FPGA based Object Tracking System for Sequential Images

Dhanesh M. S.
ECE department
RSET
Kochi, India
dhaneshmuralee@gmail.com

Ann Mary Benny
ECE department
RSET
Kochi, India
annbenny@gmail.com

Anu P Sudhahar
ECE department
RSET
Kochi, India
anusudhahar@yahoo.com

Jess John
ECE department
RSET
Kochi, India
jessjhon@hotmail.com

Abstract—**Object tracking is a popular operation in defense as well as surveillance and robotics. This paper presents a novel approach to an FPGA based object tracking system for sequential images which also supports an implementation in high speed hardware. The paper develops an efficient tracking algorithm as well as an online learning module incorporated to improve the coherence of the process. The paper implements a Sobel-edge based feature extraction for dimensionality reduction of images for easy processing. Sobel masks are also efficient in extracting edges of images in all four directions – horizontal, vertical, along +45^O, and along -45^O directions. The information from each image frame is encapsulated into a set of 16-D vectors, corresponding to different sections of the image frame. These feature vectors are correlated with the feature vector of the object to be tracked and depending on the degree of resemblance, a statistical approach is taken to direct the tracking. With the online learning process, the number of templates available for correlation is updated as newer object versions are procured during the tracking process. Thus, this hardware implementation presents an object tracker that can adapt to the environment dependent complexities like screen illumination changes, partial occlusion, slight deformations, etc.**

Keywords—Directional edge feature, Sobel edge detection, field-programmable gate array (FPGA) implementation, object tracking, online learning.

I. INTRODUCTION

Object tracking plays an important role in many applications, such as video surveillance, human–computer interface, vehicle navigation, and robot control. It is generally defined as a problem of estimating the position of an object over a sequence of images. In practical applications, however, there are many factors that make the problem complex, such as illumination variation, appearance change, shape deformation, partial occlusion, and camera motion. Moreover, lots of these applications require a real-time response. Therefore, the development of real-time working algorithms is of essential importance. In order to accomplish such a challenging task, a number of tracking algorithms [1]–[7] and real-time working systems [8]–[13] have been developed in recent years.

These algorithms usually improve the performance of the object tracking task in two major aspects, i.e., the target object representation and the location prediction. Despite the better performance of these algorithms with more complex structures, they suffer from the high computational cost that prevents their implementation from working in real time.

Some implementations using dedicated processors always result in power-hungry systems [11], [15]. Many implementations parallelize the time-consuming part of algorithms, thus increasing the processing speed to achieve real-time performance [16]–[18]. These solutions depend heavily on the nature of algorithms and the performance enhancement would be limited if the algorithms are not designed for efficient hardware implementation. Some specific implementations can be employed to speed up a certain part of the algorithm, such as feature extraction [19] or localization [20]. In this case, it is necessary to consider how to integrate them into the total system most efficiently. Several problems may arise when building parallel systems, such as transmission of large amount of data.

In this paper, we have explored a solution to the object tracking task that considers an efficient implementation as the first priority. A hardware-friendly tracking framework has been established and implemented on field-programmable gate array (FPGA), thus verifying its compatibility with very large-scale integration (VLSI) technology. Several problems that limit the hardware performance, such as complex computation, data transmission, and cost of hardware resources, have been resolved.

Since our solution provides a high flexibility in its configuration, it can be integrated into a lot of other more complex intelligent systems as their subsystems. In tracking algorithms, how to represent the target image is of particular importance because it greatly influences the tracking performance under certain tracking framework. Colour, edge, and texture are typical attributes used for representing objects. A number of other features, including active contour, scale-invariant feature transform (SIFT) feature, oriented energy, and optical flow, are also used in many works. Some works also combine these features or incorporate online learning of the model of an object and background.

It is well known that animals have excellent ability in visual tracking, but the biological mechanism has not yet been clarified. However, it was revealed that the visual perception of animals relies heavily on directional edges [26]. In this paper, therefore, the directional-edge-based image feature representation algorithm based on Sobel edge detection is employed to represent the object image. The purpose of this paper is to develop an object tracking system that is robust against disturbing situations like illumination variation, object shape deformation, and partial occlusion of target images. By employing the directional-edge based feature vector representation, the system has been made robust against illumination variation and small variation in object shapes. In order to achieve real-time performance in tracking, a VLSI hardware-implementation friendly algorithm has been developed. It employs a direct approach, in which candidate locations are identified during tracking. The basic idea was inherited from the particle filter and Multiple Candidate Regeneration (MCR), but the algorithm has been greatly modified and simplified from the original particle filter and MCR so that it can be implemented in smaller scale VLSI hardware very efficiently. The algorithm has been proposed and the performance has been verified by simulation. In order to further enhance the robustness of the tracking ability, an online learning technique has been introduced to the system. When the target object changes its appearance beyond a certain range, the system autonomously learns the altered shape as one of its variations, and continues its tracking. As a result, for a large variation in the shape and for partial occlusion, the system has also shown a robust performance.

This paper is organized as follows. The directional edge features and the tracking algorithm are explained in Section II. The implementation of this tracking algorithm on hardware is described in Section III. Finally, conclusions are drawn in Section IV.

II. Algorithm

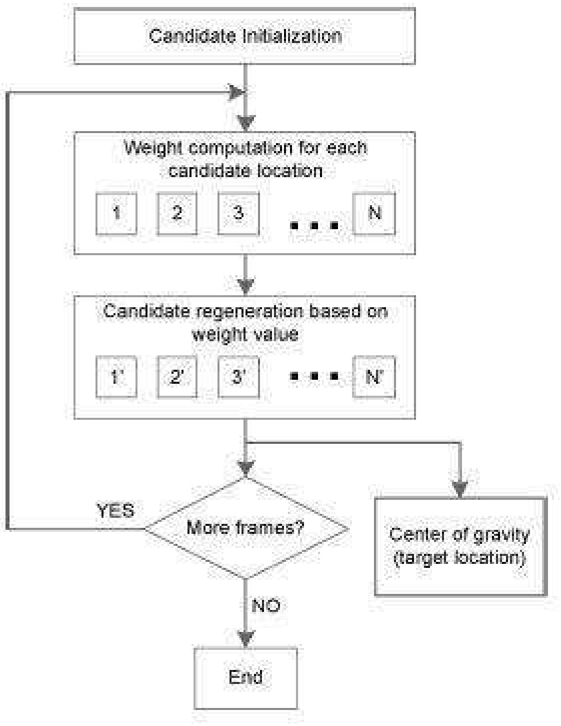
The most essential part of the algorithm in [1] is a recursive process called multiple candidate regeneration (MCR), which is similar to the prediction and update in the particle filter. Being a complex algorithm, we have simplified the concept used in the MCR algorithm to obtain a tracking procedure that determines the position of the object through direct comparison with the candidate image locations.

At the very beginning, the target image is specified manually, which acts as a template for the tracking process. The similarity between the target image and the local image is determined and a weight is assigned for each of the local image. Depending on the value of the weight, the location of the object is identified, if present in that particular image frame.

The online learning process is incorporated by specifying a template container that is updated during the tracking process. In this paper, we have implemented the online learning as an initial part of the tracking process. The similarity between the object template and the local image is estimated in the online learning section. This similarity gives an indication of the presence of the object, and also if it has been deformed. If there is considerable deformation, this local image is stored as a new template.

A. Algorithm Structure

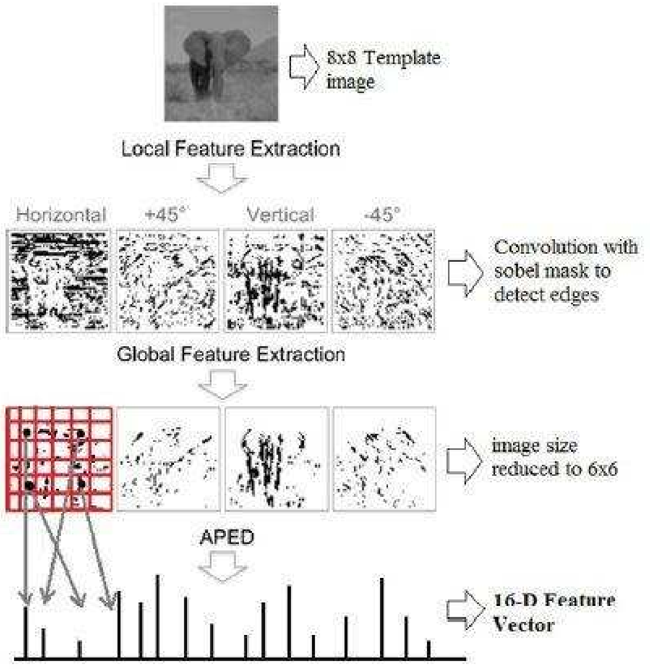
Fig. 1 shows the structure of the algorithm. The algorithm starts with the calculation of the similarity between the object template and the local image. If the local image is very similar to the object, a larger weight is assigned and the tracking is carried out. If the similarity is not much, a smaller weight is assigned and the object is not present. If the object template already available in the container and the local image under consideration are dissimilar to a certain extent, the feature vector of the local image is stored as a new template into the container.


Candidate Initialization
Weight computation for each candidate location
1
2
3
N
Candidate regeneration based on weight value
1'
2'
3'
N'
YES
More frames?
Center of gravity (target location)
NO
End

B. Object Representation

As explained in Section II-A, in order to calculate the weight of each candidate, we need to evaluate the similarity between the candidate image and the template image. This is done by calculating the distance between the two feature vectors representing the two images. Therefore, employing a suitable feature representation algorithm is very important. We employed the directional-edge-based image representation algorithm [31]–[33] that was inspired by the biological principle found in the animal visual system [26]. The edge detection was done using the Sobel edge detection algorithm as it was found to be more efficient than other edge detection algorithms. This method needs only the grayscale information of an image as input and the output is a 16-D feature vector. It consists of two successive steps: local feature extraction (LFE), and averaged principal-edge distribution (APED) [31]. Fig. 2 shows the function of each step.

1) *Local Feature Extraction:* The function of LFE is to extract the edge and its orientation at each pixel location in an 8x8 portion of a 64x64 image. The outer two rows and columns of pixel values of the image frame are discarded as they do not generate accurate edge detection results. For every pixel location, the convolutions of a 3×3 pixel region with four directional filtering kernels of the Sobel algorithm (horizontal, +45°, vertical, −45°) are calculated as shown in Fig. 5. Then, the absolute values of these four convolution results are compared, and the maximum value is stored as the gradient at this pixel location.


8x8 Template image
Local Feature Extraction
Horizontal
+45°
Vertical
-45°
Convolution with sobel mask to detect edges
Global Feature Extraction
image size reduced to 6x6
APED
16-D Feature Vector

2) *Averaged Principal-Edge Distribution:* Although the information has been compressed by extracting edges in LFE, the amount of information is still massive in quantity. Therefore, a method called APED [31] is employed to reduce the four edge maps into a 64-D vector. In the APED vector representation, each edge map is divided into 4 square bins consisting of 9 elements and the number of edge flags in each bin is summed up, which constitutes an element of the vector. The 16-D feature vector is the final output of the feature extraction

C. Weight Computation and Candidate Regeneration

Since the basic principle has already been explained, how to implement it is described here. In order to make all computations easily and efficiently implementable in the VLSI hardware, each mathematical operation was replaced by a hardware-implementation friendly analogue, which are different from that in the regular particle filter algorithm. The local image taken from each candidate location is converted to a feature vector and the Manhattan distances are calculated with template vectors. In this algorithm, there are more than one templates in the template container to represent the target. The first template is generated at the initialization step, while others are generated during the online learning process. Therefore, the minimum Manhattan distance is utilized to determine the weight of this candidate described as follows:

$$\mathrm{MD}_{i,j} = \sum_{k=1}^{n} \left| V_{Ci}[k] - V_{Tj}[k] \right| \tag{1}$$

$$D_i = \min(\mathrm{MD}_{i1}, \mathrm{MD}_{i2}, \ldots, \mathrm{MD}_{in}) \tag{2}$$

$$W_i = \begin{cases} 0, & (D_i \geq C) \\ \mathrm{INT}[N_0 \times (1 - D_i/C)], & (D_i < C). \end{cases} \tag{3}$$

Here, $\mathrm{MD}_{i,j}$ stands for the Manhattan distance between the candidate i and the template j, and $VC_i[k]$ and $VT_j[k]$ denote the kth element of the candidate vector VC_i and the template vector VT_j, respectively. D_i is the minimum distance of candidate i with all the templates and W_i represents the weight for the candidate i. N_0 is a constant value determining the scale of the weight. In (3), C is a threshold defining the scale of weight values, which is determined by experiments. INT means taking the integer component of the value. In this manner, those candidates that have at least one Manhattan distance value smaller than the threshold C are all preserved to regenerate new candidates in the next frame. At the same time, larger weight values are assigned to candidates with smaller distances.

D. Online Learning

In many practical applications, the target we are concerned about is a non-rigid object, which may change its appearance and size. In addition, sufficient knowledge about the target is, in general, not available before tracking. This problem causes tracking failure if the algorithm does not flexibly learn the appearance change in the target. An online learning method is introduced to solve this problem in this paper. The learning process begins after the estimation of the target location. One feature vector is generated from the image at the target location in the present frame. Then the Manhattan distances between this feature vector and all the templates are calculated and the minimum distance is found. If the minimum distance is larger than a certain threshold, it is interpreted as the target that has changed its appearance substantially, and the feature vector is stored as a new template in the template container.

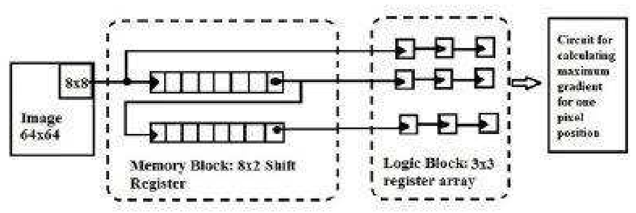
III. IMPLEMENTATION

A. Feature Extraction

The feature extraction stage is implemented in three serially connected functional blocks: LFE, GFE, and vectorisation.

The structure of LFE block is shown in Fig. 3.This block input 8x8 image portion of 64x64 scale image. There are two 8-bit shift registers, serially connected, and the output of each register is inputted to the respective row of a 3x3 register array. It shifts pixel data of 8 bits. The shift register stores the size of image data necessary for computation. The 3x3 register array works as a buffer between the shift register and the logic block. The combinational logic block deals with all the logic processing needed to calculate the gradient, including doing convolution with four directional 3x3 kernels, taking their absolute values. The intensity values of an image are sent into the first row of the shift register and, at the same time, into the top row of the register array pixel by pixel. The two rows of data in the shift register are shifted-in to the corresponding lines of the 3x3 register array. The 3x3 register array finds the gradient value of centre pixel in register array in all four directions. In this manner, the 3x3-pixel filtering kernel block scans the entire image pixel by pixel and generates a directional gradient map.

The GFE block stores the LFE values for all original image pixels excluding the outer two rows and columns in a memory


8x8
Image 64x64
Memory Block: 8x2 Shift Register
Logic Block: 3x3 register array
Circuit for calculating maximum gradient for one pixel position

block of size 36 elements. The memory block 1x36 size with elements of size 8 bits. Four such memory blocks are created for storing four directional gradient map reduces the 8x8 image size to 36 element vector. The output of GFE is a binary map that contains the edge information in four directions. In the following step, this edge information is compressed effectively into a feature vector representation in APED.

APED block divides the each of the four GFE memory blocks to four square bins with 9 elements each and the largest of the 9 elements in each square bin is found out by comparing.Thus from each memory block in GFE four dominant pixel values are extracted forming a total of 16- D values from the four different memory blocks.This is the dimensionally reduced image representation.For an 8x8 image portion to be reduced to a 16-D feature vector requires 64 clock cycles.Thus for a 64x64 full image requires 4096 clock cycles.

B. Object Tracking

The input to the algorithm is a 16D vector. This is the feature vector of the part of the image frame. The feature vector of the object (8*8) is stored in a template container. The first part of the algorithm is calculation of Manhattan distance between the image feature vector and template feature vector. The next part of algorithm is assigning weights based on the Manhattan distance. Larger weights are assigned when the Manhattan distance is smaller and vice versa. The next part of the algorithm is for calculation of maximum weight. The next part is to decide whether the portion of the image has the target object or not. If the target object is present then the location of that portion of the image is displayed. This algorithm is repeated continuously over the entire image.

1) Address Block: The address block contains the address of the incoming image feature vector. The address is represented using four corner locations (w,x,y,z) of the image. When the next part of the image enters the block the address automatically updates. This is done using an address enable signal. The address block is coded considering the image size to be 64*64 pixels. The four corner address locations are sent as output based on the signal from the enable block.

2) Manhattan Distance and weight calculation: The local image taken from each candidate location is converted to a feature vector and the Manhattan distances are calculated with template vectors [1]. The Manhattan distance is calculated as the difference between the image feature vector and the template feature vector. Manhattan distance is utilized to determine the weight of this candidate.

3) Maximum Weight Calculation and Enable Block: Maximum weight is calculated after 64 weights are assigned to different parts of the image [1]. The Maximum weight will be the input to the next part of the algorithm. The maximum weight is calculated using comparators. The maximum weight is given to the enable block. If the weight is greater than a threshold then enable signal is made high else the enable signal is made low. The threshold value is determined by experiments.

4) MCR Block: This block is the top module.It combines all the sub blocks involved.A common clock signal and reset is used for synchronisation of all the blocks.When the enable signal becomes high the four corner locations or address of the part of the image is sent as the output.This indicates that the object is present in that part of the image.If the target is not present,then the output will be zero.

C. Online Learning

The input to the algorithm is a 16D vector. This is the feature vector of the candidate object of the image frame. The feature vector of the object (16*8) is stored temporarily in the template container. The first part of the algorithm is the calculation of Manhattan distances between this image feature vector and already available template feature vectors in the container. The next part of the algorithm is the estimation of the minimum among these distances. The next part is to decide whether the portion of the image has the target object or not, and if it has, whether the appearance of the object has transformed. If there is considerable deviation, as can be understood from the Manhattan distance, the feature vector that had been stored initially is written into the template container as a new template of the object. This algorithm is repeated continuously over the entire image and for every image frame.

1) Template Container: The template container is a memory block with five slots for storing templates of object in the form of 16-D feature vectors. The original template is stored in the first slot and the remaining slots are filled as newer versions of the object is detected. An enable signal is used to determine whether the writing process is to be initiated. This signal is generated in the decision block. The templates stored in the container are serially output from the container to the next block, i.e., the Manhattan distance calculator.

2) Manhattan Distance Calculator: The image feature vector and all the five template feature vectors are compared using Manhattan distance. The Manhattan distance module is instantiated five times for the distance calculation between each template and the image vectors.

3) Minimum Manhattan Distance Estimator: The Manhattan distance between all the templates and image vector are input into this block and the minimum from these values is estimated. The minimum value is taken as it implies the case when the image vector is most correlated with the available templates. This minimum value is then routed to the decision block.

4) Decision block: The minimum Manhattan distance is compared with a threshold as determined from previous

experimental results. If the distance lies between a prescribed range, the enable signal to the template container is made high, so that the image feature vector is stored in the container as a new vector. If the threshold is not satisfied, the enable signal is not set and the container is not modified.

IV. SIMULATION RESULTS

Feature extraction coding was done using verilog ISE Design Suite 14.2 for direct implementation in Spartan E FPGA kit. An 8x8 image pixel values are given as input and 16-D vector obtained as output. For 64x64 image size, total of 4096 cycles are required. Fig. shows the feature vector generation in 64 clock cycles of an 8x8 image.

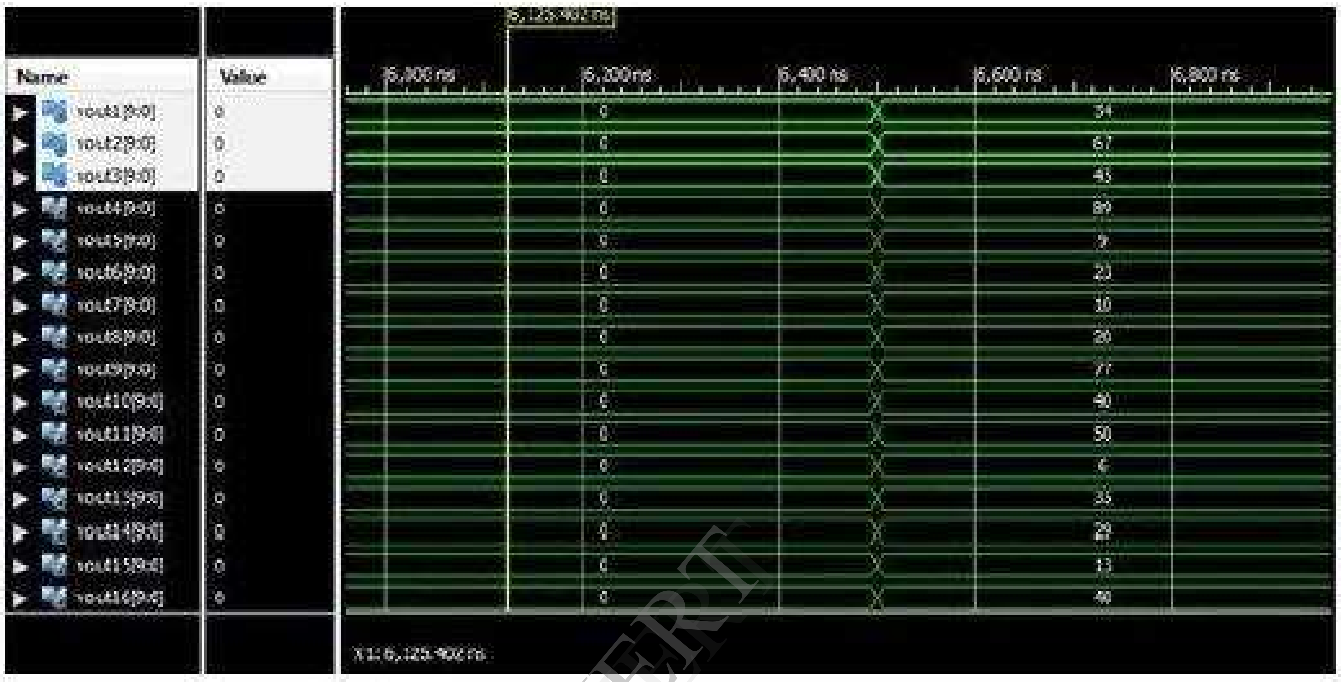
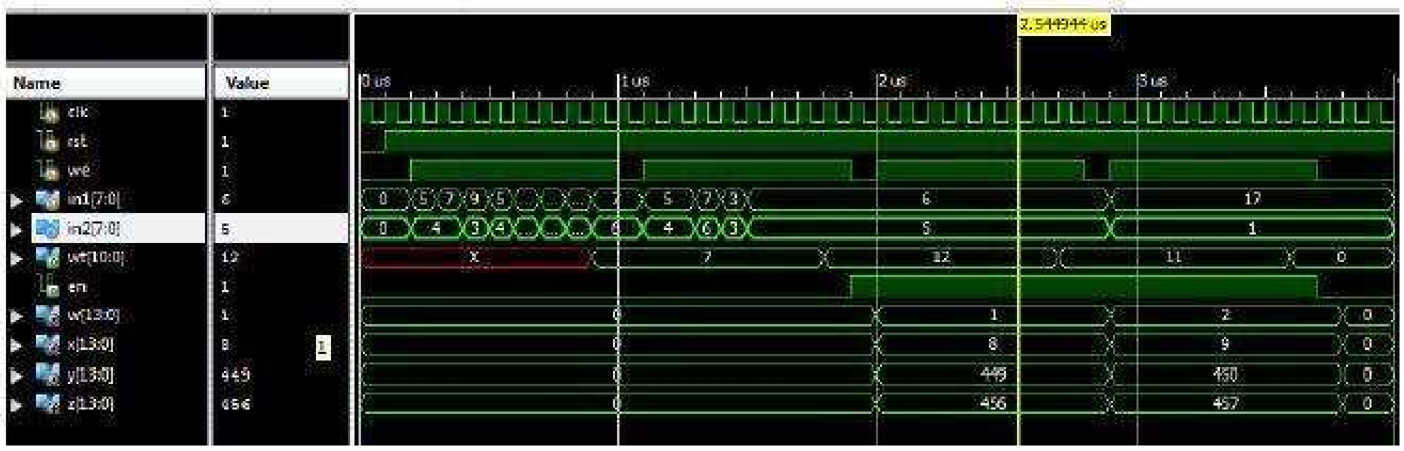
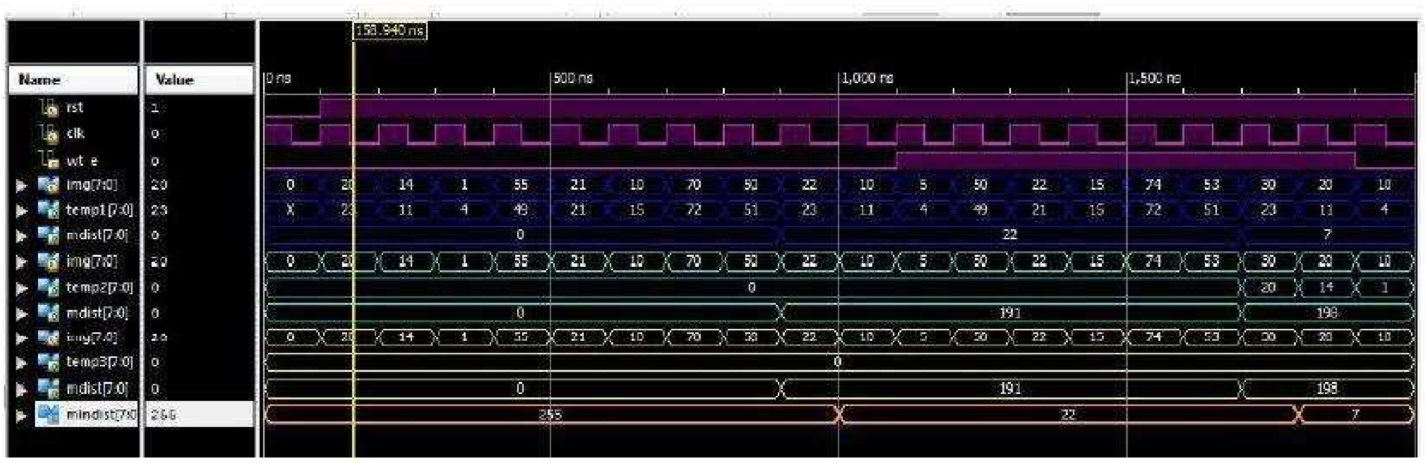


Figure below is the simulation result of the tracking algorithm. The output of the multiple candidate regeneration block is the address location (w,x,y,z) if the object is present in the image frame. If the object is not present then the output of this block is zero.



The following figure is the simulation result of the online learning algorithm. When the Manhattan distance of the oncoming feature vector lies between a certain range of values, the next vacant template in the template container is updated with this feature vector. If the object is detected, but there is no considerable deformation to the object, then the template container is not updated.



V. CONCLUSION

In this paper, an object prediction system with an onlinelearning module has been proposed. The system has been designed for an image frame of 64*64 pixels and objecttemplate of size 8*8 pixels. With the online learning algorithm,the efficiency of tracking has been improved. The design wascoded in verilog and simulated successfully.

REFERENCES

[1] Pushe Zhao, Hongbo Zhu, He Li, and Tadashi Shibata, A Directional-Edge-Based Real-Time Object Tracking System Employing Multiple Candidate-Location Generation, IEEE transactions on circuits and systemsfor video technology, vol. 23, no. 3, march 2013.J. ClerkMaxwell, A Treatise on Electricity and Magnetism, 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp.68-73.

[2] A. Yilmaz, O. Javed, and M. Shah, "Object tracking: A survey," *ACMComput. Surveys*, vol. 38, no. 4, pp. 1–45, 2006.

[3] H. Wang, D. Suter, K. Schindler, and C. Shen, "Adaptive object tracking based on an effective appearance filter," *IEEE Trans. Patt. Anal. Mach. Intell.*, vol. 29, no. 9, pp. 1661–1667, Sep. 2007.

[4] B. Han, Y. Zhu, D. Comaniciu, and L. Davis, "Visual tracking by continuous density propagation in sequential Bayesian filtering framework," IEEE Trans. Patt. Anal. Mach. Intell., vol. 31, no. 5, pp. 919–930, May 2009.

[5] Y.-J. Yeh and C.-T. Hsu, "Online selection of tracking features using AdaBoost," IEEE Trans. Circuits Syst. Video Technol., vol. 19, no. 3,pp. 442–446, Mar. 2009.

[6] Q. Chen, Q.-S. Sun, P. A. Heng, and D.-S. Xia, "Two-stage object tracking method based on kernel and active contour," *IEEE Trans.Circuits Syst. Video Technol.*, vol. 20, no. 4, pp. 605–609, Apr.2010.

[7] Z. Khan, I. Gu, and A. Backhouse, "Robust visual object tracking using multi-mode anisotropic mean shift and particle filters," *IEEE*Trans. Circuits Syst. Video Technol., vol. 21, no. 1, pp. 74–87, Jan.2011.

[8] J. U. Cho, S. H. Jin, X. D. Pham, J. W. Jeon, J. E. Byun, and H. Kang, "A real-time object tracking system using a particle filter," in *Proc.*IEEE/RSJ Int. Conf. Intell. Robots Syst., Oct. 2006, pp. 2822–2827.

[9] H. Medeiros, J. Park, and A. Kak, "A parallel color-based particle filterfor object tracking," in Proc. IEEE Comput. Soc. Conf. CVPRW, Jun. 2008, pp. 1–8.

[10] Z. Kim, "Real time object tracking based on dynamic feature grouping with background subtraction," in *Proc. IEEE Conf. CVPR*, Jun. 2008,pp. 1–8.

[11] T. Ishiguro and R. Miyamoto, "An efficient prediction scheme for pedestrian tracking with cascade particle filter and its implementationon Cell/B.E.," in *Proc. Int. Symp. ISPACS*, Jan. 2009, pp. 29–32.

[12] E. Norouznezhad, A. Bigdeli, A. Postula, and B. Lovell, "Robust object tracking using local oriented energy features and its hardware/software implementation," in Proc. 11th Int. Conf. Contr. Automat. Robot. Vision *(ICARCV)*, Dec. 2010, pp. 2060–2066.

[13] S.-A. Li, C.-C. Hsu, W.-L. Lin, and J.-P. Wang, "Hardware/software codesign of particle filter and its application in object tracking," in *Proc. ICSSE*, Jun. 2011, pp. 87–91.

[14] A. Doucet, "On sequential simulation-based methods for Bayesian filtering," Dept. Eng., Univ. Cambridge, Cambridge, U.K., Tech. Rep.CUED/F-INFENG/TR.310, 1998.

[15] H. Medeiros, X. Gao, R. Kleihorst, J. Park, and A. C. Kak, "A parallel implementation of the color-based particle filter for object tracking," inProc. ACM SenSys Workshop Applicat. Syst. Algorithms Image Sensing(ImageSense), 2008.

[16] D. Cherng, S. Yang, C. Shen, and Y. Lu, "Real time color based particle filtering for object tracking with dual cache architecture," in *Proc. 8th IEEE Int. Conf. AVSS*, Aug.–Sep. 2011, pp. 148–153.

[17] X. Lu, D. Ren, and S. Yu, "FPGA-based real-time object tracking for mobile robot," in *Proc. ICALIP*, 2010, pp. 1657–1662.

[18] S. Liu, A. Papakonstantinou, H. Wang, and D. Chen, "Real-time object tracking system on FPGAs," in *Proc. SAAHPC*, 2011, pp. 1–7.

[19] Y.-M. Lin, C.-H. Yeh, S.-H. Yen, C.-H. Ma, P.-Y. Chen, and C.-C. Kuo, "Efficient VLSI design for SIFT feature description," in *Proc. ISNE*, 2010, pp. 48–51.

[20] H. El, I. Halym, and S. E.-D Habib, "Proposed hardware architectures of particle filter for object tracking," *EURASIP J. Adv. Signal Process.*,vol. 2012, no. 1, p. 17, 2012.

[21] P. Li, "An adaptive binning color model for mean shift tracking," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 18, no. 9, pp. 1293–1299, Sep.2008.

[22] J. Wang and Y. Yagi, "Integrating color and shape-texture features for adaptive real-time object tracking," *IEEE Trans. Image Process.*, vol. 17, no. 2, pp. 235–240, Feb. 2008.

[23] S. Fazli, H. Pour, and H. Bouzari, "Particle filter based object tracking with sift and color feature," in *Proc. 2nd ICMV*, Dec. 2009, pp. 89–93.

[24] S. Avidan, "Support vector tracking," *IEEE Trans. Patt. Anal. Mach. Intell.*, vol. 26, no. 8, pp. 1064–1072, Aug. 2004.

[25] S. Avidan, "Ensemble tracking," *IEEE Trans. Patt. Anal. Mach. Intell.*, vol. 29, no. 2, pp. 261–271, Feb. 2007.

[26] D. Hubel and T. Wiesel, "Receptive fields of single neurones in the cat's striate cortex," *J. Physiol.*, vol. 148, no. 3, pp. 574–591,1959.

[27] T. Shibata, M. Yagi, and M. Adachi, "Soft-computing integrated circuits for intelligent information processing," in *Proc. Int. Conf. Inform. Fusion*, vol. 1. 1999, pp. 648–656.

[28] H. Zhu and T. Shibata, "A real-time image recognition system using a global directional-edge-feature extraction VLSI processor," in *Proc. ESSCIRC*, Sep. 2009, pp. 248–251.

[29] N. Takahashi, K. Fujita, and T. Shibata, "A pixel-parallel self-similitude processing for multiple-resolution edge-filtering analog image sensors," IEEE Trans. Circuits Syst. I, Reg. Papers, vol. 56, no. 11, pp. 2384–2392, Nov. 2009.

[30] H. Zhu, P. Zhao, and T. Shibata, "Directional-edge-based object tracking employing on-line learning and regeneration of multiple candidate locations," in *Proc. IEEE ISCAS*, Jun. 2010, pp. 2630–2633.

[31] Y. Suzuki and T. Shibata, "Multiple-clue face detection algorithm using edge-based feature vectors," in *Proc. IEEE ICASSP*, vol. 5. May 2004, pp. 737–740.

[32] A. Nakada, T. Shibata, M. Konda, T. Morimoto, and T. Ohmi, "A fully parallel vector-quantization processor for real-time motion-picture compression," *IEEE J. Solid-State Circuits*, vol. 34, no. 6, pp. 822–830, Jun. 1999.

[33] M. Yagi and T. Shibata, "An image representation algorithm compatible with neural-associative-processor-based hardware recognition systems," *IEEE Trans. Neural Netw.*, vol. 14, no. 5, pp. 1144–1161, Sep.2003.

[34] D. Lowe, "Distinctive image features from scale-invariant keypoints," *Int. J. Comput. Vision*, vol. 60, no. 2, pp. 91–110, 2004.

[35] F.-C. Huang, S.-Y. Huang, J.-W. Ker, and Y.-C. Chen, "HighperformanceSIFT hardware accelerator for real-time image featureextraction," IEEE Trans. Circuits Syst. Video Technol., vol. 22, no. 3,pp. 340–351, Mar. 2012.

[36] B. Babenko, M.-H. Yang, and S. Belongie, "Robust object tracking withonline multiple instance learning," *IEEE Trans. Patt. Anal. Mach. Intell.*,vol. 33, no. 8, pp. 1619–1632, Aug. 2011.

[37] G. Tsagkatakis and A. Savakis, "Online distance metric learning forobject tracking," IEEE Trans. Circuits Syst. Video Technol., vol. 21,no. 12, pp. 1810–1821, Dec. 2011.

[38] Z. Pylyshyn and R. Storm, "Tracking multiple independent targets:Evidence for a parallel tracking mechanism," *Spatial Vision*, vol. 3, no.3, pp. 179–197, 1988.