Indian Journal of Science and Technology, Vol 9(39), DOI: 10.17485/ijst/2016/v9i39/97339, October 2016
ISSN (Print) : 0974-6846
ISSN (Online) : 0974-5645

Improved FPGA Implementation of Real Time Modified Mean Shift Tracking Algorithm

Rajesh Rohilla* **and Rajiv Kapoor**

Department of Electronics and Communication Engineering, Delhi Technological University, Delhi, India; rajesh@dce.ac.in, rajiv@dce.ac.in

Abstract

Objectives: To modify and implement color based mean shift object detection and tracking algorithm utilizing both the parallel and sequential capabilities of Xilinx ZYNQ ZC-702 SoC in order to speed up tracking. **Method/ Statistical Analysis:** The parallel and sequential processing capabilities of Field Programmable Logic Array (FPGA) and Processing System (PS) respectively are utilized in order to have a standalone system that can be faster, reliable and efficient while tracking the object in real time. Operations such as reading and writing video, grabbing kernel from frame and mean shift vector computation are sequential in nature and are best suited for processing system where as the operations parallel in nature are best fit for FPGA and may include estimation of histogram, computation of weights and estimation of weighted histogram. Executing them in parallel helps in reducing the machine cycles and enhances the fps. **Findings:** The high computational power of the algorithm is met by collective use of hardware and software while keeping the resources available on FPGA in check. The modified mean shift tracking method helps in exploiting the parallel computation capability of the FPGA. The paper compares the results with various techniques implemented on different embedded boards and the frame processing rate is much better with proposed FPGA implementation of modified mean shift tracking algorithm. Further, the window size can be varied without affecting fps. **Application/Improvements:** The frame rate achieved by using both hardware and software simultaneously is considerably higher than achieved with earlier implementations.

Keywords: Hardware/Software, Hardware Implementation (FPGA), Mean Shift, Real Time

1. Introduction

Object tracking is the process of locating a moving object (or multiple objects) over time using a camera. Various algorithms have been proposed to achieve the objective with applications ranging from face detection, character recognition, visual position and tracking to aerial target location and identification etc. Visual tracking algorithms are computation intensive and many of these algorithms are less suitable for real time applications. Some of these algorithms are discussed in[1-4].

Though many of the algorithms have already been implemented on personal computer and embedded boards which may include anything from Digital Signal Processing (DSP) boards, Graphical Processing Unit (GPU), to Application Specific Integrated Devices (ASIC), and Field Programmable Gated Arrays (FPGA); yet the combination of the two (processor's sequential nature and embedded board's parallel nature) will be more practical. Combining the two spectrums will not only yield better results but will also be a standalone system that can cater to the real time application of the vision algorithms. Further, the resources required for the resource hogging and highly computational algorithms will be easily met. Many of the previously proposed techniques have been either done on sequential machines or on the hardware. Although, few have implemented using both but they have used higher resources with less fps. Contrary to this, our implementation achieves higher fps while keeping the options open for future expansion of the application.

The objective of the paper is to implement colour based mean shift object detection algorithm utilizing both the parallel and sequential capabilities of Xilinx ZYNQ ZC-702 SoC. FPGA can be configured to the

*Author for correspondence

required hardware to compute the task that may be parallel in nature where as the dedicated processor present on the SoC can process the task sequential in nature which on the other hand may not be parallelised easily. Not only the dedicated Processor eliminates the need of soft processor implemented using Programmable Logic on FPGA, it can also perform many arithmetic calculations faster as compared to the soft processor especially involving FLoating OPerations (FLOPs). Along with this, the resources taken to implement soft processor Intellectual Property (IP) are reduced which allows the designer to build an IP that may be resource hogging or the designer can also implement multiple IP simultaneously. Mean shift tracking algorithm in[1] is highly computational and resource consuming algorithm. In our implementation of the algorithm on FPGA, the grabbing of two consecutive frames from the video, formation of window on both the frames and computation of mean shift vector is done by RISC processor. On the other hand, making of the histograms, division and other arithmetic calculations to yield weighted histogram is performed by the hardware burnt on FPGA. This enables us to meet the available resources on FPGA and also allows us to achieve higher frames per second (fps).

Mean shift object detection is a non-parametric feature based algorithm that finds the displacement of an object in two consecutive frames by computing mean shift vector which essentially represents the displacement of the object in the second frame. The algorithm explained in[1] extracts the window from each frame and computes the histogram of each window. The two histograms are then used to find the weighted histogram which when convolved with the second window yields mean shift vector.

Implementing Mean shift tracking algorithm has been a tough challenge considering the computation power required while keeping it real time. Due to the high complexity of general purpose processors, sequential nature of Digital signal processors and higher power consumption of Graphical Processing Unit, they all do not serve well for the purpose. But the flexibility to reconfigure and extract the parallelism of FPGA has made it much more suitable to vision applications. The basic structure of FPGA and also the problems faced while executing the vision algorithms on the same is described in detail in[5]. Keeping in with FPGA[6] present a soft processor based architecture configured on it. It is capable of running mean shift tracking algorithm with two colour components in real-time at 25 fps in case of PAL and 30 fps in case of NTSC. But the soft processor essentially behaves as a sequential processor and thus the parallel capabilities of FPGA are not fully utilized. The mean-shift algorithm is implemented in[7] to utilize the advanced parallelism of FPGA EP1C6 of Altera Co and processed real-time PAL video of 720*576 at 25 fps for mobile robot application. The fps observed is nearly same but[7] essentially use the parallel nature of FPGA[8] proposed the system which incorporated XC2V8000-4CFF1152 FPGA mentioning 56. 38 fps while tracking multiple objects which is twice the fps observed by[6] and[7] however, to detect the object particle filter is used which consumes more resources and slows down the fps. This is further supported by operation time of 0. 0167s observed by[9] where the particle filter was implemented on FPGA to track the object. In[10] FPGA-based particles filter that takes advantage of evolutionary computation in order to estimate motion patterns is presented. The evolutionary algorithm, which has been included inside the re-sampling stage, mitigates the known sample impoverishment phenomenon, very common in particle-filtering systems. In addition, a hybrid mutation technique using two different mutation operators, each of them with a specific purpose, is used in order to enhance estimation results and make a more robust system. However, it takes high implementation time if we want higher particle numbers. In[11] Kalman filter has been implemented on ALTERA Cyclone II for tracking and can process 25 frames per second engaging 45% resources of Cyclone II. In[12] Xilinx Virtex-4 SX FPGA is used to process 25 frames per-second at a low resolution 128*128 pixels monochrome image. In[12] moving vehicles have been extracted from real-time camera images for the evaluation of traffic parameters, such as the number of vehicles, their direction of movement and their approximate speed, using low power hardware of a sensor network node. But the problem of low fps still persists. Along with this, the low resolution of images is a hindrance to its real time applications. Thus techniques used for implementation of vision algorithms on FPGA for real time applications either were not exploiting the parallel nature of FPGA (as in the case of soft processor) or were using lower solution video as in[9] or were having low fps. In[13] the parallel nature of FPGA was exploited and combined it with the sequential nature of processor to achieve higher fps with resolution of 360*288. It is proved by the fact that[13] achieved fps of 833 by employing colour based mean shift on Spartan 3E

using hardware configured on FPGA and soft processor simultaneously. He further showed the reduction in the amount of the resources required to achieve the objective. The main strategy thus is to explore parallelism of FPGA along with the sequential nature of processors to speed up the performance.

Our design uses both the sequential nature of processor and parallel nature of FPGA. The algorithm is implemented on Xilinx ZYNQ ZC-702 SoC which has a dedicated RISC processor unlike soft processor used by[13]. The objective is to implement the operations that consume high resources on dedicated processor which processes data faster than soft processor.

The paper is organized as follows: Section 2 discusses the design approach followed by processor and hardware implementation in section 3. Section 4 compares the performance with the previous implementations on FPGA. Finally, Section 5 discusses the experimental results and the paper is concluded in Section 6.

2. Design Approach

The Xilinx ZYNQ ZC-702 SoC is broadly classified into Processing Section (PS) and Programmable Logic (PL). Processing section consists of a dedicated sequential processor ARM Cortex A-9 to perform the software operations while the user custom designed hardware can be configured on PL. Using PS alone will decrease the speed while preferring only PL for vision algorithm may be resource hogging. Thus to implement algorithm fast and efficiently, the design on the used SoC is carefully divided between PS and PL to consume low resources while making sure to get higher speed.

Video used during our implementation has a frame resolution of 320*240 while window /kernel size used can be varied for better results. Window size used during implementation is 20*20. The mean shift vector is computed after 5 mean shift iterations instead of 20 used by[13] resulting in higher fps. Along with high fps, the resources are overcome by reducing the histogram bins from 256 to 16. However, the reduced histogram bins did not affect the results and provided satisfactory outcomes. Along with this, the multiplication operations performed to obtain mean shift vector is done on PS rather than on PL which may have not only consumed higher resources but would have been slower than the dedicated multiplier available on PS. Figure 1 shows the design overview of the partition of operations between PS and PL.

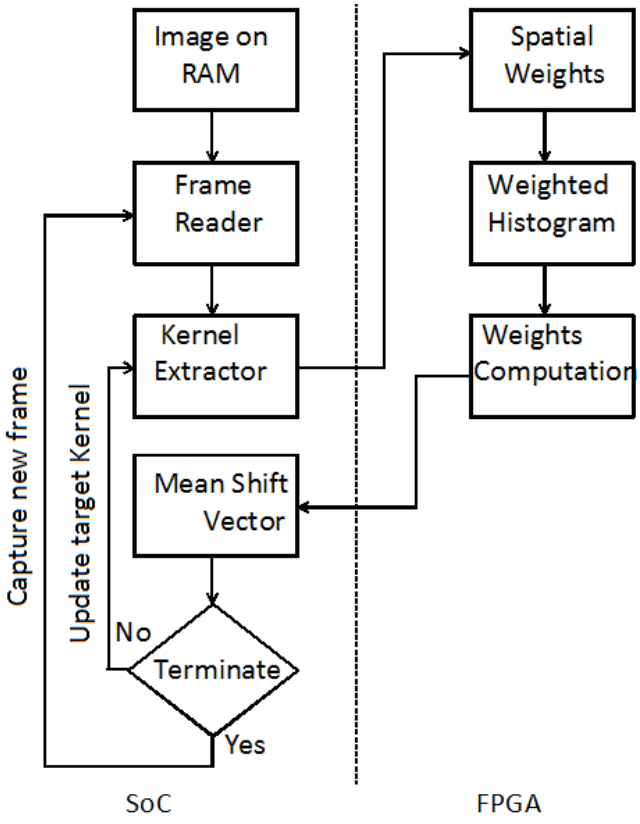

Figure 1. Hardware/software partitioning

3. Implementation

3.1 Processing Section

PS is best at performing sequential task faster than FPGA/PL and exploiting the sequential nature of PS will not only save a lot of operation time but the resources utilized are also less. Operations like reading video, grabbing frame from video, extracting candidate or target kernel from the captured frame and the decision of performing next mean shift iteration are all sequential tasks. Configuring FPGA to perform these tasks will not enhance the fps but may consume more resources. Same goes with the final computation of mean shift vector which involves a lot of multiplication operations. Using the dedicated multiplier of processor, we can obtain the values much faster. Figure 2 shows the operations performed by PS to send the kernel to PL and finally evaluate mean shift vector.

3.1.1 Kernel Extractor

PS loads the video from RAM of FPGA and then reads the video frame by frame. It then identifies the Region Of Interest (ROI) that essentially consists of the object to

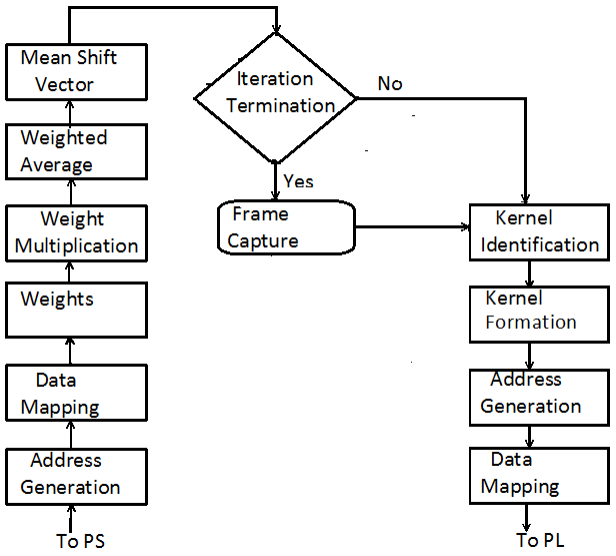

Figure 2. Processing section

be tracked. The frame captured contains candidate kernel of size 20*20 which is framed around ROI by the processor. The processor via AXI-4 Lite bus sends the kernel to FPGA by mapping the data onto the specific addresses generated by address generator. The address generator employed by PS involves a lot of shift operations which would have used a lot of resources had it been the soft processor. The subsequent frame is then read to generate target kernel of size 20*20 and the same operation is performed to send the target kernel to FPGA using address generator and mapping the data. Thus a lot of shift operations are used and using SoC, the task is done faster. The kernel sent by PS to FPGA is used to compute the weights explained in the next section.

3.1.2 Mean Shift Vector Computation

Once the processor obtains the weights from FPGA, it uses its dedicated multiplier to compute the mean shift vector. The intensity of each pixel in second frame is multiplied with its corresponding weight obtained from FPGA by PS. PS uses the result of multiplication to calculate mean shift vector. Following the computation of mean shift vector, the decision to perform the next iteration is taken by PS which might have used comparators if would have been configured on FPGA. Additionally, the process of decision making is a sequential operation and configuring FPGA for it gives no advantage of parallel nature of FPGA. Thus division of task is done carefully between SoC and FPGA while maintaining high fps but consuming less resources at the same time. Not only SoC performs the sequential task faster, the resources which might have been used by configuring soft processor on FPGA are not used now and provide us the flexibility to easily expand the IP in future.

3.2 Programmable Logic

The crux of any FPGA is the reconfigurable PL which can be programmed to any user hardware. Not only the custom hardware suits the user requirements by performing the task in parallel, it also gives user the flexibility to expand the window size or add newer operations in future. All this expansion comes at the cost of using more resources but unlike PS, it does not affect fps if hardware is designed properly. Therefore exploiting the parallel nature of FPGA will give higher fps.

1. ***Spatial Weight:*** The histogram of a kernel represents the number of pixels to the corresponding intensity. Thus in a normal histogram, each pixel of a particular intensity contributes a factor of 1. However, in a weighted histogram, not only the intensity but the spatial location also decides the contribution factor. The scalar factor decided on the basis of spatial location of a pixel is referred to as spatial weight. Thus pixel at the canter should be preferred over a pixel at the edges even if they have the same colour. The pixels laying at the centre gets their colour value multiplied by 4 indicating the higher preference they got over other pixels. The pixels farther away from the centre are multiplied by 2 and those which are at the edges are multiplied by factor of 1.

   Figure 3 shows the variation of spatial weight with respect to the distance from the central pixel. As the distance from the central pixel increases, the spatial weight decreases, there by indicating an inverse relationship between the two parameters. Figure 4 also illustrates this by assigning spatial weight of '4' to D8 pixels, a value of '2' to D16 pixels and a value of '1' to the remaining pixels. D8 pixels are closest to the central pixel and constitute the immediate pixels in horizontal, vertical or in diagonal directions directly in contact with the central pixel. D8 pixels form a square around central pixel with side equivalent to 3 pixels. Similarly, D16 pixels are the next neighbouring pixels forming a square around central pixel as shown in Figure 4.

   The weight assigned to the D8 pixels is higher than D16 pixels which in turn is higher than other pixels

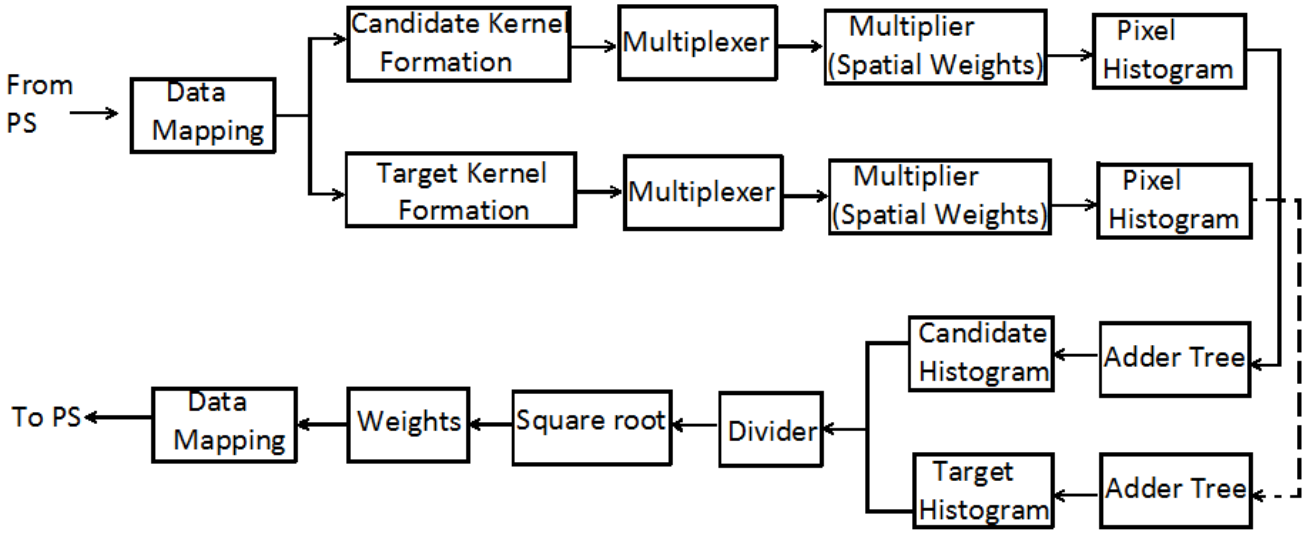

Figure 3. Block diagram showing weight computation logic over hardware

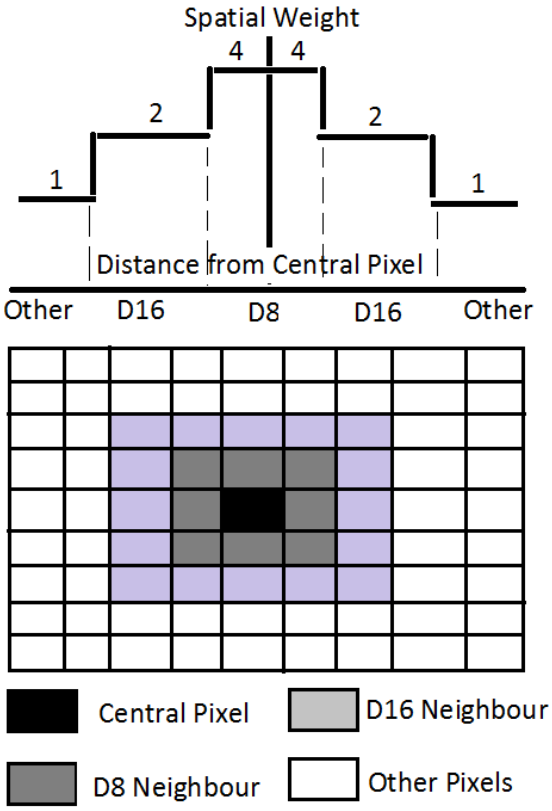

Figure 4. Spatial weight

present at the edges. This can be reasoned by more information present at the centre than at the edges. Hence, more weightage is given to the pixels at the centre than at the edges. The weightage of a pixel with respect to its distance from the central pixel is mapped by Gaussian function. However, the Gaussian function is a continuous function and cannot be realized on FPGA and is approximated by the function shown in Figure 4.

2. ***Histogram:*** In our implementation of hardware on FPGA, candidate and target kernel obtained from PS are used to form the candidate and target weighted histogram. The histogram computation starts with identifying the intensity value of each pixel with the help of multiplexer and a histogram bin is assigned to that pixel. The histogram of each pixel referred to as "Pixel Histogram" in Figure 5 is used to maximize the parallelization. Rather than extracting intensity value of each pixel one by one, all pixels have their corresponding multiplexer that evaluates intensity value in one cycle. This concept eliminates the shared multiplexer used by[13] and parallelizes the operation of computing the weighted histogram resulting in less operation time. The parallel nature obtained by computing histogram of all pixels of target and candidate kernel in one machine cycle would not have been possible on SoC and saves nearly 1500 machine cycles (799 multiplexer and 799 spatial weight multiplication cycles) in kernel of size 20*20. The histogram bin of each pixel has 16*2 bits. The 2 bits take in to consideration the contribution of spatial weight. This also explains the less number of mean shift iterations required over others providing an advantage of higher fps.
3. ***Back to PS:*** The histograms obtained of each pixel are added to give weighted histogram for both candidate and target kernel. The adder tree used here performs

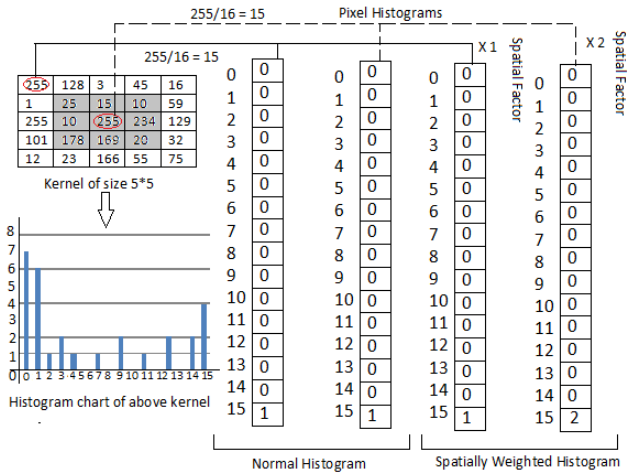

Figure 5. Pixel histogram

operations in nearly 10 machine cycles and saves 1000 addition machine cycles. The weighted histograms obtained from adder tree undergo division and square root operations on hardware and are executed in one division and one square root machine cycle. On PS, it would have taken15 additional division and square root machine cycles which internally involve lot of addition and multiplication operations. The output of square root is the weights which are fed to PS to finally compute the mean shift vector.

Coming back to the parallel nature of FPGA, our configured hardware performs all the operations-identifying pixel value, spatial weight multiplication and adding pixel histogram to obtain weighted histogram in one machine cycle. Additionally, each bin of two weighted histogram undergoes division and square root parallel to give final weights.

4. Performance Comparison

Various tracking implementations are summarised in Table 1.[13] achieved 25 fps in case of PAL and 30 fps in case of NTSC employing mean shift on sequential processor.

[7]achieves 25 fps on FPGA EP1C6 of Altera Co by again using mean -shift algorithm but did not extract full parallelization capability of FPGA. Using particle filter[8] mentioned 56.38 fps performed on XC2V8000-4CFF1152 FPGA whereas [12]used Xilinx Virtex-4SX FPGA on prototype board Virtex-4 Evaluation Kit from Avnet to process 25 frames per second data at low resolution 128*128 pixels monochrome image[13] achieved the highest fps of 833 using mean shift on Spartan 3-E

Table 1. Performance comparison

Device	Algorithm	Frame/s	Reference
Xilinx Virtex-4	Particle Filter	25	Marek Wjcikowski[12]
Altera EP1C6	Mean Shift	25	Xiaofeng Luand DiqiRen[7]
Xilinx Microblaze	Mean Shift	30	Usman Aliand M.B. Malik[6]
XC2V8000	Particle Filter	57	Jung Uk Cho[8]
Xilinx Spartan3-E	Mean Shift	833	Usman Aliand Mohammad Bilal Malik[13]
Xilinx ZYNQ ZC-702	MeanShift	2500	Our Algorithm

by incorporating both the processor and hardware to do so. The work done by us is also similar to[13] with the difference of: a) partitioning of hardware and software between SoC and FPGA. b) Along with the different mechanism of partitioning, soft processor employed by [13]is replaced by the on chip dedicated RISC processor. This allowed us to achieve operation time of 0. 08s for 200 frames resulting in 2500 fps, c) the number of iterations is also limited to 5 as compared to 20 by [13]due to alteration in evaluating weights, and d) lastly, the histogram bins are not using fixed representation but have flexible digits representation resulting in less binary values to be operated upon. All these factors contributed to low usage of resources with higher fps.

5. Experimental Results

Figure 6 shows the results applied on different test cases in which a man is tracked in various environments. The test results come out to be satisfactory and successful. Implementation of mean shift algorithm on Xilinx ZYNQ ZC-702 consumes considerable amount of resources available on FPGA. This can be attributed to the large amount of parallelization achieved at cost of higher resource consumption with high fps as the outcome.

Table 2 shows the resource utilization in detailed numbers. Each histogram to a pixel of target/kernel is represented by 16*2 bits. Weighted histogram requires 16*8 bits and weights are denoted by 16*16 bits. The algorithm is implemented on a man walking in front of a static background. The background includes a building, green grass, black roads and other people. The algorithm successfully tracked the man walking from one side of the building to the other side.

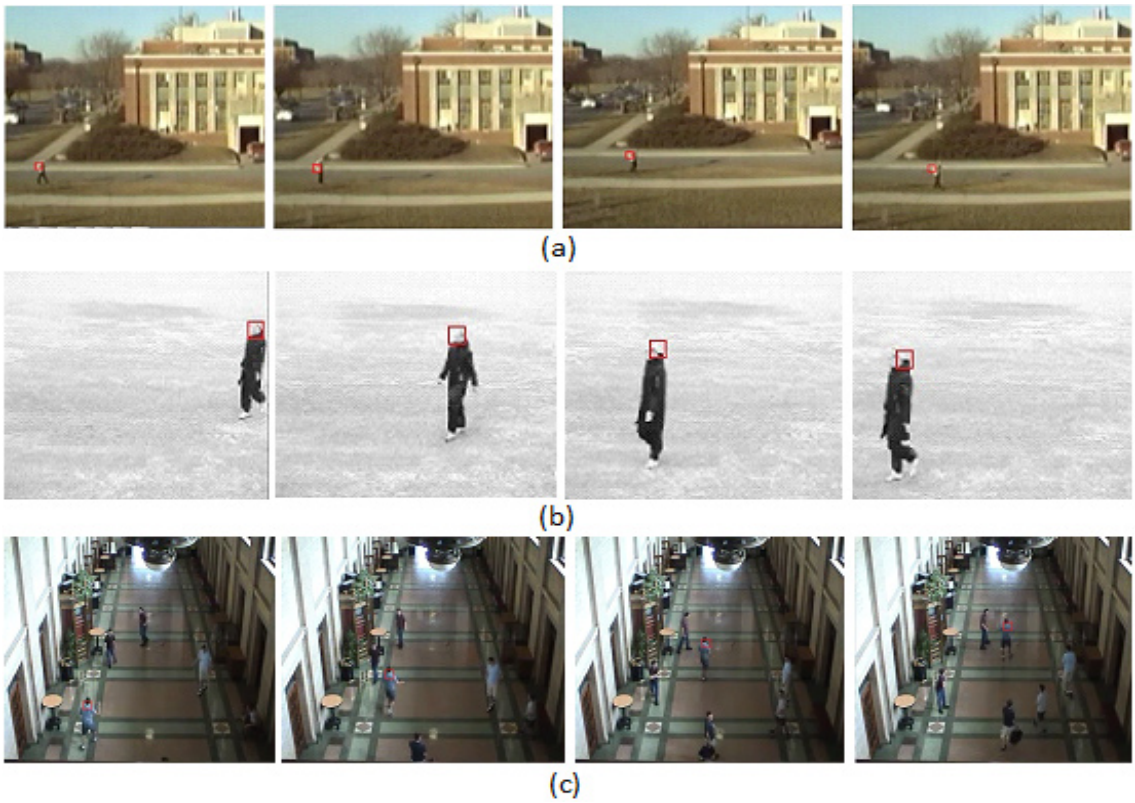

Figure 6. Frames showing the tracked man in different test cases

Table 2. Resource utilization

Resource	Available	Utilized	Percentage
LUT	53200	15965	30
BUFG	32	4	12
Register	106400	5008	5
Slice	13300	8090	60

6. Conclusion

In this paper, we have proposed a new hardware/software approach to the implementation of mean shift tracking on SoC. It uses the dedicated processor's sequential capability to extract kernel from frame and compute the mean shift vector quickly. The parallel nature of FPGA is also exploited to get the weights in fewer machine cycles. Thus combination of the two spectrums provided us the ability to keep resources low while getting higher fps. It is also expected that increase in size of kernel will not affect the fps provided the resources are kept in check.

Though the fps achieved is quite high and can be used for real time applications, the future possibilities always lead to improved efficiency and performance. The resolution of video used can be improved from 240p to 360p or 480p. Higher resolution will pave way for applications which require large details like number plate detection, face recognition in crowd etc.

7. References

1. Comaniciu D, Ramesh V, Meer P. Real-time tracking of non-rigid objects using mean shift. In: *Computer Vision and Pattern Recognition, Proceedings. IEEE Conference* on USA.2000, 2, p. 142–49.
2. Agarwal VK, Sivakumaran N, Naidu V. Six Object Tracking Algorithms: A Comparative Study. *Indian Journal of Science and Technology.* 2016; 9(30):1–9.
3. Altaf A, Raeisi A. Presenting an effective algorithm for tracking of moving object based on support vector machine. *Indian Journal of Science and Technology*. 2015 Aug 21; 8(17):1-7.
4. Chandrajit M, Girisha R, Vasudev T, Hemesh M. Data Association and Prediction for Tracking Multiple Objects. *Indian Journal of Science and Technology*. 2016 Sep 16; 9(33):1–13.
5. Johnston CT, Gribbon KT, Bailey DG. Implementing image processing algorithms on FPGAs. In: *Proceedings of the Eleventh Electronics New Zealand Conference, ENZCon*' New Zealand. 2004, p. 118–23.

6. Ali U, Malik MB, Munawar K. FPGA/soft-processor based real-time object tracking system. In: *Programmable Logic, SPL. 5th Southern Conference* on Pakistan, 2009 Apr, p. 33–37.
7. Lu X, Ren D, Yu S. FPGA-based real-time object tracking for mobile robot. In: *Audio Language and Image Processing (ICALIP), International Conference* on Shanghai University, 2010 Nov, p. 1657–62.
8. Cho JU, Jin SH, Pham XD, Jeon JW. Multiple objects tracking circuit using particle filters with multiple features. In: *Robotics and Automation, IEEE International Conference* on Korea, 2007 Apr, p. 4639–44.
9. Lu X, Wang S, Du Z, Pei D, Zheng D, Zuo T. Parallel Particle Filter Algorithm and Its FPGA Implementation. In: *International Conference on Computer, Communications and Information Technology* Atlantis Press China, 2014, p. 1–4.
10. Rodriguez A, Moreno F. Evolutionary Computing and Particle Filtering: A Hardware-Based Motion Estimation System. *IEEE Transactions on Computers*. 2015 Nov 1; 64(11):3140–52.
11. El Hajjouji I, El Mourabit A, Asrih Z, Mars S, Bernoussi B. FPGA based real-time lane detection and tracking implementation. In: *2016 International Conference on Electrical and Information Technologies (ICEIT),* IEEE, 2016 May 4, p. 186–190.
12. Wojcikowski M, Zaglewski R, Pankiewicz B. FPGA-based real-time implementation of detection algorithm for automatic traffic surveillance sensor network. *Journal of Signal Processing Systems*. 2012 Jul; 68(1):1–8.
13. Ali U, Malik MB. Hardware/software co-design of a real-time kernel based tracking system. *Journal of Systems Architecture*. 2010 Aug; 56(8):317–26.