International Journal of Current Engineering and Technology
E-ISSN 2277 – 4106, P-ISSN 2347 – 5161

Available at http://inpressco.com/category/ijcet

Research Article

Evaluation of Moving Object Tracking Techniques for Video Surveillance Applications

Mahmoud A. Aldosokey†*, Wael A. Mohamed† and Mahmoud F. Mahmoud†

†Electrical Engineering Department, Benha Faculty of Engineering, Benha University, Qalubia, Egypt

Accepted 16 Nov 2015, Available online 26 Nov 2015, **Vol.5, No.6 (Dec 2015)**

Abstract

This paper presents a comparison between the performance of Kalman filter and particle filter in the single object tracking system. The performance parameters of interest are the processing time of the algorithm and the accuracy of the tracking system. The algorithms are implemented using MATLAB and tested using two video sequences, and the results are compared and discussed.

Keywords: *Object tracking, Kalman filter, Particle filter*

1. Introduction

Video surveillance is an active research topic in computer vision that detects, recognizes and tracks objects over a sequence of images and it also understand and describe the object behavior by replacing the aging old traditional method of monitoring cameras by human operators. Object detection and tracking are important and challenging tasks in many computer vision applications such as surveillance, vehicle navigation and autonomous robot navigation.

Object tracking is the process of locating an object or multiple objects over time using a camera. There are three key steps in video analysis, detection interesting moving objects, tracking of such objects from each and every frame to frame, and analysis of object tracks to recognize their behavior (Rupesh Kumar Rout, 2013).

Moving object detection is the basic step for many video analysis tasks (M. Heikkila and M. Pietikainen, 2006). The performance of this step is particularly significant because subsequent processing of the video data is greatly dependent on this step. Moving object detection aims at extracting moving objects that are of interest in video sequences. The problems with dynamic environmental conditions make moving object detection very challenging. Some examples of these problems are shadows, sudden or gradual illumination changes, rippling water, and repetitive motion from scene clutter such as waving tree leaves (R. Radke *et al*, 2005).

The next step in the video analysis is object tracking. This problem can be formulated as a hidden state estimation problem given available observations (Li and Chellappa, 2002).

*Corresponding author: **Mahmoud A. Aldosokey**

The final step of an "intelligent" visual surveillance system is to analyze the content of the video and to identify important events in a scene. This step provides human operators with high level analyses to assist them with making their decisions more accurately, effectively, and efficiently. Events in a scene can be defined by the behavior of an individual, several persons, or a crowd.

2. System Description

A block diagram of the system is given in Figure 1. An image sequence captured by a still camera is the input to the system. We first perform motion segmentation to extract moving objects in the current frame.

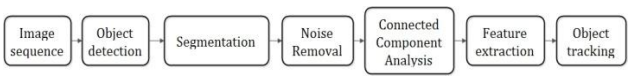


Fig.1 Block diagram of the Object tracking system

Following the segmentation of the image frame into moving and non-moving pixels, it comes filtering phase. In order to fuse narrow breaks and long, thin gulfs, eliminates small holes, and fills gaps in the contour, a morphological operation is applied to the image. As a result, small gaps between the isolated segments are erased and the regions are merged. The BLOBs with a size smaller than a selected threshold are likely to be noise and are deleted.

To extract the bounding boxes of detecting objects, connected component analysis was used.

Features are extracted from the detected objects to distinguish between objects and to enable tracking of these objects through the image sequence. The features

we use throughout this thesis is the moving object position (two dimensional). Finally, several object tracking techniques are implemented and the results are compared.

3. Gaussian Mixture Background Modeling

Gaussian mixture background modeling technique is commonly used for performing background segmentation. Stauffer and Grimson *et al* (1999), have proposed a probabilistic approach using a mixture of Gaussian for identifying the background and foreground objects.

P. Wayne Power and Johann A. Schoonees (2002), have described a practical implementation of the Stauffer and Grimson algorithm and provide values for all model parameters. The paper also shows what approximations to the theory were made and how to improve the standard algorithm by redefining these approximations.

If each pixel resulted from a particular surface under particular lighting, a single Gaussian would be sufficient to model the pixel value while accounting for acquisition noise. If only lighting changed over time, a single, adaptive Gaussian per pixel would be sufficient. In practice, multiple surfaces often appear in the view of a particular pixel and the lighting conditions change. Thus, multiple, adaptive Gaussians are necessary. We use a mixture of adaptive Gaussians to approximate this process.

In this approach the color values (pixel history) of a particular background pixel is modeled by a mixture of k adaptive Gaussian components (k= 3 or 4 or 5), (Stuthi Eddula *et al*, 2012). For each new frame each pixel is compared to the k components we have. Based on the persistence and the variance of each of the Gaussians of the mixture, we determine which Gaussians may correspond to background colors. Pixel values that do not fit the background distributions are considered foreground until there is a Gaussian that includes them with sufficient, consistent evidence supporting it.

- At each iteration gaussians are evaluated using a simple heuristic to determine which ones are most likely to correspond to the background.
- Pixels that do not match with the background gaussians are classified as foreground.

The probability of observing the current pixel value is:

$$P(X_t) = \sum_{k=1}^{K} \omega_{k,t} * \eta\,(X_t, \mu_{k,t}, \Sigma_{k,t}) \quad (1)$$

Where:

$$\eta(X_t, \mu_{k,t}, \Sigma_{k,t}) = \frac{\exp\left[-0.5(X_t - \mu_{k,t})^{\mathrm{T}} \Sigma_{k,t}{}^{-1} (X_t - \mu_{k,t})\right]}{(2\pi)^{0.5|X_t|}\,|\,\Sigma_{k,t}|^{0.5}} \quad (2)$$

$$\sum_{k=1}^{K} \omega_k = 1 \quad (3)$$

Where:

k: is the number of the Gaussian components (***k*** **=** 3 or 4 or 5)

$\omega_{k,t}$: represents the weight of the k^{th} Gaussian in the mixture, i.e. (what portion of the data is accounted for by this Gaussian).

$\mu_{k,t}$: representing the mean value of each ***k***-Gaussian in the mixture at time t.

$\Sigma_{k,t}$: is the covariance matrix of the k^{th} Gaussian in the mixture at time t.

η: is a Gaussian probability density function.

The weight indicates the influence of the k^{th} Gaussian and time t. Since it is an iterative process that all parameters are updated, with the inclusion of every new pixel. Before update take place the new pixel is compared to see if it matches any of the ***k*** existing Gaussian. A match is determined if:

$$|\,X_t - \mu_{k,t}| < 2.5\,\sigma$$

Where X_t is the pixel value at time t and σ correspond to the standard deviation of the Gaussian.

The match present three cases for the update. For each case, X_t is a value for the pixel (i_t, j_t) in frame t. The three cases are discussed below:

The first case is for a matching k-Gaussian in the mixture for pixel (i_t, j_t). In this case we update the weight, mean, and variance from the following equations:

$$\omega_{k,t} = (1 - \alpha) * \omega_{k,t-1} + \alpha \quad (4)$$

$$\mu_{k,t} = (1 - \rho) * \mu_{k,t-1} + \rho * Xt \quad (5)$$

$$\sigma^{2}_{k,t} = (1 - \rho) * \sigma^{2}_{k,t-1} + \rho * (X_t - \mu_{k,t})^{\mathrm{T}} * (X_t - \mu_{k,t}) \quad (6)$$

Where:

α : is the learning rate parameter.

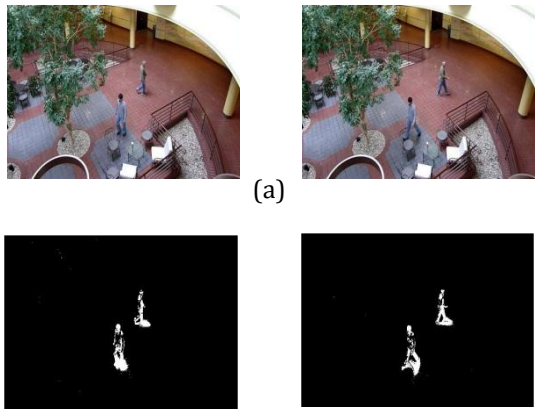
ρ: is the Gaussian probability density function scaled by α with the following parameters:

$$\rho = \alpha * \eta(X_t, \mu_{k,t}, \Sigma_{k,t}) \quad (7)$$

The second case is for a non-matching k-Gaussian. Here, we only update the weight with the following equation so the weight is decreased. The mean and variance parameters do not change.

$$\omega_{k,t} = (1 - \alpha) * \omega_{k,t-1} \quad (8)$$

The third case is when none of the k-Gaussians match for our pixel (it, jt). This implies that the pixel value X_t is not close to any of the distributions in the mixture, ie. X_t is potentially a foreground pixel and is marked as such.



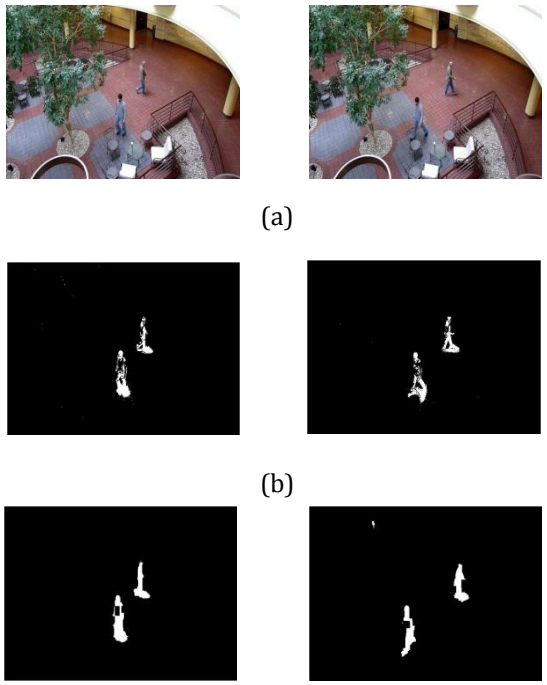
(a)

(b)

Fig. 2 (a) original video frames, (b) output of the Gaussian mixture model

4. Morphological Operations

The morphological operations used are opening (is an erosion process followed by a dilation process) and closing (is a dilation process followed by an erosion).



(a)

(b)

(c)

Fig. 3 (a) original video frames, (b) output of the Gaussian mixture model and (c) Output after applying morphological operations

The structuring elements used in the opening and closing operations are rectangular structuring elements of size 2*2 and 13*13. The simulation results of applying morphological operations to the output of the Gaussian mixture model are shown on Figure 3.

5. Kalman Filter

The Kalman filter is used to estimate the state of the system, based on the current and previous states, that tend to be more precise than those based on a single the measurements alone.

The Kalman filter averages a prediction of a system's state with a new measurement using a weighted average. The purpose of the weights is that values with better (i.e., smaller) estimated uncertainty are "trusted" more. The weights are calculated from the covariance, which is a measure of the estimated uncertainty of the prediction of the system's state. The result of the weighted average is a new state estimate that lies between the predicted and measured state, and has a better estimated uncertainty than either alone. This process is repeated every time step, this means that the Kalman filter works recursively and requires only the last "best guess", rather than the entire history, of a system's state to calculate a new state.

Kalman Filter Phases

Kalman filter has two steps: the prediction step, where the next state of the system is predicted given the previous measurements, and the update step, where the current state of the system is estimated given the measurement at that time step.

System Model

The system model describes the transition from one state to the next, for the standard Kalman filter, the state transition from *it* to *t+1* can be expressed by the equation:

$$x_{t+1} = A\,x_t + w_t \quad (9)$$

Where A is referred to as the state transition matrix and w_t Is the process noise which is assumed to be Gaussian random variable with zero mean and a covariance matrix Q. The covariance matrix Q accounts for possible changes in the process between t and $t+1$ that are not already accounted for in the state transition matrix. Another assumed property of w_t is that it is independent of the state x_{t+1}.

Measurement Model

Assume we have a measurement (z_{t+1}) at current time step (t+1) which is used to update the system state estimate (x_{t+1}). It's assumed that the measurement is linearly related to the state estimate by an equation of the form:

$$z_{t+1} = H\,x_{t+1} + v_{t+1} \quad (10)$$

Where H is the measurement sensitivity matrix which relates the state to the measurement and v_{t+1} Is the measurement noise which is assumed to be Gaussian

random variable with zero mean and a covariance matrix *R*.

Prediction Phase (time update)

The state estimate from the previous time step is used to produce an estimate of the state at the current time step. This predicted state estimate is also known as the *a priori* state estimate because, although it is an estimate of the state at the current time step, it does not include observation information (current measurement z_{t+1}) from the current time step.

$$\underline{x}_{t+1} = A*x_t \quad (11)$$

Where:
$\underline{x}_{t+1}$: Is the prior (predicted) current state estimate before seeing the measurement.
x_t: Previous state estimate.
A: State transition matrix

$$\underline{P}_{t+1} = A*P_t*A^T + G*Q*G^T \quad (12)$$

Where:
$\underline{P}_{t+1}$: prior (predicted) error covariance matrix before seeing the measurement.
P_t: error covariance matrix of previous state.

Correction Phase (measurement update)

$$K_{t+1} = \underline{P}_{t+1} * B^T * inv(B * \underline{P}_{t+1} * B^T + R) \quad (13)$$
$$x_{t+1} = \underline{x}_{t+1} + K_{t+1} * (z_{t+1} - B * \underline{x}_{t+1}) \quad (14)$$
$$P_{t+1} = (I - K_{t+1} * B) * \underline{P}_{t+1} \quad (15)$$

x_{t+1}: Current state estimate after seeing the measurement.
z_{t+1}: Is the current (noisy) measurement.
P_{t+1}: Estimated error covariance matrix after seeing the measurement.
Q: The noise covariance matrix. It accounts for possible changes in the process between t and t+1 that are not already accounted for in the state transition matrix.

6. Particle Filter

Particle filters have an advantage over Kalman filters in that they can model any multimodal distribution, whereas Kalman filters are constrained by assumptions that the state and sensory models are linear, and the noise and posterior distribution are Gaussian.

Particle filters use a set of samples (particles) with associated weights to approximate the posterior PDF. Like a Kalman filter, the process contains two major steps each time step, prediction and update.

The state of the filter at time t, is represented by x_t, and its history $X_t = (x_1, x_2, \ldots, x_t)$. The observation (measurement) at time t is z_t, and its history is $Z_t = (z_1, z_2, \ldots, z_t)$. It is assumed that the object dynamically from a temporal first order Markov chain so that the next state is totally dependent on directly previous state.

$$p(x_t \mid X_{t-1}) = p(x_t \mid x_{t-1}) \quad (16)$$

Observations are considered to be mutually independent, the observation process is defined by specifying the conditional density, $p(z_t \mid x_t)$ at each time step t:

$$p(Z_t \mid X_t) = \prod_{i=1}^{t} p(z_i \mid t_i) \quad (17)$$

The conditional state density at time t is defined as:

$$P_t(x_t) = p(x_t \mid Z_t) \quad (18)$$

State density is propagated over time, according to the rule:

$$p(x_t \mid Z_t) = k_t\, p(z_t|x_t)\, p(x_t|Z_{t-1}) \quad (19)$$

In our computational environment, we approximate the posterior $p(x_t \mid Z_t)$ as a set of N samples, $\{s^i_{t=1\ldots\ldots N}\}$, where each sample has an importance weight ω^i_t. When the filter is initialized, this distribution is drawn from the prior density *p(x).* At each time step the distribution is re-sampled to generate an un-weighted particle set. Re-sampling is done according to the importance weights.

The implemented particle filter algorithm is outlined below and illustrated in figure 4.

- Initialization: at t = 0 (or when the object first appears).

For i = 1……N, select samples s^i_0 From the prior distribution $p(x_0)$.

- Iterate for t = 0,1,2,….T

1. Importance sampling

(a) Predict the samples next position:

$$s^i_t = p(x_t|x_{t-1} = s^i_{t-1}) \quad (20)$$

The prediction process is governed by the needs of the individual system, but typically involves adjusting the position according to pre-defined system dynamics.

(b) Evaluate importance weights based on the measured features (observation), z_t:

$$\omega^i_t = p(z_t|x_t = s^i_t) \quad (21)$$

(c) Normalize the importance weights:

$$\omega^i_t = \frac{\omega^i_t}{\sum_j^N \omega^j_t} \quad (22)$$

2. Re-sampling

(a) Resample from s_t^i So that samples with a high weight, ω_t^i are emphasized (sampled multiple times) and samples with a low weight are suppressed (re-sampled few times)

(b) Set ω_t^i to $\frac{1}{N}$ for i = 1……N.

(c) The resultant sample set can be used to approximate the posterior distribution.

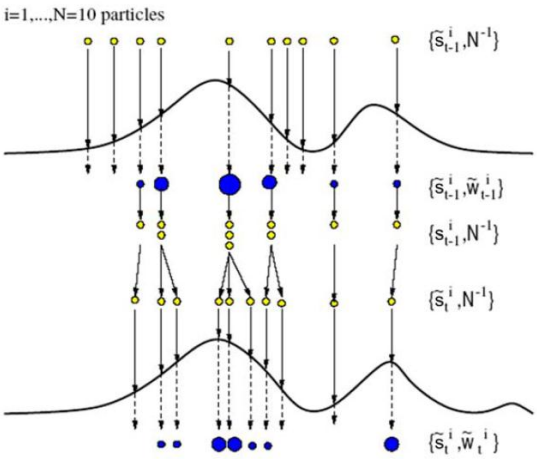


Fig. 4 The Particle Filter Algorithm

The re-sampling step (2(a)) is required to avoid degeneracy in the algorithm by ensuring that all weights does not associated with only one particle. Sequential importance re-sampling (SIR) is a commonly used re-sampling scheme that uses the following steps to select a new sample. The steps are applied for n = 1…..N.

1. Generate a random number, r ε [0,1].
2. Find the smallest j which satisfies $\sum_1^j \omega_t^j \geq r, j \varepsilon [1 \ldots N]$
3. Set $s_t'^i = s_{t-1}^j$

Particles with a higher weight are more likely to be sampled (particles are replicated in proportion to their weights), resulting in higher weighted being given greater emphasis in the next time step.

7. Simulation Results

7.1 Single Object Tracking Results Using Kalman Filter

The implemented system for single moving object tracking using the Kalman filter is tested by two video sequences. The first video sequence has a resolution of 480 * 360 pixels and frame rate 15 frames per second. On the first video sequence the moving object does not undergo any occlusion. The tracking system succeeded to track the moving object from the moment the object appears till the object exits the scene.

The tracking system performance is illustrated in figure 5. The measured trajectory and the predicted trajectory in the x direction and y direction are shown in figure 6 and 7 respectively.

- Video sequence 1 frame 52, 60

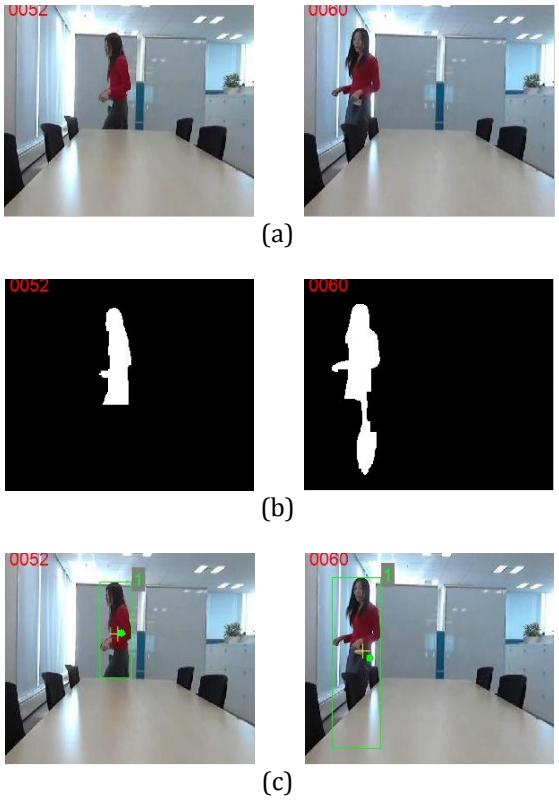


Fig 5(a) Original video frames, (b) Output of segmentation stage (c) Output of the tracking system

The average processing time per frame for the tracking stage is calculated (5.559 msec). Also the relative root mean square error for the tracking system in x and y directions are calculated. These results will be suitable when including the tracking system in real time applications.

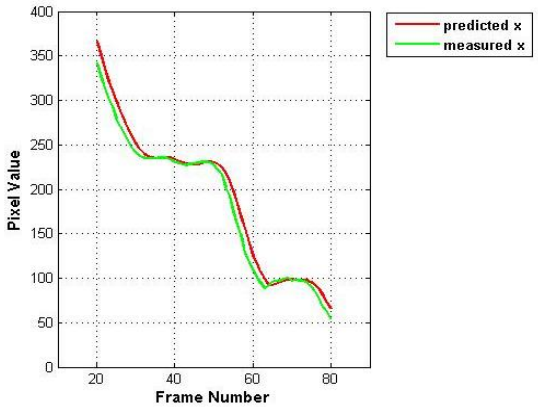


Fig. 6 The moving object trajectory in x direction, measured trajectory with green curve and estimated trajectory with red curve

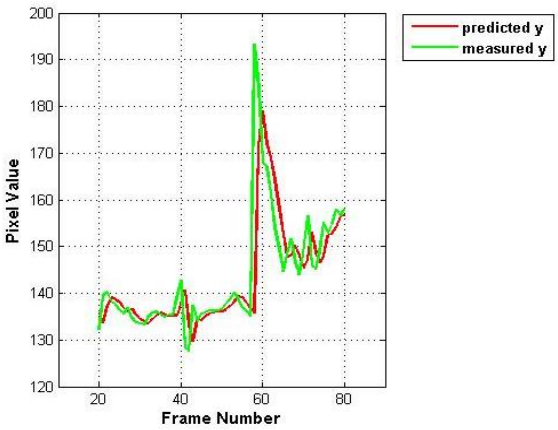


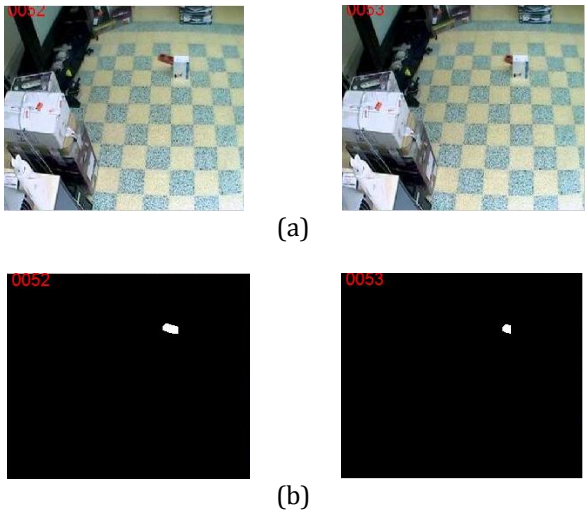
Fig. 7 The moving object trajectory in y direction, measured trajectory with green curve and estimated trajectory with red curve

Processing time per frame: 5.559 msec
Relative mean square error in x direction: 3.9064 %
Relative mean square error in y direction: 1.7807 %

The other video sequence used to test the implemented system for single moving object tracking using the Kalman filter has a resolution of 480 * 360 pixels and frame rate 25 frames per second. This video sequence involves object occlusion, the moving object becomes occluded at frame 54. When the object occluded there is no measurement is supplied to the tracking system, the tracking system assumes that the object current position during occlusion is the same as the last measurement detected. The tracking system succeeded to track the moving object from the moment the object appears till the object exits the scene.

The tracking system performance is illustrated in figure 8 and figure 9. The measured trajectory and the predicted trajectory in x direction and in y direction are shown in figure 10 and 11 respectively.

- Video sequence 2 frames 52, 53



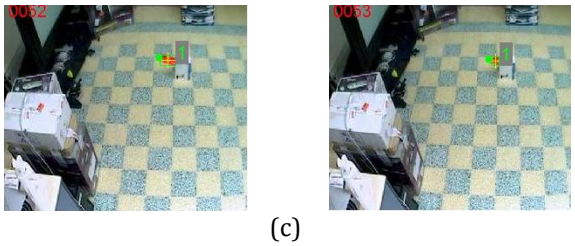


Fig. 8 (a) Original video frames, (b) Output of segmentation stage (c) Output of the tracking system

Video sequence 2 frames 54, 56

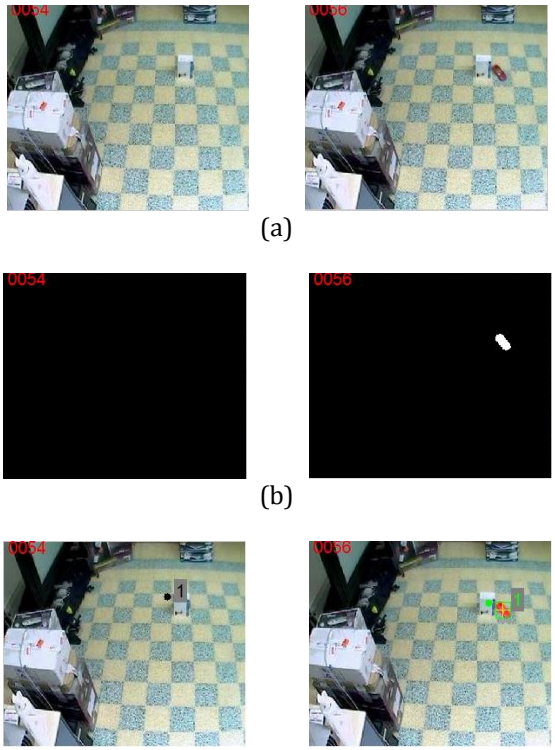


Fig.9 (a) Original video frames, (b) Output of segmentation stage (c) Output of the tracking system

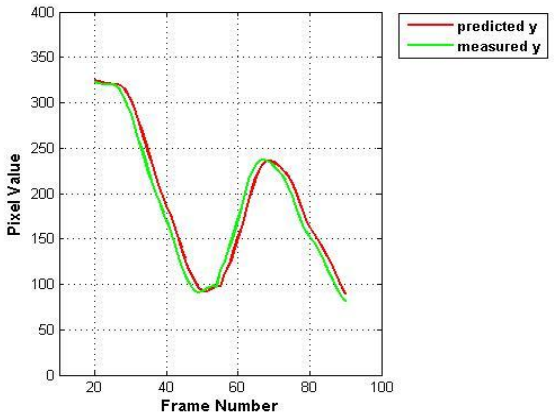


Fig. 10 The moving object trajectory in x direction, measured trajectory with green curve and estimated trajectory with red curve

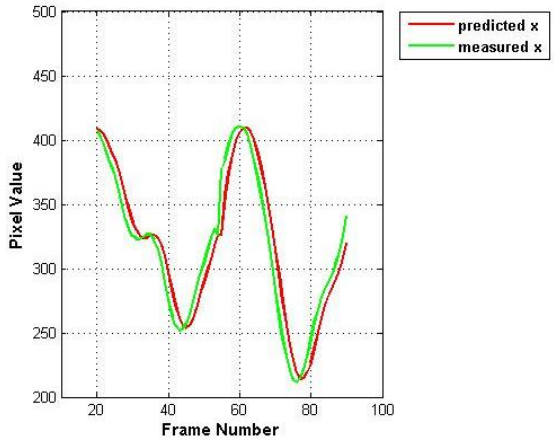


Fig. 11 The moving object trajectory in y direction, measured trajectory with green curve and estimated trajectory with red curve

Processing time per frame: 5.0913 msec
Relative mean square error in x direction: 3.3067 %
Relative mean square error in y direction: 5.2771 %

7.2 Single Object Tracking Results Using Particle Filter

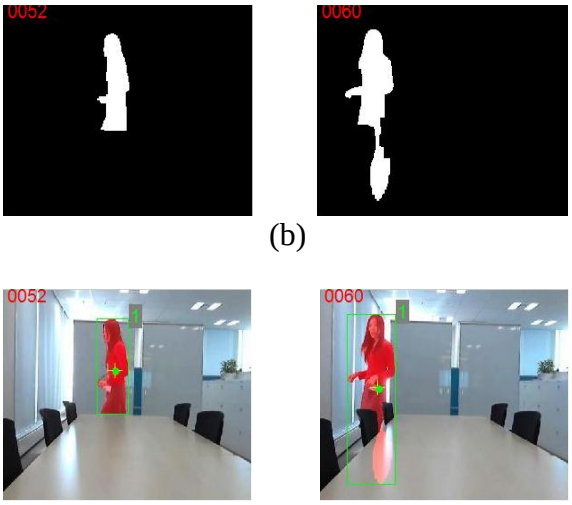
The implemented system for single moving object tracking using the particle filter is tested by the two video sequences we use to earlier to test the system implemented with the Kalman filter. The tracking system succeeded to track the moving object from the moment the object appears till the object exits the scene.

The tracking system performance is illustrated in figure 12. The measured trajectory and the predicted trajectory in x direction and in y direction are shown in figure 13 and 14 respectively.

The average processing time per frame for the tracking stage is calculated (135.9419 msec). Also the relative root mean square error for the tracking system in x and y directions are calculated. Comparing these results with the results obtained from the system implemented with Kalman filter for the same video sequence, we notice that the processing time per frame for the system based on the Kalman filter is much lower than the processing time per frame for the system implemented by the particle filter.

- Video sequence 1 frame 52, 60

(a)



(b)

(c)

Fig. 12 (a) Original video frames, (b) Output of segmentation stage (c) Output of the tracking system

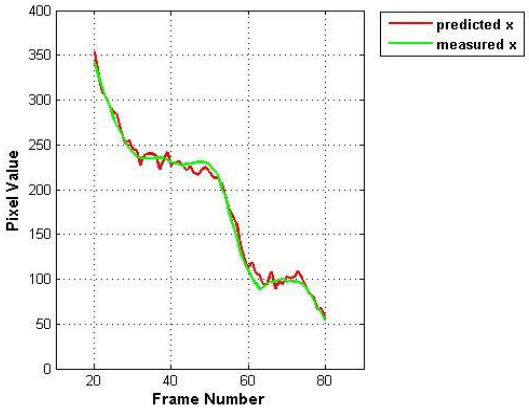


Fig. 13 The moving object trajectory in x direction, measured trajectory with green curve and estimated trajectory with red curve

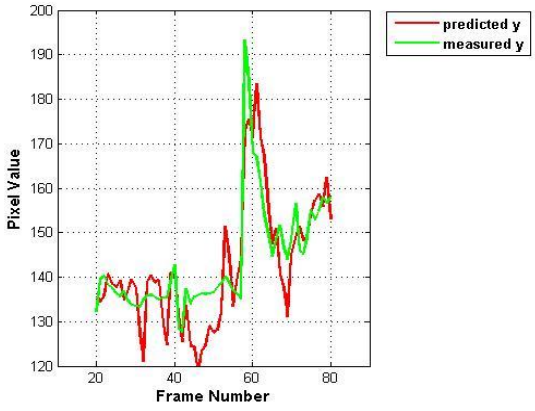


Fig. 14 The moving object trajectory in x direction, measured trajectory with green curve and estimated trajectory with red curve

Number of particles: 700
Processing time per frame: 135.9419 msec
Relative mean square error in x direction: 3.6681 %
Relative mean square error in y direction: 3.6449 %

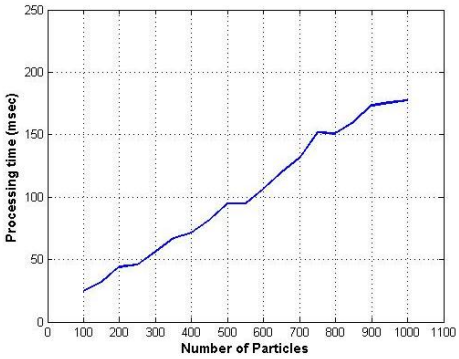


Fig. 15 The processing time per frame (msec) with respect to the number of particle we set in the particle filter algorithm

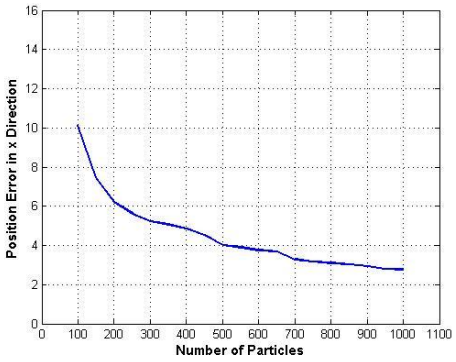


Fig. 16 The position error in x – direction with respect to the number of particles we set in the particle filter algorithm

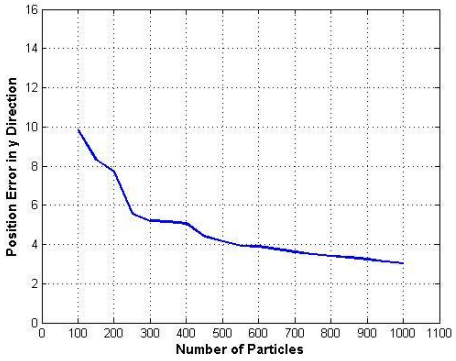


Fig. 17 The position error in the y – direction with respect to the number of particles we set in the particle filter algorithm

The other video sequence used to test the implemented system for single moving object tracking using the particle filter has a resolution of 480 * 360 pixels and frame rate 25 frames per second. This video sequence involves object occlusion, the moving object becomes occluded at frame 54. When the object occluded there is no measurement is supplied to the tracking system. The tracking system failed to track the moving object from the moment the object appears till the object exits the scene.

- Video sequence 2 frames 52, 53

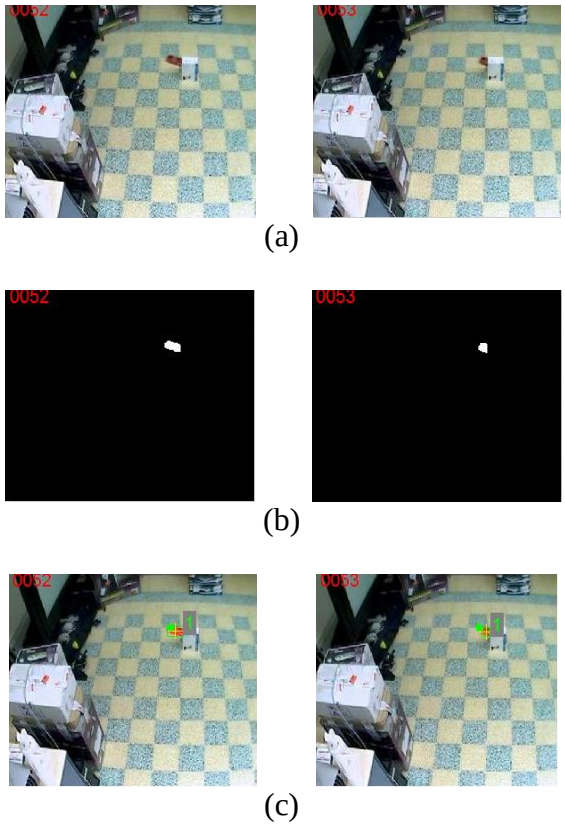
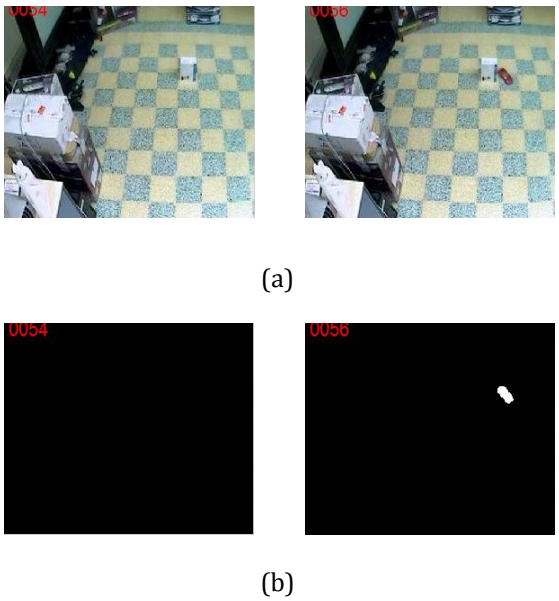


Fig. 18 (a) Original video frames, (b) Output of segmentation stage (c) Output of the tracking system

The tracking system performance is illustrated in figure 18 and figure 19. As shown occlusion occurred at frame 54, before occlusion at frames 52, 53 the moving car is labeled as 1 after occlusion at frame 56 the moving car is labeled as 2.

- Video sequence 2 frames 54, 56



(c)

Fig. 19 (a) Original video frames, (b) Output of segmentation stage (c) Output of the tracking system

Conclusions

1) The performance of the system based on the Kalman filter is much better than the system based on the particle filter from the processing time point of view. Also the system based on the particle filter will not be suitable when including the tracking system in real time applications.
2) The tracking error for the system implemented by the particle filter decreases significantly with increasing the number of particles used, but the processing time also increases, this makes the particle filter more suitable to be used in the off line application, i.e. not with real time applications.

References

Rupesh Kumar Rout, (2013) A Survey on Object Detection and Tracking Algorithms, master thesis, Computer Science and Engineering Department, National Institute of Technology Rourkela, Rourkela, India.

Stuthi Eddula, P Chandana, Prof. Ben Santhi Swaroop, (2012), Implementing Gaussian Mixture Model for Identifying Image Segmentation, International Journal of Application or Innovation in Engineering & Management, Volume 1, Issue 3, November 2012.

M. Heikkila and M. Pietikainen, (2006), A texture-based method for modeling the background and detecting moving objects, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 28, no. 4, pp. 657–662.

R. Radke, S. Andra, O. Al-Kofahi, and B. Roysam, (2005), Image change detection algorithms: a systematic survey, IEEE Transactions on Image Processing, vol. 14, no. 3, pp. 294–307.

Li and Chellappa, (2002), A generic approach to simultaneous tracking and verification in video, IEEE Transactions on Image Processing, vol. 11, pp. 530–544.

Chris Stauffer and W Eric L Grimson, (1999), Adaptive background mixture models for real-time tracking. In Computer Vision and Pattern Recognition, 1999. IEEE Computer Society Conference on Computer Vision and Pattern Recognition., volume 2.

P. Wayne Power and Johann A, (2002), Schoonees. Understanding Background Mixture Models for Foreground Segmentation. Proceedings Image and Vision Computing New Zealand 2002 University of Auckland, Auckland, New Zealand.