

Real Time Object Tracking: Simulation and Implementation on FPGA Based Soft Processor

Manoj Pandey¹, Dorothi Borgohain², Gargi Baruah², J.S. Ubhi³,
and Kota Solomon Raju²

¹ B K Birla Institute of Engineering and Technology, Pilani-333031

² DSG, Council of Scientific and Industrial research (CSIR) –Central Electronics Engineering Research Institute (CEERI) CSIR-CEERI, Pilani-3330311,

³ ECE, Sant Longowal Institute of Engineering and Technology, Longowal, Sangrur, Pb
{manoj2pandey, js_ubhi}@yahoo.com, dorothi.s@gmail.com,
solomon@ceeri.ernet.in

Abstract. Adaptive systems are being easy to design using reconfiguration facility on Field programmable gate arrays (FPGAs). In this paper, Kernel based Mean shift algorithm is used for tracking a moving object. First it is simulated on Matlab and then implemented on microblaze soft processor based FPGA board. Tracking is observed for two similar objects crossing each other moving with uniform speed in a stored video as well as real time video. Object tracking, when it comes to implement on pure software (SW) in real time becomes difficult task due to certain limitations of SW. This paper shows how the mean shift algorithm is implemented on Xilinx Spartan 6 FPGA board using EDK. Once the complete algorithm is implemented on microblaze soft processor then some of the mathematical functions of algorithm are calculated on hardware to use HW-SW co-designing methodology to enhance the performance of the system.

Keywords: Kernel, Mean Shift, Real Time Tracking, EDK, FPGA, Spartan6.

1 Introduction

Object tracking is one of the fundamental component of computer vision that can be very beneficial in applications such as unmanned vehicles, surveillance, automated traffic control, biomedical image analysis and intelligent robots, to name a few. Tracking aims to generate the trajectory of objects across video frames. Object tracking is used for identifying the trajectory of moving object in video frame sequences. Like most computer vision tasks, object tracking involves intensive computation in order to extract the desired information from high volume video data. In addition, the real time processing requirements of different computer vision applications stress the need for high performance object tracking implementations. Implementation of vision systems in real time requires high performance HW with flexibility to incorporate the change after the design has been freed. GPPS and DSPs give flexibility but not high performance while ASICS gives performance but not flexibility.

Latest technologies give performance and flexibility of FPGAs more and rugged with the use of Reconfigurable Computing System (RCS) facilities. Additionally,

most FPGAs support dynamic partial reconfiguration (DPR) [1] to provide higher flexibility. With this benefit it is possible to reconfigure a part of the FPGA during run time. The other part of the FPGA is unaffected by this process and continues to run. Reconfigurable Computing offers cost-effective solution for computationally intensive application through reuse of hardware. Using dynamic programming, RCS is efficient in terms of hardware utilization without degrading its performance [2].

In this paper, we presents the simulation of Kernel based mean shift algorithm for object tracking and implementation on FPGA board. Objects as a person is used for the tracking, considering the case of overlapping and scale changing. The aim is to allow designers of applications that benefit from FPGA implementation, to leverage this capability for reconfigurable architecture. In Section 2, the tracking algorithms and its FPGA implementations are presented along with a review of existing hardware (HW) approaches. Section 3 gives basic steps of algorithm and 4 introduce about the HW IPs and their FPGA oriented design. Section 5, presents simulation and implementation results. Finally, Section 6 concludes this paper.

2 Related Work

Several implementations of object tracking for FPGAs exist. A soft-processor based object tracking system on FPGAs are carried out in ref [3, 4]. In this paper Xilinx 32 bit Microblaze soft processor is used to implement the tracking algorithm. The rest of the FPGA is used for the frame grabber and visualization of the video stream. A multi-object-tracking architecture for FPGAs or ASICs based on image segmentation is used in [5]. The algorithm is fixed one for a given constraints. If any constraints change after the design, the system does not work effectively. These systems have some restrictions. The main disadvantage is the restriction of the systems to one algorithm. If the constraints are changing the settings of an algorithm has to be changed. In the worst case the algorithm gets completely improper. To avoid these problems, the implementation of the system with reconfiguration facility is able to provide required change by replacing with additional flexibility in algorithm. Author in [6] implemented a hardware detection system based on the Active Shape Model (ASM) algorithm, and they reported speedup up to 15X compared to software execution. However, their implementation does not include tracking. Schlessman in [7] discusses a practical design based on a hardware/software co-design to realize an optical flow tracking system, which puts the KLT portion that consumes much processing time to FPGA-based hardware implementation as optimization. Christopher in [8] introduces an object tracking hardware design based on color features. Due to the advantages offered by FPGAs in compute intensive applications, several object tracking algorithms have been implemented on reconfigurable devices in recent researches. Nevertheless, one of the biggest challenges of custom hardware implementations is mapping complex algorithms onto reconfigurable fabric architectures that can offer good performance under rigid resource constraints.

For object tracking purposes, numerous algorithms have been proposed in the literatures and are implemented on FPGAs. Object tracking is a complex task which comprises two main subtasks: i) object detection and ii) tracking. In [9] object detection algorithms are classified into point detection, background subtraction

techniques, and supervised learning techniques. Furthermore, the tracking portion of object tracking can be performed either separately or jointly with object detection. Alper Yilmaz [10] has characterized tracking algorithms across three main categories: i) point tracking, ii) kernel tracking and iii) silhouette tracking. Using these tracking techniques various tracking algorithms are developed /used for clustering and tracking of non rigid objects. Some of these popular algorithms are KLT-tracker [11], mean shift [12], mean shift with motion vector [13], kernel based Mean shift [14], eigen tracking [15], optical flow tracking [16], fast object tracking using adaptive block matching [17], heuristic methods for object tracking [18], Kalman filter [19], particle filter [20] etc, which are used for various image processing applications instead of only object tracking. The preliminary work for the system presented in this paper was published in [10]. The proposed work was purely SW based and implement on EDK based design.

3 Mean Shift Algorithm

Mean shift is a nonparametric density gradient statistical method which considers feature space as an empirical probability density function (PDF). If the input is a set of points, then mean shift considers them as sampled from the underlying density function. In mean shift trackers, objects of interest are characterized by the probability density functions (pdfs) of their colour or texture features. A spatially-smooth similarity function between the original object as candidate and target are defined as a masking distributions with a monotonically decreasing kernel. Mean shift iterations are then used as a local maximum of this similarity function as an indicator of the direction of target's movement. In order to apply a mean shift calculation, the set of histogram values is weighted by Epanechnikov kernel to yield a smoothed set of values. It defines an ellipsoidal region and gives more weights to pixel closer to the center of the kernel. The rationale for using a kernel to assign smaller weights to pixels farther from the centre is that those pixels are the least reliable, since they are the ones most affected by occlusion or interference from the background. A kernel with Epanechnikov profile was essential for the derivation of the smooth similarity function between the distributions. Since its derivative is constant; the kernel masking lead to a function suitable for gradient optimization, which gave us the direction of the target's movement. The search for the matching target candidate in that case is restricted to a much smaller area and therefore it is much faster than the exhaustive search. The number of bins in the histogram representation for the target is defined by the user. This allows for simpler histograms in cases where the image sequence features highly distinctive colors and is devoid of collision events between like-colored objects. Likewise a larger number of bins may be used if the range of colors is limited or the target and background colors are nearby in normalized RGB. Thus algorithm completes its processing mainly in six steps. In First step we initialize the location of target in current frame. In step two, it computes PDF for target and candidate. In the same step it also computes the similarity between them using Bhattacharya Coefficient. In step third and forth, computes the weights and then apply the mean shift to find the new location respectively. Finally, in step five finds the new location of object in reference to the centre candidate and in step sixth iterate the algorithm for all the frames of video streaming.

4 FPGA Implementation

4.1 Base System Builder

All EDK designs are built on a Base System Builder (BSB) platform which provides a common base and building blocks. Each of the EDK reference designs included with the IVK is built from the base platform. The Base Platform is not a separate design that is delivered with this kit, rather it is the starting point from which all the other designs were built. The board we have used is Xilinx Industrial Video Processing Kit (IVK) Spartan-6 XC6SLX150T-3FGG676C- based embedded platform as shown in fig. 1. It provides two FMC LPC general-purpose I/O expansion connectors, and a memory of 128 MB DDR3 SDRAM. For communication we used RS-232 serial port. The hardware for the implementation of mean shift is made by adding various IPs and peripherals. These are Micro Blaze™ 32-bit soft microprocessor, Local Memory Bus (LMB), LMB Block RAM controller, Block RAM block memory, Processor Local Bus (PLB46), XPS UARTlite, Xilinx Platform Studio (XPS) General Purpose Input/Output (GPIO), XPS Inter-Integrated Circuit (IIC) Controller, External Multi-port Memory Controller (MPMC), MDM MicroBlaze Debug Module, Clock Generator, Processor System Reset and finally the DDR3 block representing the external memory.

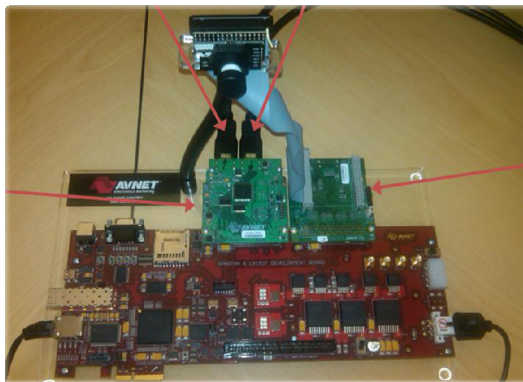


Fig. 1. Xilinx Spartan6 IVK FPGA board setup

The image sensor video input source enters the Camera Input PCORE [21]. This PCORE decodes the BT656 codes to generate synchronization signals and formats the video as an XSVI bus interface. The Video Detect PCORE does not alter the video, but monitors the VSYNC and ACTIVE VIDEO signals to determine the dimensions of the active video streaming through the FPGA. It also generates Video DMA compatible bus interface used to write video data to external memory. The Video DMA PCORES, in collaboration with the Video Frame Buffer Controller (VFBC) [21] interfaces on the Multi-Port Memory Controller (MPMC), perform the actual transfers to/from external memory. These cores are extremely flexible and are configured via the Micro Blaze processor. The GENLOCK port indicates where the

first Video DMA has written the incoming frames. The second Video DMA reads video frames from memory based on the GENLOCK information. After that the histogram calculation IP and later the mean shift block gets the pixel data and takes the RGB values, each of 8 bit. It takes the pre-calculated kernel values and finds out the histogram of the target and the candidate model and they compute the displacement in the mean shift block of the target object in each frame. Since the output frame rate is higher than the input frame rate, frames are duplicated when necessary. The Video Generate PCORE, under control of the Micro Blaze, generates video timing for the output. It also generates a Video DMA compatible bus interface used to read video data from external memory. The DVI Output PCORE takes an XSVI bus interface as input and optionally drives the pins of the DVI output interface. This output to the FMC connector will only be driven once the FMCIMAGEOV module has properly been identified. The video capture is at 1280x720P @ 30Hz and video playback at 1280x720P @ 60Hz. These resolutions are configured by the embedded processor (Micro Blaze) and can be modified to support other resolutions (limited by the image sensor used).

4.2 Compute Displacement (HW)

To accelerate the computing faster, compute displacement function is replaced with HW designed IP compute_dis_25 as shown in fig 2. The displacement dx and dy is calculated with the input functions Kernel derivative function (S1) and weight function (S2). These values are stored in BRAM segmented memory locations (160 X 80) To store values S1 and S2 before computing the displacement. The counter i logic and j logic are used to multiply the element by element of S1 and S2 and then added the pixel values as shown in block diagram in fig 2.

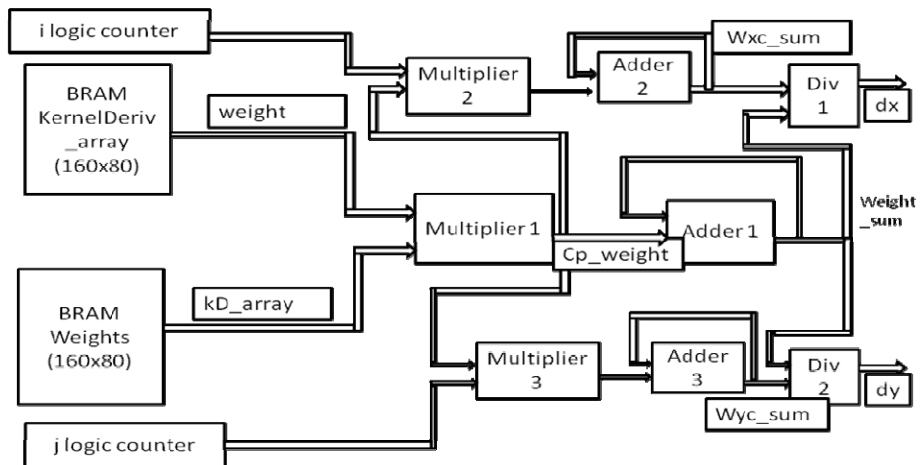


Fig. 2. Data flow diagram of Compute displacement module

5 Simulation and Implementation Results

5.1 Simulation Results

To evaluate and compare the system performance of tracking algorithm, first the algorithm is simulated on Matlab tool. In the simulation of the tracker, we have chosen RGB color space as a feature space, in which the chosen feature space was quantized into $5 \times 5 \times 5$ bins. The set of histogram values is weighed by Epanechnikov kernel which yields a smoothed set of values. A constant kernel derivative is used in the calculation of the kernel profile. The value of $Cd = 2$ and $d=1$ is used. The algorithm is executed comfortably at 75 frames per second (fps) on 2.70 GHz PC, Matlab (version 7.60). The Matlab simulation results are shown in fig 3.1, fig 3.2 and fig 3.3. The first result in fig 3.1, states that when we increase bin number from 5 to 25 to 50 of the target, then there is a slight difference between the trajectory of the object in different bins condition but this difference is too less. It means on increasing the bins there is a separation between the object being tracked and background, which shows that the algorithm is robust to bin size. Graph in fig 3.2 shows the metric distance between the frames 320 to 340 is very less compared to frames 450 to 470, which means that there is maximum similarity (i.e. minimum distance) between the target window and the candidate window in the successive frames from 320 to 340. It signifies the movement of the object in these consecutive frames is slow in respect to frames 450 to 470. It means that the change in the histogram does also affect the similarity (i.e. distance) between the target model and the candidate model. The graph in fig 3.3 shows the Mean Shift iteration. The maximum distance shows, the more illuminated area is there in the frames ranging from 450 to 500. It means

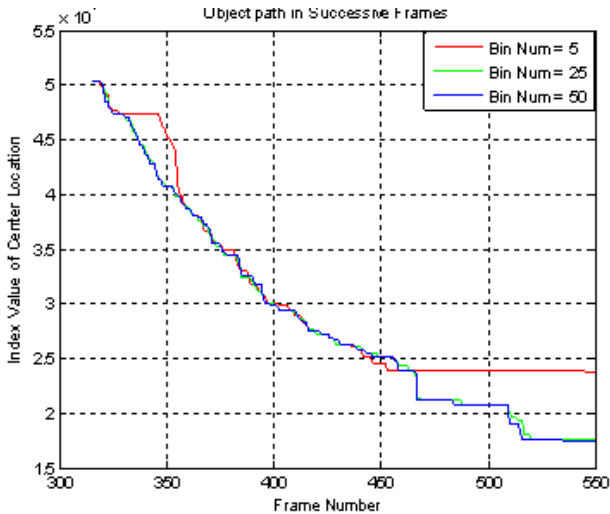


Fig. 3.1. Measurement of estimated location in successive frames

when the object is being tracked and passes from a well illuminated region to variable illuminated region, the model histogram is not indicative of the target very well. Ultimately, tracker shifts from the actual object. As a result, if the two similar objects are overlapped in a scene the tracker move to other similar object. The tracking results of two similar objects are shown in fig 3.4 in three different frames. From top to bottom: (a) before overlapping of objects (b) overlapping of objects in middle and (c) tracking to another similar object at bottom. Object is tracked well before the overlapping of both the objects.

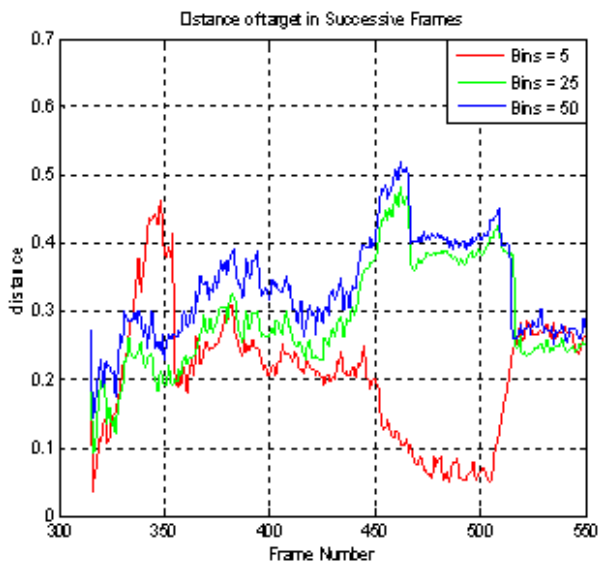


Fig. 3.2. The minimum value of distance function of the frame index

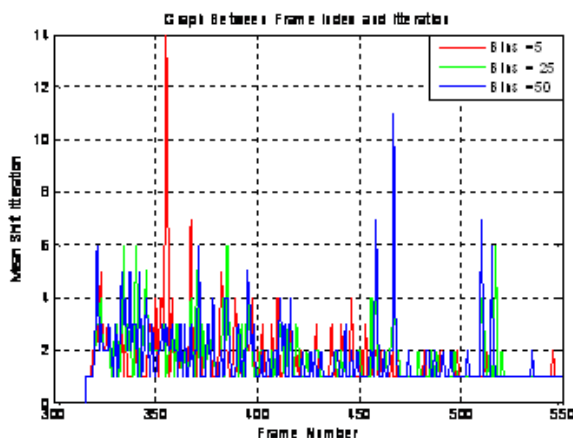


Fig. 3.3. The number of mean shift iterations function of the frame index



Fig. 3.4. Tracking frames of a moving person in top figure, in middle overlapping of similar object at bottom tracker shifts to other similar object after overlapping

5.2 Implementation Results

Table 1. The table shows the resources used and its percentage utilization

Device Utilization Summary (estimated values)			
Logic Utilization	Used	Available	Utilization
Number of Slice Registers	11,437	184,304	6%
Number of Slice LUTs	10980	92,152	11%
Number used as memory	846	21,680	3%
Number of bonded IOBs	95	386	23%
Number of BUFG/BUFGCTRLs	10	32	31%
Number of DSP48Es	18	128	4%

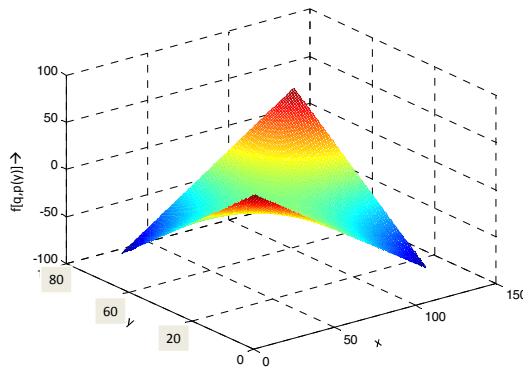


Fig. 4.1. Similarity function $f [q, p(y)]$

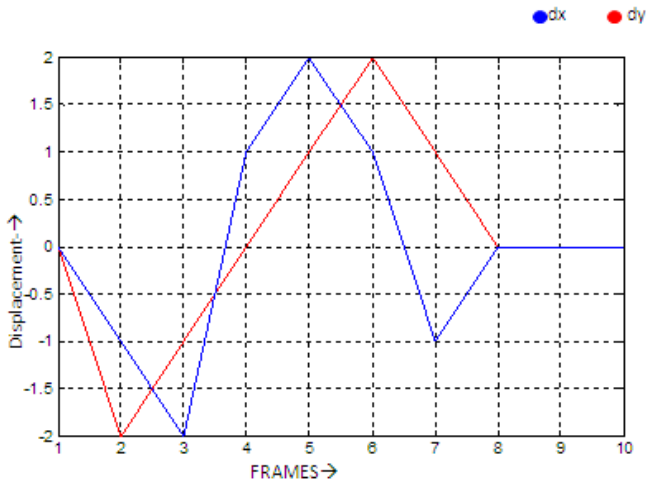


Fig. 4.2. Displacement of the target object in 10 consecutive frames

6 Conclusion

The simulated result of Kernel based mean shift algorithm is found to be smoothly tracking a specified object if there is a good separation between the objects. If the objects are found similar and overlapped then simple Mean Shift may track a wrong object. So this shortcoming may overcome with combination of other algorithms or with filtering in future work. On the other hand, for the implementation in real time tracking some of the frames are lapsed due to high complexity of computation if algorithm is executed on only general purpose soft processor. Solution for real time implementation of algorithm is to bring the complete execution with HW/SW co- design methodology to accelerate the execution. In this reference a compute displacement function is be replaced with hardware which is presented in this paper as an IP.

References

- [1] Hsiung, P.-A., Santambrogio, M.D., Huang, C.-H.: Reconfigurable System Design and Verification. CRC Press © Taylor & Francis Group, London (2009)
- [2] Compton, K., Hauck, S.: Reconfigurable Computing: A Survey of Systems and Software. ACM Computing Surveys 34(2), 171–210 (2002)
- [3] Ali, U., Malik, M.B., Munawar, K.: FPGA/Soft- Processor based real-time object tracking system. In: Proceedings IEEE, Fifth Southern Programmable Logic Conference, pp. 33–37 (2009)
- [4] Raju, K.S., Baruah, G., Rajesham, M., Phukan, P.: Computing Displacement of Moving Object in a Real Time Video using EDK. In: International Conference on Computing, Communications, Systems And Applications (ICCCSA), Hyderabad, March 30-31, pp. 76–79 (2012) ISBN:978-81-921580-8-2

- [5] Rummele-Werner, M., Perschke, T., Braun, L., Hübner, M., Becker, J.: A FPGA based fast runtime reconfigurable real-time Multi-Object-Tracker. In: IEEE International Symposium on Circuits and System (ISCAS) (May 2011)
- [6] Xu, J., Dou, Y., Li, J., Zhou, X., Dou, Q.: FPGA Accelerating Algorithms of Active Shape Model in People Tracking Applications. In: Proc. 10th IEEE Euromicro Conference on Digital System Design Architectures, Methods and Tools (DSD 2007) (2007)
- [7] Schlessman, J., Chen, C.Y., Ozer, B., Fujino, K., Itoh, K., Wolf, W.: Hardware/software Co-design of an FPGA based Embedded Tracking System. In: Proceedings of the IEEE Conference on Computer Vision and Pattern 1662 Recognition Workshop, pp. 123–133 (2006)
- [8] Johnston, C.T., Gribbon, K.T., Bailey, D.G.: FPGA based Remote Object Tracking for Real-time Control. In: Proceeding 1st International Conference on Sensing Technology, November 21–23, pp. 66–71 (2005)
- [9] Yilmaz, A., Javed, O., Shah, M.: Object Tracking: A Survey. *ACM Computing Surveys* 38(4), Article 13 (December 2006)
- [10] Raju, K.S., Baruah, G., Rajesham, M., Phukhan, P., Pandey, M.: Implementation of moving object tracking using EDK. *International Journal of Computer Science Issues (IJCSI)* 9(3), 43–50 (2012)
- [11] Shi, J., Tomasi, C.: Good features to track. In: Proceeding IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR), pp. 593–600 (1994)
- [12] Comaniciu, D., Ramesh, V., Meer, P.: Real-time tracking of non-rigid objects using mean shift. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Hilton Head, vol. 2, pp. 142–149 (2000)
- [13] Tian, G., Hu, R.-M., Wang, Z.-Y., Zhu, L.: Object Tracking Algorithm Based on Meanshift Algorithm Combining with motion vector analysis. In: Proceeding, First International Workshop on Education Technology and Computer Science, vol. 01, pp. 987–990 (2009)
- [14] Comaniciu, D., Ramesh, V., Meer, P.: Kernel-Based Object Tracking. *IEEE Trans. on Pattern Analysis and Machine Intelligence* 25(5), 564–577 (2003)
- [15] Lucas, B.D., Kanade, T.: An iterative image registration technique with an application to stereo vision. In: Proceedings of the 7th International conference on Artificial Intelligence (IJCAI), August 24–28, pp. 674–679 (1981)
- [16] Barron, J., Fleet, D., Beauchemin, S.: Performance of optical flow techniques. *Int. J. Comput. Vision (IJCV)* 12(1), 43–77 (1994)
- [17] Hariharakrishnan, K., Schonfeld, D.: Fast object tracking using adaptive block matching. *IEEE Transaction on Multimedia* 7(5) (October 2005)
- [18] Ronfard, R.: Region based strategies for active contour models. *Int. J. Comput. Vision* 13(2), 229–251 (1994)
- [19] Zhong, J., Sclaroff, S.: Segmenting foreground objects from a dynamic textured background via a robust kalman filter. In: Proceeding of the Ninth IEEE International Conference on Computer Vision (ICCV), October 13–16, vol. 1, pp. 44–50 (2003)
- [20] Zhou, S., Chellapa, R., Moghadam, B.: Adaptive visual tracking and recognition using particle filters. *IEEE Transactions on Image Processing* 13(11), 1491–1506 (2004)
- [21] Spartan-6 Industrial Video Processing Kit – EDK Reference Design Tutorial, Xilinx Inc., <http://www.xilinx.com>