Implementation of Hybrid Model of Particle Filter and Kalman Filter based Real-Time Tracking for handling Occlusion on Beagleboard-xM

Jharna Majumdar[1], Parashar Dhakal[2], Nabin Sharma Rijal[2], Amar Mani Aryal[2], Nilesh Kumar Mishra[2]
[1]Dean R&D, Prof & Head, Department of CSE (PG)
[2]Final year, Undergraduate Students, Department of ECE
Nitte Meenakshi Institute of Technology, Bangalore, India

ABSTRACT
Particle filter is considered as one of the most robust and accurate techniques in object tracking because of its capability to handle non-linear and non-Gaussian problems. However this technique fails whenever the tracked object is occluded by other objects. In order to solve this problem, in this paper, we have proposed a computer vision based target tracking algorithm that combines both particle filter and Kalman filter. When the target is visible, particle filter is used for tracking the target but whenever there is occlusion, Kalman filter is used to predict and estimate the state of the occluded target. Hence, the proposed algorithm provides accurate results during both visible and occluded conditions. In order to verify the validity and effectiveness of the proposed algorithm, implementation on BeagleBoard-xM, an ARM based embedded platform, has been done. Integration of tracking algorithm on embedded platform paves the way for many real world applications like automated surveillance, human computer interaction, robotics, traffic monitoring etc.

Keywords
ARM, BeagleBoard-xM, Distance Measures, Embedded Computer Vision, Gstreamer, Kalman Filter, Particle filters, Re-Sampling, SDL, Target Tracking.

1. INTRODUCTION
Real time target detection and tracking, because of its wide range of applications, has been a very active area of research over the past two decades with popularly increasing its prospect in the field of computer vision. Target tracking is the process of segmenting an object of interest from a video scene and extracting useful information from it by keeping track of its motion, orientation, state, occlusion etc.

During the past two decades, researchers have made significant efforts on target tracking and developed some tracking algorithms, such as particle filter, Kalman filter and so on. Particle Filter is a powerful tool for Bayesian state estimation in non-linear and non-Gaussian systems by approximating a posterior distribution over unknown state variables by a set of particles, drawn from this distribution. The flaw in this technique can be observed whenever the tracked target is occluded and precise tracking of the moving target becomes erroneous. In order to solve this problem, a number of techniques have been proposed by researchers. We have used Kalman filter in our proposed algorithm for handling the occlusion.

In this paper, Kalman filter is introduced into the standard particle filter for handling occlusion [1] during target tracking. The latest states of the tracked target are memorized during the object tracking process. When occlusion occurs, the position of the tracked target is predicted by the states stored in the memory space. The hybridization of particle filter and Kalman filter helps us in overcoming various shortcomings such as non-linear and non-Gaussian models along with occlusion and makes our algorithm efficient in diverse environment.

2. PLATFORM OVERVIEW AND SPECIFICATIONS
Implementation of computationally challenging computer vision based algorithms for real time tracking requires fast processing of the input video feed. Low-power ARM based embedded system-on-chips (SoCs) combine various co-processors including a Vectorized Floating Point Unit (FPU), a Graphics Processing Unit (GPU) and a Digital Signal Processor (DSP) on a single chip to realize tracking algorithms on hardware. Although, an embedded system has limited power and memory resources, it is adequate to meet the demands of miniaturization and real time computation. Hence, selection of the right processor and optimization of the tracking algorithms should be paid close attention.

2.1 Embedded Platform
The BeagleBoard-xM [2] is a low-cost, low power, fan-less open source hardware, with no additional cooling and heat sinks due to its low power dissipation. It comprises of a 512LPDDR RAM and can be operated through a micro-sd card. The processor in BeagleBoard-xM is the DM3730CBP, featuring an ARM Cortex™-A8 running at up to 1GHz and delivering more than 2,000 Dhrystone MIPS of performance via superscalar operation and comes in a 0.4 mm pitch pop package.POP (Package on Package) is a technique where memory is mounted on top of the processor. The DM3730 is a high-performance, multi-media application device integrated onto TI's advanced 45-nm process technology. Along with a 600 MHz cortex-A8 core, the DM3730 integrates TI's TMS320C64X core, a high end DSP (digital signal processor) clocked at 430 MHz.

The board has been equipped with a minimum set of features to overcome size, cost and power constraints. The board is equipped with four-port high-speed USB 2.0 hub with 10/100 Ethernet, which helps in establishing direct connection.The board is compatible with processors like OMAP35x processor, DaVinci™ DM37x processor, Sitara AM37x processor. The board consists of different subsystems each handling different operations:

2.1.1 MPU subsystem

Multiprocessor Unit (MPU) subsystem handles the transaction within the ARM core, NEON SIMD coprocessor, and cache memory and interrupt controller (INTC). The MPU subsystem integrated the ARM sub chip with additional logic for floating point operations acceleration, emulation, interrupt handling.

2.1.2 IVA2.2 Subsystem

BeagleBoard-xM includes the high-performance Texas Instruments image video and audio accelerator (IVA2.2) based on the TMS320DMC64X+ VLIW digital signal processor (DSP) core (800 MHz up to 720p @30 fps)

2.1.3 SGX Subsystem

The 2D/3D graphics are supplied by an imagination SGX 2D/3D graphics processor supporting dual independent displays. The SGX graphics accelerator can processes Pixel data, Vertex data and Video data. The 2D/3D graphics accelerator enables efficient and concurrent processing of the real time video inputs.

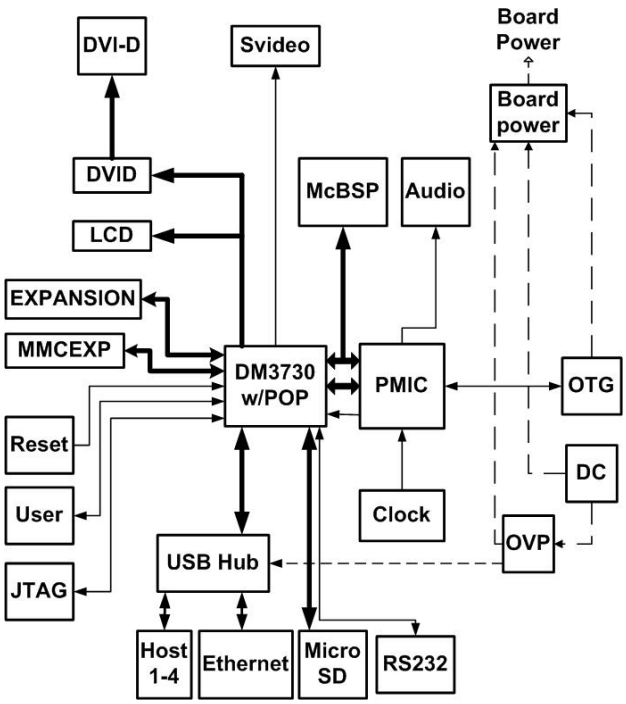


Fig 1: Beagle-xM Architecture

3. PARTICLE FILTER

In particle filter [3] we are concerned with the distribution $p(x_t | z_{1:t})$ where x_t is the unobserved state at time‘t’ and $z_{1:t}$ is the sequence of observation from time 1 to time t.

The pdf (probability density function) of a current state in particle filter is given by:

$$p(x_t | z_{1:t}) = \alpha\, p(z_t | x_t) \int p(x_{t-1} | z_{1:t-1})\, p(x_t | x_{t-1})\, dx_{t-1} \quad (1)$$

Where $p(z_t | x_t)$ expresses likelihood function that describes measurement model, $p(x_{t-1} | z_{1:t-1})$ is the pdf from last time step, $p(x_t | x_{t-1})$ = motion model, α = normalization constant.

For each time step in particle filter loop has 3 phases prediction, update and resample.

3.1 Prediction

Given the prior density $p(x_0)$, and assuming that pdf $p(x_{t1} | z_{1:t1})$ is available at time t-1, the prior pdf $p(x_t | z_{1:t-1})$ can be obtained at time‘t’ by:

$$p(x_t | z_{1:t-1}) = \int p(x_t | z_{1:t-1}) p(x_t | z_{1:t-1}) dx_{t-1} \quad (2)$$

Where, x_t is the state at time‘t’ and $z_{1:t-1}$ is the observation sequence from time 1 to time t-1.

3.2 Update

Here, the weight associated with each particle $p(z_t | x_t)$ is normalized so that all weight sum to 1.If z_t is the observation at time t, the prior pdf $p(x_t | z_{1:t-1})$ is updated with the new measurement z_t using Bayes’ rule to obtain posterior over x_t .

$$p(x_t | z_{1:t}) = \int \frac{p(z_t | x_t) p(x_t | z_{1:t-1})}{p(z_t | x_t) p(x_t | z_{1:t-1}) dx_t} \quad (3)$$

Where, x_t is the state at time t and $z_{1:t}$ is the observation sequence from 1 to time t.

3.3 Resample

Resampling [4] is a method to overcome the degeneracy of particles by replicating particles in proportion to their weights. A pre-defined threshold is used to compare against the weight of every particle. If the weight exceeds the threshold, the position of the corresponding particle is retained, while the position of the remaining particles is chosen arbitrarily in the next frame.

4. OCCLUSION CONDITION

An occlusion occurs when the maximum weight in the particle set is less than pre-defined threshold (Th) i.e.

$$\max\left(\frac{w_i}{\sum_{m=1}^{N} w_m} \right) < Th; i \in [1:N] \quad (4)$$

Where,

w_i = Weight of i^{th} particle

$\sum_{m=1}^{N} w_m$ = Sum of weight of all particles in current frame
N = number of particles

5. KALMAN FILTER

The Kalman filter [5] involves the use of linear square estimation based technique for the purpose of tracking moving targets using a series of observations taken over time to estimate the values of unknown state variables. The aforementioned algorithm is predominantly a statistical model which takes inaccuracies and other uncertainties caused due to non-planar, non-Gaussian noise into account. Since, the next state is estimated on the basis of the current state, it is essentially a memory based estimation model.

The Kalman filter is very useful for tracking objects under occlusion. The state table which comprises of relevant information regarding the displacement, velocity, etc of the moving object is continuously updated. Although the object may not be visible, the values stored in the state table enable the system to track the target. At the instance of occlusion, the next tate is predicted using the equation:

$$y_{k+1} = \Phi_k + w_k \quad (5)$$

Where, y_k is the state vector of the process at time k, Φ is the state transition matrix of the process from the state at k to the state at k + 1, and is assumed stationary over time and w_k is the associated white noise process with known covariance.

Observations on this variable can be modeled in the form:

$$m_k = Hy_k + v_k \quad (6)$$

Where, m_k is the actual measurement of y at time k, H is the noiseless connection between the state vector and the measurement vector, and is assumed stationary over time and v_k is the associated measurement error. This is again assumed to be a white noise process with known covariance and has zero cross-correlation with the process noise.

The co-variances of the two noise models are assumed stationary over time and are given by:

$$Q = E[w_k, w_k^T] \quad (7)$$

$$R = E[v_k, v_k^T] \quad (8)$$

Assuming the prior estimate of $\hat{y}_k$ is $\hat{y}'_k$ called and was gained by knowledge of the system. It possible to write an update equation for the new estimate, combining the old estimate with measurement data thus

$$\hat{y}_k = \hat{y}'_k + K_k + j_k \quad (9)$$

Where, K_k is the Kalman gain, j_k is the innovation or measurement residual and is given by $j_k = m_k - H\hat{y}_k$.

The Kalman filter works recursively on noisy streams of input data. This algorithm can be better understood through the help of the following flow diagram:

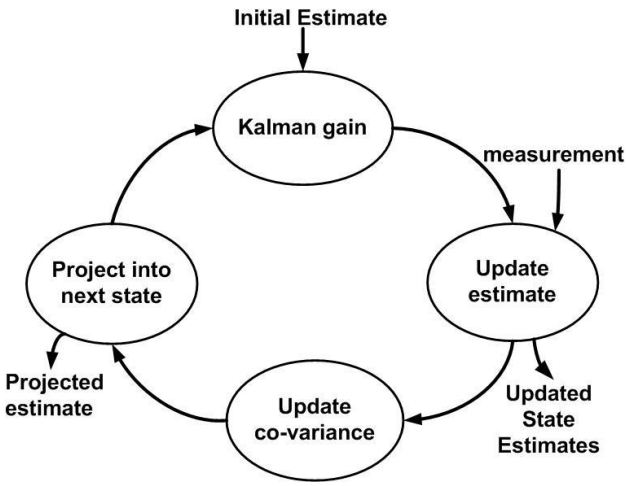


Fig 2: Flow diagram of the Kalman filter

6. THE PROPOSED ALGORITHM

The diagram of proposed algorithm is shown in fig. 3 and is described as follows:

Step 1: Consider an initial point (X_t) at position (x, y) on the reference image in the first frame (t =1).

Step 2: Extract the template image using the initial position and calculate the reference histogram.

Step 3: Generate a set of N particles $\{X_m^t\}_{m=1,2,\ldots\ldots,N}$ around the object or entire frame.

Step 4: Compute the weight for each particle using particle histogram and reference histogram using one of the following distance measures [6]:

(i) Bhattacharya Coefficient

$$\rho[H1,H2] = \sum_{u=1}^{m} \sqrt{H_{1u} * H_{2u}} \quad (10)$$

Where, u =histogram bin index, m=number of bins.
For two identical normalized histograms we obtain $\rho = 1$, indicating a perfect match. To quantify the distance [7] between two distributions, the distance'd' is defined as

$$d = \sqrt{1 - \rho[H1,H2]}$$

(ii) Correlation Coefficient Histogram Method

$$d(H_1, H_2) = \frac{\sum H'_1(i).H'_2(i)}{\sqrt{\sum H'^2_1(i).H'^2_2(i)}} \quad (11)$$

(iii) Intersection Distance

$$d(H_1, H_2) = \sum \min(H_1(i), H_2(i)) \quad (12)$$

Where, H_1 = Reference Histogram
H_2 = Particle Histogram

$$H'_1 = H_1 - \overline{H_1}$$

$$H'_2 = H_2 - \overline{H_2}$$

The value of`d' should lie between 0 and 1 and for a perfect match; value of 'd' should be close to 1 for some distance measures while it should be close to 0 for others .

Step 5: Normalize the weights by dividing weight of each particle by total weight.

$$w_{ni} = \left[\frac{w_i}{\sum_{m=1}^{N} w_m} \right]; i \in [1:N] \quad (13)$$

Where, w_{ni} = Normalized weight of i^{th} particle

w_i = Weight of i^{th} particle

$\sum_{m=1}^{N} w_m$ = Sum of weight of all particles in current frame

N = no of particles

Step 6: Decide whether the object is occluded or not from the equation 4.

Step 7: If there is occlusion, use Kalman filter approach else use particle filter approach to estimate the new position of the target.

Step 8: Resample the particles for next iteration.

Step 9: Set $t = t + 1$ and go to step 3.

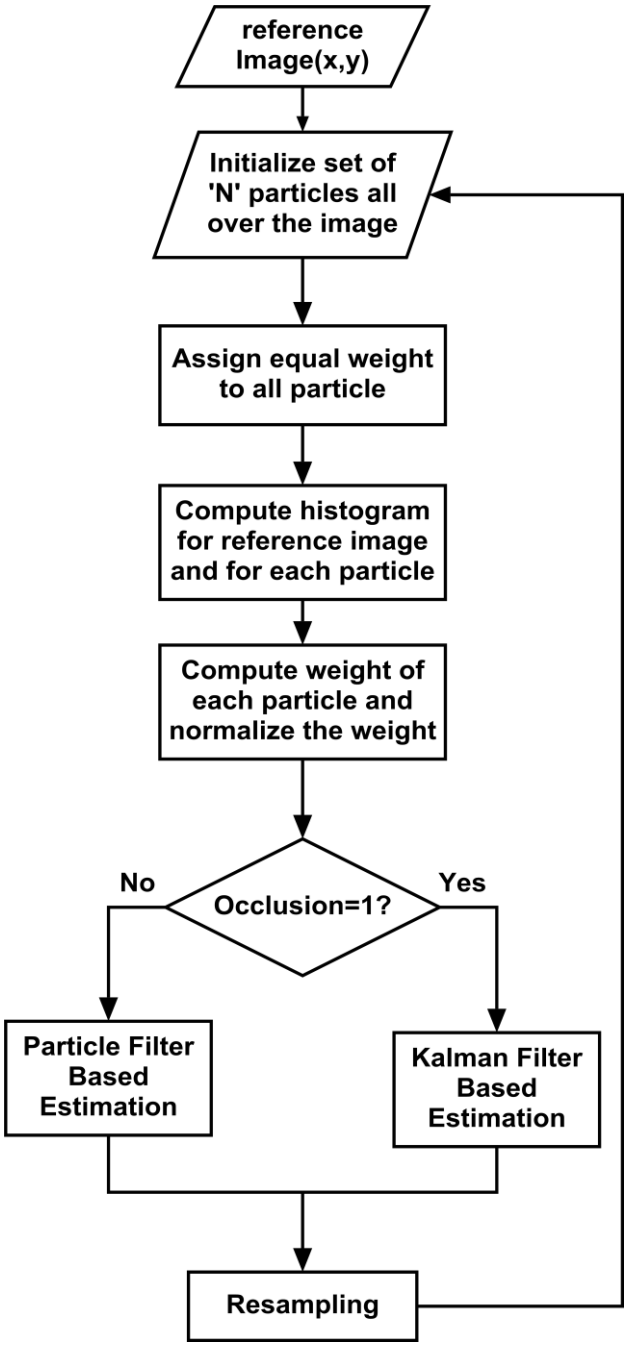


Fig 3: Flowchart of the proposed algorithm

7. OVERVIEW OF EMBEDDED SYSTEM FRAMEWORK

A minimal version of Angstrom Linux 2011.03 is setup on the BeagleBoard-xM to completely utilize the 1GHz clock frequency of the board and to provide a stable kernel. The customized software image is generated using Narcissus and is installed on the SD card for booting the device. The necessary toolchains (gcc), software libraries (sdl, gstreamer) along with their development headers and kernel modules (dsplink, cmem, uvcvideo) are integrated into the generated image to save the burden of later manual compilation and installation of the same.

The Digital Video SDK from TI is installed on a 32-bit intel PC with Ubuntu 12.10 for accessing the C6Accel [8] framework

used for compiling the DSP side kernels and the ARM [9, 10] side wrappers for the algorithms implemented. The generated objects are finally linked with the target tracking application on the board. The output of the system can be displayed on a monitor via the DVI-D output. A USB keyboard and mouse are connected to the system for user input. The entire tracking system comprises of four modules, namely:

7.1 Image Acquisition Module

A sequence of images is obtained either from a live video feed connected through a USB port or from stored videos. We make use of GStreamer [11], which is a pipeline-based multimedia framework that allows a programmer to create a variety of media-handling components for image acquisition. Moreover, additional plugins are installed to enhance the performance of the processor.

7.2 Pre-processing Module

The RGB to HSV conversion is done on the input frames in order to reduce computation time and to counter the varying light intensity problem. An efficient integer method is employed for the aforementioned conversion.

7.3 Tracking Module

The tracking module utilizes the DSP core (TI's TMS320C64x+) on the board to calculate computationally intensive tasks like RGB to HSV conversion, calculation of correlation values and distance measures. C6Accel framework provides linkage between the DSP and primary ARM processor to develop the DSP kernels, which can be called from the ARM side.

7.4 Display And User Interface Module

The input stream of frames is continuously displayed on a DVI monitor connected to the board. The user has the freedom to precisely specify the object on the input frame to be tracked through a mouse input. A template is extracted around the clicked co-ordinate and is used as a reference for the tracking module. The tracked target is also elucidated on the displayed image by a colored rectangle. The Simple Direct Media Layer (SDL) [12], a cross-platform development library, provides a common framework to handle mouse input and 2D pixel operations.

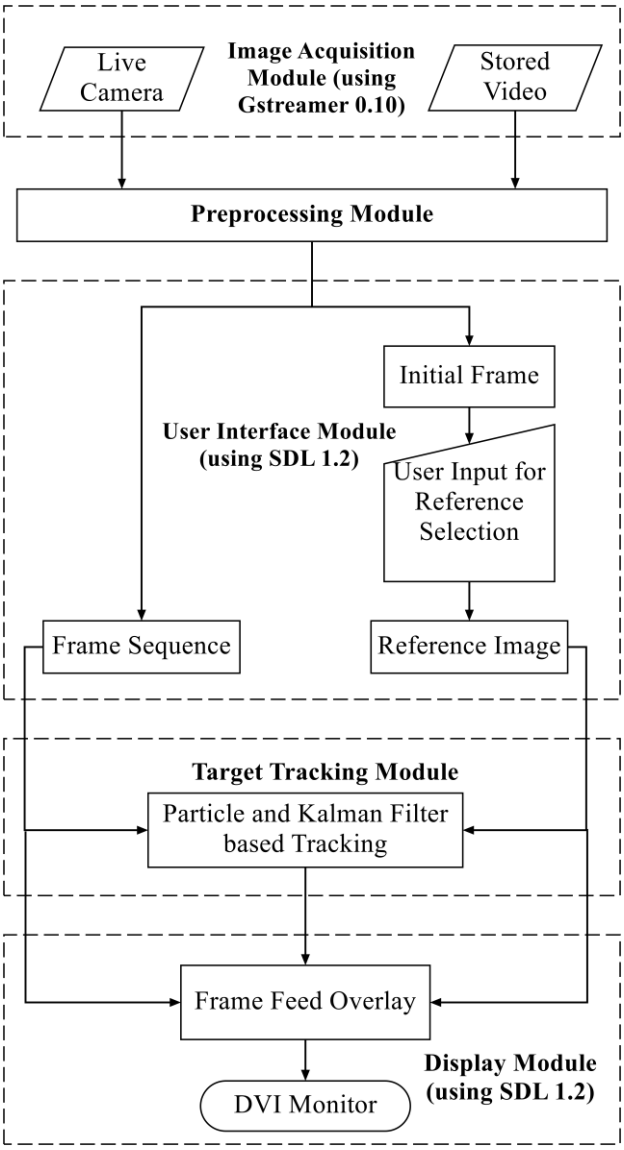


Fig 4: Flowchart of the embedded Tracking System

8. TIMING ANALYSIS AND RESULTS

8.1 Timing Analysis

The execution time per frame for various distance measures with and without the utilization of DSP processor on the board is compared in Table 1.The performance of the proposed algorithm is improved by 3 times and above after utilizing the computation power of the onboard DSP core since all the computationally intensive tasks like calculation of distance measure and RGB to HSV conversions were executed parallely on the DSP (TMS320c64X+) supplied in the board.

As depicted in table 1, the execution time for Bhattacharya distance is found to be the highest among the three distance measures used while the intersection distance consumes the least computation time. When the onboard DSP is utilized, all the distance measures have provided a performance fairly above the real-time requirement of 30 fps. Bhattacharya distance was found to be the most accurate measures among the three and was considered in the final implementation.

Table1: Timing results for different distance measured

Distance Measures	Execution time per frame(ms)	
	Without DSP	With DSP
Bhattacharya	42	13
Correlation	35	11
Intersection	31	10

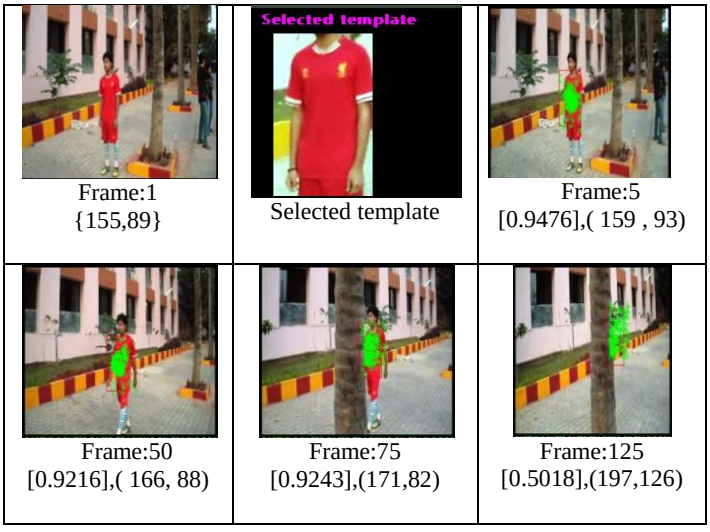
8.2 Results

The failure of general Particle filter during occlusion is depicted in figure 5.In the figure, occlusion has occurred at frame number 125 and the object to be tracked is not visible in the video frame. The particle filter based Algorithm hence has lost track of the object to be tracked and has given an undesirable match.

The results obtained for the same video when both Particle and Kalman filters are used is shown in figure 6. It is clearly visible from the figure that even when the object is not visible after frame no 125, Kalman filter is able to estimate the likely position of the target. The green rectangle as depicted in the figure shows the prediction done by the Kalman filter and the red rectangle denotes the final updated co-ordinates. When the object is visible, Particle filter gives the final updated co-ordinates.

Figure 7 depicts the results when Bhattacharya distance was used in the final implementation. It is clearly visible that the match value has significantly increased when compared to figure 5 and figure 6 where correlation distance was used to find the histogram similarity.

Frame size: 320x240
Template Size: 40x80
Number of particles: 250



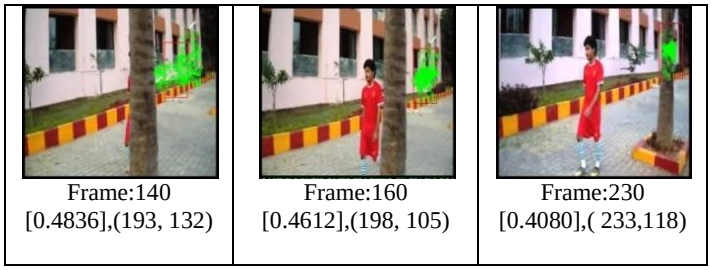


Fig 5: Failure of Particle Filter during occlusion.

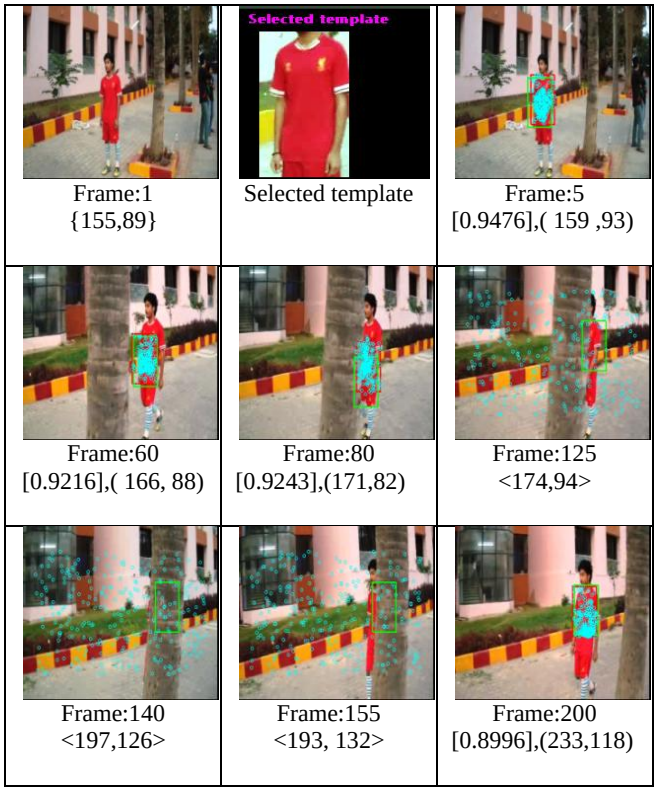
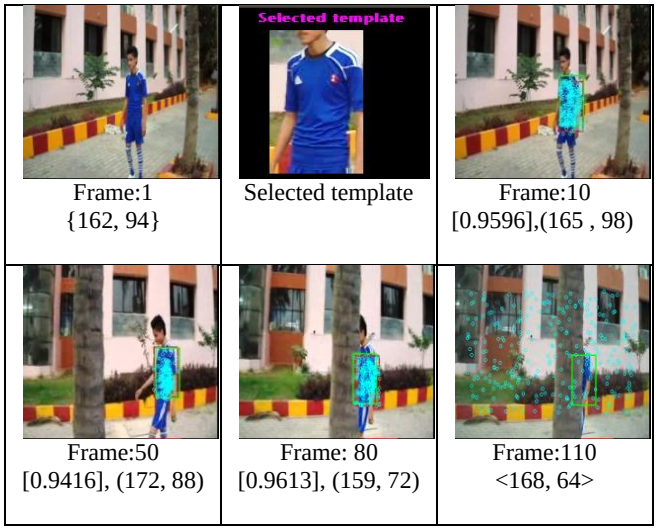
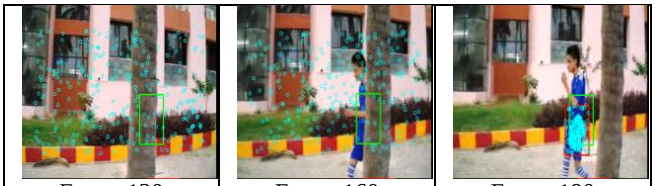


Fig 6: Overcome of occlusion using proposed algorithm.





Frame:130	Frame:160	Frame:190
<166, 67>	<210, 72>	[0.9458], (193, 55)

Fig 7: Kalman filter predicts the position of occluded object.

Note:
(i) The target coordinates obtained for each frame are indicated by parenthesis (x, y).
(ii) The match value computed for each frame is indicated by square brackets [ ].
(iii) The initial coordinates obtained for the first frame is indicated by curly brackets {x, y}.
(iv) The predicted values obtained using the Kalman Filter is indicated by < x, y >.

9. CONCLUSION AND FUTURE SCOPE

The introduction of Kalman Filter in the general Particle filter is able to track the object in both normal and occluded conditions. However, Kalman Filter used in the proposed algorithm is able to predict the occluded object only for duration of few seconds. As the error in the process increases with occlusion duration, prediction fails after a certain time. Also the Kalman Filter provides only the linear estimate of the occluded object.
Hence, one major aspect of the future work is to use the extended Kalman filter for nonlinear state estimation which, in turn, increases the time for which the likely position of the occluded object is predicted successfully.

10. REFERENCES

[1] A. P. Li, Z. L. Jing, and S. Q. Hu, “Robust Observation Model for Visual Tracking in Particle Filter”, International Journal of Electronics and Communication, Vol. 61, No.3, pp. 186-194, 2007

[2] BeagleBoard-xM Rev C System Reference Manual – http://www.beagleboard.org

[3] Rajbabu Velmurugan,”Implementation Strategies For Particle Filter Based Target Tracking”, School of Electrical and Computer Engineering,Georgia Institute of TechnologyMay 2007.

[4] Jeroen D. Hol, Thomas B. Sch¨on, Fredrik Gustafsson, “On Resampling Algorithms For Particle Filters”, Division of Automatic Control,Department of Electrical Engineering, Link¨oping University,SE-581 83, Link¨oping, Sweden.

[5] Maria Isabel Ribeiro, “Kalman and Extended Kalman Filters:Concept, Derivation andProperties”,Institute for Systems and Robotics, Instituto Superior T´ecnico, Av. Rovisco Pais, 11049-001 Lisboa Portugal,2004.

[6] Ling, H., Okada, K, “An Efficient Earth Mover’s Distance Algorithm for Robust Histogram Comparison”, PAMI, 2007.

[7] S´everine Dubuisson, “The computation of the bhattacharyya distance between histograms without histograms”, Laboratoire d’Informatique de Paris 6, Universit´e Pierre et Marie Curie.

[8] C6Accel Advanced Users Guide http://processors.wiki.ti.com/index.php?title=C6Accel_Advanced_Users_Guide

[9] Lian Pan, Xiaoming Liu, "The Study of Target Tracking Based on ARM Embedded Platform", Journal of Computers, Vol 7, No 8, pp 2015-2023, August 2012.

[10] Shen Khang, Vooi Voon Yap, Patrick Sebastian, "Implementation and Optimization of Human Tracking System using ARM Embedded Platform", International Conference on Intelligent & Advanced Systems, Vol 1, pp 353 – 356, June 2012

[11] GStreamer Manual – http://www.gstreamer.net/

[12] Simple Directmedia Layer (SDL) – http://www.libsdl.org/