrated the mean shift (MS) algorithm and KF to develop a visual tracking method based on adaptive video filtering. MS-KF achieved a tracking performance of 38 fps at 75 MHz while implementing on FPGA [54].

2.2 DEEP LEARNING BASED ACCELERATION FOR OBJECT DETECTION

Regional Convolutional neural network (RCNN) demands a significant number of memory accesses and calculations. The development of RCNNs have resulted in a major breakthrough in computer vision applications. Frequent access to off-chip memory result in slow computation and high power consumption. FPGAs are logically reconfigurable hardware processors that offer significant advantages in terms of performance and power consumption, making them an excellent option for deploying a deep convolutional network. When compared to GPUs, FPGAs are able to provide higher performance in deep learningbased applications where latency is critical [55]. Comparison of ASICs, FPGAs and GPUs for deep learning applications are listed in Table 2.2.

Table 2.2 Comparison of ASICs, FPGAs and GPUs for Deep Learning Applications

Criteria	ASICs	FPGAs	GPUs
Processing peak power	High	Very high	High
Power consumption	Low	Very low	Very high
Flexibility	Low	Very high	Moderate
Training	Potential but not trained yet	Not so efficient	Highly efficient
Inference	Poor	Best	Average

FPGAs provide hardware customisation with built-in AI and can be configured to behave similarly to a GPU or an ASIC. The advantages of FPGAs over GPUs are as follows.

Greater performance with low latency: By directly ingesting video into the FPGA and bypassing a CPU, FPGAs can provide low latency as well as deterministic latency for real-time applications such as video stream, interpretation, and gesture detection. Designers can create a neural network from scratch and organise the FPGA to best suit the model.

Cost-efficiency and value: Because FPGAs can be reprogrammed for numerous functions and data formats, they are one of the most cost-effective hardware solutions available. Furthermore, FPGAs can be utilised for purposes other than AI. Designers can save money and board space by combining multiple functions into the same chip. Because FPGAs have extended product life cycles, FPGAbased hardware designs can have a long product life, measured in years or decades. Because of this, they are suited for application in the industrial defence, medical, and automotive markets.

YOLOv4 uses single convolutional neural network predicts the bounding box coordinates, the class probabilities for these boxes and confidence of the object from the whole image. The evaluation metrics are mean average precision (mAP), precision, recall, loss measurements used in learning mechanisms. For different image resolutions, mAP is shown in Figure 2.4. FPGAs perform reconfiguration which offer better trade-offs between 23 performance and flexibility. Thus FPGAs deserve high priority than ASICs as well as GPUs for performing deep learning techniques [64].

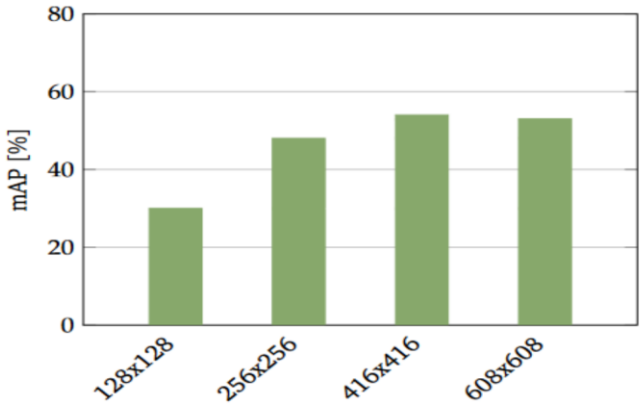

Figure 2.3 mAP for Different Input Resolutions.

FPGAs are well-suited for deep learning algorithms, there are numerous surveys available with respect to design and implementation [65-71]. There are several FPGA design have been proposed for the implementation of different versions of YOLO algorithms shown in Table 2.4. Ding et al. proposed a framework (REQ-YOLO) for object detection implemented on FPGA and developed new processing element (PE) structure [72]. However, in literature [72] achieved very high percentage of resource utilisation. For effective resource utilisation, the fast FIR algorithm (FFA) is introduced in literature [73] for computation of CNN models and is implemented on Xilinx Virtex VC707 and Zynq ZC706 boards. Mini-YOLOv3 is developed for real-time embedded applications tested on MS-COCO dataset and proposed a Multi-Scale Feature Pyramid Network (MSFPN) for feature extraction [74]. The architecture in literature [74] used two types of convolution processes such as group convolution and depth-wise convolution, yet the accuracy and speed is not high when compared to recently proposed methods.

FPGA-YOLO hardware acceleration for object detection is developed in [79] which utilized maximum number of on-chip resources. The overall performance of hardware acceleration is quite low in [80] for implementing CNN on VC707 FPGA board at 61.62 GigaFLOPS (100 MHz). The object detection implemented on FPGAs and GPUs is carried out in [81] without any optimization of hardware resource

utilization for both platforms. D. Pestana et al. developed acceleration of YOLO tiny architectures algorithm on FPGAs achieving 31 fps with efficient resource utilization [82]. For hardware acceleration of object detection algorithms, resource utilization and prediction time are the significant parameters to evaluate the research work.

Several existing CNN based accelerators failed to evaluate both the factors. Despite the fact that existing FPGA accelerators outperform generic hardware, the exploration of design space of hardware accelerators is still a major concern. The most significant issue is the computation speed which may not be sufficient to meet the memory bandwidth given by an FPGA platform. As a result, present techniques are not capable to reach optimal performance due to overexploitation of hardware resources and memory bandwidth. At the same time, the rising complexities and flexibility of deep learning systems aggravates the challenge.

III.3.METHODOLOGY

Researchers are turning their attention to the development of dedicated hardware for image and video processing applications as technology improves exponentially. To begin, software acceleration is a set of instructions for the CPU that allows numerous libraries to be executed for optimization. The implementation of these algorithms on programmable logic (PL) provided by SoC device is the second phase. These connectors provide efficient resource utilization, user management, real-time processing, and a variety of other benefits. Remote sensing, high performance computing, histology, medical diagnostics, broadcast, cryptography, wireless communication, and other applications are supported by FPGA hardware architectures. The advancements in microchip technologies allow for the integration of a wide range of functionality on host semiconductor chips. As a result, embedded SoC designers are dealing with design requirements such as adaptability, power consumption, performance, and cost.

FPGAs, which are gaining prominence in reconfigurable computing, are well suited for the development of embedded hardware design. FPGAs provide a solid foundation for image and video processing based hardware architecture mechanisms, with performance comparable to specialised ICs. These computer vision systems are mostly used in fields ranging from security to space in combination with developing technologies. Implementing image processing techniques on reconfigurable hardware can boost performance, accelerating common methods that shall simplify debugging and validation processes. The development time and expense are affected by these arguing limits. Therefore, there is a requirement for flexible solutions that provide greater precision in the development process and may be reused after final implementation.

FPGA implementation of filter algorithms are discussed in this chapter. SobelFeldman, threshold and posterize filter algorithms are implemented on Xilinx Zynq-7000 SoC with effective hardware resources utilization are also explained.

3.1 OVERVIEW OF EDGE DETECTION ALGORITHMS

Sobel-Feldman or Sobel filtering is a standard edge-detection technique that creates input data that highlights edges and transitions in real-time image and video processing [6]. The illustration of Sobel-edge filtering algorithm is shown in Figure 3.1.

Figure 3.1 Sobel Filter Illustration

Sobel filter represents the pixel values as x and y derivatives which is depicted in Figure 3.2. The convolution of Image (M) and two 3 x 3 kernels with vertical (Gx) and horizontal (Gy) derivative approximations is represented in Equation (3.1).

3.2 ACCELERATION OF EDGE DETECTION ALGORITHMS

The proposed architecture uses Xilinx XC7Z020-1CLG484C SoC device as major hardware, as illustrated in Figure 3.4. To connect the hardware blocks to the computing system, AXI bus connectivity protocols are exploited. Accessing hardware configuration registers and broadcasting input image processing data from IP core to frame buffer in DDR3 component memory is done via the AXI4-Lite bus and AXI4 stream block also converts the input image data to AXI4 stream interface. Test pattern generator in the FPGA receives video signals from the visual timing controller in the hardware part. VDMA 29 is used to send the video frames to memory controller, and then output is shown via external HDMI output interface.

Direct streaming and frame buffer streaming are the two methods of streaming architectures. The information of each pixel in the PL is passed to the video processing component and is transmitted to output interfaces in direct streaming architecture. However, with frame buffer streaming architecture, image data is originally retained by dedicated memories before video streams are buffered related to appropriate memory bandwidth. With the reprogrammable architecture of Zynq SoCs and software and hardware acceleration, multiple image and video processing operations can be accomplished.

The complete operating system is controlled by embedded hardware frame buffers and filters incorporated into PL component, in addition to software algorithms. In this architecture, video signals are sent from the processor to the video processing pipelined through Advanced Extensible Interface (AXI) interconnect. These ports and communication with the ARM CPU are shared by the USB camera. The memory controller stores the image data. Once the image is processed, it is transferred through the output linked with HDMI display, and the process pipeline is repeated during several iterations.

3.3 INTEGRATION OF VIVADO AND OPENCV

IP cores are set up with dimensions (width x height) of the input data. All video frames are received and processed when the IP core has been initialised in the input interface. The processed frame is then transferred over the output interface. Simultaneously, the core receives the next video frame, which is ready to process without any delay and next video frames are processed at regular intervals. This process is continued once the entire pipeline is executed and the image filters will be turned off. The video format used in the streaming design is 16-bit 4:2:2 YUV. The GUI layers of the display component are controlled by 32-bit RGB. For image processing applications, Vivado HLS includes a large set of functions [4]. To name a few, cv: : Mat class one of the library functions to indicate images in video processing system represented as,

Xilinx Video Timing Controller (VTC) can be utilized to determine dimensions of the input image. AXI4 stream interfaces, on the other hand, share any data through the output interfaces. The design approach for acceleration based on the video library functions for the Vivado HLS Application Program Interface (API) is shown in Figure 3.5. The library functions used in proposed design flow are AXIvideo2Mat and Mat2AXIvideo for conversion.

• AXIvideo2Mat function – converting from AXI4 Streams to hls::Mat representations.

• Mat2AXIvideo function – converting from hls::Mat representations to AXI4 video streams.

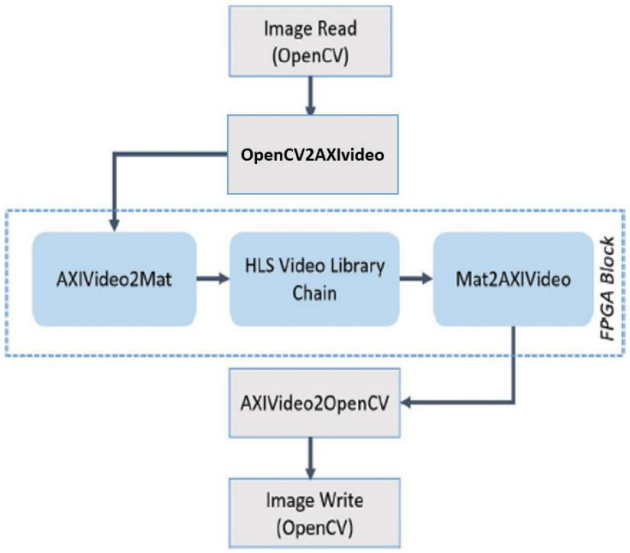

Figure 3.2. Design Flow of Vivado HLS Video Library Functions.

IV. RESULTS AND DISCUSSION

On a dual-core ARM PS, filtering algorithms are implemented. While input image data is executed and kept in output memory at the same time, various edge detection filters can be applied. Sobel-Feldman, threshold and posterize filtering techniques are illustrated among the detection filter algorithms used in this design are shown in Figure 3.6. Vivado 2018.2 is used to generate simulation and synthesis results. The complete system is synthesized and implemented at a desirable frequency of about 100 MHz. While analyzing performance and flexibility, some physical limits such as DSP blocks exhaustion are obtained. The resources of the FPGA SoC are utilized based on design which may vary depending on the design complexities of FPGA implementation.

4.1 Resource Utilization

The resources utilized for the proposed hardware acceleration is listed in Table 3.1. For input image with 1920 x 1080 resolution, three different types of filter designs are implemented. The utilization of hardware resources was investigated and achieved as 23.63%, 51%, 15%, 29.47% and 45.6% of DSP blocks, LUTs, BRAMs, FFs and logic cells respectively. The placement and routing mechanisms influence how logic cells are used. From the simulation results, the proposed implementation of filter algorithms consumed 9% and 4% fewer number of FFs and BRAMs respectively [6] as well as 41% and 30% fewer amount of logic cells and LUTs respectively [22]. The resource utilization comparison with other existing implemented systems is shown in

4.5 SUMMARY

Xilinx Zynq-7000 SoC was used to deploy hardware acceleration of SobelFeldman filtering, posterize and threshold filtering techniques for 1920 x 1080 image resolutions. Vivado 2018.2 was used to generate simulation and synthesis outcomes. Based on the simulation results, the proposed implementation used 29% and 40% less LUTs and logic cells respectively [6], as well as 10% and 3% fewer flip-flops and BRAMs, respectively [22]. Filter algorithms are executed concurrently in the proposed approach, providing flexibility and parallelism for hardware acceleration.

V. CONCLUSION AND FUTURE WORK

5.1 CONCLUSION

Object tracking and detection are important tasks in computer vision applications. The real-time performance of these processes can be achieved by implementing on FPGAs, GPUs and multi-core architectures. The primary objective of this research work provides different hardware accelerations for image processing techniques such as object tracking

and detection to achieve high performance, effective utilization FPGA resources and prediction accuracy.

We have implemented edge detection filter algorithms such as Sobel-Feldman, posterize and threshold on Zynq SoC for 1920 x 1080 image resolutions with effective resource utilization provides primary objective for other accelerations. The synthesis and simulations were carried out using Vivado 2018.2 and libraries were utilized from OpenCV functions. The outcomes of this work were compared with similar implementations in terms of hardware resource utilization.

In case of tracking, multiple object tracking (MOT) is the challenging processes when it is performed in real-time mode for high degree of accuracy and speed. MOT aims to estimate trajectories of all objects under various factors for example occlusions and distracters. To overcome this issue, MDKF algorithm is proposed and it is trained on various benchmark datasets such as OTB-100, UAVDT and MOT and ablation studies proved that MDKF achieved with 70.3% of precision and 44.7% of AUC. These results demonstrated that the proposed tracking algorithm outperformed other state-of-the art trackers.

The tracking speed of MDKF is achieved as 49 fps in software approach using MATLAB. The hardware acceleration of MDKF achieved 91 fps at 100 MHz with 780 mW consumption of power which shows that the proposed FPGA-based acceleration of MDKF achieved high tracking speed than the software approach. In order to analyze the 82 performance of MDKF, path tracking is carried out for linear systems. For estimation, state extrapolation, covariance and Kalman gain equations are updated based on linear measurements. The simulation is carried out using MATLAB/Simulink

Further the research is fueled by the application of deep learning in the image processing applications. Among several deep learning based object detections algorithms, RCNN algorithm is chosen and acceleration was attempted. In addition, a hardware based neural network for object detection was designed. The model is trained on MS-COCO benchmark dataset and outcomes are compared with existing implementations. For proposed acceleration of real-time detection, the prediction time is about 10.12 ms which is faster than existing SoC accelerations.

5.2 FUTURE PERSPECTIVES

The research work can be extended with the intention to incorporate full reconfiguration or partial reconfiguration (PR) which are the main features for FPGAs. The most difficult problems are programming for reconfigurable architectures and effective virtualization of FPGA resources for PR. With the development of FPGA technology, it is possible to implement reconfigurability for real-time image processing applications.

Modern FPGAs have greater capacity and faster memory speeds than in the past, allowing for more design space. In our research, we discovered that there may be a performance difference of up to 95% between two different solutions that use the identical logic resource of an FPGA. It is not trivial to settle with one optimal solution, particularly when the computation resource and memory bandwidth of an FPGA platform are taken into account. Therefore, if an accelerator structure is not designed properly, its compute performance will be insufficient to satisfy the memory band-width requirements enabled by FPGAs. It denotes that performance has suffered as a result of insufficient usage of either logic resources or memory bandwidth.

Finally, the future work is focused on implementing object detection algorithms on different types of hardware platforms to analyze various parameters like power consumption, speed and resource utilization. However, FPGA implementations surpasses software implementations in terms of timing accuracy and efficiency, their deployment is complex and time-consuming. Moreover, developers with specialised

expertise are required for development and customization of the FPGA algorithm. The upcoming platforms that can generate system configuration from software requirements could be a solution.

VI. REFERENCES

[1]. S.A. Fahmy, K. Vipin, FPGA dynamic and partial reconfiguration: A survey of architectures, methods, and applications. Comput. Surveys 51, pp. 1-39, 2018.

[2]. Xilinx, ZC702 Evaluation Board for the Zynq-7000 XC7Z020 SoC: User Guide (2017).[Online].https://www.xilinx.com/support/documentation/boards_and_kits/zc702_zvik/ug850-zc702-eval-bd.pdf. (Accessed on 23rd March, 2022)

[3]. Xilinx Inc.: Zynq-7000 all programmable SoC technical reference manual. (2021). (Accessed on 23rd March, 2022) Available at: https://www.xilinx.com/support/documentation/user_guides/ug585-Zynq-7000- TRM.pdf

[4]. Xilinx Inc, "Vivado Design Suite tutorial high level synthesis, UG871 (v 2014.1) May 6, 2014," UG871 (v 2014.1) May 6, 2014. [Online]. Available at: https://www.xilinx.com/support/documentation/sw_manuals/xilinx2019_1/ug871- vivado-high-level-synthesis-tutorial.pdf. (Accessed on 23rd March, 2022)

[5]. P. Babu, E. Parthasarathy, "Reconfigurable FPGA Architectures: A Survey and Applications," J. Inst. Eng. India Ser. B 102, pp. 143–156, 2021.

[6]. J. C. Mora, E. C. Gallego and S. S. Solano, "Hardware/software co-design of video processing applications on a reconfigurable platform," in Int. Conf. on Industrial Technology (ICIT), Seville, Spain: IEEE, pp. 1694–1699, 2015.

[7]. K. F. Kong Wong, V. Yap and T. P. Chiong, "Hardware accelerator implementation on FPGA for video processing," in IEEE Conf. on Open Systems (ICOS), Kuching, Malaysia, pp. 47–51, 2013.

[8]. A. L. Sangiovanni-Vincentelli et al. "Defining Platform-Based Design," In EEDesign. Available at www.eedesign.com/story/OEG20020204S0062). (Accessed on 23rd March, 2022) 84

[9]. L. Kechiche, L. Touil and B. Ouni, "Real-time image and video processing: Method and architecture," in 2nd Int. Conf. on Advanced Technologies for Signal and Image Processing (ATSIP), IEEE, Monastir, Tunisia, pp. 194–199, 2016.

[10]. J. G. Pandey, A. Karmakar and S. Gurunarayanan, "Architectures and algorithms for image and video processing using FPGA-based platform," in 18th Int. Sym. on VLSI Design and Test (VDAT), IEEE, pp. 1, 2014.

[11]. J. Rettkowski, A. Boutros and D. Göhringer, "HW/SW co-design of the HOG algorithm on a Xilinx Zynq SoC," Journal of Parallel and Distributed Computing, vol. 109, pp. 50–62, 2017.

[12]. S. Madhava Prabhu and S. Verma, "A Comprehensive Survey on Implementation of Image Processing Algorithms using FPGA," 2020 5th IEEE International Conference on Recent Advances and Innovations in Engineering (ICRAIE), 2020, pp. 1-6, doi: 10.1109/ICRAIE51050.2020.9358384.

[13]. Ali Azarian, Mahmood Ahmadi, "Reconfigurable Computing Architecture: Survey and introduction," in 2nd International Conference on Computer Science and Information Technology, IEEE, Beijing, China, pp. 269–27, 2009.

[14]. A. DeHon, "Reconfigurable Architectures for General-Purpose Computing", Technical Report Massachusetts Institute of Technology, 1996.

[15]. I. Kuon, R. Tessier and J. Rose, "FPGA Architecture: Survey and Challenges", J. Found. and Trends in Electronic Design Automation, vol. 2, no. 2, pp. 135-253, 2008.

[16]. K. Compton and S. Hauck, "Reconfigurable computing: A survey of systems and software", ACM Computing Surveys, vol. 34, no. 2, pp. 171-211, 2002.

[17]. R. Cumplido, M. Gokhale and M. Huebner, "Guest Editorial: Special issue on Reconfigurable Computing and FPGA technology," Journal of Parallel and Distributed Computing, vol. 133, pp. 359–361, 2019.

[18]. C. Claus, W. Stechele and A. Herkersdorf, "Autovision – A run-time reconfigurable MPSoC architecture for future driver assistance systems," IT- Information Technology, vol. 49, no. 3, pp. 181–187, 2007. 85

[19]. C. Khongprasongsiri, P. Kumhom, W. Suwansantisuk, T. Chotikawanid, S. Chumpol et al., "A hardware implementation for real-time lane detection using high-level synthesis," in International Workshop on Advanced Image Technology (IWAIT), Chiang Mai, Thailand: IEEE, pp. 1–4, 2018.

[20]. D. G. Bailey, "Image processing using FPGAs," Journal of Imaging, vol. 5, no. 53, pp. 1–4, 2019.

[21]. M. Kowalczyk, D. Przewlocka and T. Krvjak, "Real-time implementation of contextual image processing operations for 4K video stream in Zynq UltraScale+ MPSoC," in Conf. on Design and Architectures for Signal and Image Processing (DASIP), Porto, Portugal, pp. 37–42, 2018.

[22]. A. B. Amara, E. Pissaloux and M. Atri, "Sobel edge detection system design and integration on an FPGA based HD video streaming architecture," in 11th Int. Design & Test Sym. (IDT), Hammamet, Tunisia, pp. 160–164, 2016.

[23]. E. Onat, "FPGA implementation of real time video signal processing using Sobel, Robert, Prewitt and Laplacian filters," in 25th Signal Processing and Communications Applications Conf. (SIU), Antalya, Turkey, pp. 1–4, 2017.

[24]. R. Tessier, I. Kuon, J. Rose, "FPGA architecture: survey and challenges," Found. Trends Electron. Des. Autom. 2(2), pp. 135–253, 2008.

[25]. Y. Fang, L. Yu and S. Fei, "An Improved Moving Tracking Algorithm With Multiple Information Fusion Based on 3D Sensors," in IEEE Access, vol. 8, pp. 142295-142302, 2020, doi: 10.1109/ACCESS.2020.3008435.

Cite this article as :

R. Ajay Krishnaraju, J. Poovarasan, S. Santhies Kumar, S. Surya, P. Tamilselvan, "Object Detection Using Adaptive Block Partition and RCNN Algorithm", International Journal of Scientific Research in Science and Technology (IJSRST), Online ISSN : 2395-602X, Print ISSN : 2395-6011, Volume 10 Issue 3, pp. 1009-1023, May-June 2023. Available at doi : https://doi.org/10.32628/IJSRST523103196
Journal URL : https://ijsrst.com/IJSRST523103196