ISSN: 0976-9102 (ONLINE)
DOI: 10.21917/ijivp.2016.0200

REAL-TIME OBJECT DETECTION IN PARALLEL THROUGH ATOMIC TRANSACTIONS

K. Sivakumar[1] and P. Shanmugapriya[2]
[1]*Department of Computer Science Engineering, Sri Chandrasekharendra Saraswathi Viswa Mahavidyalaya University, India*
E-mail: [1]mailtosivacse@gmail.com
[2]*Department of Information Technology, Sri Chandrasekharendra Saraswathi Viswa Mahavidyalaya University, India*
E-mail: [2]priya_prakasam@yahoo.co.in

Abstract

Object detection and tracking is important operation involved in embedded systems like video surveillance, Traffic monitoring, campus security system, machine vision applications and other areas. Detecting and tracking multiple objects in a video or image is challenging problem in machine vision and computer vision based embedded systems. Implementation of such an object detection and tracking systems are done in sequential way of processing and also it was implemented using hardware synthesize tools like verilog HDL with FPGA, achieves considerably lesser performance in speed and it does support lesser atomic transactions. There are many object detection and tracking algorithm were proposed and implemented, among them background subtraction is one of them. This paper proposes an implementation of detecting and tracking multiple objects based on background subtraction algorithm using java and .NET and also discuss about the architecture concept for object detection through atomic transactional, modern hardware synthesizes language called Bluespec.

Keywords:

Object Detection, Atomic Transaction, Parallel, Background Subtraction, AdaBoost.

1. INTRODUCTION

Real-time object detection and tracking in real time video and static images means extraction of information from image or a sequence of frames, processing of information, determining whether information contains in a particular object and its exact location in the image [3]. Real time object detection is important operation used in various applications such as campus security system, automatic product quality and quantity checking, intelligent transportation system, [15], video surveillances and other areas [4]. Implementing these kind of operations is computationally intensive, need high performance computing and several attempts have been made to design hardware-based object detection algorithms, especially in the context of embedded and real time systems [4].

Traditional way of objects detection and tracking operations are done using sequential processing in hardware as well as software. These are very time consuming and cannot be used in high speed computing [2] like navigational, intelligent transportation systems, quality and quantity checking in manufacturing system and video surveillance system and many more.

There are various types, approaches, models and performances [19] [22] for object detection, recognition, and tracking were introduced and studied.

In video surveillance tracking multiple human objects in video is unavoidable, usually this could be done through connecting multiple camera and it should processed with serial one [11]. In Unmanned Aerial Vehicle for Road detection kind of work [14][18], may need fast processing and improvised algorithm in terms of parallel and hardware and software approaches. In [21], they proposed object detection based on the identifying foreground and background subtraction, then subtracted image could be used to track the object using connected components. This method achieves 79% accuracy and it eliminates shadows. This work was implemented in sequential processing.

Keeping this in mind there were various hardware and software based approaches have been proposed and implemented to detect and track the real time objects [1]. However many of the proposed methods were developed for object detection and tracking in hardware based [1], FPGA with verilog HDL approach over software approach [1]. The most of the proposed work uses FPGA-Field Programmable Gate Array with Xilinx for object detection implementation is concern. Through this we can reconfigure the system whenever we need, achieve good performance with lesser time and lower power consumptions.

So that the Hardware must always works highly in atomic transactions, it means that hardware elements should work with highly independent. An atomic transaction is the core of the hardware design technology. Atomic transactions simplify complex concurrency, improve the communication between modules, and elevate the description and synthesis of system, control [6]. Bluespec [6] is the only technology providing such a solution and also it provide hardware modeling, verification and rapid prototype design [6]. The bluespec also having concurrent rule execution and scheduling rules into clocks [13]. To get maximum performance of any operations using bluespec, we might use to execute many rules as possible concurrently [13]

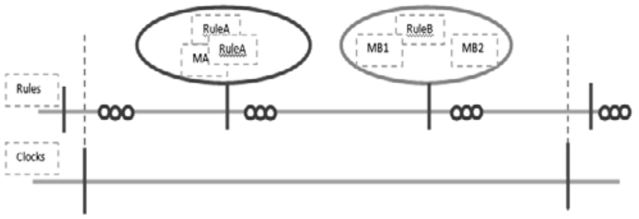


Fig.1. Rules and number clocks

In BSV, we abstract out the hardware-specific constraints into a simple semantic model of scheduling constraints on pairs of methods. The Fig.1 shows, how more than one rules executing from many methods concurrently that means many rules running within a single clock [13].

There is various object detection and tracking algorithms [19][22] exited, background subtraction is the one among many and most popular robust method for detecting moving objects and it is also a fundamental task in surveillance applications and transportations system [5]. Background subtraction algorithm uses a background image which is stored in memory [1] and foreground images which is dynamic scene read from video stream. Input video stream is transformed into frames and each frame is subtracted from the background image to detect the moving object region [1]. But there is challenges in background subtraction are still far from solved due to dynamic background, sudden changes in illumination [15][5] and many more.

The proposed method presents parallel architecture [16] for detecting moving and static object based on background subtraction algorithm which utilize adaBoost learning algorithm introduced by Freund and Schapire [7][8]. This work also extends to implementation of adaBoost algorithm using Bluespec. The adaBoost [3] work is based on a massively parallel computation of the classification engine using a systolic array implementation which gives extremely high detection frames per second (fps) [3], an also it is designed in such a way as to boost parallel computation of the classifiers used in the algorithm, and parallelize integral image computation, reducing the frequency of memory access [3]. To make the architecture scalable in terms of image sizes, we utilize an image pyramid generation module in conjunction with the systolic array.

The proposed paper chapters are organized as follows. In the section 1, detailed introductions are given, section 2, describe the related work in details and its merits and demerits discussed. Detailed background subtraction algorithm given, proposed architecture details are explained in section 3, Algorithm and implementation are given in section 4. Experimental results and analysis are given in section 5; conclusion and future directives are given in the section 6.

2. RELATED WORK

There are lot of proposed work, based on background subtraction methods has been carried out in real-time object detection field, but most of the works are related to traditional way of computing that is processing each frame by frame in sequential ways; both in hardware and software approach. This will increase execution time of object detection this will lead slowdown the object recognition and other image operations. In this section we will see about the related works which is already implemented and also discuss about the merits and demerits of the each work.

In [1], their work FPGA based moving object detection based on the background subtraction algorithm was adapted, and sobbed edge detection added in the architecture. This works could also extend to classify the objects based on their object shapes, but it was not implemented in parallel detection.

In [9] Y. Dedeoglu in his work is capable of classifying the objects based on its shape [9]. This system is works well in indoor and outdoor environments and also capable of detecting object while changing illuminious conditions [9]. This proposed work detects the object by processing frame by frame in sequential way. But it does not support parallel processing in both hardware and software approach.

In Yi Zhang, Tao Li and Jungang Han [10], their work background subtraction, Haar features detection and PAAG method are employed. They did not consider the motion vectors, and it takes too much time to process foreground object in the video frames at the same time. But there is in need for do some optimization in implementation.

In [12][1], his implementation, the work is capable of processing frames at very high rate (high fps) in a single low cost FPGA chip which suits for many real time motion detection application. But it was implemented in low cost FPGA board, which has less memory, BRAM and others.

In [17], used adaptive background subtraction and eliminate the shadows using RGB color space. This work was implemented in for both indoor and outdoor environments. VGA CCTV, infrared camera was used to implement this work. It uses intel core 2 duo processor running on 2.26 GHz, 2GB RAM for implementation and running time is less than 40 milli seconds.

The main objective of this work is to a). Implement Object detection using java and .net through serial and parallel computing approach and compare time efficiency. b). Propose a new methods to implement a very efficient background subtraction algorithm in a parallel processing way by applying AdaBoost based algorithm [3]. c). These works also propose an introduction of an atomic transaction technology which is available in bluespec. d). Reduce memory access time by applying efficient algorithm. e). Detect multiple object at the same time with high speed with good accuracy. e). compares the execution time of object detection done by using Java with the expected execution time from bluespec. In the next section we will see the architecture and it working principle.

3. PROPOSED METHOD AND ARCHITECTURE

3.1 GENERAL BACKGROUND SUBTRACTION ALGORITHM

Let us consider the background image pixel at *bx*, *by* co-ordinate at the time *tb* as *BItb*(*bx*,*by*), foreground image pixel at *fx*, *fy* co-ordinate at the time *tf* as *FItf*(*fx*,*fy*), Difference between background and foreground pixel at *dx*, *dy* co-ordinate at the time *td* as *DItd*(*dx*,*dy*) is as follows. Calculate the difference between background and foreground pixel value. If the difference is greater than threshold *T*, then the pixel will be consider as background pixel, else if the difference is greater than or equal threshold then the pixel will be foreground.

$$DIt(xd,yd) = 1, \text{ if } FI(fx,fy) - BIi\text{-}1(bx,by) <= T$$

$$DIt(xd,yd) = 0, \text{ if } FI(fx,fy) - BIi\text{-}1(bx,by) >= T$$

3.2 GENERAL ARCHITECTURE

In the proposed architecture, there is an input section, where the two input are feed in to the sliding windows method. These two inputs are, background image is the one where the camera is fixed and Foreground image is the one where objects are moving. Then sliding windows will split the windows into equal division of size 8×8, 16×16, 24×24, and maximum level as possible, it depends upon the size of the image.

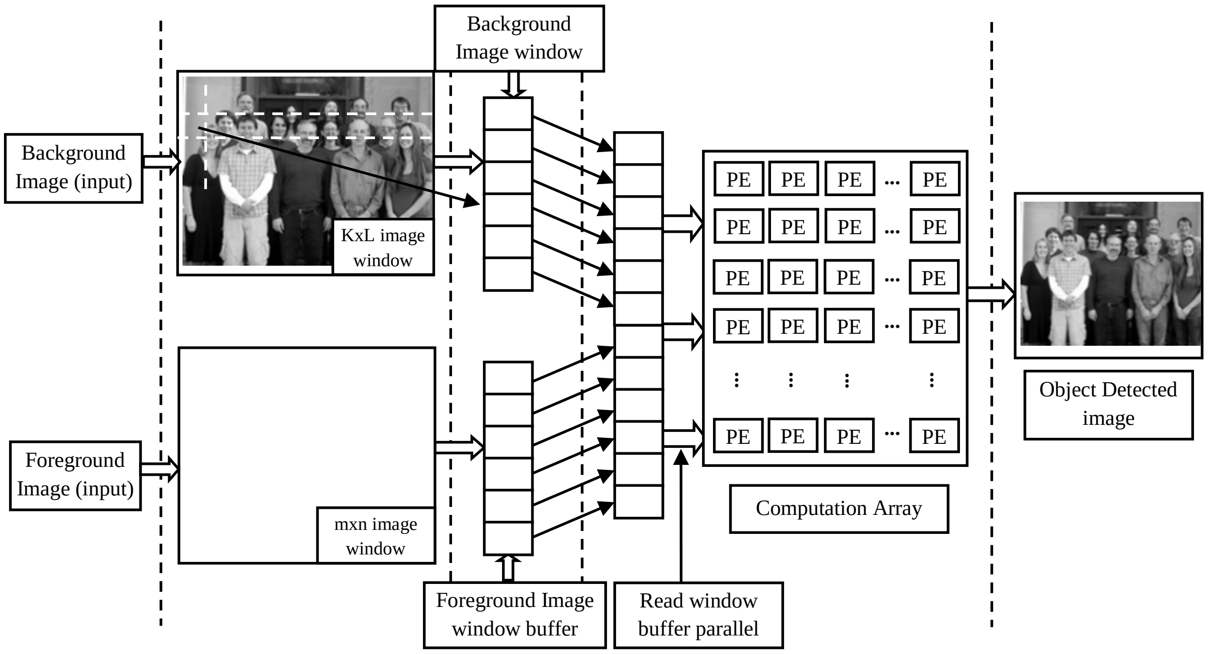
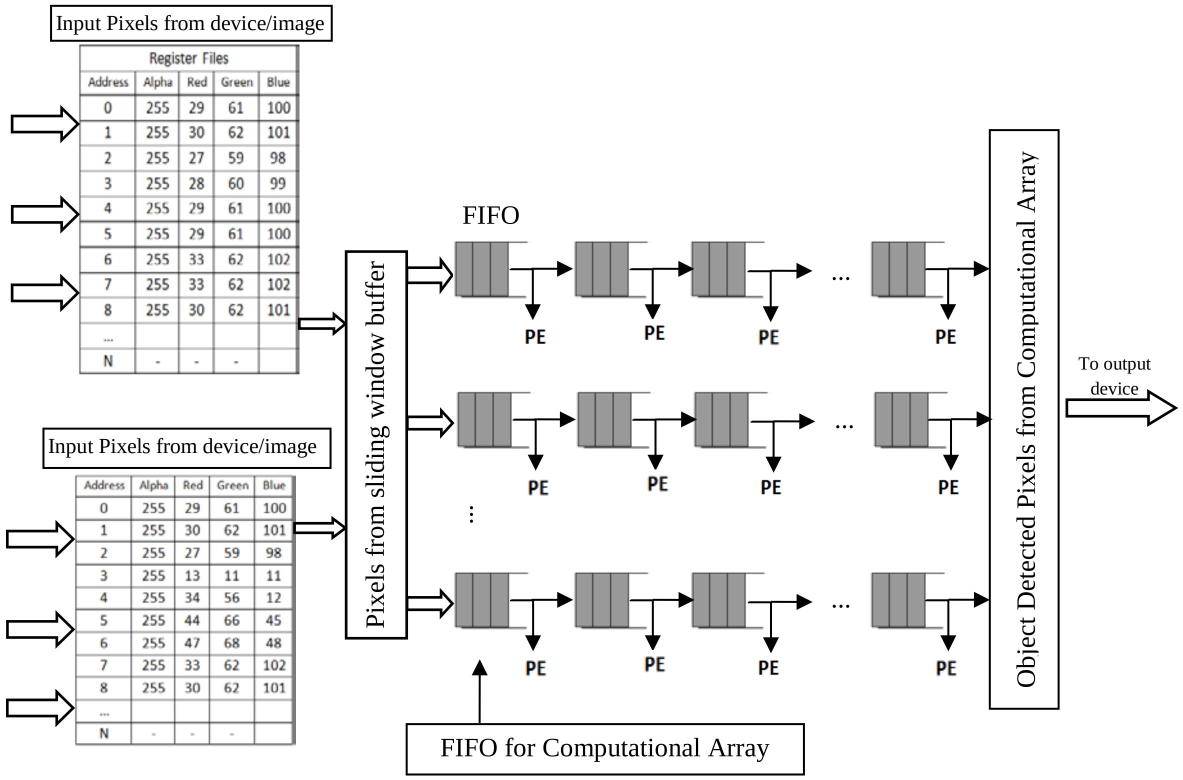


Fig.2. General Architecture


Input Pixels from device/image
Pixels from sliding window buffer
FIFO
PE
FIFO for Computational Array
Object Detected Pixels from Computational Array
To output device

Input Pixels from device/image

Register Files				
Address	Alpha	Red	Green	Blue
0	255	29	61	100
1	255	30	62	101
2	255	27	59	98
3	255	28	60	99
4	255	29	61	100
5	255	29	61	100
6	255	33	62	102
7	255	33	62	102
8	255	30	62	101
...				
N	-	-	-	

Input Pixels from device/image

Address	Alpha	Red	Green	Blue
0	255	29	61	100
1	255	30	62	101
2	255	27	59	98
3	255	13	11	11
4	255	34	56	12
5	255	44	66	45
6	255	47	68	48
7	255	33	62	102
8	255	30	62	101
...				
N	-	-	-	

Fig.3. Parallel Process and Pipeline

Example: 17×51, 255×255, 255×500, 1024×1024 etc. The windows of $M \times N$ size of the background image pixels and $K \times L$ size of foreground image pixels are feed into windows buffer for further read by the computational array buffer in parallel. Then the processing elements will read all the data from the buffer, and then it perform the background and foreground subtraction operations in parallel. All the processing elements are performing subtraction operations concurrently. Then it will check with threshold values, based on the result the objects are detected and images constructed.

3.3 PARALLEL PROCESSING COMPONENT DESIGN

The processing elements are placed in the computational array or systolic array, in row by column fashion and it is also an important concept involved in this proposed work, because it perform the subtraction operations in parallel. The parallel processing elements need FIFO. It is the one of the important component, which can be used to feed pixels from window buffer to the processing elements and also can be used to implement pipeline processing. The pipeline will increase the performance of the pixels processing at large, in lesser time. In bluespec, the pipeline concept can be implemented very easy, because it supports atomic transaction [6]. The processed pixels elements are feed to output buffer that will again feed to the output device for display.

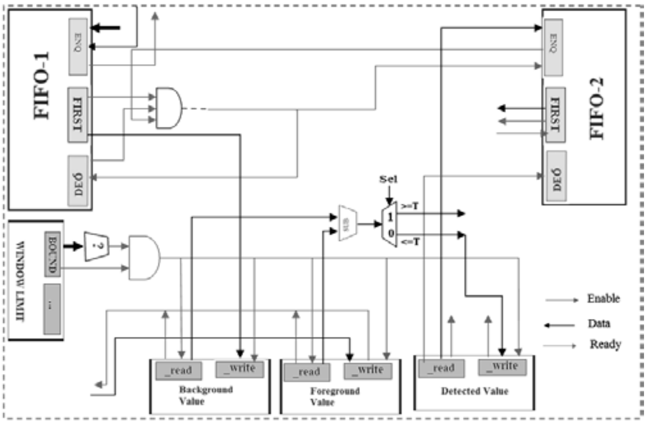


Fig.4. Atomic transactions elements and its signals

3.4 PROCESSING ELEMENT DESIGN

The processing elements consist of FIFO's to receive and feed data between FIFO's and processing elements; there is window limit module which will check the limit of window of m×n size [3]. The bound limit will enable the ready signal to read/write the background and foreground image pixels register. Background and foreground image pixels could also be read when the ready signal activated in the conditional module. The subtraction unit is used to find out the difference between background and foreground value, then it is feed to selection units for checking with threshold value for decision making, whether it is foreground or background value. Whenever the background and foreground register ready signal is on, then the value will be return to FIFO's, the same operations are repeated for all the pixels values being processed, which are present in the sliding windows.

3.5 STORE PROCESSED PIXELS INTO REGISTER FILES

Finally all the processed pixels need to store and feed into display units, for that we used register files. The register file consists of clock, ReadData, WriteData, ReadSel and WriteSel signals. These signals are used to read and write the data from to the register files, some register file have one or two read port and one write port. Parallel read and write operations are coordinated by rules and its conditions.

4. ALGORITHMS AND IMPLEMENTATION

This section discusses about the object detection using Java and extends the work by proposing new architecture using bluespec simulator for object detection.

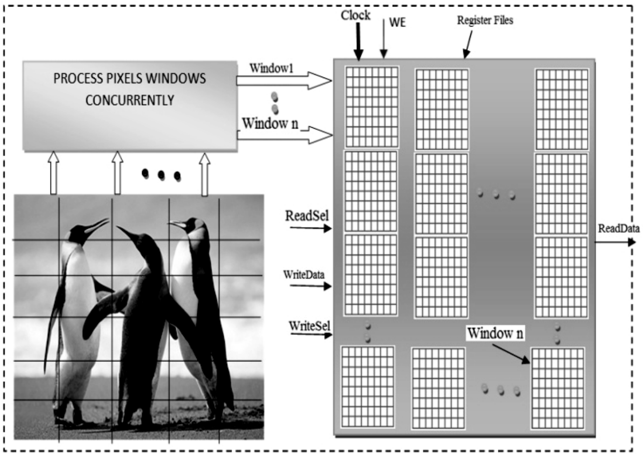


Fig.5. Processed pixels and register files

This proposed method can use java as the front end tools to give input images and give signals to the bluespec simulator [6] to start process and stop.

The atomic transactions are the key success to the bluespec [6], with the help of the atomic transaction we will coordinate the signals. The Bluespec is used for central processing unit to perform all the operations by splitting image into many individual equal sized window, subtracting background image pixel with foreground pixels, checking subtracted result with threshold value, construct the detected image from the result. The result is feed in to the register file using FIFO's for display purpose. These operations are performed concurrently to achieve high performance.

4.1 PESUDO ALGORITHM

4.1.1 Read Pixels from Image and Write into Register File:

```
package pk_readpixels
module mod_readpixels
rule rl_readpixels
for i=0 i<=image_width; i++)
for i=0 i<=image_height; i++)
getpixels(i,j)
put pixel into (registerfile(i))
end for
end for
endrule
endmodule
endpackage
```

4.1.2 Read Pixels from Register File and Process:

```
Package pk_split_image_window
Module mod_split
Rule rl_readpixels
For i=0 i<=image_width; i++)
For i=0 i<=image_height; i++)
getpixels from register file one by one and construct equal sized
Window and it is feed into FIFO in pipeline
```

```
process each windows pixels separate rule
End for
End for
endrule
endmodule
endmodule
endpackage
```

4.1.3 Read Each Window Pixels and Apply Subtraction:

```
package pk_process_window
module mod_process
Rule rl_readwindow(check the boundary formfirst to last window)
For i=0 i<=image_win_width; i++)
For i=0 i<=image__win_height; i++)
Read each pixel from window and apply subtraction
Check result with threshold value
If result>=T then
Pixel is background
Write pixel into register file
Call display image into the output
else if result<=T
Pixel is foreground
Call display image into the output
End for
end for
endrule
endmodule
endmodule
endpackag
```

The given algorithm is not included all the operation like clock, readsel, writesel and many other signal which is related to the bluespec atomic transactions is concerned. During implementation we can use all the signals to coordinate to achieve highly parallel operations.

5. EXPERIMENTAL RESULTS AND ANALYSIS

The experimental setup can design and implemented using java and .net c#. The input pixels data is read from the source using java front end interface that will be used for input to the proposed method. The rule construct can be used to read all the data from the text file generated from the java. This pixels data area loaded in to the register file. The M×N sizes of window data are constructing in parallel, and then these data will feed in to the FIFO. The FIFO is responsible for parallelizing the windows sized pixels being processed. The processing element will read the data and perform background subtraction and store the result into the data register file again for display purpose. The following diagram shows the Experimental result done from completely from java and Java. Then the execution times are noted for the fixed size image. The image size and running times are tabulated.

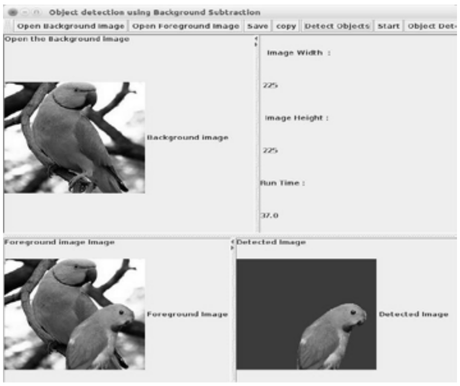


Image Width :

225

Image Height :

225

Run Time :

37.0

Fig.6. Object Detected Image

More number of Images is tested and analyzed its execution time for varying sizes of images. This method gave considerable performance. We tested with 255×255, 300×300, 500×400, 1024×1024 sizes of images it gave considerable result. We can implement this experiment for the very large images by scaling up and scaling down the images sizes during window split operations. This work was implemented using Intel and AMD processor, 32, 64 bit machine with the following configurations, Intel core i5 processor 3.1 GHz, 4 GB RAM 64 bit OS, AMD Dual core processor 2.10 GHz 2 GB RAM with 32-bit OS.

The Table.1 shows the execution time of the object detection using java and proposed execution column show that the expected execution time of the object detection using java and .net in sequential and parallel mode. The same could be used to implement in bluespec for real-time systems.

5.1 ANALYSIS OF TIME EFFICIENCY BETWEEN JAVA AND .NET

From the Table.2 and Fig.7 we can observed that the running time of java and .net for background subtraction gave equal some point it less than the .net. From Fig.8 we could observed that the running time of the .net for both sequential and parallel (multi-threading) processing. Parallel approaches gave highest performance than the sequential processing. This work was compared with the existing work which was implemented in various processors. We noted that if we increase the number of core the performance of both sequential and parallel algorithms for object detection.

6. CONCLUSIONS AND FUTURE WORK

Implementation of object detection using java and .NET achieves good result. This method uses fixed and variable size images and Videos, both foreground and background image size should be equal. The implementation of the object detection in hardware is left to the future development using bluespec. This proposed method of background subtraction based object detection and tracking in parallel can achieves good result and reduces the processing time and power consumption, because of multi-threading concept used in .NET the same could be used to implement using bluespec to achieve highest performance through atomic transactions. The main advantages of this work is to process all the pixels from the image or videos in parallel by splitting the whole image or video frames in to equal sized window and processed windows separately in parallel, instead of what we have done in traditional way of sequential processing. In real-time moving camera, there is changing of illumination variation in the background is present; to avoid this we can change the threshold value during run-time. This work is left to the development and enhancement of object detection in parallel for region based; histogram based optical flow, frame difference and other methods. There are numbers of object detection and tracking implementation can be possible using bluespec with FPGA technologies to accelerate video processing in real time embedded system.

Table.1. Object Detection Execution and Result using java and .NET with sequential and parallel

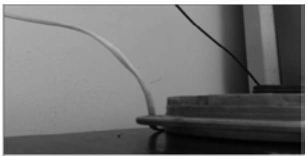
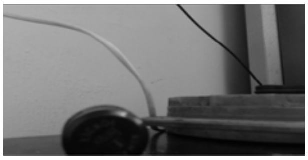
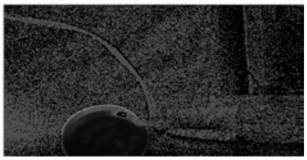
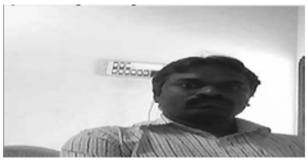
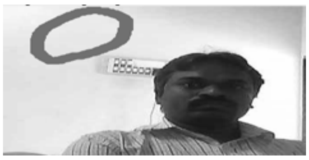
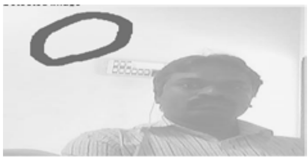
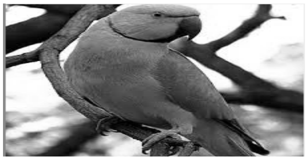
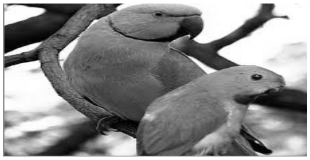
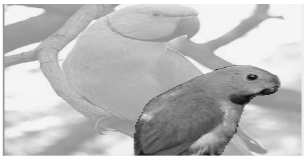
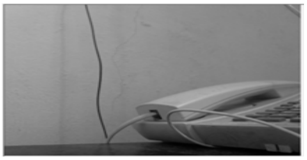
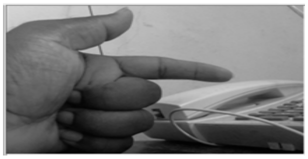
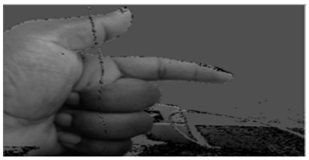
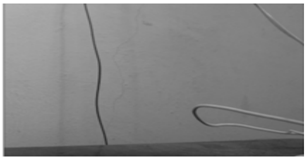
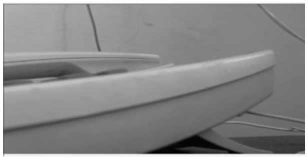
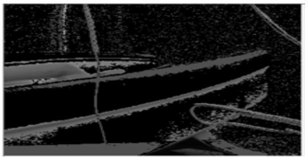
Ref.no	Image Type	Image Size	Background Image	Foreground Image	Detected Image
1	Small box	320×240			
2	Person image	352×352			
3	Bird	225×225			
4	Hand	320×240			
5	Telephone	320×240			

Table.2. Execution and Result Analysis

Ref. No.	Image Type	Image Size	Language Used for Implementation	Running Time in Milliseconds	
				Sequential Processing	Parallel Processing
1	Person Image	352×352	Java	31	--
			.NET	37	16
2	Parrot	225×225	Java	22	--
			.NET	21	11
3	Plastic Round Box	320×240	Java	132	--
			.NET	105	65
4	Hand	320×240	Java	195	--
			.net	186	53
5	Telephone	320×240	Java	180	--
			.net	161	47
6	List of Books	320×240	Java	199	--
			.net	180	59

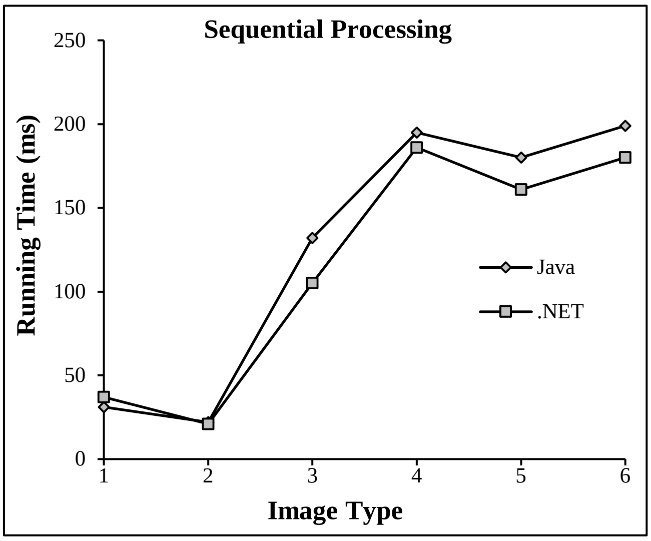


Fig.7. Result Analyses (Java & .NET)

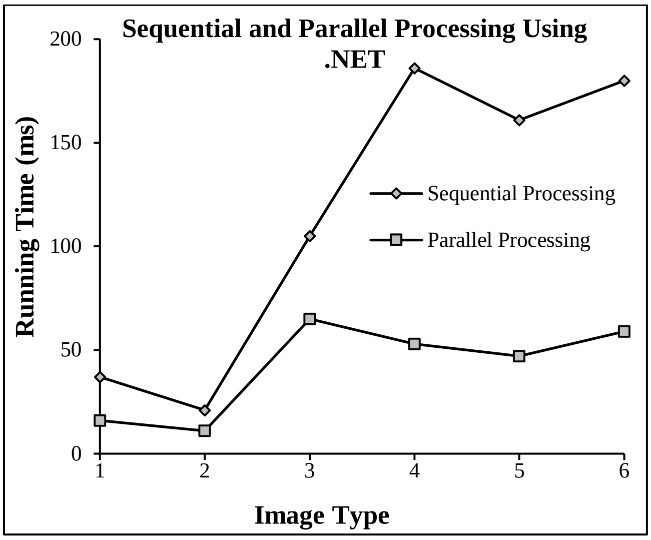


Fig.8. Result Analysis (.NET-Parallel and Sequential)

REFERENCES

[1] Gujrathi, Poonam, R. Arokia Priya, and P. Malathi, "Detecting moving Object using Background Subtraction Algorithm in FPGA", *Proceedings of 4th International Conference on Advances in Computing and* Communications, pp. 117-120, 2014.

[2] S.M. Rasiq and S. Krishnakumar, "Parallel Processors for High Speed Multiple Objects Recognition", *Proceedings of nternational Conference on Contemporary Computing and Informatics*, Vol. 19, No. 6, pp. 651-656, 2014.

[3] Christos Kyrkou and Theocharis Theocharides, "A Flexible Parallel Hardware Architecture for AdaBoost-based Real-Time Object Detection", *IEEE Transactions on Very Large Scale Integration Systems*, pp. 1034-1047, 2011.

[4] Fabien Moutarde, Bogdan Stanciulescu and Amaury Breheret, "Real-Time Visual Detection of Vehicles and Pedestrians with New Efficient Adaboost Features", *Proceedings of IEEE International Conference on Intelligent Robots Systems*, 2008.

[5] Deepak Kumar Panda and Sukadev Meher. "Detection of moving Objects using Fuzzy Color Difference Histogram based Background Subtraction", *IEEE Signal Processing Letters*, Vol. 23, No. 1, pp. 45-49, 2016.

[6] Bluespec, Available at: http://www.bluespec.com/technology.html

[7] Y. Freund, R. Schapire and N. Abe, "A Short Introduction to Boosting", *Journal of Japanese Society for Artificial Intelligence*, Vol. 14, No. 5, pp. 771-780, 1999.

[8] Robert E. Schapire, "*The Boosting Approach to Machine Learning: An Overview*", Available at: http://www.lira.dist.unige.it/teaching/SINA/slides-current/boosting-survey.pdf.

[9] Yigithan Dedeoglu, "Moving Object Detection, Tracking and Classification for Smart Video Surveillance", Ph.D. Dissertation, Department of Computer Engineering and the Institute of Engineering and Science, Bilkent University, 2004.

[10] Yi Zhang, Tao Li and Jungang Han, "Parallel Foreground Object Detection on a Polymorphous Computer", *Proceedings of 10th IEEE International Conference on High Performance Computing and Communications*, pp. 1784-1789, 2013.

[11] A.K.S. Kushwaha, C.M. Sharma, M. Khare, R.K. Srivastava and A. Khare, "Automatic Multiple Human Detection and Tracking for Visual Surveillance System", *Proceedings of International Conference on Informatics, Electronics and Vision*, pp. 326-331, 2012.

[12] E.M. Saad, A. Hamdy and M.M. Abutaleb, "FPGA-based Implementation of a Low Cost and Area Real-Time Motion Detection", *Proceedings of 15th International Conference on Mixed Design of Integrated Circuits and Systems*, pp. 249-254, 2008.

[13] Joel S. Emer and Murali Vijayaraghavan, "Computer Architecture: A Constructive Approach- using Executable and Synthesizable Specifications", http://csg.csail.mit.edu/6.375/6_375_2016_www/resources/archbook_2015-08-25.pdf

[14] Hailing Zhou, Hui Kong, Lei We, Douglas Creighton and Saeid Nahavand, "Efficient Road Detection and Tracking for Unmanned Aerial Vehicle", *IEEE Transactions on Intelligent Transportation Systems*, Vol. 16, No. 1, pp. 297-309, 2015.

[15] Alvarez, José M. Álvarez and Antonio M. Lopez, "Road Detection based on Illuminant Invariance", *IEEE Transactions on Intelligent Transportation Systems*, Vol. 12, No. 2, pp. 184-193, 2011.

[16] Christos Kyrkou and Theocharis Theocharides, "A Parallel Hardware Architecture for Real-Time Object Detection with Support Vector Machines", *IEEE Transactions on Computers*, Vol. 61, No. 6, pp. 831-842, 2012.

[17] Saeed Vahabi Mashak, Behnam Hosseini, Musa Mokji, S.A.R. Abu-Bakar, "Background Subtraction for Object Det ection Under Varying Environments", *Proceedings of International Conference on Soft Computing and Pattern Recognition*, pp. 123-126, 2010.

[18] Hailing Zhou, Hui Kong, Lei Wei, Douglas Creighton, and Saeid Nahavandi, "Efficient Road Detection and Tracking for Unmanned Aerial Vehicle", *IEEE Transactions on*

Intelligent Transportation Systems, Vol. 16, No. 1, pp. 297-309, 2015.

[19] H.S. Parekh, D.G. Thakore and U.K. Jaliya, "A Survey on Object Detection and Tracking Methods", *International Journal of Innovative Research in Computer and Communication Engineering*, Vol. 2, No. 2, pp. 2970-2979, 2014.

[20] Sanjivani Shantaiya, Keshri Verma, and Kamal Mehta, "A Survey on Approaches of Object Detection", *International Journal of Computer Applications*, Vol. 65, No. 18, pp. 14-20, 2013.

[21] Asad Abdul Malik, Amaad Khalil and Hameed Ullah Khan, "Object Detection and Tracking using Background Subtraction and Connected Component Labeling", *International Journal of Computer Applications*, Vol. 75, No. 13, pp. 1-5, 2013.

[22] J. Komala Lakshmi and M. Punithavalli, "A Survey on Performance Evaluation of Object Detection Techniques in Digital Image Processing", *International Journal of Computer Science Issues*, Vol. 7, No. 6, pp. 86-94, 2010.