A Stochastic Resampling Based Selective Particle Filter for Robust Visual Object Tracking

Neilay Khasnabish

A Thesis Submitted to

Indian Institute of Technology Hyderabad

In Partial Fulfillment of the Requirements for

The Degree of Master of Technology

Department of Electrical Engineering

June 2016

Declaration

I declare that this written submission represents my ideas in my own words, and where ideas or words of others have been included, I have adequately cited and referenced the original sources. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission. I understand that any violation of the above will be a cause for disciplinary action by the Institute and can also evoke penal action from the sources that have thus not been properly cited, or from whom proper permission has not been taken when needed.

Neilay Khasnabish

(Signature)

Neilay Khasnabish

(Neilay Khasnabish)

EE14MTECH11036

(Roll No.)

Approval Sheet

This Thesis entitled A Stochastic Resampling Based Selective Particle Filter for Robust Visual Object Tracking by Neilay Khasnabish is approved for the degree of Master of Technology from IIT Hyderabad

[illegible]

(Dr. Vineeth N Balasubramanian) Examiner
Department of Computer Science and Engineering
IITH

Soumya Jana

(Dr. Soumya Jana) Examiner
Department of Electrical Engineering
IITH

[illegible]

(Dr. Ketan P. Detroja) Adviser
Department of Electrical Engineering
IITH

[illegible]

(Dr. K. Sri Rama Murty) Chairman
Department of Electrical Engineering
IITH

Acknowledgements

First I would like to thank my advisor Dr. Ketan P. Detroja of Department of Electrical Engineering at IIT Hyderabad for his guidance and motivation. I bow my sincere thanks to all my teachers from whom I learned a lot throughout the years of my study. At the same time, I gratefully acknowledge [1, 2, 3] as I used their benchmark test videos to evaluate the performance of my algorithm. Finally I express my gratitude towards my parents and my friends for encouraging me all the time.

Dedication

For my parents, the source of my inspiration

Abstract

In this work, a new variant of particle filter has been proposed. In visual object tracking, particle filters have been used popularly because they are compatible with system non-linearity and non-Gaussian posterior distribution. But the main problem in particle filtering is sample degeneracy. To solve this problem, a new variant of particle filter has been proposed. The resampling algorithm used in this proposed particle filter is derived by combining systematic resampling, which is commonly used in SIR-PF (Sampling Importance Resampling Particle Filter) and a modified bat algorithm; this resampling algorithm reduces sample degeneracy as well as sample impoverishments. The measurement model is modified to handle clutter in presence of varying background. A new motion dynamics model is proposed which further reduces the chance of sample degeneracy among the particles by adaptively shifting mean of the process noise. To deal with illumination fluctuation and object deformation in presence of complete occlusion, a template update algorithm has also been proposed. This template update algorithm can update template even when the difference in the spread of the color-histogram is especially large over time. The proposed tracker has been tested against many challenging conditions and found to be robust against clutter, illumination change, scale change, fast object movement, motion blur, and complete occlusion; it has been found that the proposed algorithm outperforms the SIR-PF (Sampling Importance Resampling Particle Filter), bat algorithm and some other state-of-the-art tracking algorithms.

Object tracking is the estimation of object trajectory using an array of sensors. The sensors can be RADAR, SONAR or video camera etc. Object tracking has diverse application in the domains like defense, aerospace, robotics, cell biology, surveillance and ubiquitous computing etc. Visual object tracking is a special case where a camera or a network of cameras is used as a sensor. Visual object tracking is widely used in various fields like human computer interaction, automation, defense and industry etc. The major application oriented research areas of visual object tracking are mobile robotics, robotic arm control, vehicle control and navigation, video surveillance, autonomous landing, low altitude positioning, dropping of payloads and obstacle avoidance, and cell biology, etc.

In Chapter 1, a general introduction to object tracking is given. The idea is further visualized by some motivating examples. Then a detailed overview of visual object tracking is given along with their challenges. Finally the statement of our project work is explained.

In Chapter 2, a detail review of all the key algorithms in 2D object tracking is made. The algorithms described in this chapter are: region based tracking, covariance based tracking, template based tracking, and particle filter based tracking. In region based tracking, smoothened probabilistic histogram of target is formed; then it is matched with the reference one. Next the similarity metric between them is maximized using MeanShift algorithm. In covariance based tracking, the features of the target are encapsulated in a covariance matrix for tracking; encapsulation of feature in covariance matrix makes the tracker more robust. In the described template based tracking, a novel Kalman filtering is used to update the template which makes the system immune to severe occlusion. Finally, particle filter is introduced. In particle filter based tracking, both object and likelihood modeling is introduced along with SIR-PF (Sampling Importance Resampling Particle Filter) algorithm.

In Chapter 3, a detail review of data association algorithms are explained. In general, data association algorithms belong to a different class of algorithm which are widely used for multiple object tracking. Here a brief review of data association algorithms like NNF (Nearest Neighbour Filter), PDA (Probabilistic Data Association) and JPDA (Joint Probabilistic Data Association) are

introduced.

In Chapter 4, multiple camera tracking methods are reviewed. Multiple camera multiple object tracking includes both overlapped and non-overlapped camera networks. 3D object tracking is also considered as a sub-category of multi camera object tracking. Stereo camera network is widely used in depth estimation using geometry. Single camera can also locate objects in 3D using the intrinsic and extrinsic parameters of the camera, which are computed by calibration. Hence, under this sub-section, 3D object tracking is also reviewed.

In Chapter 5, the proposed algorithm is explained in detail. The proposed algorithm combines a modified version of SIR-PF (Sampling Importance Resampling Particle Filter) and a modified version of bat algorithm for re-sampling. This combination improves particle filter in many aspects which are explained in this section. Further this proposed tracker is enhanced using several other extra modules to handle clutter, occlusion and illumination fluctuation problem, etc.

Chapter 6 includes the experimental results. The proposed tracker is compared against SIR-PF (Sampling Importance Resampling Particle Filter), bat algorithm and other state-of-the-art algorithms. A large number of videos are tested. The videos include different challenging conditions like background clutter, abrupt illumination change, scale change of object, fast movement of the object, motion blur and complete occlusion. The videos are tested several times and over the complete sequence. The quantitative results are also listed in this chapter for every video sequence. From the test results, the performance of the proposed algorithm is found to be satisfactory.

The document is concluded with the complete list of the referred documents. The literature review sections give complete idea on the recent trends in visual object tracking. Then the proposed algorithm is explained and evaluated under several critical test situations. Thus this document gives the detail literature review of visual object tracking and the proposed approach, concluding with the referred literature.

Contents

Declaration . . . ii
Approval Sheet . . . iii
Acknowledgements . . . iv
Abstract . . . vi

1 Introduction to Visual Object Tracking 1
1.1 Introduction . . . 1
1.2 Object Tracking Examples . . . 1
1.2.1 Aerospace Monitoring . . . 1
1.2.2 Video Surveillance . . . 2
1.2.3 Cell Biology . . . 2
1.3 Visual Object Tracking . . . 2
1.4 Objective of the Project . . . 4
1.5 Summary . . . 4

2 Review of Single Camera Object Tracking Algorithms 5
2.1 Introduction . . . 5
2.2 Region Based Tracking . . . 5
2.2.1 Introduction . . . 5
2.2.2 Target Modeling . . . 6
2.2.3 Target Candidate Modeling . . . 7
2.2.4 Bhattacharyya Coefficient Maximization . . . 7
2.2.5 Algorithm . . . 9
2.2.6 Critical Analysis . . . 9
2.3 Covariance Based Tracking . . . 10
2.3.1 Introduction . . . 10
2.3.2 Covariance Matrix Formulation . . . 10
2.3.3 Computing the Best Match . . . 11
2.3.4 Critical Analysis . . . 11
2.4 Template Based Tracking . . . 11
2.4.1 Introduction . . . 11
2.4.2 Algorithm . . . 12
2.4.3 Critical Analysis . . . 13

2.5 Particle Filter Based Tracking . . . 13
2.5.1 Introduction . . . 13
2.5.2 Particle Filters . . . 14
2.5.3 System Model . . . 15
2.5.4 Color Model . . . 16
2.5.5 Algorithm . . . 17
2.5.6 Critical Analysis . . . 17
2.6 Summary . . . 17

3 Review of Multiple Visual Object Tracking Algorithms 19
3.1 Introduction . . . 19
3.2 Comparison of Data Association Algorithms . . . 19
3.3 Nearest Neighbor Data Association . . . 20
3.4 Probabilistic Data Association . . . 20
3.5 Particle Filter Data Association . . . 23
3.6 Summary . . . 23

4 Review of Multi-camera and 3D Object Tracking 24
4.1 Introduction . . . 24
4.2 Camera Calibration . . . 24
4.3 Multi-camera Object Tracking . . . 25
4.3.1 Disjoint Field of View Tracking . . . 25
4.3.2 Overlapped Field of View Tracking . . . 25
4.3.3 Tracking Human Motion . . . 26
4.3.4 Critical Analysis . . . 26
4.4 Depth Estimation and 3D Tracking . . . 26
4.4.1 Introduction . . . 26
4.4.2 Literature Review : Different Approaches to 3D Tracking . . . 27
4.5 Summary . . . 27

5 Proposed Algorithm 28
5.1 Introduction . . . 28
5.2 Related Works . . . 29
5.3 Proposed Particle Filter . . . 30
5.3.1 Sampling Importance Resampling Particle Filter . . . 30
5.3.2 Proposed Algorithm . . . 32
5.4 Proposed Measurement Model . . . 33
5.4.1 Other Related Works . . . 34
5.4.2 Proposed Measurement Model . . . 34
5.5 Proposed Motion Dynamics Model . . . 35
5.6 Proposed Resampling Algorithm . . . 38
5.6.1 Bat Algorithm . . . 38
5.6.2 Proposed Resampling Algorithm . . . 40
5.6.3 Computational Cost Analysis . . . 42

5.7 Proposed Template Update Mechanism 43
5.7.1 Related Works 43
5.7.2 Proposed Template Update Algorithm 43
5.8 Summary 45

6 Test Results 47
6.1 Introduction 47
6.2 Comparison with SIR-PF and Bat Algorithm 48
6.2.1 Case Study 1 : Tracking Object Under Illumination Variation, Occlusion and Clutter 48
6.2.2 Case Study 2 : Tracking Object Under Clutter, Fast Motion and Scale Change 49
6.2.3 Case Study 3 : Tracking Object Under Motion Blur, Fast Motion and Scale Change 54
6.2.4 Case Study 4 : Tracking Object Under Occlusion, Fast Motion and Clutter . 57
6.2.5 Case Study 5 : Tracking Object Under Occlusion and Clutter 60
6.2.6 Case Study 6 : Tracking Object Under Occlusion, Fast Motion and Scale Change 63
6.2.7 Case Study 7 : Tracking Object Under Illumination Variation and Clutter . . 63
6.2.8 Case Study 8 : Tracking Object Under Scale Change, Partial Occlusion and Clutter 69
6.2.9 Case Study 9 : Tracking Object Under Complete Occlusion and Clutter . . . 72
6.2.10 Case Study 10 : Tracking Object Under Illumination Variation and Motion Blur 75
6.2.11 Analysis 75
6.3 Computational Cost Analysis 78
6.4 Comparison in Subtracted Color Space 80
6.4.1 Case Study 1 : Tracking Object Under Illumination Variation, Occlusion and Clutter 81
6.4.2 Case Study 2 : Tracking Object Under Clutter, Fast Motion and Scale Change 81
6.4.3 Case Study 3 : Tracking Object Under Motion Blur, Fast Motion and Scale Change 82
6.4.4 Case Study 4 : Tracking Object Under Occlusion, Fast Motion and Clutter . 82
6.4.5 Case Study 5 : Tracking Object Under Occlusion and Clutter 82
6.4.6 Case Study 6 : Tracking Object Under Occlusion, Fast Motion and Scale Change 83
6.4.7 Case Study 7 : Tracking Object Under Illumination Variation and Clutter . . 83
6.4.8 Case Study 8 : Tracking Object Under Scale Change, Partial Occlusion and Clutter 83
6.4.9 Case Study 9 : Tracking Object Under Complete Occlusion and Clutter . . . 84
6.4.10 Analysis 84
6.5 Comparison with Other State-of-the-art Trackers 84
6.5.1 Case Study 1 : Tracking Object Under Illumination Variation, Occlusion and Clutter 85
6.5.2 Case Study 2 : Tracking Object Under Clutter, Fast Motion and Scale Change 85
6.5.3 Case Study 3 : Tracking Object Under Motion Blur, Fast Motion and Scale Change 90
6.5.4 Case Study 4 : Tracking Object Under Occlusion, Fast Motion and Clutter . 90

6.5.5 Case Study 5 : Tracking Object Under Occlusion and Clutter 95
6.5.6 Case Study 6 : Tracking Object Under Occlusion, Fast Motion and Scale Change 95
6.5.7 Case Study 7 : Tracking Object Under Illumination Variation and Clutter . . 95
6.5.8 Case Study 11 : Tracking Object Under Deformation, Clutter, Scale Variation and Fast Motion 99
6.5.9 Case Study 12 : Tracking Object Under Clutter, In-plane Rotation and Out-plane Rotation 99
6.5.10 Case Study 13 : Tracking Object Under Clutter, Object Deformation and Occlusion 105
6.5.11 Case Study 14 : Tracking Object Under Illumination Variation, Motion Blur, Fast Motion, Background Clutter and Low Resolution 105
6.5.12 Analysis 109
6.6 Summary 109

7 Conclusion **111**

References **112**

Chapter 1

Introduction to Visual Object Tracking

1.1 Introduction

Object tracking refers to the estimation of object location and their trajectory using sensors like RADAR, SONAR, camera and microphone etc. The key objective of the object tracking is to estimate the number of objects as well as their states like position, velocity and acceleration. As for example, tracking of an aircraft using RADAR; in this context, the problem is to estimate the number of aircrafts present, their type or class, their location and speed, using RADAR.

There are lots of uncertainties in tracking objects that make the task complicated. As for example, object motion can be mixed by random disturbances, their motion can remain undetected by the sensors and the quantity of the objects traversing the field-of-view of the sensor or the sensor network can be changed randomly. The sensor measurements are get coupled with random noise, making the estimation incorrect. Sometimes the objects may be overlapped and cannot be distinguished so easily. Also sensors may give measurements while there is no object either.

In this chapter, a complete overview of different object tracking research areas is given in detail, including the detail problem statement for this project.

1.2 Object Tracking Examples

Object tracking refers to the estimation of the kinematic states (position, velocity and acceleration), using measurements from sensors or a network of sensors in presence of noise and clutter. A typical object tracking system consists of a network (array) of sensors and a processor. Some applications of object tracking are reviewed as follows:

1.2.1 Aerospace Monitoring

This might be a very common example of object tracking. RADAR is used from domestic aircraft tracking to military surveillance applications. RADAR basically gives the measurement; but due to the uncertainties in the measurements available from RADAR, the tracking become quite challenging.

Birds and clouds etc. are sources of noise present in this measurement. Another challenge is due to the maneuvering feature of aircrafts. Also, when a number of fighter aircrafts form an array, the problem refers to multi object tracking.

1.2.2 Video Surveillance

This is another popular example where the sensor the network of cameras which are placed away from each other. This sort of facilities are nowadays available with all banks, airports and malls etc. With the help of high speed wired or wireless networks, the network of cameras are growing larger and larger. This leads to the problem of automatic detection and tracking of a person or a vehicle in frames, and then, from this data, the object tracking algorithm has to infer their behavior (unusual or criminal behavior); this is another trending research area. US military used this sort of application in the battle-field. Another modification to this approach is crowd-detection.

1.2.3 Cell Biology

In cell biology, the birth as well as death rates and the motion of the biological-cells are needed to be observed continuously. As for example, in the analysis of anti-inflammatory diseases, the speed and the acceleration of the lymphocyte cells are continuously monitored by taking their periodic images may be over a number of days. The speed, division and death-rate can be estimated by tracking the cells; these parameters can be calculated from the track initialization as well as termination probabilities; this makes the algorithm enough intelligent to discover any unobserved behaviour. Data association algorithms are used to identify cells and to track.

1.3 Visual Object Tracking

Visual object tracking uses camera network as sensor. Visual object tracking is a popular research area in computer vision. The existence of fast processors, high quality, but inexpensive video cameras and their high demanding need for automatic video analysis has increased the need for research in the field of object tracking algorithms. In visual analysis of objects, the key steps are: detection, tracking and recognition.

Visual object tracking widely used in various fields like Human Computer Interaction, automation, defense and industry etc. The major application oriented research areas of visual object tracking are Mobile Robotics, Robotic Arm Control, Vehicle Control and navigation, Surveillance, Autonomous Landing, Low Altitude Positioning, Dropping of Payloads and Obstacle Avoidance etc.

Both mobile robotics and robotic arm refer to Control and Automation. In both these applications, estimation of object location and depth are very important. Once the 3D coordinate is calculated, the robotic arm can pickup the object. But, while picking up the object, the robotic arm need to track the object continuously whether the object is moving or standing still. This can be achieved by using a network of cameras mounted on the arm. Similarly vehicle control also needs continous tracking and depth estimation using a set of cameras.

Apart from single camera object tracking, lots of research is going on in the field of multi camera object tracking also. The examples of recent trends in multi camera tracking are 3D track formation, depth estimation and occlusion resistant tracking. Multi camera multi object tracking is widely used in surveillance, 3D track formation, space and shuttle fleet inspection (NASA), etc. The key research challenges are to attain higher accuracy, to track/form 3D shape of fast moving objects with low resolution cameras and to estimate depth (widely used in mobile robots, vehicle security systems etc.) etc. Literature [4] describes how multi camera network can be used for easy living. Occlusion resistant tracking basically uses different homo-graphic approaches while, in multi-camera surveillance, multiple objects are stitched in camera-views, using transition probability and histogram matching.

Different algorithms have been proposed till now regarding object tracking. Those algorithms can be classified in two broad categories: one is *target representation and localization based*, which is basically an optimization algorithm, and the other one is *data-association and filtering based.* Both these approaches have their own advantages and disadvantages depending on their application. [5, 6, 7] describe all the recent trends in single as well as multi camera object tracking. State-space formulation is frequently used for modeling discrete-time systems in object tracking.

The main difficulties faced in visual object tracking are:

1. Loss of information due to 3D to 2D projection
2. Noisy and blurred reproduction of images
3. Complex and overlapped motion of multiple objects
4. Non-rigidity of objects
5. Partial or complete occlusion
6. Abrupt illumination change
7. Fast and real-time processing

Lots of algorithms have been developed, but most of them are application oriented. Their differences lie in the fact that:

1. How objects are modeled
2. Which features are used for modeling
3. What is the application they are intended for

The popular appearance representations regarding visual object tracking are:

1. Probabilistic modeling: It can be Gaussian, Gaissian Mixture model or Parzen window based. As for example [8] uses Parzen window and Histogram for object modeling.

2. Templates modeling: They are based on geometry of the objects.

3. Multiview appearance models: Here different views of the same object is encoded in differrent subspace views. Subspace approaches are like PCA (Principal Component Analysis) based Eigen-face approach.

The common visual features used in object tracking are:

1. Color
2. Edge
3. Texture

As mentioned earlier, visual analysis consists of object detection and tracking, the few object detection methods are as follows:

1. Point Detection: They are used to find interest points in the images using some descriptor. The examples of point detectors are SIFT (Scale Invariant Feature Transform) and Harris Corner Detector etc.

2. Background Subtraction: This refers to modeling background using Gaussian or Mixture of Gaussian for outdoor scenarios or HMM (Hidden Markov Model) to subtract foreground and background to detect new incoming objects

3. Segmentation: It is the grouping of similar pixels. Several algorithms like MeanShift etc. are available for segmentation

The key tracking categories are:

1. Point tracking: Every point/single-point/every feature point of the object is used for tracking. as for example, Kalman or particle filtering. For multi object tracking, these filters are extended by using some additional algorithms like data association techniques like PDA (Probabilistic Data Association), JPDA (Joint Probabilistic Data Association), or NNF (Nearest Neighbor Filter) etc.

2. Kernel tracking: It is apparently done by estimating the motion of an object, which is modeled by any method, from one frame to another [8].

1.4 Objective of the Project

In this work, a new algorithm has been proposed. The proposed approach includes a modified SIR-PF (Sampling Importance Resampling Particle Filter) and a modified bat algorithm. This improves SIR-PF (Sampling Importance Resampling Particle Filter) in many aspects. Particle filters suffer from two major problems: sample degeneracy and sample impoverishment. To solve these sample degeneracy and sample impoverishment, the resampling algorithm for this proposed particle filter has been derived by modifying the bat algorithm. The measurement model is modified to handle clutter in presence of varying background, and a new motion dynamics model is proposed which further reduces the chance of sample degeneracy among the particles by adaptively shifting mean of the process noise. To deal with illumination fluctuation and object deformation in presence of complete occlusion, a template update algorithm has also been proposed. This template update algorithm can update template even when the difference in the spread of the color-histogram is especially large over time. Various datasets, which are collections of benchmark videos on visual object tracking, are taken from [1, 2, 3]. The videos include different challenging conditions like heavy background clutter, abrupt illumination change, scale change of object, fast movement of the object, motion blur, and complete occlusion. The proposed algorithm tested several times and over the complete sequence. The quantitative results are also listed in this chapter for every video sequence. The case studies show that the proposed algorithm is compared against SIR-PF (Sampling Importance Resampling Particle Filter), bat algorithm and other state-of-the-art algorithms, and the proposed algorithm outperforms many of them. Thus the proposed tracker is found to be robust against clutter, illumination fluctuation, scale change, fast object movement, motion blur and complete occlusion.

1.5 Summary

This chapter describes the various aspects and key application areas where visual object tracking can be the crucial part of the research. The challenges of visual object tracking are also well explained. Then the objective of the project is described. In the next chapters, the major tracking algorithms are described.

Chapter 2

Review of Single Camera Object Tracking Algorithms

2.1 Introduction

In this chapter, several major single object visual tracking algorithms have been introduced. Here both the advantages and disadvantages of each algorithm are figured out. The algorithms described fall under both deterministic and probabilistic categories; each category has their own advantages and disadvantages. As for example, deterministic tracking is faster while probabilistic tracking is more robust in presence of partial occlusion and clutter.

All the major algorithms like region based tracking, feature based tracking, adaptive template based tracking and particle filter based tracking etc. are reviewed and analyzed. In deterministic search, the current frame is searched using a template. There are existing algorithms for updating the template, which makes the tracking robust. These sort of template based tracking algorithms are simply optimization algorithms. The ultimate objective is to minimize the distance between the reference model and the target model. In most of these deterministic algorithms, tracking works fine until the background color matches with the object color or the object is occluded for a pretty long time. It has been tried to figure out intuitively why this happens. On the other hand, probabilistic tracking can almost eliminate these problems; as for example, particle filter based approach etc.

The selection of the algorithm depends on the application. As for example, face tracking in crowd much more dependent on target representation rather than on target dynamics. In contrast to this, in aerial video surveillance, target as well as camera motion are more important. Finally it can be concluded that, whatever the application is, the complexity of the tracker must be as less as possible to make it real-time.

2.2 Region Based Tracking

2.2.1 Introduction

In this section, the tracking algorithm explained is based on color histogram and MeanShift optimization. This algorithm can track non-rigid objects. The feature histogram of the targets are masked with kernel, which makes the function smooth and suitable for gradient-based optimization algorithms, which searched for the local maxima. A metric has been derived from the Bhattacharyya coefficient to measure similarity, and then MeanShift is used to optimize the similarity metric. This algorithm successfully deals with camera movement,clutters, partial occlusions and target dimension variations. Further, this algorithm can be integrated with filters and data association algorithms.

In this section, it has been tried to infer reasons of some drawbacks of this algorithm and described in 2.2.6. In this approach, first model of the object is made using RGB histogram, then it is localized by using extended MeanShift algorithm (a machine learning based non-parametric algorithm) as discussed earlier.

[8] describes a method where histogram matching is done between target model and target representation, using Bhattacharya coefficient. Then, the peak of Bhattacharyya matching coefficient is reached by using MeanShift algorithm. Hence, the objective is to maximize a likelihood-type function. Although [8] claims that this new approach to object tracking is immune to camera motion, partial occlusion, clutter and target scale variants, we could figure out few disadvantages in this algorithm. It is the inherent feature of MeanShift algorithm to stop at the local maxima (optima). Hence, if background is similar to the object, the tracker may get stuck at an incorrect location rather that at the centroid of the real object. In this paper [8], the authors provide a solution to this problem by providing background modeling.

2.2.2 Target Modeling

[8] uses m-bin ($r = 1...m$) joint histogram for target modeling or reference modeling. Target model is represented by ellipsoidal region by individually rescaling the row and column by h_x and h_y so that the pixel locations are normalized to make the histogram rotation invariant. Let $\mathbf{n}_l^\star$ be the normalized pixel location of the l-th pixel and Epanechnikov is the kernel is used to make the distribution smooth.

Let, m be the total number of bins in the joint color histogram (in case of RGB, it is three dimensional), r be the bin of the joint histogram, $\mathbf{n}_l^\star$ be the l-th normalized pixel location, δ be the Kronecker delta function, n be the total number of pixels present in the template, $K(.)$ be the Epanechnikov kernel and $\mathbf{y}$ be location of the template.

The target or reference model is:

$$\hat{\mathbf{q}} = \{\hat{q}_r\}_{r=1...m} \qquad \sum_{r=1}^{m} \hat{q}_r = 1 \tag{2.1}$$

The target candidate at location $\mathbf{y}$ is:

$$\hat{\mathbf{p}}(\mathbf{y}) = \{\hat{p}_r(\mathbf{y})\}_{r=1...m} \quad \sum_{r=1}^{m} \hat{p}_r = 1 \tag{2.2}$$

The joint-histogram is formed as follows:

$$\hat{q}_r = C \sum_{l=1}^{n} K\left(\|\mathbf{n}_l^\star\|^2\right) \delta\left[b(\mathbf{n}_l^\star) - r\right] \tag{2.3}$$

where the normalization constant C is:

$$C = \frac{1}{\sum_{l=1}^{n} K\left(\|\mathbf{n}_l^\star\|^2\right)} \tag{2.4}$$

This probabilistic method using which the model is built is called Parzen window based approach which is a popular method for non-parametric density estimation. There are several winodws used to measure the density like Gaussian window, Epanechnikov kernel etc.

$$density_{PARZEN} = \frac{\frac{Tumberofsamplesfallingwithinwindow}{Totalnumberofsamples}}{Volume} \tag{2.5}$$

Equation (2.3) depicts that the histogram is immune to rotation. Unlike in normal histogram, where only number of bins are counted, each bin is weighted by Epanechnikov kernel according their distance from the origin (midpoint) of the template in this literature [8]. The pixel near the center have higher weight and border pixels have less weight. Epanechnikov kernel is used because it is faster in convergence rather than Gaussian one although Gaussian kernels are smoother than other ones.

The similarity function is made smooth by masking with the help of an isotropic kernel, like Epanechnikov kernel, in its spatial domain. Although Gaussian kernels can provide more smoothing effect, Epanechnikov kernels are faster in convergence rate.

2.2.3 Target Candidate Modeling

This modeling is the same as that of the target modeling. The difference is that the normalized pixel locations in the region of target candidate are centered at $\mathbf{y}$. Let, m be the total number of bins in the joint color histogram (in case of RGB, it is three dimensional), r be the bin of the joint histogram, $\mathbf{n}_l^\star$ be the normalized pixel location of the l-th pixel, h be the bandwidth, δ be the Kronecker delta function, n_h be the total number of pixels present in the template, $K(.)$ be the Epanechnikov kernel and $\mathbf{y}$ be location of the template.

The histogram model is as follows:

$$\hat{p}_r(\mathbf{y}) = C_h \sum_{l=1}^{n_h} K\left(\left\|\frac{\mathbf{y} - \mathbf{n}_l^\star}{h}\right\|^2\right) \delta\left[b(\mathbf{n}_l^\star) - r\right] \tag{2.6}$$

where the normalization constant C_h is given as:

$$C_h = \frac{1}{\sum_{l=1}^{n_h} K(\|\frac{\mathbf{y}-\mathbf{n}_l^\star}{h}\|^2)} \tag{2.7}$$

2.2.4 Bhattacharyya Coefficient Maximization

Bhattacharyya distance measures the distance between two probabilistic distribution models, and this is the metric to measure the similarity between two probabilistic distributions. Hence, the histograms must be in probabilistic form. Once the probabilistic histogram models are formed for target model and target candidate, their similarity can be compared as follows using Bhattacharyya coefficient:

$$\rho = \sum_{r=1}^{m} \sqrt{\hat{p}_r(\mathbf{y})\hat{q}_r} \tag{2.8}$$

To localize the target in the current frame, (2.8) must be maximized. The localization starts from the location of the target in the previous frame knows as the model. Although RGB histogram is used here, texture and edges features can also be used under this framework. Now the general extended MeanShift optimization algorithm is used to maximize the Bhattacharyya coefficient (2.8). Since search for new target-location starts at $\hat{\mathbf{y}}_0$, Taylor's expansion at $\hat{\mathbf{y}}_0$ (which is the initial condition of each iteration at k-th frame) is done as follows:

$$\rho \approx \frac{1}{2}\sum_{r=1}^{m} \sqrt{\hat{p}_r(\hat{\mathbf{y}}_0)\hat{q}_r} + \frac{1}{2}\sum_{r=1}^{m} \hat{p}_r(\mathbf{y})\sqrt{\frac{\hat{q}_r}{\hat{p}_r(\hat{\mathbf{y}}_0)}} \tag{2.9}$$

Using (2.6) and (2.9), the following can be derived:

$$\rho \approx \frac{1}{2}\sum_{r=1}^{m} \sqrt{\hat{p}_r(\hat{\mathbf{y}}_0)\hat{q}_r} + \frac{C_h}{2}\sum_{l=1}^{n_h} w_l K\left(\left\|\frac{\mathbf{y}-\mathbf{n}_l^\star}{h}\right\|^2\right) \tag{2.10}$$

where weight at each pixel $l = 1...n_h$ is:

$$w_l = \sum_{r=1}^{m} \sqrt{\frac{\hat{q}_r}{\hat{p}_r(\hat{\mathbf{y}}_0)}}\ \delta\ [b(\mathbf{n}_l^\star) - r] \tag{2.11}$$

To optimise, MeanShift is used, which searches for the local maxima:

$$\hat{\mathbf{y}}_1 = \frac{\sum_{l=1}^{n_h} \mathbf{n}_l^\star w_l g\left(\left\|\frac{\hat{\mathbf{y}}_0-\mathbf{n}_l^\star}{h}\right\|^2\right)}{\sum_{l=1}^{n_h} w_l g\left(\left\|\frac{\hat{\mathbf{y}}_0-\mathbf{n}_l^\star}{h}\right\|^2\right)} \tag{2.12}$$

where derivative of Epanechnikov kernel is ($n^\star$ is the pixel distance):

$g(n) = -\frac{dK(n^\star)}{dx}$

Epanechnikov kernel is defined as follows:

$K(n) = \frac{1}{2}c_d^{-1}(d+2)(1-n^\star)$ if $n \leq 1$ else $K(n) = 0$

where d is the dimension and c_d is the volume of the dimension. As discussed earlier, [4] uses this kernel as it is pretty fast as compared to Gaussian kernel.

To eliminate background clutter, [8] also describes background weighted color histogram. Background template is defined around the foreground or object template or target area. The area of the background template is application dependent and has been taken as three times than that of the target area in [8]. The pixels outside the target area but inside the background area are considered as background pixels. Let $\{\hat{o}_r\}_{r=1...m}$ be the discrete background histogram of the background space and $\hat{o}^*$ be its smallest non-zero value. Hence, the weight for each bin r is defined as:

$$\left\{v_r = \min\left(\frac{\hat{o}^*}{\hat{o}_r}, 1\right)\right\}_{r=1...m} \tag{2.13}$$

The weight v_r is multiplied with the target histogram as follows:

$$\hat{q}_r = C v_r \sum_{l=1}^{n} K(\|\mathbf{n}_l\|^2)\delta\left[b(\mathbf{x}_l) - r\right] \tag{2.14}$$

where the normalization constant C is:

$$C = \frac{1}{\sum_{l=1}^{n} K(\|\mathbf{n}_l\|^2) \sum_{r=1}^{m} v_r \delta\left[b(\mathbf{n}_l) - r\right]} \tag{2.15}$$

Similarly, the weight v_r is also multiplied with the target candidate histogram as follows:

$$\hat{p}_r(\mathbf{y}) = C_h v_r \sum_{l=1}^{n_h} K\left(\left\|\frac{\mathbf{y} - \mathbf{n}_l}{h}\right\|^2\right)\delta\left[b(\mathbf{n}_i) - r\right] \tag{2.16}$$

where the normalization constant C_h is:

$$C_h = \frac{1}{\sum_{l=1}^{n_h} K(\|\frac{\mathbf{y}-\mathbf{n}_l}{h}\|^2) \sum_{r=1}^{m} v_r \delta\left[b(\mathbf{n}_l) - r\right]} \tag{2.17}$$

2.2.5 Algorithm

The steps of the algorithm described in [8] are as follows:

1. Initialize the target model $\hat{\mathbf{q}}$ and its initial location $\hat{\mathbf{y}}_0$ in initial frame using (2.3).

2. Initialize initial location in k-th frame as $\hat{\mathbf{y}}_0$ inhereted from $(k-1)$-th frame and Bhattacharyya coeficient at k-th frame using (2.8).

3. Derive weight w_l for all $l = 1...n_h$ using (2.11) at $\hat{\mathbf{y}}_0$ in k-th frame.

4. Run (2.12) iteratively until it converges. If error between $\hat{\mathbf{y}}_0$ and $\hat{\mathbf{y}}_1$ is less than a threshold ϵ then stop, else assign $\hat{\mathbf{y}}_0 = \hat{\mathbf{y}}_1$, and keep the loop running until it converges. Or it can be made to run this loop for a maximum number of iterations. In average, after 20 iterations, it converges.

5. Annotate k-th frame at new location: $\hat{\mathbf{y}}_1$, assign $\hat{\mathbf{y}}_0 = \hat{\mathbf{y}}_1$ and pass this $\hat{\mathbf{y}}_0$ to the $(k+1)$-th stage.

5. Go to step 2 and statrt tracking at $(k+1)$-th frame.

2.2.6 Critical Analysis

Since MeanShift searches for the local maxima, it may get stuck when the background and object have same color distribution; the same has been observed from simulation. To get rid of it to some extent, [8] suggests background modeling. The convergence accuracy and time depends on h (bandwidth) parameter. If h is too high, accuracy will be less and, if h is too small, time of convergence will be pretty high. The illumination and occlusion immunity of the algorithm is highly dependent upon the discriminating power of the histogram. To solve occlusion problem, [8] proposes to merge this algorithm with other data-association and filtering algorithms. Different tracking cues can be converted into a histogram, then can be tracked using this algorithm. Another problem in MeanShift based tracking is that it is highly illumination sensitive. If Bhattacharyya matching coefficient has several peaks or optima due the illumination fluctuation, it may get stuck at a wrong location. [9] proposes an approach which uses DC coefficient of the DCT (Discrete Cosine Transform) to measure illumination change. Then, a weighted histogram is proposed which nullifies this change in DC energy or illumination. This makes the algorithm robust against illumination. MeanShift is popular only because its high speed; the accuracy is increased by adding some extra weight to the histogram. The inherent drawbacks of the original MeanShift based tracking algorithm is solved by [10]; [10] resolves these problems by mixing edge, color and texture information all together. In this approach, the tracker is splitted into several fragments to utilize the spatial information. Similarly, [11] modifies MeanShift for scale variant, illumination variant and occluded object tracking. [12] uses MeanShift for perpetual user interface.

2.3 Covariance Based Tracking

2.3.1 Introduction

It has been showed that popular MeanShift algorithms's efficiency and accuracy depends on the histogram-formation. So researchers have tried other descriptors like HoG (Histogram of Gradient), DCT (Discrete Cosine Transform) and SIFT (Scale Invariant Feature Transform) with MeanShift, and each of them has their own advantages and disadvantages; as for example, SIFT (Scale Invariant Feature Transform) is invariant to rotation, but slow as it computes at different pyramid levels. [13] describes the method to formulate DCT (Discrete Cosine Transform) histogram. The color histogram's complexity increases exponentially with its dimension, and, if the number of bins are increased, sparsity will also increase.

Hence, in this section, another method has been introduced using which the object can be modeled and then tracked. [14] explains this novel algorithm to model as well as match objects using covariance matrix. This covariance matrix based approach is a unique method to mix multiple features within a small dimension. While averaging during the covariance matrix computation, a large amount of the mean-zero noise is removed. This covariance matrix based feature representation is also invariant to scale and illumination.

2.3.2 Covariance Matrix Formulation

[14] describes the covariance based object descriptor and the lie algebra based update mechanism. The covariance matrix helps fusion of different modalities and features (as for example, color intensity, gradient, etc.) inside a small dimension, and, it is not necessary to take any assumption on the measurement noise. This descriptor can be used as non-stationary camera model also. Literature [14] describes a descriptor, not any tracking algorithm.

The formulation is quite simple. Let a square template R_T is formed the dimension of which is: $M \times N$. For every pixel l, let x and y be their locations in x-axis and y-axis. $\mathbf{I}$ be the intensity value at x, y, and $\mathbf{I}_x$ be the $1st$ derivative of $\mathbf{I}$ along x-axis direction. Thus, for every pixel in the template R_T, the following vector is formed:

$$\mathbf{f}_l \quad = \quad [x \quad y \quad \mathbf{I}(x,y) \quad \mathbf{I}_x(x,y) \quad \ldots \quad] \tag{2.18}$$

or can be arranged as roation-invariant form:

$$\mathbf{f}_l^{\mathbf{r}} \quad = \quad \lceil \; \|(x', y')\| \quad \mathbf{I}(x,y) \quad \mathbf{I}_x(x,y) \quad \ldots \; \rceil \tag{2.19}$$

where

$$\|(x', y')\| = \sqrt{(x'^2 + y'^2)}, \quad (x', y') = (x - x_0, y - y_0) \tag{2.20}$$

and x_0, y_0 is the pixel location of the centre of the template R_T.

The covariance matrix $\mathbf{C}_{R_T}$ is defined as:

$$\mathbf{C}_{R_T} = \frac{1}{MN} \sum_{k=1}^{MN} (\mathbf{f}_k - \boldsymbol{\mu}_{R_T})(\mathbf{f}_k - \boldsymbol{\mu}_{R_T})^T \tag{2.21}$$

where $M \times N$ be the dimension of the rectangular patch R_T and $\boldsymbol{\mu}_{R_T}$ be the mean of the corresponding feature throughout the entire patch R_T.

2.3.3 Computing the Best Match

Let $\mathbf{C}_{R_1}$ and $\mathbf{C}_{R_2}$ be the two feature covariance matrices formed using (2.21), in patches or templates R_1 and R_2. Since covariance matrix does not lie on the Euclidean space, the dissimilarity or distance between these two matrices is measured using Forstner's method as follows:

$$\rho_a(\mathbf{C}_{R_1}, \mathbf{C}_{R_2}) = \sqrt{\sum_{t=1}^{d} ln^2 \lambda_t(\mathbf{C}_{R_1}, \mathbf{C}_{R_2})} \tag{2.22}$$

where d be the dimension and $\lambda_t(\mathbf{C}_{R_1}, \mathbf{C}_{R_2})$ be the generalised Eigen values ($t = 1...d$).

2.3.4 Critical Analysis

Since Eucleadian distance cannot be used to match the covariance matrices, this algorithm [14] cannot be made compatible for Bhattacharyya metric and then extended to MeanShift, according to the way described in [8]. [15] uses the covariance matrix concept in some other way. [15] uses the same covariance descriptor like [14], but uses a different metric based on Riemannian manifolds. They propose a probabilistic novel tracking algorithm with a covariance tensor learning mechanism.

2.4 Template Based Tracking

2.4.1 Introduction

Novel template based tracking is described in [16, 17]. Basically, template based searches are done using correlation matrices. A template is formed and searched throughout the image space and computes the correlation coefficient between the target na the model. But this simple approach is not suitable for cases where occlusion occurs, object is non-rigid and illumination change is intense. Hence, [16] uses a novel template update mechanism by robust Kalman filters. So it is resistant to severe occlusions, abrupt illumination change, etc. This algorithm embeds both template update and tracking mechanism together. In this section, literature [16] is described. Unlike the common application of Kalman filter to predict and update position (state), in [16], it has been used for prediction and smoothing of the intensity values of the pixels inside the template. Then, this template is fed to an optimization algorithm to match the template-transformation-parameters the best. The method deals with severe occlusion and changing object orientation. The key features of this algorithm described in [16] are summarized below:

1. Severe occlusions and illumination changes are solved
2. Fast and abrupt change of object orientation is solved
3. Background clutter is eliminated
4. Can be mounted on a movable camera

2.4.2 Algorithm

The steps of the algorithm described in [16] is summarized below:

1. A template R_T is taken manually. Every pixel is considered as a feature point, and one Kalman filter is alloted for each pixel l to predict. Hence, $l = 1...M \times N$ where M, N is the dimension of the template R_T. Let $I_k(R)$ is the intensity value of a pixel at k-th stage in R-th color channel (in case of RGB) and so defined for other two channels. The feature vector $\mathbf{f}_k$ denotes state of a pixel at k-th frame is defined as follows:

$$\mathbf{f}_k = [I_k(R)\ I_k(G)\ I_k(B)]^T \tag{2.23}$$

2. State of feature vectors is predicted as:

$$p(\mathbf{f}_k|\mathbf{f}_{k-1}) \sim \mathcal{N}(\mathbf{f}_{k-1}, \mathbf{W}) \tag{2.24}$$

$\mathbf{W}$ be the error covariance matrix and it is same for each pixel; it has to be initialized.

3. Using $\mathbf{f}_{k-1}$, an *optimization task* is done to find the best match transform (as for example, affine transformation etc.) and to extract the transformation parameters $\hat{\mathbf{a}}_k$ at k-th stage by summing and optimizing over all the pixels in template R_T as follows:

$$\hat{\mathbf{a}}_k = \arg\min_{\mathbf{a}} \sum_{l \in R_T} \rho\left(\epsilon\left(\mathbf{I}_k(\varphi(l;\mathbf{a})), \hat{\mathbf{f}}_{k-1}(\mathbf{x})\right)\right) \tag{2.25}$$

where $\mathbf{I}_k(\varphi(l;\mathbf{a}))$ is the feature vector observed at the point $\varphi(l;\mathbf{a})$ in k-th frame.

$$\hat{\epsilon} = \epsilon(\mathbf{z}_k, \hat{\mathbf{f}}_k) \tag{2.26}$$

$$\epsilon(\mathbf{z}_k, \hat{\mathbf{f}}_k) = \sqrt{[\mathbf{z}_k - \hat{\mathbf{f}}_k]^T {\mathbf{R}_1}^{-1} [\mathbf{z}_k - \hat{\mathbf{f}}_k]} \tag{2.27}$$

where $\hat{\mathbf{f}}_k$ is the predicted feature vector at k-th stage. At the initial few stages $\mathbf{R}_1$ is a large positive definite matrix. After some frames, it gets updated as given in [16].

4. Once $\hat{\mathbf{a}}_k$ is got, the template can be shifted to a new location and the measurement $\mathbf{z}_k$ is obtained at this k-th frame

5. Predict and update feature vector for the next (k+1)-th frame as follows:

$$\hat{\mathbf{f}}_k = [\psi(\hat{\epsilon}){\mathbf{R}_1}^{-1} + (\mathbf{C}_{k-1} + \mathbf{W})^{-1}]^{-1}[\psi(\hat{\epsilon}){\mathbf{R}_1}^{-1}\mathbf{z}_k + (\mathbf{C}_{k-1} + \mathbf{W})^{-1}\hat{\mathbf{f}}_{k-1}] \tag{2.28}$$

where $\mathbf{C}$ be the covariance matrix of the approximated Gaussian output distribution:

$$p(\mathbf{f}_k|\mathbf{z}_{1:k-1}) = \int\limits_{\mathbf{f}_{k-1}} p(\mathbf{f}_k|\mathbf{f}_{k-1})p(\mathbf{f}_{k-1}|\mathbf{z}_{1:k-1}) = \mathcal{N}(\hat{\mathbf{f}}_{k-1}, (\mathbf{C}_{k-1} + \mathbf{W})^{-1} \tag{2.29}$$

and other parameters are defined as:

$$\rho(\epsilon) = \epsilon^2/2 \qquad \text{if}|\epsilon| < c \tag{2.30a}$$

$$\rho(\epsilon) = c(|\epsilon| - c/2) \qquad \text{otherwise} \tag{2.30b}$$

$$\psi(\hat{\epsilon}) = 1 \qquad \text{if}|\epsilon| \leq c \tag{2.31a}$$

$$\psi(\hat{\epsilon}) = \frac{c}{|\epsilon|} \qquad \text{if}|\epsilon| > c \tag{2.31b}$$

where c is a threshold obtained from chi-square distribution.

7. $\mathbf{f}_k$, $\mathbf{W}$ and $\mathbf{C}_k$ is passed on to the next (k+1)-th frame

2.4.3 Critical Analysis

In [16], when $\epsilon > c$, template pixel is regarded as outliers. Hence, occlusion is detected pixel-by-pixel. Occlusion is detected when percentage of outliers exceeds some predefined threshold like 30 percent. At that time, tracking must be stopped for some predefined unit of time. The template is reinitialized after that predefined time period.

2.5 Particle Filter Based Tracking

2.5.1 Introduction

Another completely different but popular approach to object tracking is Bayesian tracking. Use of particle filter in object tracking has brought a revolutionary change in the concept of tracking. Whereas Kalman filters follow only Gaussian distributions, particle filters propagate in a more general distributions, which is more realistic; this solves many ambiguous situations faced in visual object tracking. Another important advantage of particle filters is that it allows fusion of information to track objects better in much more complex situations. Hence, in this section, particle filter based tracking is introduced in datail.

The particle filter based tracking is described in [18] which uses the same method depicted in [8] to model the object in a probabilistic framework rather than in a deterministic framework. Color-modeled trackers are said to be robust with a decent computational cost. They are efficient in tracking even when objects to be tracked shows variability on spatial structures where most space-dependent trackers will fail. Hence, it can track when there is drastic change in the spatial appearance of the object. [18] apparently describes Monte Carlo tracking using the same color histogram modeling. [18] uses particle filter which eliminates clutter and solves complete occlusion much better. [18] formulates a likelihood function based on color histogram distances just like [8]. Argument is given that, since likelihood is non-linear as well as multi-modal in visual tracking, particle filter is preferred to Kalman filter. [19], [20] and [21] also explain particle filter in detail. This probabilistic approach is also better in the sense that it does not need much tuning parameters.

The key philosophy in Bayesian tracking is to update the pre-knowledge of the hidden state with the set of measurement including the latest one. Many real-world systems exist which follow the Markov property that the present state depends on the immediate past state only. In object tracking also, this happens.

2.5.2 Particle Filters

Here the SIR (Sampling Importance Re-sampling) particle filter is reviewed. Both Kalman and particle filter based tracking falls under the Bayesian tracking category. In order to make estimation, both accurate system model and a measurement model are needed.

In Bayesian approach, one creates posterior density function (pdf) of the states to be estimated based on received measurement and all other information. Now the objective is to maximize the posterior estimation. This maximization is done using some filters. These filters have two steps: Prediction and Update. The latest measurement is used to update the prior or predicted ones.

Let, k be the stage, $\mathbf{x}$ be the state, $\mathbf{z}$ be the measurement vector. Thus prior (predicted ones) is updated with a likelihood and a normalization factor as follows using Baye's rule:

$$p(\mathbf{x}_k|\mathbf{z}_{1:k}) = \frac{p(\mathbf{z}_k|\mathbf{x}_k)p(\mathbf{x}_k|\mathbf{z}_{1:k-1})}{p(\mathbf{z}_k|\mathbf{z}_{1:k-1})} \tag{2.32}$$

where
$\mathbf{x}_k$ = estimated state at k-th stage
$\mathbf{z}_k$ = measurement at k-th stage
Predictions occur as follows:

$$p(\mathbf{x}_k|\mathbf{z}_{1:k-1}) = \int p(\mathbf{x}_k|\mathbf{x}_{k-1})p(\mathbf{x}_{k-1}|\mathbf{z}_{1:k-1})\, d\mathbf{x}_{k-1} \tag{2.33}$$

$$p(\mathbf{z}_k|\mathbf{z}_{1:k-1}) = \int p(\mathbf{z}_k|\mathbf{x}_k)p(\mathbf{x}_k|\mathbf{z}_{1:k-1})\, d\mathbf{x}_k \tag{2.34}$$

Let, $\mathbf{v}_{k-1}$ be the process noise. Then the philosophy of the approach is that the state is predicted using some *dynamics* as follows:

$$\mathbf{x}_k = \mathbf{f}_k(\mathbf{x}_{k-1},\, \mathbf{v}_{k-1}) \tag{2.35}$$

Since state is hidden, this predicted state is not correct. Hence, it is then corrected using (2.32). The measurement model is as follows:

$$\mathbf{z}_k = \mathbf{H}_k\mathbf{x}_k + \mathbf{n}_k \tag{2.36}$$

where $\mathbf{H}_k$ is the measurement matrix and $\mathbf{n}_k$ is the measurement noise.

Kalman filter provides optimal state estimation, assuming system to be ideally linear and noise is Gaussian-zero-mean; hence, the posterior distribution is unimodal. On the other hand, extended Kalman filter and particle filter provide suboptimal solution. Particle filters work with a number of particles each having some weight. In SIR-PF (Sampling Importance Re-sampling Particle Filter) algorithm, each particles are transmitted through some model state space equations, then likelihood is computed, particles having higher weight are replicated while others are suppressed and this process goes on. As mentioned earlier, particle filters have many versions proposed to solve the following two problems mainly:

1. To reduce sample degeneracy problem : After some iterations, many particles have negligible weight. These particles are discarded while higher weightage particles are replicated. The degeneracy is calculated using the following equations:

$$\widehat{N_{eff}} = \frac{1}{\sum\limits_{i=1}^{N_s} (w_k^i)^2} \tag{2.37}$$

where i be the number of particle, w_k^i be the weight associated with i-th particle which is a measure of posterior at the point and N_s be the total number of particles generated. If $\widehat{N_{eff}}$ is more than a threshold, re-sampling is required. While, in another variant of particle filter called SIR-PF (Sampling Importance Resampling Particle Filter), re-sampling is done at every stage k.

2. To reduce sample impoverishment : While re-sampling, particles get concentrated around higher weight which reduces diversity among the particles. Even in case of very little noise, particles will merge to a single point after some iterations. One solution may be a good choice of importance density to minimize variance among the particle weights. Another solution is to use MCMC sampling.

A good choice of selecting particles is to increase their variance: $Var(\omega_k^i)\ for\ i = 1...N_s$. There are many variants of particle filtering; SIR-PF (Sampling Importance Resampling Particle Filter) filter is one of them. It is a Monte Carlo algorithm which can be applied to the recursive Bayesian filtering problems easily. The advantage of using SIR-PF (Sampling Importance Resampling Particle Filter) is its weak assumptions. To implement this algorithm, state dynamics or system model is needed to be known along with the measurement matrix. The original algorithm of particle filtering called Sequential Importance Sampling (SIS). The key assumption is that since it is difficult to draw particles from probability density p(.), sample are so generated from some other density called q(.) (*importance density*). Hence, the weight for i-th particle:

$$w_k^i \propto \frac{p(\mathbf{x}_{0:k}^i|\mathbf{z}_{1:k})}{q(\mathbf{x}_{0:k}^i|\mathbf{z}_{1:k})} \tag{2.38}$$

$$w_k^i \propto \frac{p(\mathbf{z}_k|\mathbf{x}_k^i)p(\mathbf{x}_k^i|\mathbf{x}_{k-1}^i)p(\mathbf{x}_{0:k-1}^i|\mathbf{z}_{1:k-1})}{q(\mathbf{x}_k^i|\mathbf{x}_{0:k-1}^i, \mathbf{z}_{1:k})q(\mathbf{x}_{0:k-1}^i|\mathbf{z}_{1:k-1})} \tag{2.39}$$

$$w_k^i \propto w_{k-1}^i \frac{p(\mathbf{z}_k|\mathbf{x}_k^i)p(\mathbf{x}_k^i|\mathbf{x}_{k-1}^i)}{q(\mathbf{x}_k^i|\mathbf{x}_{k-1}^i, \mathbf{z}_k)} \tag{2.40}$$

SIR-PF (Sampling Importance Resampling Particle Filter) is easy to implement as the weight is proportional to the likelihood, but the re-sampling has to be taken at each step.

2.5.3 System Model

In probabilistic tracking, system and measurement models are very crucial. The models are given below:

$$\mathbf{x}_k = \mathbf{F}_k\mathbf{x}_{k-1} + \mathbf{v}_{k-1} \tag{2.41}$$

$$\mathbf{z}_k = \mathbf{H}_k \mathbf{x}_k + \mathbf{n}_k \tag{2.42}$$

where

$\mathbf{F}_k =$

$$\begin{pmatrix} 1 & 0 & dt & 0 \\ 0 & 1 & 0 & dt \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{2.43}$$

$\mathbf{H}_k =$

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix} \tag{2.44}$$

where, dt=time difference between two frames. This model is called constant-velocity model. And, the states are $\mathbf{x} = [x \quad y \quad \dot{x} \quad \dot{y}]^T$ where x, y be the pixel locations along x-axis and y-axis and $\dot{x}, \dot{y}$ be their respective velocities. $\mathbf{v}_{k-1}$ be the process noise at $(k-1)$-th stage and $\mathbf{n}_k$ be the measurement noise at k-th stage.

2.5.4 Color Model

A measurement model based on joint m-bin histogram ($r = 1...m$) as reference model as well as target candidate has been proposed [8]. The center of the template is taken as the origin. *Epanechnikov* kernel used in this model makes border pixels have lesser weight. Let $\hat{q}_r$ and $\hat{p}_r(\mathbf{y})$ ($r = 1...m$) be the reference and target candidate color histograms, respectively as given in [8]. $\mathbf{y}$ be the location of the target histogram. Now, these two histograms are compared by Bhattacharyya coefficient [8]. Bhattacharyya coefficient approximates the chi-square statistics eliminating the singularity problem while dealing with empty bins [22]. The equation is given in (2.45).

$$\rho = \sum_{r=1}^{m} \sqrt{\hat{p}_r(\mathbf{y})\hat{q}_r} \tag{2.45}$$

The distance between the two distributions is given by [8]:

$$D(\mathbf{y}) = \sqrt{1-\rho} \tag{2.46}$$

Hence, likelihood of the i-th particle at $\mathbf{y}$ in k-th stage is given by:

$$p(\mathbf{z}_k^i|\mathbf{x}_k^i) \propto \exp(-\lambda D(\mathbf{y})) \tag{2.47}$$

where, λ is a tuning parameter [18].

For higher accuracy, [18] proposes that the tracked region of the object is divided into N number of different patches having distinct colors. This makes the tracking more robust wit the help of the spatial information. This method also increases robustness to occlusion. The likelihood becomes:

$$p(\mathbf{y}_t|\mathbf{x}_t) \propto \exp(-\lambda \sum_{t=1}^{N} D(\mathbf{y})_t) \tag{2.48}$$

Background modeling is also introduced in [18]. The background need to be static and available offline. Similarly, [8] derives another way to fuse M different cues (like motion, color and acoustic cues, etc.) together. Their derived equation is given below:

$$p(\mathbf{z}|\mathbf{x}) = \prod_{t=1}^{M} p(\mathbf{z}^t|\mathbf{x}) \tag{2.49}$$

where $\mathbf{z}^1, \mathbf{z}^2, ...\mathbf{z}^M$ are different available measurements

2.5.5 Algorithm

Let $\mathbf{x}$ be the state, i be the particle, w be the weight, $\mathbf{z}$ be the measurement and N_s be the number of particles. Then the steps in SIR-PF (Sampling Importance Resampling Particle Filter) algorithm are shown in Algorithm 2.

Algorithm 1 SIR-PF Algorithm

```
1: procedure SIR-PF($(\mathbf{x}^i_{k-1}, w^i_{k-1})^{N_s}_{i=1}, \mathbf{z}_k$)
2:     for $i = 1 : N_s$ do
3:         Draw: $\mathbf{x}^i_k \sim p(\mathbf{x}^i_k|\mathbf{x}^i_{k-1})$
4:         Compute: $w^i_k = p(\mathbf{z}_k|\mathbf{x}^i_k)$
5:     Normalize: $w^i_k$ for $i = 1...N_s$
6:     Systematic Resampling: Discard particles with low weight and replicate particles with higher
   weight as given in [19]
7:     return $(\mathbf{x}^i_k, w^i_k)^{N_s}_{i=1}$
```

2.5.6 Critical Analysis

Particle filters, also known as Bayesian trackers, are very efficient trackers. They can handle clutter and complete occlusion inherently. There are many variants of particle filters. SIR-PF (Sampling Importance Resampling Particle Filter) is one of them and is the simplest one. Although particle filters are more robust as compared to deterministic trackers, they are slower compared to deterministic trackers like MeanShift trackers. Hence, lots of work has been done on MeanShift algorithm to make it equivalent to particle filters. Yet particle filters are key research topics from computer vision researchers. Work [18] also describes multiple object tracking scheme. But it is simple up to two objects; for higher interacting objects, complexity increases. In case of deterministic trackers, number of trackers are run in parallel to track multiple objects. This scheme works fine as long as the situation is quite simple. When number of objects increases and they start interacting with each other (overlapping), this parallelly running multiple tracker scheme fails. Generally, in case of probabilistic trackers, others techniques are adopted for multiple object tracking.

2.6 Summary

In this chapter, a set of major algorithms ranging from template matching to Bayesian tracking has been discussed. Bayesian tracking includes different forms of particle filters and Kalman filters.

Optimal Kalman filters are suitable for ideally linear model and Gaussian posterior and noise distribution. On the other hand, extended Kalman filter and uncented Kalman filter, are the sub-optimal ones. In case of Bayesian tracking, modeling is a crucial part of algorithm development. This modeling can be either single model or multiple model. In multiple model approach, estimations are made individually, and then they are clubbed together to provide the final estimation; as for example, Interacting Multiple Model Kalman Filter. The algorithms described in this chapter form the basis of other algorithms which are designed for more complicated scenario. In the next chapter, multiple object tracking algorithms are reviewed.

Chapter 3

Review of Multiple Visual Object Tracking Algorithms

3.1 Introduction

Up to this chapter, deterministic and probabilistic visual object tracking algorithms have been introduced for single object mainly. Although the previously described algorithms can be extended for multiple object tracking, in this chapter, another different approach called data association technique has been introduced. Although the algorithms explained here are for visual object tracking, they are also compatible for non-visual object tracking as explained like RADAR based object tracking, etc. Data association algorithms are more robust in case the objects are close each other or overlapping on each other as compared to other algorithms. Most deterministic trackers fail in this situation. In data association method, multiple measurements are assigned to multiple objects. As number of objects is known, the data association step computes the origin of the measurements probabilistically. It is based on the hypothesis of the origin of measurement. In single object tracking, there are two hypothesis: one measurement is from object and one is from clutter. But in multiple object tracking, such hypothesis is more.

The key philosophy of data association algorithms is that, under probabilistic framework, a predicted state is updated with the help of available measurements; either the best measurement is picked up or all the measurements are used together with some weight assigned to each measurement. The most common data association algorithms are: Nearest Neighbor Filter (NNF), Probabilistic Data Association (PDA), Joint Probabilistic Data Association (JPDA), MHT (Multiple Hypothesis Tracking), PMHT (Probabilistic Multiple Hypothesis Tracking) and MCMC-DA (Markov Chain Monte-Carlo Data Association) etc. The selection of data association algorithm is application specific.

3.2 Comparison of Data Association Algorithms

GNN (Global Nearest Neighborhood), which picks up the best one from all the available measurements for updating the predicted state, is simple and fast. But, its disadvantages are: less efficient in low SNR (Signal to Noise Ratio), less efficient in ambiguous situation and less efficient when objects are interacting. On the other hand, JPDA (Joint Probabilistic Data Association) is a modified version of PDA (Probabilistic Data Association) where joint probability is computed across all the targets. But, as JPDA (Joint Probabilistic Data Association) is less efficient in low SNR (Signal to Noise Ratio) situation, ML-JPDA (Maximum Likelihood Joint Probabilistic Data Association) has been proposed for some cases. The drawbacks of JPDA (Joint Probabilistic Data Association) are: it is less efficient in low SNR (Signal to Noise Ratio), or when objects are interacting or objects are closely spaced. Moreover, JPDA (Joint Probabilistic Data Association) cannot cannot detect new objects automatically. MHT (Multiple Hypothesis tracker) uses the concept of batch processing where later measurements are used prior correlation association. Another version of MHT (Multiple Hypothesis tracker) is PMHT (Probabilistic Multiple Hypothesis tracker) which works at low SNR, but needs higher computational load. In MHT (Multiple Hypothesis tracker), number of hypothesis increases exponentially with number of measurements.

3.3 Nearest Neighbor Data Association

It is the simplest data association strategy although it has some other modified variants also. All data association algorithms are designed for Kalman Filter. Hence, the equations used in Kalman Filter are not modified. Let, $\mathbf{x}$ be the state, $\mathbf{z}$ be the measurement, k be the stage, $\mathbf{P}$ be the state error covariance matrix, $\mathbf{Q}$ be the process noise covariance matrix, $\mathbf{R}$ be the measurement noise covariance matrix, $\mathbf{F}$ be the system matrix, $\mathbf{H}$ be the measurement matrix and d be the number of measurements. $KF_{Predict}$ denotes all the stages of Kalman Filter prediction. The best measurement is picked up for the update stage as follows:

1. Prediction:

$$[\mathbf{x}_{k|k-1}, \mathbf{P}_{k|k-1}] = KF_{Predict}[\mathbf{x}_{k-1}, \mathbf{P}_{k-1}, \mathbf{Q}, \mathbf{F}] \tag{3.1}$$

2. Distance computation:

$$\mathbf{z} = argmin[\mathbf{z}_k(t) - \mathbf{H}\mathbf{x}_{k|k-1}]^T \mathbf{S}_{k|k-1}{}^{-1}[\mathbf{z}_k(t) - \mathbf{H}\mathbf{x}_{k|k-1}] \tag{3.2}$$

where t=1 ...d and $\mathbf{S}_{k|k-1} = \mathbf{H}\mathbf{P}_{k|k-1}\mathbf{H}^T + \mathbf{R}_k$

3. Pickup the best measurement as the closest to predicted one and that measurement is used for updating the Kalman filter.

For multiple object tracking, a matrix $\mathbf{L}$ is formed. Rows contain d_1 measurements and colums contain d_2 targets. Thus $\mathbf{L}$ is a $d_1 \times d_2$ matrix. If $\mathbf{L}$ is a square matrix, data association is done using Hungarian method. Otherwise, Munker's method is used for non-square matrix $\mathbf{L}$.

3.4 Probabilistic Data Association

In comparison with NNF (Nearest Neighbor Filter), the PDA (Probabilistic Data Association) uses more than one measurement to update the state of the target. It forms a gate and considers those measurements which are within the gate. Unlike NNF (Nearest Neighbor Filter), it assigns weight to all these measurements inside the gate and uses them to compute updated stage.

For PDA (Probabilistic Data Association), the computational requirements are around fifty percent more than that of the normal Kalman filter. The steps in PDA (Probabilistic Data Association) algorithm is described briefly from [23]. The key assumptions are:

A) One target of interest is present

B) The track has been already initialized

C) Every time a measurement validation region is set up around the predicted measurement

D) The target detections are time independent with detection probability P_D

Let $\mathbf{x}$ be the state, $\mathbf{F}$ be the system matrix, $\mathbf{H}$ be the measurement matrix, $\mathbf{z}$ be the measurement, k be the stage, $\mathbf{P}$ be the state error covariance matrix, $\mathbf{Q}$ be the process noise covariance matrix and $\mathbf{R}$ be the measurement noise covariance matrix. The key steps of the basic algorithm are described as follows:

1. Predict using Kalman filter:

$$\mathbf{x}_{k|k-1} = \mathbf{F}_{k-1}\mathbf{x}_{k-1|k-1} \tag{3.3}$$

$$\mathbf{z}_{k|k-1} = \mathbf{H}_k\mathbf{x}(k|k-1) \tag{3.4}$$

$$\mathbf{P}_{k|k-1} = \mathbf{F}_{k-1}\mathbf{P}_{k-1|k-1}\mathbf{F}'_{k-1} + \mathbf{Q}_{k-1} \tag{3.5}$$

$$\mathbf{S}_k = \mathbf{H}_k\mathbf{P}_{k|k-1}\mathbf{H}'_k + \mathbf{R}_k \tag{3.6}$$

where $\mathbf{x}_{k|k-1}$ be the predicted state at k-th instant, $\mathbf{z}_{k|k-1}$ be the predicted measurement, and $\mathbf{P}_{k|k-1}$ be the predicted state error covariance matrix.

2. A gate is formed with volume $\mathcal{V}$; the validation region is formed as an ellipse. The gate probability (probability that gate having volume $\mathcal{V}$ contains true measurement) be P_G and the detection probability be P_D. The gate volume is as follows:

$$\mathcal{V} = c_{n_z}|\gamma\mathbf{S}_k|^{1/2} = c_{n_z}\gamma^{\frac{n_z}{2}}|\mathbf{S}_k|^{1/2} \tag{3.7}$$

where γ is the threshold and $\mathcal{V}$ is the gate volume. c_{n_z} is the volume of measuement dependent n_z dimensional unit hemisphere. Thus measurements within this gate are considered:

$$\{\mathbf{z} : [\mathbf{z} - \mathbf{z}_{k|k-1}]\mathbf{S}_k^{-1}[\mathbf{z} - \mathbf{z}_{k|k-1}] \leq \gamma\} \tag{3.8}$$

where $\mathbf{z}$ be the set of validated $m(k)$ number of measurements at k-th stage.

3. The data association probability at k-th stage and for t-th measurement is computed as follows:

$$\beta_t(k) = \frac{\mathcal{L}_t(k)}{1 - P_DP_G + \sum_{j=1}^{m(k)}\mathcal{L}_j(k)} \tag{3.9}$$

$$\beta_t(k) = \frac{1 - P_DP_G}{1 - P_DP_G + \sum_{j=1}^{m(k)}\mathcal{L}_j(k)},\ t = 0 \tag{3.10}$$

where $t = 1, \ldots, m(k)$ and $t = 0$ means clutter and

$$\mathcal{L}_t(k) = \frac{\mathcal{N}[\mathbf{z}_k(t); \mathbf{z}_{k|k-1}, \mathbf{S}_k]P_D}{\lambda} \tag{3.11}$$

Hence, measurements which are within the validation region are taken into consideration, and all these measurements are used together to update the step unlike NNF (Nearest Neighbor Filter) where only the best measurement is picked up. Hence, $\beta_t(k)$ gives the data association weight of t-th measurement with the target at k-th stage.

4. The Kalman-predicted state is updated using those validated measurements along with their computed weight respectively:

$$\mathbf{x}_{k|k} = \mathbf{x}_{k|k-1} + \mathbf{W}(k)\boldsymbol{\nu}(k) \tag{3.12}$$

The computation of combined innovation is as follows:

$$\boldsymbol{\nu}(k) = \sum_{t=1}^{m(k)} \beta_t(k)\boldsymbol{\nu}_t(k) \tag{3.13}$$

Kalman Gain is computed as:

$$\mathbf{W}(k) = \mathbf{P}_{k|k-1}\mathbf{H}'_k\mathbf{S}_k^{-1} \tag{3.14}$$

$$\mathbf{P}_{k|k} = \beta_0(k)\mathbf{P}_{k|k-1} + [1-\beta_0(k)]\mathbf{P}^c_{k|k} + \tilde{\mathbf{P}}_k \tag{3.15}$$

$$\mathbf{P}^c_{k|k} = \mathbf{P}_{k|k-1} - \mathbf{W}(k)\mathbf{S}_k\mathbf{W}(k)' \tag{3.16}$$

$$\mathbf{P}^c_{k|k} = \mathbf{P}_{k|k-1} - \mathbf{W}(k)\mathbf{S}_k\mathbf{W}'_k \tag{3.17}$$

$$\tilde{\mathbf{P}}_k = \mathbf{W}(k)\left[\sum_{t=1}^{m(k)} \beta_t(k)\boldsymbol{\nu}_t(k)\boldsymbol{\nu}_t(k)' - \boldsymbol{\nu}(k)\boldsymbol{\nu}(k)'\right]\mathbf{W}(k)' \tag{3.18}$$

The main disadvantage of PDA (Probabilistic Data Association) lies in the selection of proper validation gate. If it is too large, clutters will be included, and, if it is too small, actual measurement may be excluded. Another variant of PDA (Probabilistic Data Association) is JPDA (Joint Probabilistic dada Association). PDA (Probabilistic Data Association) is extended to JPDA (Joint Probabilistic Data Association) for multiple object tracking. There are various fast implementation algorithms of this JPDA (Joint Probabilistic Data Association). The complexity of JPDA (Joint Probabilistic Data Association) can be further reduced by using MCMCDA (Markov Chain Monte Carlo Data Association) algorithm. JPDA (Joint Probabilistic Data Association) is described in [23, 24] where measurement-to-target association is computed jointly across all the targets. JPDA (Joint Probabilistic Data Association) is used for multiple target tracking. The state estimation is calculated either separately or jointly. In decoupled estimation, marginal association probabilities are required which is computed by summing over all the joint events in which marginal event of interest occurs. Many fast sub-optimal implementations of JPDA (Joint Probabilistic Data Association) have been proposed.

3.5 Particle Filter Data Association

Although data association algorithms are designed for Kalman filters [25], they are used with particle filters also [26, 27]. In this section, how particle filters can be merged with data association algorithms is introduced. Particle filter based tracker explained in 2.5.6 can be experimented with any of the data association algorithms to analyze their performance. One way to merge particle filters with JPDA (Joint Probabilistic Data Association) is described in [26]. Particle filter with JPDA (Joint Probabilistic Data Association) is used to track multiple objects. The key philosophy in [26] is to update the *weight* of particle filters in a different way. The weight equation in [26] not only includes measurement likelihood but also the data association probability. The weight for each particle is computed described in [26] is given as:

$$w_{t_2,k} = \sum_{t_1=0}^{m_1(k)} \beta_{t_2,k}^{t_1} p(\mathbf{z}_k^{t_1} | \mathbf{x}_{t_2}^i) \tag{3.19}$$

where $m_1(k)$ be the total number of measurement and $m_2(k)$ be the total number of target at k-th stage, t_1 be the measurement, t_2 be the target, i be the particle, k be the stage. $p(\mathbf{z}_k^{t_1} | \mathbf{x}_{t_2}^i)$ be the likelihood with respect to t_1-th measurement. Thus $t_1 = 1...m_1(k)$ and $t_2 = 1...m_2(k)$.

3.6 Summary

In this chapter, different data association algorithms like Nearest Neighbor Filter, Probabilistic data Association Filter and its variant Joint Probabilistic Data Association Filter are explained along with their comparisons. It has also been described how they can be used with particle filters. Hence, single object tracking algorithms can be extended to multiple object tracking either by running a number of trackers in parallel or by using data association algorithm. As told earlier, data association algorithms are more robust when objects come too close to each other. Once multiple object tracking is done, it can be extended to multi camera object tracking depending on the requirement of application. Multi camera object tracking has tremendous applications reviewed in the next chapter in detail.

Chapter 4

Review of Multi-camera and 3D Object Tracking

4.1 Introduction

In this chapter, multiple camera object tracking as well as depth estimation techniques from video frames are reviewed. Camera networks are used in various applications including surveillance, disaster response and environmental modeling etc. In most of the cases, the data from each node (camera node) is sent to the central server for processing. In a general approach, inside the processing unit, program written for each camera first tracks multiple objects using tracking algorithms, and then, the objects tracked are stitched across different camera views using transition-probability and appearance-probability matching. On the other hand, in case of depth estimation, stereo camera based approach is found to be very popular and so it is reviewed in this chapter. Stereo 3D tracking is widely used in various applications including robotics and automation engineering. With the increment of automation in industry, the demand of humanoids is also increasing where depth estimation is a crucial part. Various stereo algorithms have been proposed so far. Another multi camera tracking application is human motion tracking. It is widely used in smart environments where analysis of human motion is very crucial. Similarly tracking human motion under multiple camera network is used in many smart applications. Tracking human motion in a camera network is quite challenging because of the highly deformable nature of human body. All these techniques are addressed in this chapter.

4.2 Camera Calibration

Camera calibration is the process of estimation of intrinsic and extrinsic parameters of the camera. Intrinsic parameters include focal length, principal point, size of pixel etc., while extrinsic parameters include camera orientation. With both intrinsic and extrinsic parameters, camera matrix can be formed. With the help of this camera matrix, any 2D image co-ordinate is converted to 3D real world coordinate and vice-a-versa. Literature [28, 29] gives details of camera calibration. A detail method of camera calibration has been found in the lecture video [30]. The main goal is to compute the camera matrix which is formed using the geometry between real 3D world coordinates and their corresponding 2D image-coordinates, considering pin-hole camera model. Once the camera matrix is formed, one can track the object in 2D image and project back its 2D coordinates to the original 3D coordinates. With MATLAB Camera Calibration app, the calibration task can also be done to get the camera matrix directly. In contrast to the reference based camera calibration, in self camera calibration, camera is moved in a predefined fashion, and then, the camera matrix is computed by clubbing rotational, translational matrices etc. 3D coordinate is multiplied with this camera matrix gives the 2D location in image plane. Thus this real world coordinate and image coordinates are correlated.

4.3 Multi-camera Object Tracking

In multi-camera object tracking, one popular approach is the homography based approach. Kalman filters are also used in multi camera object tracking using homography. Although many algorithms have been proposed so far, the main difference in multiple camera object tracking lies in the camera topology. In some topology, the field of view of two adjacent cameras can overlap. In other case, none of the cameras share their field of views with others. In this section, both overlapped and un-overlapped multiple camera object tracking is introduced.

4.3.1 Disjoint Field of View Tracking

In disjoint FoV (field of view) tracking, the camera views do not overlap. This increases the area of observation. The approach explain in [31] uses camera topology and Parzen window to find path probabilities during training period; Parzen window is generally used when the form of probability density is unknown. In this approach, the objective is to find the MAP (maximum a posteriori). The advantage of this method is that tracking does not require calibration, but the disadvantage is that the system needs to be trained occasionally. Parzen window is used to compute inter-camera space-time probabilities like probability of an object to enter a certain camera at a certain time given location, time and velocity of exit at the other camera. The probability of two objects to be same depends upon space-time information and appearance. In [31], learning is done by making one person traveling across the disjointed cameras during the training period. The appearance similarity across different cameras is calculated using the same Bhattacharyya coefficient. In some other literature, to compare appearance, BTF (Brightness Transfer Function) has been proposed for better performance.

4.3.2 Overlapped Field of View Tracking

In multi camera object tracking, generally data from each node (camera terminal) is sent to the centralized server.But, due to low bandwidth, less security and memory management, each node of camera should act in an autonomous way; this leads to the concept of autonomous processing. In autonomous processing, each node interacts with its neighbors and share information to reach a consensus. Thus each camera processes information locally. Kalman filter modified with consensus algorithm can be used in this scenario. Autonomous processing applications are distributed in nature. On the other hand, in some other application, both the distributed and centralized algorithms are mixed together. Distributed camera networks are used widely used in Ubiquitous Computing. [32]

uses Kalman consensus filter, which is widely used in distributed sensor network, to track objects in a overlapped camera network. It is a decentralized tracking system. Each camera has a Kalman filter and shares information with its neighbors, and finally reaches a consensus. The system uses homographic information. Each node (camera) shares information with its neighbor iteratively to attain a common decision. Distributed Kalman filtering is used in cooperative control based applications. Kalman consensus filter is nothing but another modified version of distributed Kalman filter. Consensus means to attain a general agreement after sharing each camera its own estimate of state with its neighbors. Hence, even if one node fails, the tracking is not lost.

4.3.3 Tracking Human Motion

In this section, review of human motion tracking is introduced. [33] describes tracking human using monocular gray-scale-images. The camera network is fixed. The key challenge is to establish correspondence among objects viewed across different cameras. This method first segments out human from static background, then establishes the correspondence. The non-rigidity of the human body is solved by matching pixels which are on the middle-line of the human body. In the first stage, single view tracking is done by segmentation and probabilistic feature association. In the second stage, multiple view (multi camera view) spatial matching is done by finding perpendicular distance from an equi-polar line formed by the other camera view; from this perpendicular distance, a Gaussian model is derived and matching is done according to this Gaussian model.

4.3.4 Critical Analysis

Both object transition-probability and object similarity-probability may fail (especially in case of non-overlapping camera topology) if object is moving in a more unpredicted way and the appearance of the object is changed in an abrupt fashion respectively. Similarly, in case of [32], the camera needs to be calibrated frequently; the accuracy of tracking is dependent upon calibration. Literature [34] uses the concept of Brightness Transfer Function (BTF) which relates how two different cameras can see the same object. Literature [35] derives an expression for the transition probability taking case of the *blind area*.

4.4 Depth Estimation and 3D Tracking

4.4.1 Introduction

Depth estimation and 3D tracking have vast application ranging from Computer Vision to Control engineering (as for example, in robotics or in outer space vehicles to form a convoy). Thus depth estimation and 3D tracking is a growing research topic. Another application of 3D tracking can be used to learn and follow the path by automated objects. An example is KUKA robot which learns using 3D path formation of the tracked object. Apart from this, 3D image information is also used in movies and computer games etc. 3D information of a tracked object can be obtained by calibrating a single or a stereo-camera. As already discussed, using the camera matrix, 3D and 2D points can be related. On the other hand, in stereo camera configuration, depth is calculated from disparity.

4.4.2 Literature Review : Different Approaches to 3D Tracking

Depth is estimated from disparity in stereo configuration. If a single camera is used, 3D tracking is done using camera matrix which is derived from the intrinsic and extrinsic parameters.

Stereo configuration is not only used for depth estimation only. It can be used for higher accuracy in tracking. [36] uses particle filter under stereo configuration. Likelihood is computed using both the stereo images. Then these two likelihood values of object appearance are clubbed together and used as a joint likelihood. [37] also uses almost the same concept. They try to use both 2D and 3D information into filtering for higher accuracy rather than depth estimation.

[38] uses single camera for 3D tracking where camera parameters are used to correspond between 3D and 2D. Literature [39] derives a novel approach to track objects in 3D. In their approach, two sets of particle filters on each of the stereo images are formed and a correspondence is made in between them. Literature [40] derives disparity information under three different conditions to estimate depth.

4.5 Summary

In this chapter, various multi camera and 3D object tracking schemes are reviewed. It can be seen that the research-trend in stereo depth estimation is to increase the depth of estimation with accuracy. Similarly, in multi camera object tracking, training the camera system has to be improved so that the objects coming from the *blind area* can be modeled properly for efficient detection. Another research area might be low resolution multi camera object tracking. Since low resolution camera network will take low bandwidth for communication, this can be used for pervasive or ubiquitous computing, which is a leading research topic in embedded systems engineering.

In pervasive computing, human machine interaction is a crucial part, where tracking objects can be a small but most important part of research. As for example, ubiquitous monitoring of old people staying in an environment; for automatic analysis of their behavior or motion, a network of embedded cameras has to be mounted the output of which has to be sent either to a central processor or has to processed at each camera node. Pervasive computing environments are donned with communication, networking, computing, Human Machine Interfacing, speech and vision facilities, etc. Pervasive computing environments can be either stand alone or mobile; as for example, mounted on a ship etc. Hence, in all these applications, multiple object tracking and establishing correspondence among them from the view of different cameras in a sensor network is very crucial.

Till this chapter, all major visual object tracking algorithms are reviewed. Now a new variant of particle filter based visual tracking algorithm has been proposed in this work which is explained in the next chapter.

Chapter 5

Proposed Algorithm

5.1 Introduction

In this chapter, a new variant of particle filter along with a stochastic resampling algorithm has been proposed. In visual object tracking, particle filters have been used popularly because they are compatible with system non-linearity and non-Gaussian posterior distribution. But the main problem in particle filtering is sample degeneracy. To solve this problem, a new variant of particle filter has been proposed. The resampling algorithm used in this proposed particle filter is derived by combining systematic resampling, which is commonly used in SIR-PF (Sampling Importance Resampling Particle Filter), and a modified bat algorithm; this resampling algorithm reduces sample degeneracy as well as impoverishments. The measurement model is modified to handle clutter in presence of varying background. A new motion dynamics model is proposed which further reduces the chance of sample degeneracy among the particles by adaptively shifting mean of the process noise. To deal with illumination fluctuation and object deformation in presence of complete occlusion, a template update algorithm has also been proposed. This template update algorithm can update template even when the difference in the spread of the color-histogram is especially large over time. The proposed tracker has been tested against many challenging conditions and found to be robust against against clutter, illumination change, scale change, fast object movement, motion blur, and complete occlusion; it has been found that the proposed algorithm outperforms the SIR-PF and bat algorithm.

Different forms of particle filters have been widely used in all sorts of visual object tracking widely. The key idea behind particle filter is to generate particles that will simulate the true posterior, then from the mean of the posterior can be easily estimated. The higher is the number of particles, the better is the simulation. In case of image data, both the likelihood and the posterior distributions are non-Gaussian. Hence, particle filters are preferred to Kalman filters in visual object tracking; Kalman filters work accurately only when posterior and noise are Gaussian, including the system to be linear.

But still there are some drawbacks in particle filter. Computationally a particle filter is less efficient compared to a Kalman filter. So, in some cases, where the process and measurement noise are almost or nearly Gaussian and system can be approximated by linearization, Kalman filters can be used, and, in such cases, Kalman filters can give optimal solution. Kalman filters are much higher computational efficiency as compared to particle filters. Large computational cost is one major drawback of particle filtering. Another problem in particle filter is called sample impoverishment which arises while resolving the degeneracy problem in particle filters. But, as compared to Kalman filters, particle filters are more robust to clutter. Hence, for this project, particle filter is selected. Lots of work has been done on particle filter based visual object tracking. Many modifications in particle filtering [18, 20, 26, 21, 27, 36, 39, 41, 42, 43, 44] for visual object tracking have been proposed to improve tracking accuracy.

In this work, a new form of particle filter has been proposed along with a new resampling algorithm. In this work, first it has been tried to reduce the inherent problems of SIR-PF (Sampling Importance Resampling Particle Filter) like sample degeneracy and sample impoverishment. Next, other additional modifications are made to make the algorithm robust against clutter, illumination change, scale change, fast object movement, motion blur, and complete occlusion. Then the algorithm is tested using a number of different challenging videos, and found to be working satisfactorily as compared with SIR-PF (Sampling Importance Resampling Particle Filter), bat algorithm and other state-of-the-art algorithms.

5.2 Related Works

Till now, numerous tracking algorithms have been proposed in the field of visual object tracking [5, 6, 7, 45, 46, 47, 44, 48, 15, 21]. As discussed earlier, these algorithms can be classified in two broad categories: i) optimization based algorithms and ii) filtering based algorithms. Both these approaches have their own advantages and disadvantages depending on the application. Traditionally, optimization based tracking methods are deterministic in nature, however recently nature inspired stochastic optimization methods [45, 41] have also been proposed for object tracking. On the other hand, filtering based tracking methods are stochastic in nature.

Compared to the deterministic trackers [8], probabilistic trackers [18] are relatively more robust in presence of short-time complete occlusion as well as background clutter. In probabilistic tracking, two most popular filters are: Kalman filter and particle filter. As discussed earlier, the Kalman filters perform well when the underlying distribution is Gaussian, however particle filters can adapt to more general distributions. Likelihood model in visual object tracking is non-linear and posterior is non-Gaussian. Therefore visual object tracking methods based on particle filters are more successful. Thus particle filters outperform Kalman filters in visual object tracking.

Although Kalman filters are not more suitable than particle filters for visual object tracking, in some applications, Kalman filters are also used [46]. The modified versions of Kalman filter are extended Kalman filter, unscented Kalman filter, etc. [49] uses unscented Kalman which is far better than extended Kalman filter. Unscented Kalman filter preserves linearity up to order three with the same computational load as extended Kalman filter, using the concept of sigma points and unscented transformation. The main disadvantage with extended Kalman filter is to find out Jacobbian matrix at every time step. [49] uses this unscented Kalman filter for contour tracking. It computes nonlinear measurement model more accurately without the help of a Jacobian matrix. So some researchers have combined unscented Kalman filter and particle filter to formulate unscented particle filter for better proposal density distribution [42]. As already discussed, one key problem in particle filtering is sample degeneracy. After a few iterations, degeneracy may occur when there are many particles having negligible weight. One can overcome this problem of sample degeneracy either by choosing better importance density [42] or by systematic resampling [19]. While re-sampling, these low-weight particles are discarded while higher weight-particles are replicated. Hence, particles get concentrated around higher weight, thus reduces diversity among the particles. But this systematic resampling may make the particles merge to a single point making them lose their diversity. This problem is known as sample impoverishment. To overcome sample impoverishment, various improvements have been proposed in the literature [41, 43]. A good choice of selecting particles is to increase their variance. Hence, ideally the particles should have diversity without any sample degeneracy. In this work, a modified resampling algorithm is proposed to address these issues simultaneously.

In another approach, rather than computing a better proposal density, particles filters are merged with nature inspired stochastic optimization algorithms. Such nature inspired stochastic optimization algorithms have been proposed in the literature [50]. In the proposed work, bat algorithm has been modified to be merged with particle filter. Amongst other contemporary stochastic optimization algorithms, bat algorithm has a fast convergence rate in the initial stages. Also the flavor of Particle Swarm Optimization (PSO) and Harmony Search (HS) can be obtained from bat algorithm by changing loudness and pulse-rate of the microbats. It has been shown in [50] that bat algorithm performs better than Particle Swarm Optimization (PSO) and under certain conditions where Particle Swarm Optimization (PSO) may fail, bat algorithm will converge to the global optimum [50].

In this proposed work, both the SIR-PF (Sampling Importance Resampling Particle Filter) and the bat algorithm are modified and then merged together to get rid of sample degeneracy and sample impoverishment mainly. The tracking efficiency of the proposed algorithm has been found to be much better than the SIR-PF (Sampling Importance Resampling Particle Filter) and bat algorithm. The proposed algorithm has been tested under many challenging conditions and found to be robust against against clutter, illumination change, scale change, fast object movement, motion blur, and complete occlusion. In this work, test results from various datasets [1, 2, 3] have been demonstrated. These data sets are collections of benchmark videos on visual object tracking.

5.3 Proposed Particle Filter

In this section, the proposed particle filter, which is a modified version of SIR-PF (Sampling Importance Resampling Particle Filter), has been described. The key idea of particle filtering is to simulate the true posterior using a number of large particles. But there are some disadvantages of particle filter. These are:

1. Sample degeneracy may occur after some iterations
2. Sample impoverishment due to systematic resampling [19]
3. Large number of particles required to simulate the true posterior correctly

In this proposed particle filter algorithm along with the modified bat inspired resampling algorithm, these problems are resolved. This increases the tracking accuracy.

5.3.1 Sampling Importance Resampling Particle Filter

First the basic equations of SIR-PF (Sampling Importance Resampling Particle Filter) is revisited. As discussed earlier, Bayesian tracker has two steps: Prediction and Update. Let, k be the stage, $\mathbf{x}$

be the state, $\mathbf{z}$ be the measurement vector. Thus prior (predicted ones) is updated with a likelihood and a normalization factor as follows using Bayes' rule:

$$p(\mathbf{x}_k|\mathbf{z}_{1:k}) = \frac{p(\mathbf{z}_k|\mathbf{x}_k)p(\mathbf{x}_k|\mathbf{z}_{1:k-1})}{p(\mathbf{z}_k|\mathbf{z}_{1:k-1})} \tag{5.1}$$

where, $\mathbf{x_k}$ = estimated state at k-th instant and $\mathbf{z_k}$ = measurement at k-th instant.

$$p(\mathbf{x}_k|\mathbf{z}_{1:k-1}) = \int p(\mathbf{x}_k|\mathbf{x}_{k-1})p(\mathbf{x}_{k-1}|\mathbf{z}_{1:k-1})\,d\mathbf{x}_{k-1} \tag{5.2}$$

$$p(\mathbf{z}_k|\mathbf{z}_{1:k-1}) = \int p(\mathbf{z}_k|\mathbf{x}_k)p(\mathbf{x}_k|\mathbf{z}_{1:k-1})\,d\mathbf{x}_k \tag{5.3}$$

In particle filtering, the posterior distribution is estimated by particles having some weight. So these particles imitate the true posterior; finally the mean of this posterior is computed. The state (samples or particles) is predicted using system dynamics as follows:

$$\mathbf{x}_k = \mathbf{f}_k(\mathbf{x}_{k-1},\, \mathbf{v}_{k-1}) \tag{5.4}$$

where, $\mathbf{v}_{k-1}$ is the process (state) noise.

There may be mismatch in the actual system dynamics and the modelled system dynamics. This model mismatch as well as the fact that states are hidden (not measured) make the predictions inaccurate. Hence it is necessary to update the predictions using (5.1). The measurement model is as follows:

$$\mathbf{z}_k = \mathbf{h}_k(\mathbf{x}_k, \mathbf{n}_k) \tag{5.5}$$

where $\mathbf{n}_k$ is the measurement noise.

In particle filtering, the key objective is to properly represent the posterior distribution $p(\mathbf{x}_k|\mathbf{z}_{1:k})$. Each particle i at stage k has weight w_k^i which gives some measure of posterior at that point:

$$w_k^i \propto \frac{\pi(\mathbf{x}^i)}{q(\mathbf{x}^i)} \tag{5.6}$$

The theory as described in [19] is: *particles are generally difficult to draw from distribution* $\pi(\mathbf{x}^i)$, *but* $\pi(\mathbf{x}^i)$ *can be computed.* Hence, samples are generated from another distribution called importance density $q(\mathbf{x}^i)$. Now the modified form of the weight at k-th instant is:

$$w_k^i \propto w_{k-1}^i \frac{p(\mathbf{z}_k|\mathbf{x}_k^i)p(\mathbf{x}_k^i|\mathbf{x}_{k-1}^i)}{q(\mathbf{x}_k^i|\mathbf{x}_{k-1}^i,\, \mathbf{z}_k)} \tag{5.7}$$

where $p(\mathbf{z}_k|\mathbf{x}_k^i)$ is the likelihood, $p(\mathbf{x}_k^i|\mathbf{x}_{k-1}^i)$ is the prior and $q(\mathbf{x}_k^i|\mathbf{x}_{k-1}^i,\, \mathbf{z}_k)$ is the importance density.

As the number of particles or samples N_s increases, the following computation tends to the true posterior:

$$p(\mathbf{x}_k|\mathbf{z}_{1:k}) \approx \sum_{i=1}^{N_s} w_k^i \delta(\mathbf{x}_k - \mathbf{x}_k^i) \tag{5.8}$$

For simplicity, importance density is taken as prior. Hence, the weight becomes:

$$w_k^i \propto w_{k-1}^i p(\mathbf{z}_k|\mathbf{x_k}) \tag{5.9}$$

Reduction of Equation (5.7) to (5.9) results in some inaccuracy in the tracking. There are many ways to to improve this. In [42], the importance density distribution $q(\mathbf{x}_k^i|\mathbf{x}_{k-1}^i, \mathbf{z}_k)$ and the prior $p(\mathbf{x}_k^i|\mathbf{x}_{k-1}^i)$ are derived from unscented Kalman filter; this forms the unscented particle filter. This helps shifting the prior towards the better likelihood. Another way of shifting the particles to better position is to use either deterministic or probabilistic optimization algorithm as stated earlier.

The steps in SIR-PF (Sampling Importance Resampling Particle Filter) algorithm are shown in Algorithm 2.

Algorithm 2 SIR-PF Algorithm

```
1: procedure SIR-PF($(\mathbf{x}_{k-1}^i, w_{k-1}^i)_{i=1}^{N_s}, \mathbf{z}_k$)
2:     for $i = 1 : N_s$ do
3:         Draw: $\mathbf{x}_k^i \sim p(\mathbf{x}_k^i|\mathbf{x}_{k-1}^i)$
4:         Compute: $w_k^i = p(\mathbf{z}_k|\mathbf{x}_k^i)$
5:     Normalize: $w_k^i$ for $i = 1...N_s$
6:     Systematic Resample: Discard particles with low weight and replicate particles with higher
   weight as given in [19]
7:     return $(\mathbf{x}_k^i, w_k^i)_{i=1}^{N_s}$
```

5.3.2 Proposed Algorithm

The key idea behind particle filtering is to simulate the true posterior using a large number of particles. The weight associated with each particle gives the measure of posterior at that state. In particle filtering, degeneracy may occur after a few iterations. When degeneracy occurs, the particles cannot simulate the true posterior properly, producing error in tracking. The most popular form of particle filter is SIR-PF (Sampling Importance Resampling Particle Filter) [19] which uses systematic resampling. This systematic resampling can eliminate sample degeneracy, but may cause another problem called sample impoverishments. So, a new variant of particle filter along with resampling algorithm has been proposed in this work, which eliminates both sample degeneracy and sample impoverishments among the particles, increasing the accuracy in tracking.

To reduce sample degeneracy, the proposed particle filter uses selective resampling. Let i be the particle, k be the stage and $\mathbf{x}$ be the state. Hence, state of i-th particle at k-th stage $\mathbf{x}_k^i$ is drawn from state of i-th particle at $(k-1)$-th stage $\mathbf{x}_{k-1}^i$ as follows:

$$\mathbf{x}_k^i \sim p(\mathbf{x}_k^i|\mathbf{x}_{k-1}^i) \tag{5.10}$$

Hence, at k-th stage, two states are available for i-th particle. One is from previous stage $\mathbf{x}_{k-1}^i$ and another one is newly drawn state: $\mathbf{x}_k^i$. But, measurement is taken only at k-th stage for both these states: $\mathbf{z}_k^i$ for the i-th particle. Thus unlike SIR-PF (Sampling Importance Resampling Particle Filter), the weight is computed twice as shown in (5.11) and (5.12):

$$w_k^i = p(\mathbf{z}_k^i|\mathbf{x}_k^i) \tag{5.11}$$

$$w_{prev}^i = p(\mathbf{z}_k^i|\mathbf{x}_{k-1}^i) \tag{5.12}$$

The two weights (5.11) and (5.12) are compared and the best one is chosen. Thus this selective resampling helps the tracker work even if the process noise model is not properly tuned. The steps of the proposed particle filter is explained in Algorithm 3 (where N_s be the total number of particles, $LostTrackTH$ is the threshold).

Algorithm 3 Proposed Algorithm

```
1: procedure PROPOSED ALGORITHM($VideoInput, InitialLocation, WindowSize$)
2:     $fullscale$, $Updating$ and others are set $true$ or $false$ depending upon the application
3:     All the threshold values are set
4:     Number of particles $N_s$ is initialized
5:     Initialize particles $x_k^i$ ($i = 1 : N_s$) around $InitialLocation$ in the initial frame
6:     Initialize reference histogram $\hat{q}_r$ of the object
7:     while Reading frame $k$-th from $VideoInput$ do
8:         for $i = 1 : N_s$ do
9:             Draw: $\mathbf{x}_k^i \sim p(\mathbf{x}_k^i|\mathbf{x}_{k-1}^i)$
10:            Compute likelihood: $w_k^i = p(\mathbf{z}_k^i|\mathbf{x}_k^i)$ using (5.20)
11:            Compute likelihood: $w_{prev}^i = p(\mathbf{z}_k^i|\mathbf{x}_{k-1}^i)$ using (5.20)
12:            if $w_k^i < w_{prev}^i$ then
13:                $\mathbf{x}_k^i = \mathbf{x}_{k-1}^i$
14:                $w_k^i = w_{prev}^i$
15:        Estimate mean $\mathbf{y}_k$ of all the states $\mathbf{x}_k^i$ for $i = 1...N_s$
16:        Compute Bhattacharyya similarity coefficient $\rho_k^\star$ at $\mathbf{y}_k$ using (5.18)
17:        if $\rho_k^\star < LostTrackTH$ then
18:            $r0 = 1$
19:        else
20:            $r0 = 0$
21:        Resample particles (Algorithm 6)
22:        Re-estimate mean $\mathbf{y}_k$ of all the updated resampled states $\mathbf{x}_k^i$ for $i = 1...N_s$
23:        Compute the final estimated location $\mathbf{y}_k$
24:        Update the template (Algorithm 7)
25:        Update the model dynamics (Algorithm 4)
26:        Annotate at $\mathbf{y}_k$ and display the frame
27:        Pass $\mathbf{y}_k$ for the next frame
```

Hence, although the computational cost of the proposed particle filter is more than that of the conventional SIR-PF (Sampling Importance Resampling Particle Filter), the proposed algorithm can solve sample degeneracy and sample impoverishment much better.

5.4 Proposed Measurement Model

Measurement model is one of the most crucial part of particle filter based tracking. A better measurement model can reduce the effect of clutter, making the tracking robust. The proposed measurement model is formulated by merging two other different measurement models [8, 18].

5.4.1 Other Related Works

Till now various measurement models, such as covariance based [14], edge based [11, 10], texture based [10], and color based [8] models have been proposed in the literature. To include spatial information in histograms, multi part modeling has been introduced in [18, 10]. Work [11] splits the MeanShift tracker into different small segments to make it more robust against occlusion. Works [8, 18, 10, 11] all use background modeling and merge them to the measurement model to make the measurement model more robust against clutter. To solve clutter, [10] merges color, edge, texture and background information all together with MeanShift making it robust against background clutter.

5.4.2 Proposed Measurement Model

A better measurement model can reduce the effect of clutter. A measurement model is proposed by combining the measurement models from literature [18] and [8]. [18] introduces the concept of multi-part modeling which adds spatial information to the histogram and [8] introduces background modeling for dynamically changing background. [18] also uses background modeling where background should be static and background information should be available offline. In [8], a background probabilistic histogram is formed, and each bin value of the histogram is compared with its smallest non-zero bin value ($\hat{o}^\star$). In the proposed approach, instead of comparing with the least probability value, a user defined threshold Th is used for comparison. The proposed measurement model is more robust when the diversity among the colors in background is less. Thus, in the proposed model, spatial information is added to the color histogram while background can be dynamic.

A four dimensional histogram is proposed. Three dimensions are for colors Red, Green and Blue, and one more dimension is for storing spatial information about the pixel (pixel location). But, if each pixel's location is stored, the dimension of the histogram will be too large to process fast. Hence, the image template is divided or quantized into number of blocks, and the block number a pixel belongs to is considered as the spatial information while generating the histogram. And, for processing the background information, the background template around the object template having the equal area is taken, and the pixels which are outside the object template but inside the background template are considered as background pixels for processing.

A measurement model based on joint m-bin histogram ($r = 1...m$) as reference model as well as target candidate has been proposed [8]. The center of the template is taken as the origin. Epanechnikov kernel used in this model makes border pixels have lesser weight. Let $\hat{q}_r$ and $\hat{p}_r(\mathbf{y})$ ($r = 1...m$) be the background-weighted reference and target color histograms, respectively as given in [8]. $\mathbf{y}$ be the location of the target histogram. $\hat{o}_r$ be the discrete probabilistic histogram of the background. Notation $\mathbf{n}_l^\star$ be the l-th normalized pixel location, δ be the Kronecker delta function and $K(.)$ be the Epanechnikov kernel. n and n_h are the number of pixels in reference and target histograms. Each bin-value r of the joint histogram is multiplied by a modified weight v_r given below :

$$\left\{ v_r = \min\left(\frac{Th}{\hat{o}_r}, 1\right)\right\}_{r=1...m} \tag{5.13}$$

$$\hat{q}_r = Cv_r \sum_{l=1}^{n} K(\|\mathbf{n}_l^\star\|^2)\delta\left[b(\mathbf{n}_l^\star) - r\right] \tag{5.14}$$

$$C = \frac{1}{\sum_{l=1}^{n} K(\|\mathbf{n}_l^\star\|^2) \sum_{r=1}^{m} v_r \delta \left[b(\mathbf{n}_l^\star) - r \right]} \tag{5.15}$$

$$\hat{p}_r(\mathbf{y}) = C_h v_r \sum_{l=1}^{n_h} K \left(\left\| \frac{\mathbf{y} - \mathbf{n}_l^\star}{h} \right\|^2 \right) \delta \left[b(\mathbf{n}_l^\star) - r \right] \tag{5.16}$$

$$C_h = \frac{1}{\sum_{l=1}^{n_h} K(\|\frac{\mathbf{y}-\mathbf{n}_l^\star}{h}\|^2) \sum_{r=1}^{m} v_r \delta \left[b(\mathbf{n}_l^\star) - r \right]} \tag{5.17}$$

Now, these two histograms are compared by Bhattacharyya coefficient [8]. Bhattacharyya coefficient approximates the chi-square statistics eliminating the singularity problem while dealing with empty bins [22]. The equation is given in (5.18).

$$\rho = \sum_{r=1}^{m} \sqrt{\hat{p}_r(\mathbf{y}) \hat{q}_r} \tag{5.18}$$

The distance between the two distributions is given by [8]:

$$D(\mathbf{y}) = \sqrt{1 - \rho} \tag{5.19}$$

Hence, likelihood of the i-th particle at $\mathbf{y}$ in k-th stage is given by:

$$p(\mathbf{z}_k^i | \mathbf{x}_k^i) \propto \exp(-\lambda D(\mathbf{y})) \tag{5.20}$$

where, λ is a tuning parameter [18].

The proposed likelihood model is expected to reduce clutter in case of dynamically changing background, and also it can suppress background color even if background has some color throughout the entire background template region.

The proposed measurement model is tested. The test result is given in Fig. 5.1 and Fig. 5.2. Fig. 5.1 shows the locations of the object templates or the windows. *Window 1* is the initial reference window. Other windows are taken over the entire frame to test the efficiency of the proposed measurement model. The dimension of the histogram is $8 \times 8 \times 8 \times 2$ where 2 is the number of the blocks the object template or window is divided into for utilizing the spatial information. The image frame is taken from [3]. Hence, the proposed likelihood model can reduce clutter in case of dynamically changing background, and also it can suppress background color even if background has same color throughout the entire background template region.

5.5 Proposed Motion Dynamics Model

Bayesian filters track dynamics of the object. Hence, many motion dynamics models have been proposed in literature [51] depending on the application. In this proposed work, instead of constant-velocity model, as used in most of the cases, zero velocity model has been proposed, and velocity is encoded as process noise. The advantage of this model is that it is application independent and the model parameters change adaptively to generate better particles which may reduce sample degeneracy.

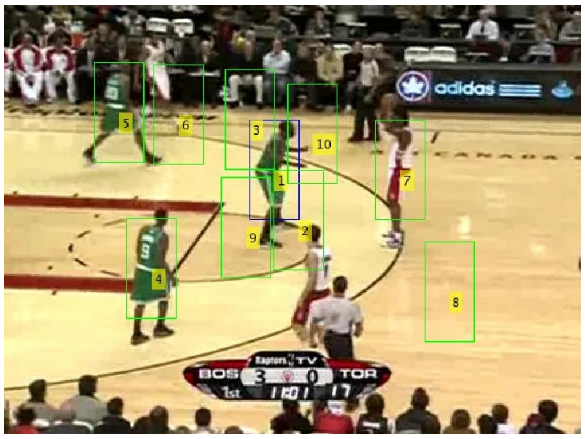

Figure 5.1: Location of the numbered windows

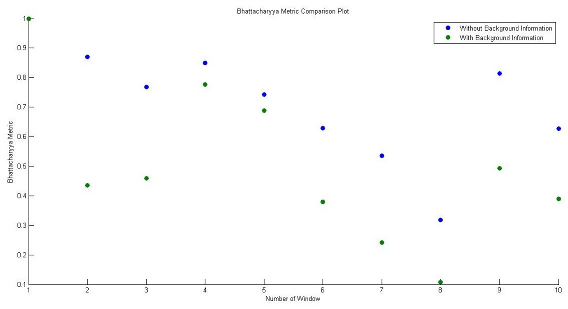

Figure 5.2: Bhattacharyya metric comparison plot

Generally the velocity (process noise) is represented by Gaussian noise $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\sigma})$ where $\boldsymbol{\mu}$, is the mean value and $\boldsymbol{\sigma}$ is the standard deviation. In this proposed adaptive approach, $\boldsymbol{\sigma}$ is kept fixed while $\boldsymbol{\mu}$ keeps changing adaptively according to (5.21):

$$\boldsymbol{\mu} = \frac{\boldsymbol{\sigma}}{2}(1 - e^{-\mathbf{d}}) \tag{5.21}$$

where $\mathbf{d} \in [0\ 3]$ is the normalized distance traveled by an object from $(k-1)$-th stage to k-th stage and $\boldsymbol{\sigma}$ is the maximum estimated velocity. When $\mathbf{d} = 0$, $\boldsymbol{\mu} = 0$, and, when $\mathbf{d} = 3$, $\boldsymbol{\mu} = 0.95 \times \frac{\boldsymbol{\sigma}}{2}$. The limiting value of $\boldsymbol{\mu}$ as $\mathbf{d} \to \infty$ is $\frac{\boldsymbol{\sigma}}{2}$.

The proposed model shifts the $\boldsymbol{\mu}$ towards the direction of motion of the object. The limiting value of $\boldsymbol{\mu}$ ensures that $\boldsymbol{\mu}$ is not completely shifted in one direction so that, if velocity of the object abruptly changes in reverse direction, the model can still track the object. The algorithm is described in Algorithm 4. $\boldsymbol{\mu}$ can be either positive or negative. $\boldsymbol{\mu}$ is divided into μ_x and μ_y. In terms of state space, particles are generated according to (5.22).

$$\mathbf{x}_k = \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \mathbf{x}_{k-1} + \begin{pmatrix} 0 \\ 0 \\ \mathcal{N}(\mu_x, \sigma_x) \\ \mathcal{N}(\mu_y, \sigma_y) \end{pmatrix} \tag{5.22}$$

where the states are $\mathbf{x} = [x \quad y \quad \dot{x} \quad \dot{y}]^T$. Let x, y be the positions of the pixels along x-axis and y-axis, and $\dot{x}, \dot{y}$ be their respective velocities.

Algorithm 4 Proposed Motion Dynamics Model

```
1: procedure MODEL–1($\rho_k^\star, \rho_{k-1}^\star, TrackOK, \mathbf{y}_k, \mathbf{y}_{k-1}$)
2:     Compute: $\mathbf{d} = \mathbf{y}_k - \mathbf{y}_{k-1}$
3:     Compute: $d_x = |d_1|$
4:     Compute: $d_y = |d_2|$
5:     if $\rho_k^\star > TrackOK\ AND\ \rho_{k-1}^\star > TrackOK$ then
6:         if $d_1 < 0$ then
7:             $\mu_x = -\frac{\sigma_x}{2} \times (1 - exp(-d_x/K1))$
8:         else
9:             $\mu_x = \frac{\sigma_x}{2} \times (1 - exp(-d_x/K1))$
10:        if $d_2 < 0$ then
11:            $\mu_y = -\frac{\sigma_y}{2} \times (1 - exp(-d_y/K2))$
12:        else
13:            $\mu_y = \frac{\sigma_y}{2} \times (1 - exp(-d_y/K2))$
14:    else
15:        Assign : $\mu_x = 0$
16:        Assign : $\mu_y = 0$
17:    return $\mu_x, \mu_y$
```

In Algorithm 4, $\rho_k^\star$ and $\rho_{k-1}^\star$ be the Bhattacharyya similarity coefficient at k-th and $(k-1)$-th stage computed using (5.18) at $\mathbf{y}_k$ and $\mathbf{y}_{k-1}$ respectively. Since image frames are 2D, distance $\mathbf{d} = \begin{bmatrix} d_1 \\ d_2 \end{bmatrix}$. $TrackOK$ is the threshold and $K1$, $K2$ are the two normalizing constants so that $\mathbf{d} \in [0\ 3]$ in (5.21). Hence, the proposed model can further reduce the chance of sample degeneracy.

Generally the velocity is represented by Gaussian noise $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\sigma})$ where $\boldsymbol{\mu}$, is the mean value and

$\boldsymbol{\sigma}$ is the standard deviation. Generally $\boldsymbol{\mu}$ is set a 0 while $\boldsymbol{\sigma}$ is tuned by the user according to the estimated velocity of the tracked object. The probability of generating a velocity $\mathbf{v}$ is given by:

$$p(\mathbf{v}) = K_1 e^{-\frac{(\mathbf{v}-\boldsymbol{\mu})^2}{2\boldsymbol{\sigma}^2}} \tag{5.23}$$

where $K_1 = \frac{1}{\boldsymbol{\sigma}\sqrt{2\pi}}$. If velocity of the object or the particles is less, a lower value of $\boldsymbol{\sigma}$ will work. But, if velocity is large, from (5.23), it can be easily inferred the increasing $\boldsymbol{\sigma}$ will not improve the probability of generating a high velocity much. Also the velocity of the object keeps changing with time along with its direction. Mathematically, say $\boldsymbol{\sigma}$ is fixed and maximum velocity $v = \sigma$ has to be produced. Hence, K_1 is also fixed. Probability of producing velocity $\mathbf{v} = \boldsymbol{\sigma}$ at $\boldsymbol{\mu} = \mathbf{0}$ is $p_1(\mathbf{v}) = K_1 e^{-0.5} = 0.606K_1$. But $\boldsymbol{\mu} = \frac{\boldsymbol{\sigma}}{2}$, Probability of producing velocity $\mathbf{v} = \boldsymbol{\sigma}$ is $p_2(\mathbf{v}) = K_1 e^{-0.125} = 0.882K_1$. So $p_2 > p_1$. Now, in case the velocity changes to $\mathbf{v} = -\boldsymbol{\sigma}$ suddenly, probability of producing velocity $\mathbf{v} = -\boldsymbol{\sigma}$ with $\boldsymbol{\mu} = 0$ is $p_3(\mathbf{v}) = K_1 e^{-0.5} = 0.606K_1$, but with $\boldsymbol{\mu} = \frac{\boldsymbol{\sigma}}{2}$, $p_4(\mathbf{v}) = K_1 e^{-1.125} = 0.324K_1$. So p_4 is not too small to produce velocity in the negative direction suddenly. Even if the number of negative velocity generated is less, the proposed particle filter with resampling algorithm (explained in the next section) will track the object immediately. The variable $\mathbf{d}$ is obtained by finding the distance between the same object in the current and previous frame and then normalizing the distance into [0 3] by diving it using the maximum assumed distance. Hence, the proposed model can further reduce the chance of sample degeneracy.

5.6 Proposed Resampling Algorithm

Resampling is essential to avoid sample degeneracy among the particles, which occurs after few iterations. With SIR-PF (Sampling Importance Resampling Particle Filter), the systematic resampling algorithm is used as described in [19] where low weight particles are discarded while higher weight particles are replicated. This again reduces the diversity among the particles causing sample impoverishment among them. Also, in case of clutter, there is a chance that all the particles may merge at a wrong point as there is no further searching in the algorithm. In this work, a new resampling algorithm is proposed. This proposed resampling algorithm reduces the degeneracy problem and also sample impoverishment. Also, if due to error in motion dynamics, tracking is lost, this resampling algorithm can immediately track the lost object. This proposed resampling algorithm utilizes both systematic resampling [19] and bat algorithm. Bat algorithm is used because it has two basic moves: *exploration* and *exploitation*. The key philosophy of the approach is that these two moves are used in two different situations as explained in this section.

5.6.1 Bat Algorithm

In 2010, Xin-She Yang introduced nature inspired Bat Algorithm [50]. Bat algorithm simulates the behavior of a group of micro bats reaching an optimum location. The key philosophy of the algorithm is that all the bats track the current best solution randomly. While tracking the current best solution, if any other better than the current best solution is found, all the bats will move toward that new better solution. In general, this is the way the micro bats will reach the optimum location. While moving, the bats will randomly switch between the two types of jumps or moves.

Let i be the particle or bat and t be the iteration. Multiplying factor $\beta \in [0\ 1]$ and f_i be the random frequency of i-th bat within the range $[f_{min}\ f_{max}]$. The type-1 jump occurs using the following equations:

$$f_i = f_{min} + (f_{max} - f_{min})\beta \tag{5.24}$$

The velocity of i-th bat is updated as:

$$\mathbf{v}_i^{t+1} = \mathbf{v}_i^t + (\mathbf{s}_i^t - \mathbf{s}^\star)f_i \tag{5.25}$$

$$\mathbf{s}_i^{t+1} = \mathbf{s}_i^t + v_i^{t+1} \tag{5.26}$$

where $\mathbf{s}_i^t$ be the location of the i-th bat at t-th instant or iteration; $\mathbf{s}^\star$ is the *present* best global solution; this $\mathbf{s}^\star$ gets updated as the bats converge. And multiplying factor $\beta \in [0\ 1]$.

Once exploration is done, the bats are allowed to search in the vicinity of the current location using the type-2 jump as follows:

$$\mathbf{s}_i^{t+1} = \mathbf{s}^\star + \epsilon A^t \tag{5.27}$$

where $\epsilon \in [-1\ 1]$ and A^t is the mean loudness among all the bats at that stage.

Loudness (sound) of i-th bat will vary according to:

$$A_i^{t+1} = \alpha A_i^t \tag{5.28}$$

where, $\alpha \in (0\ 1)$

Pulse-rate will vary according to:

$$r_i^{t+1} = r_i^0[1 - \exp(-\gamma t)] \tag{5.29}$$

where, $\gamma > 0$

The general bat search algorithm with N_B-number of bats is shown in Algorithm 5.

Algorithm 5 Bat Algorithm

```
1: procedure BAT($(b_i, v_i)_{i=1}^{N_B}, \mathbf{s}^\star$)
2:     Initialize $f_{max}$, $f_{min}$, $\beta$, $\alpha$, $\gamma$, $r_i^0$, $r_i$ and $A_i$ for $i = 1...N_B$
3:     Assign : $t = 0$
4:     while $t < Maximum\ iterations$ do
5:         Generate frequency $f_i$ using (5.24) for $i = 1...N_B$
6:         Generate new solutions using (5.25) and (5.26)
                   ▷ $rand(min, max)$ is uniformly created random number in between (min,  max)
7:         if $rand(0,1) > r_i$ then
8:             Update $\mathbf{s}^\star$ from the current best solution
9:             Generate new solution using using (5.27)
10:        if $rand(0,1) < A_i\ AND\ f(b_i) < f(b_{best})$ then
11:            Accept the new solutions
12:            Update $A_i$ and $r_i$ using (5.28) and (5.29)
13:        Find the current best among all the bats and update $\mathbf{s}^\star$
14:        $t = t + 1$
15:    return $\mathbf{s}^\star$
```

The Bat algorithm converges much faster, especially at initial stages. Convergence analysis of bat algorithm is provided in [50].

5.6.2 Proposed Resampling Algorithm

Unlike the conventional bat algorithm [50], in the modified algorithm, each bat (or particle) moves toward its *own best* rather than toward the global best. In this way each bat revolves around its own best which gets updated when a better solution is found.

Let i be the particle or bat and t be the iteration. Multiplying factor $\beta \in [0\ 1]$ and f_i be the random frequency of i-th bat within the range $[f_{min}\ f_{max}]$:

$$f_i = f_{min} + (f_{max} - f_{min})\beta \tag{5.30}$$

The velocity of i-th bat is updated as:

$$\mathbf{v}_i^{t+1} = \mathbf{v}_i^t + (\mathbf{s}_i^t - \mathbf{s}_i^\star)f_i \tag{5.31}$$

where $\mathbf{s}_i^t$ be the state of the i-th bat or particle in t-th iteration and $\mathbf{s}_i^\star$ be the most recent global optimum solution of the i-th bat or particle chosen from all the present and past moves.

The type-1 jump for the next $(t+1)$-th move:

$$\mathbf{s}_i^{t+1} = \mathbf{s}_i^t + \mathbf{v}_i^{t+1} \tag{5.32}$$

where $\mathbf{s}_i^t$ be the state of the i-th bat or particle in t-th iteration and $\mathbf{s}_i^{t+1}$ be the state of the i-th bat or particle in $(t+1)$-th iteration

Similarly type-2 jump for the next $(t+1)$-th move:

$$\mathbf{s}_i^{t+1} = \mathbf{s}_i^\star + rand(min1, max1) \tag{5.33}$$

where $min1,\ max1$ be the lower and upper limit of the uniformly generated random number.

Let N_s be the number of particles or bats. The steps of the proposed resampling algorithm are shown in Algorithm 6.

The proposed algorithm modifies the SIR-PF (Sampling Importance Resampling Particle Filter) just to move the particles to better position. In general, particle filter has three steps: generation of particles, computation of weight and resampling the particles. Similarly, the ideal bat algorithm can also be divided into three stages namely Stage-I, Stage-II and Stage-III as follows:

Stage-I: This stage is the first stage. Loudness is set as 1 and pulse-rate as 0. Bats are generated, and the best bat is picked up. Let all the bats move toward that best bat location. If any other best location found, update the new best location. Relocate the bats around this best located bat (5.33) as pulse-rate is 0. Accept this new location of the bats if better than previous ones as loudness is set as 1.

Stage-II: This stage includes all the stages after the first and before the last stage. Loudness is decreasing and pulse-rate is increasing. New bats are generated near the best with some changing probability. New locations are also accepted with some changing probability. If new locations are not accepted, the distance between their old location and the updated best is used cumulatively to increase search space. This slowly causes zooming.

Algorithm 6 Proposed Resampling Algorithm

```
1: procedure RESAMPLING($(\mathbf{x}_k^i, w_k^i)_{i=1}^{N_s}, N_s, r0$)
2:     Initialize $f_{max}$, $f_{min}$, $\beta$, $r_i = r0$ for $i = 1...N_s$
3:     Assign : $\mathbf{s}_i^\star = \mathbf{x}_k^i$ for $i = 1...N_s$
4:     Assign : $t = 0$
5:     Assign : $\mathbf{v}_i^t = 0$ for $i = 1...N_s$
6:     Assign : $\mathbf{s}_i^t = \mathbf{x}_k^i + rand(min, max)$ for $i = 1...N_s$  ▷ Particles are slightly deviated from the
       actual state by uniformly distributed random numbers in the range $min, max$
7:     while $t < Maximum\ iterations$ do
8:         for $i = 1$ to $N_s$ do
9:             if $rand(0,1) > r_i$ then
10:                $\mathbf{s}_i^{t+1} = \mathbf{s}_i^\star + rand(min1, max1)$
11:            else
12:                Generate new solution $\mathbf{s}_i^{t+1}$ using (5.30), (5.31), (5.32)
           Compute: $w_{new}^i = p(\mathbf{z}_k^i|\mathbf{s}_i^{t+1})$ using (5.20)
13:            if $w_k^i < w_{new}^i$ then
14:                Update : $w_k^i = w_{new}^i$
15:                Update : $\mathbf{x}_k^i = \mathbf{s}_i^{t+1}$
16:                Update : $\mathbf{s}_i^\star = \mathbf{s}_i^{t+1}$
17:            Assign : $\mathbf{s}_i^{t+1} = \mathbf{s}_i^t$
18:        $t = t + 1$
19:    if $r0 = 1$ then
20:        Systematic Resample : Discard particles with low weight and replicate particles with
       higher weight as given in [19]
21:    return $(\mathbf{x}_k^i, w_k^i)_{i=1}^{N_s}$
```

Stage-III: This stage is the last stage. Ideally loudness is set as 0 and pulse-rate as 1. Neither bats are generated around best (5.33) nor locations are updated. Only the best is updated and the velocity of each bat is cumulatively added up. This causes only exploration.

To summarize, in case of conventional bat algorithm, at the initial stage, due to both exploration and exploitation, all bats move toward the same target and all the better new solutions are accepted. As number of stage increases, chance of exploration increases while the chance of exploitation decreases; also the probability of accepting new better solutions also decreases. In the final stage, no exploitation occurs. Only exploration occurs and none of new solutions are accepted; only the global best value is updated as usual. Hence, it can be visualized that, at the initial stage the bats tend to converge, and then the bats keeps searching around the converged location and this search space increases with the number of stages.

As shown in Algorithm 6, this bat algorithm is modified to reduce the computational complexity and sample impoverishment. Particles are generated with a motion dynamics following the real object and there is some probability that particles will follow the real object. Then Bhattacharyya metric (5.18) at deterministically estimated mean location says if the particles are converged. If converged, particles are just moved to better location using (5.33). This increases the convergence and reduces sample impoverishment because the best values are individual for each bat, not a common one. When the object is lost (detected by Bhattacharyya metric (5.18), it cannot be decided which particle is near the lost or occluded object. Hence, it is assumed that each particle is equally probable to be near the object. Now the Stage-III described above is directly used to explore the region by each particle independently. After exploration, once the best particle is found (the accuracy of finding the best particle is further increased by the proposed anti-clutter object model described in 5.4), the particles are converged using systematic resampling described in [19]. Thus the modified bat algorithm uses either exploitation or the exploration move. This reduces the computational cost and sample impoverishment. To summarize, when object is lost either due to clutter or complete occlusion, exploration move – type-1 jump (5.32) – gets activated. Otherwise, type-2 jump (5.33) simply rearranges the particles.

5.6.3 Computational Cost Analysis

It can be shown that the proposed resampling algorithm is computationally less expensive compared to the bat algorithm. Analysis of the computational complexity of the proposed resampling algorithm and the bat algorithm is carried out. Let c_1 be the average cost of computing all the stages of type-1 jump (including frequency, cumulative velocity computation, etc.), c_2 be the average cost of evaluating the likelihood, c_3 be the average cost of rearranging the bats and accepting the best value, c_4 be the average cost of increase of pulse-rate and decrease of loudness, c_5 be the average cost of computing all the stages of type-2 jump, c_6 be the average cumulative cost for resampling, N_s be the number of bats and K_s be the number of iterations. It can be shown that c_5 (type-2 jump) is computationally more efficient than c_1 (type-1 jump) as type-1 jump contains some extra computation to compute frequency and velocity, which are not in type-2 jump. Hence, the computational complexity of the bat algorithm in every frame: every frame:

$$Cost_{BA} = \sum_{i=1}^{N_s}\sum_{t=1}^{K_s}(c_1 + p_1(c_2 + c_5) + c_2 + p_2c_4 + c_3) \tag{5.34}$$

Now for the proposed algorithm, the computational cost in every frame:

$$Cost_{PRO} = \sum_{i=1}^{N_s}\sum_{t=1}^{K_s}(p_1c_5 + (1 - p_1)(c_1 + c_6) + c_2 + c_3) \tag{5.35}$$

where, when $p_1 = 1$, type-2 jump is selected, otherwise type-1 jump is selected.It is observed that the sum of the computational cost of likelihood and rearranging the bats and accepting the best value is far more than other costs, as likelihood model accesses more memory locations and perform more arithmetic and logical operations. Hence, (5.34) and (5.35) can be reduced as, respectively:

$$Cost_{BA} \approx \sum_{i=1}^{N_s}\sum_{t=1}^{K_s}((p_1 + 1)c_2 + c_3) \tag{5.36}$$

$$Cost_{PRO} \approx \sum_{i=1}^{N_s}\sum_{t=1}^{K_s}(c_2 + c_3) \tag{5.37}$$

where p_1 can be 0 or 1 randomly throughout the frames. Hence, approximately the average computational cost of the proposed bat inspired resampling algorithm is less than that of the conventional bat algorithm throughout the entire frames. So it is better to use the proposed resampling algorithm rather than using the conventional bat algorithm directly for resampling the particles.

5.7 Proposed Template Update Mechanism

As time increases, many changes in the histogram of the template takes place. As for example, changes can be in scale, shape and illumination mainly. Hence, to keep tracking the object accurately, the template needs to be updated.

5.7.1 Related Works

One major problem in template update is the presence of occlusion. If a template is completely updated under prolonged occlusion, it will loose tracking the right object. Hence, templates cannot be updated when complete occlusion occurs for prolonged time. So literature[16] turns off tracking when number of outliers in the template pixel increases beyond a certain percentage; this outliers are detected using Mahalanabis distance formula between the measured and predicted pixels, and the comparison is made pixel to pixel. On the other hand, [11] updates the template by simply copying the past template if foreground is more prominent than background and tracking is correct. Thus, in the works [52, 53, 54, 11], different template update mechanisms are described. As discussed earlier, [11] decides the template update policy depending on foreground and background pixel count; then a particular fragment is updated by directly replacing the current model. While [16] uses Kalman filter in a novel way to update the template.

5.7.2 Proposed Template Update Algorithm

As time increases, many changes in the histogram of the template may take place. As for example, changes may occur in scale, shape and illumination. Hence, to keep tracking the object accurately, the template needs to be updated. One problem in template update is the presence of occlusion. If a template is updated under prolonged occlusion, it may loose tracking the right object. Hence, a template update algorithm is proposed which can work in presence of prolonged complete occlusion. Also this algorithm includes a mechanism to adaptively change the range of histogram to increase the discrimination among the colors without increasing sparsity in the histogram. The update equation is given as:

$$qu_k = (1-\alpha)qu_{k-1} + \alpha pu_k; \tag{5.38}$$

where qu_{k-1} is the reference histogram inherited from $(k-1)$-th stage and used for tracking objects in k-th stage. pu_k is the target histogram formed at present tracked location $\mathbf{y}_k$ in k-th stage. pu_k contains the new information of the tracked object which may be different from that of qu_{k-1}. Now it has been observed that, if pu_k is directly assigned to qu_k, the change in qu_k (which is used for $(k+1)$-th stage) can be abrupt. This abrupt update is not desirable in many cases as it may cause drift. Hence, (5.38) will help smooth this abrupt change using a smoothing parameter α. This will stop drifting the tracking, which arises in direct assignment. Here $\alpha \in [0\ 1]$. Generally less weight is given to pu_k.

In the proposed algorithm, the range of histogram can be chosen by setting $fullscale$ as true or false. As the template is not getting updated immediately, Bhattacharyya metric (5.18) can be used to detect occlusion, rather than comparing pixel-by-pixel. The proposed mechanism is described in Algorithm 7.

In Algorithm 7, $xbin$, $ybin$, $zbin$ are the numbers of joint histogram bins of R, G and B channel of

Algorithm 7 Template Update Algorithm

```
1: procedure UPDATE($qu_k, qu_{k-1}, pu_k, \rho_k^\star, \rho_{k-1}^\star,$
       $Updating, \alpha, fullscale, xbin, ybin, zbin, \mathbf{y}_k, \mathbf{y}_{k-1}$)
2:     Initialize Threshold Values $UpdateTH$, $simiTH$
3:     Compute : $qu_k = (1-\alpha)qu_{k-1} + \alpha pu_k$
4:     Compute $\rho^\star$ using $qu_0$, $qu_k$ and (5.18)
5:     if $\rho_k^\star > UpdateTH\ AND\ \rho_{k-1}^\star > UpdateTH\ AND\ \rho^\star > simiTH\ AND\ Updating == true\ AND\ fullscale == true$ then
6:     else
7:         Assign : $qu_k = qu_{k-1}$;
                  ▷ Or can be initialized to $qu_0$ depending on application and occlusion type
8:     if $\rho_k^\star > UpdateTH\ AND\ \rho_{k-1}^\star > UpdateTH\ AND\ \rho^\star > simiTH\ AND\ Updating == true\ AND\ fullscale == false$ then
9:         Choose an area around the location $\mathbf{y}_k$ at $k$-th stage
10:        Compute Maximum Value in R-channel $maxR$
11:        Compute Maximum Value in G-channel $maxG$
12:        Compute Maximum Value in B-channel $maxB$
13:        Compute : $xstep = (maxR + threshold) \div xbin$
14:        Compute : $ystep = (maxG + threshold) \div ybin$
15:        Compute : $zstep = (maxB + threshold) \div zbin$
16:        Compute $pu_k$ using $xstep$, $ystep$, $zstep$, $\mathbf{y}_k$ and others at $k$-th stage
17:        Compute : $qu_{k-1}$ using this new $xstep$, $ystep$, $zstep$, $\mathbf{y}_{k-1}$ and others at $(k-1)$-th frame
18:        Compute : $qu_k = (1-\alpha) * qu_{k-1} + \alpha * pu_k$
19:        Compute : $qu_0$ with this new $xstep$, $ystep$, $zstep$ for the $first$ frame
20:        Compute $\rho^\star$ using $qu_0$, $qu_k$ and (5.18)
21:        if $\rho^\star > simiTH$ then
22:        else
23:            Assign : $qu_k = qu_{k-1}$;
24:    return $qu_k$                         ▷ $qu_k$ is be used to search in the $(k+1)$-th frame
```

the 4D probabilistic histogram respectively including $block$ (number of blocks the template is divided into) which are all fixed. qu_0 is the reference histogram in the initial frame. If $\rho^\star < simiTH$, severe drifting takes place, and so the reference histogram is assigned to the previous or initial one. Selection of $threshold$ is application dependent. If $fullscale = false$ then, in the first frame also, the step size of the joint histogram should be set as:

$$\begin{aligned} xstep &= (maxR + threshold) \div xbin \\ ystep &= (maxG + threshold) \div ybin \\ zstep &= (maxB + threshold) \div zbin \end{aligned} \tag{5.39}$$

where $maxR$, $maxG$, $maxB$ are the maximum values of the R, G and B planes of the frame respectively, and $threshold$ is a user defined value. Hence, the proposed algorithm can update the template under severe prolonged occlusion keeping the tracking accurate in presence of large variation of scale, shape and illumination.

5.8 Summary

In most of the cases, importance density of particle filters is taken as state transition prior for simplicity. Since transition prior does not have the information about present observation, particles may stay in the region where likelihood is less. The most optimal distribution minimizes the variance and variance reduces with increasing number of particles. Ideally, as the number of particles reaches infinity, the weights reach the true posterior density. But, although using a large number of particles will reduce variance, it will increase the computational cost. This is the main disadvantage of particle filtering. In practice, a trade-off between the computational cost and the variance is made while selecting the number of particles. As number of particles increases, variance reduces, but computational cost increases. In SIR-PF (Sampling Importance Resampling Particle Filter), particles are generated following the dynamics, and then, using the resampling algorithm, all low weighted particles converge toward the best particles ideally; but these best particles may not be at the real best location due to many reasons. On the other hand, the original bat algorithm converges in the beginning, and then starts exploring the state space; it does not follow the dynamics. Since all bats move toward the frequently changing global best, there may be a chance that Bats move toward the clutter. Hence, to increase search space more bats are required. Now the proposed algorithm first follows the dynamics of the tracked object, and then searches for the best locations, and finally the particles will converge; this increases the probability of increasing variance around the much better locations than that of the SIR-PF (Sampling Importance Resampling Particle Filter) at the cost of some extra computation. As each particle (can be considered as a bat also) moves towards its own best, it increases the search space by making it more robust against occlusion and clutter. Hence, although the overall computational cost of the proposed algorithm is more than that of the SIR-PF (Sampling Importance Resampling Particle Filter) and the bat algorithm, the proposed algorithm takes less number of particles to track accurately as compared to SIR-PF (Sampling Importance Resampling Particle Filter) and bat algorithm. The proposed algorithm is also compared with other state-of-the-art algorithms.

The proposed algorithm eliminates two major problems in particle filtering : sample degeneracy as well as sample impoverishment problem. The proposed particle filter has been modified to reduce the degeneracy among the particles. This has been tested in different test videos and has been found to be working satisfactorily. Hence, the proposed algorithm can generate good particles having enough diversity among them. The key advantage of using bat algorithm is that the probability of jumps of the bats can be switched between type-1 jump and type-2 jump by changing tuning parameter called pulse-rate. Hence, in the proposed algorithm, when object is moving slow and there is no occlusion present, movement of the particles according to type-2 jump is given higher priority and, when there is complete occlusion, movements of the particles according to type-1 jump is given higher priority by tuning the pulse-rate parameter properly. This feature of bat algorithm is merged with the modified particle filter, which can restrict the movement of the particles from going to a worse location, has been proven robust against many tracking scenario.

In the proposed algorithm, the bat algorithm is modified to be merged with the modified particle filter. The key reason of modifying bat algorithm is that, if it used with SIR-PR (Sampling Importance Resampling Particle Filter) as a replacement of the systematic resampling algorithm, it works slowly, and finally all the particles converge to a single point, reducing the verity among the particles. Hence, the conventional bat algorithm has been modified. In the modified bat algorithm, the exploration and exploitation are separated. It has been observed that exploration moves are suitable when track is lost while exploitation moves are suitable just reach a better location. While exploration, the particles do not track global best. Rather it tracks it's own best; this increases the search space and finally when the better solution is found for each particle separately, the systematic resampling algorithm is used to pickup the best among the better ones and to converge to that best location. Hence, this algorithm is robust in case of complete occlusion. Under occlusion, the object may get lost, but as soon as it comes under view, the tracking algorithm immediately tracks it. The algorithm has been tested with a number of different videos.

Also, it has been showed that the adaptive motion model increases the probability of generating the particles towards the moving object rather than away from the object. It works even better in case the object moves in one direction uniformly. The parameters are defined in such a way so that, if object is moving haphazardly, it can also track. This module has been found to be working satisfactorily.

Finally, the update mechanism along with the dynamic histogram ranging can solve vast illumination change while tracking object. The algorithm has been tested for both fast and slow illumination change. This update mechanism can resolve scale change and blurring effect also.

The algorithm has been tested with a number of challenging video sets over the complete frames. The test results are given in the next chapter. The test results are analyzed both analytically and quantitatively. From the test results it has been found that the proposed algorithm is robust against clutter, illumination fluctuation, scale change, fast object movement, motion blur and complete occlusion. The proposed algorithm is compared with other state-of-the-art algorithms, and found to be working satisfactorily.

Chapter 6

Test Results

6.1 Introduction

This chapter contains test results of the proposed algorithm. To demonstrate the effectiveness of the proposed algorithm, case studies involving clutter, large illumination fluctuation, scale change, fast object movement, motion blur and complete occlusion are made. The quantitative test results of the proposed algorithm are compared with that of SIR-PF (Sampling Importance Resampling Particle Filter), bat algorithm and other state-of-the-art algorithms, and it has been observed that the proposed algorithm works satisfactorily. The tests have been done a number of times for each video and their average values are noted. The tests are run over the complete frames for each video input, and the targets are manually initialized. Error in tracking is measured in terms of Euclidean distance of the tracked centroid and the ground-truth centroid. Testing benchmark [50] defines precision plot for evaluating tracking performance. This plots what percentage of frames is within a given threshold Euclidean distance from the ground-truth. [50] uses 20 pixel distance as the threshold for their evaluation purpose. But, in this work, instead of 20 pixels, 15 pixels are taken as threshold. Then a Success Rate is defined as follows:

$$Success\ Rate = \frac{Positive\ Frames}{Total\ Number\ of\ Frames} \times 100 \tag{6.1}$$

Success Rate is computed as percentage by considering those tracking results (positive frames) where this Euclidean distance between ground truth pixel location and tracked pixel location is within the above mentioned threshold limit, against the total number of frames.

The parameter setting is kept fixed for all the test cases. $8 \times 8 \times 8$ RGB histogram is used throughout the experiments for bat and SIR-PF (Sampling Importance Resampling Particle Filter). For the proposed algorithm, $8 \times 8 \times 8 \times 2$ RGB histogram is used throughout the experiment where 2 is the number of *block*s the template has been vertically divided for utilizing the spatial information in the proposed measurement model. μ_x, μ_y, σ_x and σ_y are taken as 0, 0, 20 and 20 respectively for all the videos. For type-2 jump, $max1 = 2$ and $min1 = -2$. For the proposed algorithm, range of $[f_{min}\ f_{max}]$ is taken as $[-5\ 5]$. Remaining other threshold values are set as: $LostTrackTH = 0.85$, $UpdateTH = 0.85$, $TrackOK = 0.85$, $simiTH = 0.7$, $\alpha = 0.1$, $block = 2$, and $Th = 0.001$. Also $fullscale = true$ and $Updating = true$ is set. All parameters are kept same for all the video sequences except for *Case Study 10.* The tests are carried out a number of times, and their average values are noted. First the proposed algorithm is compared with SIR-PF (Sampling Importance Resampling Particle Filter) and bat algorithm only. With the same initial conditions, the tests are sun for several times for the complete sequence of each video. For SIR-PF (Sampling Importance Resampling Particle Filter), 400 particles are initialized, for bat algorithm, 40 bats with 10 iterations for each one are initialized, and for the proposed algorithm, 20 particles with 5 iterations for each one is set.

6.2 Comparison with SIR-PF and Bat Algorithm

First the proposed algorithm is compared with SIR-PF (Sampling Importance Resampling Particle Filter) and bat algorithm. In this section, the results are listed. Each subsection tabulates the results and then explains the results. The tables list success rate, mean error, standard deviation of error and maximum error, which are computed in average, for each video. Each subsection also contains the marked images for these three algorithms so that the difference among the performance can be easily analyzed. Only *Case Study 10* is different in terms of parameter settings. In other cases, the parameters are kept same. Although the parameters can be tuned separately for each type of video using off-line training data. Thus, if the object to be tracked is known beforehand, the parameters can be tuned using a number of training videos. This will enhance the performance of tracking.

6.2.1 Case Study 1 : Tracking Object Under Illumination Variation, Occlusion and Clutter

In this case study, the video is taken from [1, 2]. The video contains a number of basketball players. Among them, a specific player is tracked. The video has been chosen for the case study because a number of players wearing clothes of the same color, and, out of them, only one specific player has to be tracked. This example proves the robustness of the proposed algorithm against clutter. There is illumination fluction also in some frames, along with occlusion; the illumination change is abrupt. Table. 6.1 lists the results of tracking and Fig. 6.1, Fig. 6.2, Fig. 6.3, Fig. 6.4 shows sample sequences of tracking. It has been seen that the proposed algorithm can track the object better than SIR-PF (Sampling Importance Resampling Particle Filter) and bat algorithm, and there is large difference in the results. As the object (player) moves in one direction in most of the time, the proposed motion dynamics model adaptively shifts the mean of the process noise accordingly; this reduces the degeneracy among the particles generated. Further, in presence of clutter, the background weighted spatial-histogram track better by reducing the effect of clutter which SIR-PF (Sampling Importance Resampling Particle Filter) and bat algorithm do not. Hence, the proposed tracker does not get trapped at any local optima. But, even if it gets trapped at a local optima anyhow, the proposed resampling algorithm helps the proposed tracker come out of the trapped local optima; this is because, when the object gets trapped at the local optima, the weight is obviously still lower than that of the actual location, and so, type-1 jump gets activated and the it searches throughout the entire space. The same thing happens in case of occlusion also. Hence, the algorithm comes out of the trapped or wrong location much faster as compared to SIR-

PF (Sampling Importance Resampling Particle Filter) and bat algorithm. Also maximum error in case of the proposed algorithm is less because of the proposed particle filter which lets the particles move to better places only. The appearance change is captured by the template update mechanism. Hence, in overall, the proposed algorithm works much better than SIR-PF (Sampling Importance Resampling Particle Filter) and bat algorithm.

Table 6.1: Case Study 1

—	**SIR-PF**	**Bat Algo.**	**Proposed Algo.**
Mean Error	78.95	93.44	14.03
Maximum Error	346.69	344.94	161.45
Std. Deviation	103.19	103.51	18.53
Frames per Second	1.79	1.50	1.30
Success Rate	51.17	41.66	86.82

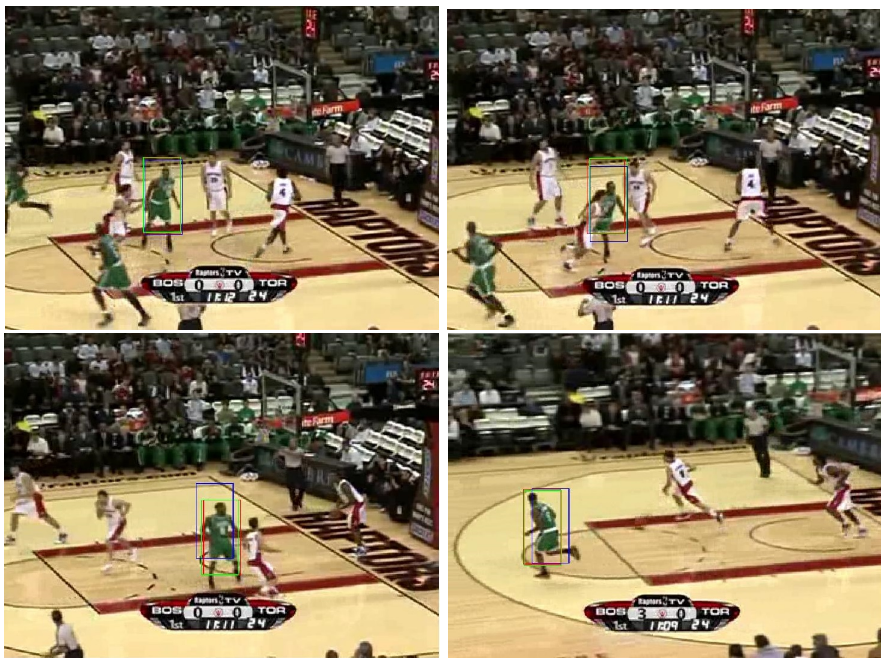

Figure 6.1: Tracking sequence: Frame 5, 15, 38, and 101 are shown (starting from the top left one in clockwise direction). Red box is for SIR-PF, Green box is for bat algorithm and Blue box is for the proposed algorithm for Table. 6.1

6.2.2 Case Study 2 : Tracking Object Under Clutter, Fast Motion and Scale Change

In this case study, the video is taken from [1, 2]. In the video, a runner is tracked under the presence of clutter, fast motion and scale change. The proposed algorithm can track the object, and there is significant difference in the test results among the trackers. Table. 6.2 lists the results of tracking and Fig. 6.5, Fig. 6.6, Fig. 6.7, Fig. 6.8 shows sample sequences of tracking. The accuracy of the proposed algorithm is much higher. Because the object, which is a runner here, runs in one

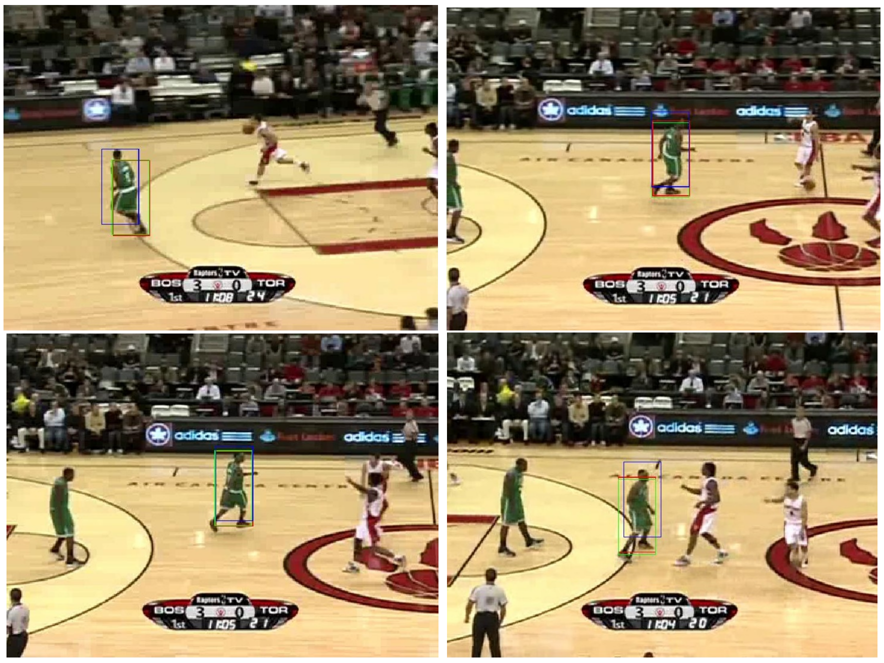

Figure 6.2: Tracking sequence: Frame 114, 193, 210, and 244 are shown (starting from the top left one in clockwise direction). Red box is for SIR-PF, Green box is for bat algorithm and Blue box is for the proposed algorithm for Table. 6.1

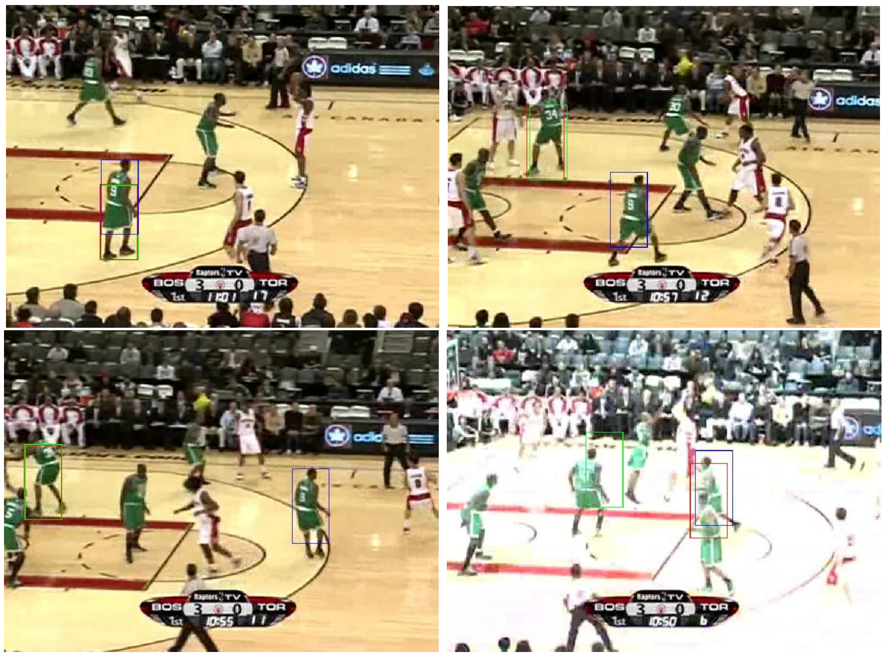

Figure 6.3: Tracking sequence: Frame 334, 460, 506, and 651 are shown (starting from the top left one in clockwise direction). Red box is for SIR-PF, Green box is for bat algorithm and Blue box is for the proposed algorithm for Table. 6.1

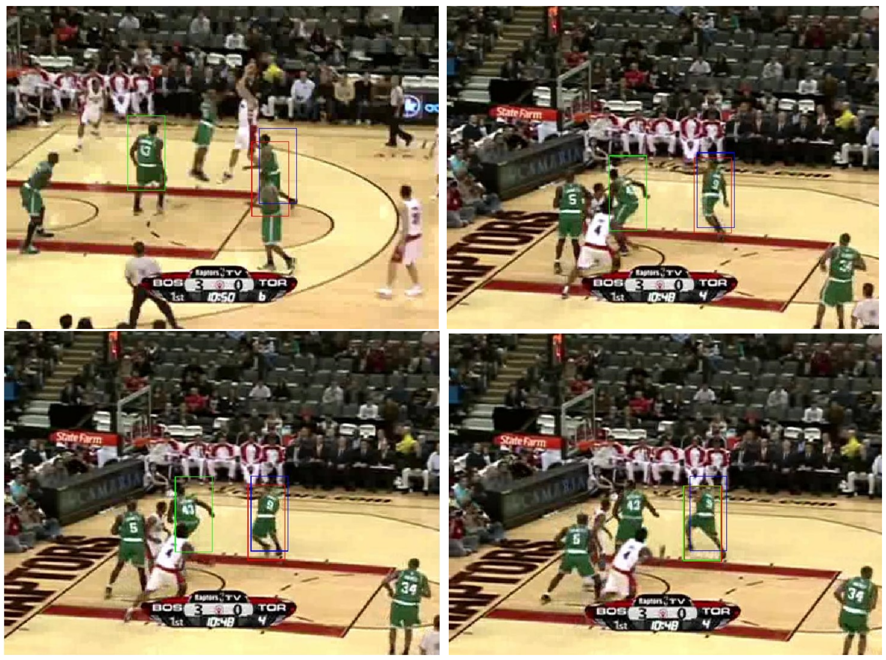
Figure 6.4: Tracking sequence: Frame 652, 719, 722, and 725 are shown (starting from the top left one in clockwise direction). Red box is for SIR-PF, Green box is for bat algorithm and Blue box is for the proposed algorithm for Table. 6.1

direction in most of the time, the proposed motion dynamics model adaptively shifts the mean in the of the process noise right direction accordingly. So the degeneracy among the particles generated is reduced in the proposed algorithm as compared to SIR-PF (Sampling Importance Resampling Particle Filter). When there is no degeneracy, the particles are moved toward some better location using type-2 jump. This may be called as rearrangement of the particles. Further, in presence of clutter, the background weighted spatial-histogram computes weight for each bin, reducing the effect of clutter which SIR-PF (Sampling Importance Resampling Particle Filter) and bat algorithm do not. The proposed tracker does not get trapped at any local optima for this reason, or the chance of such trapping is comparatively lesser. But, again even if it gets trapped at a local optima anyhow, the proposed resampling algorithm helps the tracker come out of the trapped local optima; this is because, when the object gets trapped at the local optima, the weight is obviously still lower than that of the actual location, and so, type-1 jump gets activated and then it searches throughout the space. The same thing happens in case of occlusion also. Hence, the algorithm comes out of the trapped or wrong location much faster as compared to SIR-PF (Sampling Importance Resampling Particle Filter) and bat algorithm. Also maximum error in case of the proposed algorithm is less because of the proposed particle filter which lets the particles move to only better places, and hence, mean error and standard deviation is also less. The appearance change is captured by the template update mechanism. Hence, the proposed algorithm works much better than SIR-PF (Sampling Importance Resampling Particle Filter) and bat algorithm.

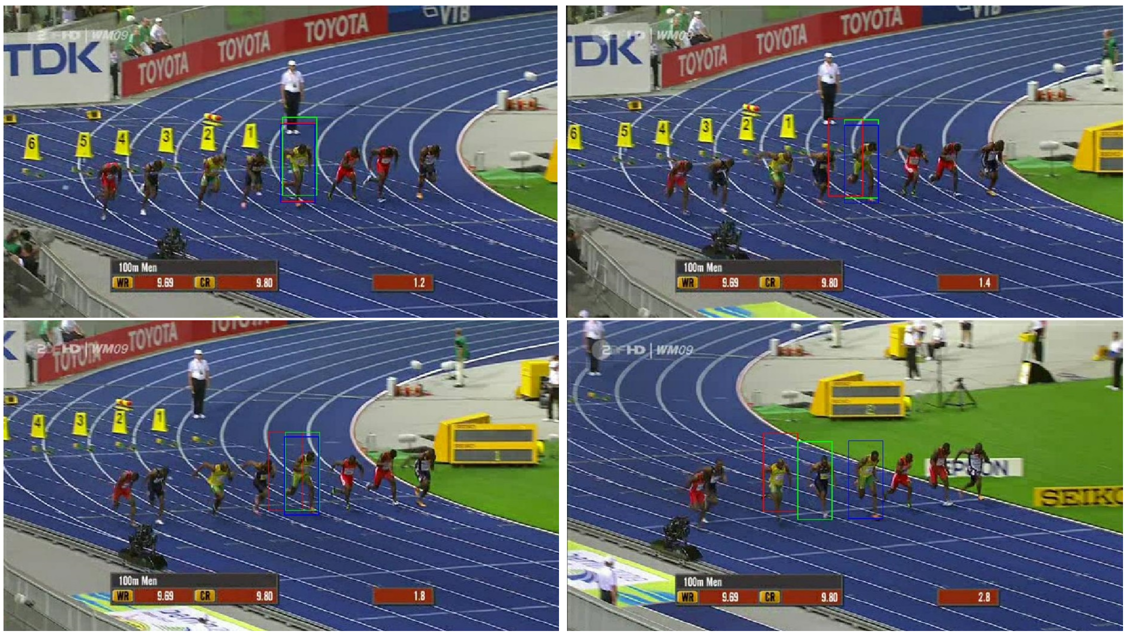
Figure 6.5: Tracking sequence: Frame 27, 31, 42, and 64 are shown (starting from the top left one in clockwise direction). Red box is for SIR-PF, Green box is for bat algorithm and Blue box is for the proposed algorithm for Table. 6.2

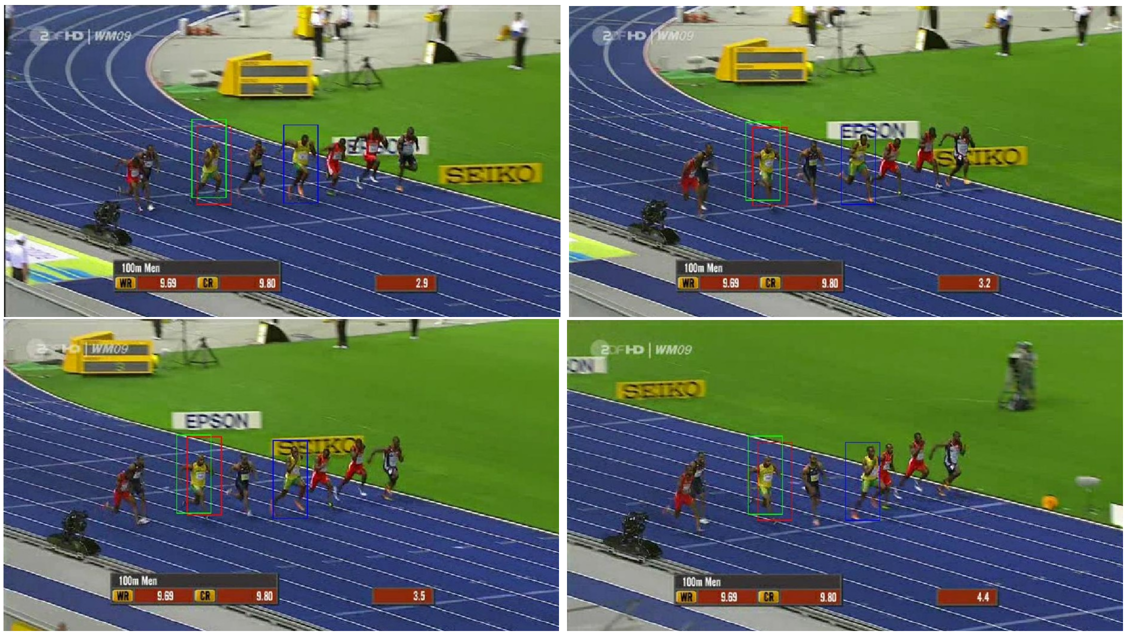
Figure 6.6: Tracking sequence: Frame 67, 74, 83, and 103 are shown (starting from the top left one in clockwise direction). Red box is for SIR-PF, Green box is for bat algorithm and Blue box is for the proposed algorithm for Table. 6.2

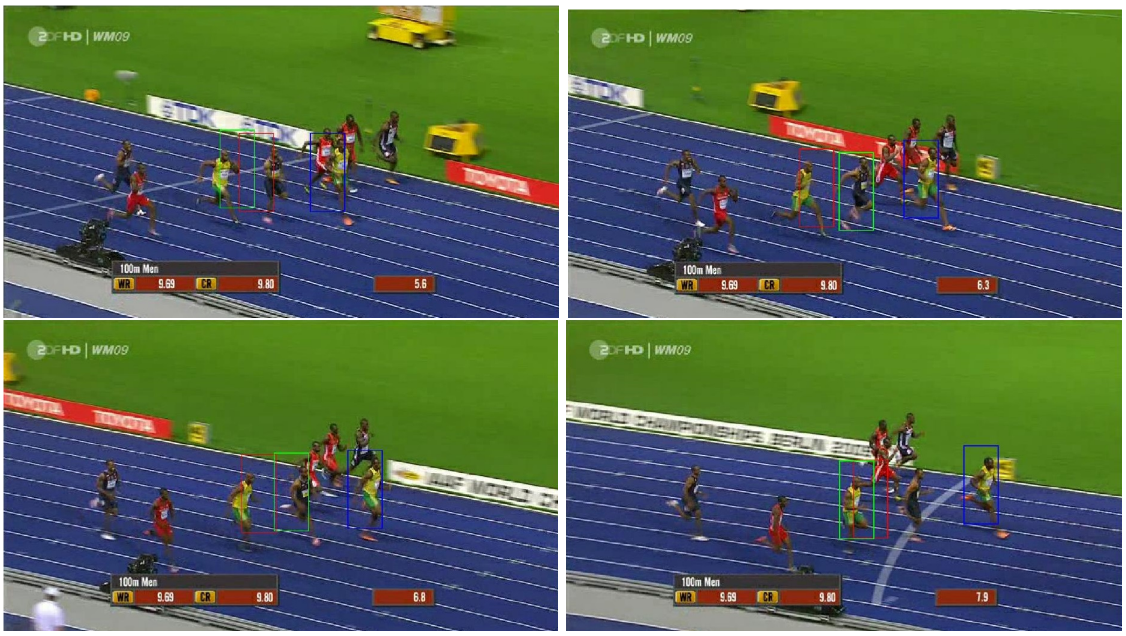

Figure 6.7: Tracking sequence: Frame 133, 148, 161, and 187 are shown (starting from the top left one in clockwise direction). Red box is for SIR-PF, Green box is for bat algorithm and Blue box is for the proposed algorithm for Table. 6.2

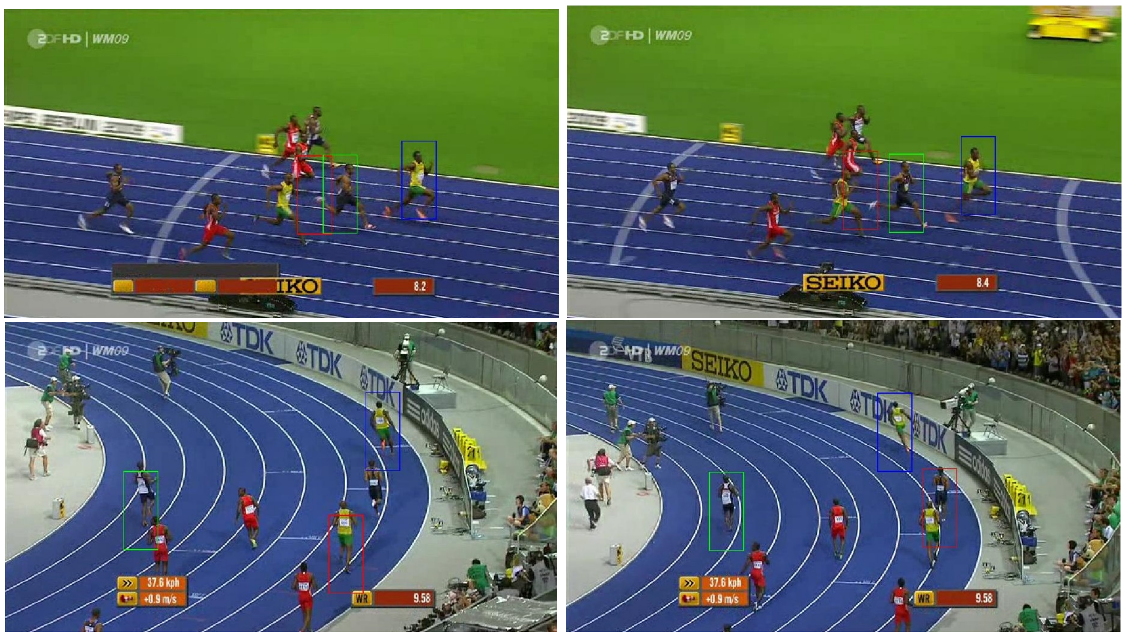

Figure 6.8: Tracking sequence: Frame 195, 200, 330, and 350 are shown (starting from the top left one in clockwise direction). Red box is for SIR-PF, Green box is for bat algorithm and Blue box is for the proposed algorithm for Table. 6.2

Table 6.2: Case Study 2

—	SIR-PF	Bat Algo.	Proposed Algo.
Mean Error	54.66	102.45	12.54
Maximum Error	146.65	253.11	22.67
Std. Deviation	27.58	67.86	16.42
Frames per Second	3.76	2.63	2.80
Success Rate	49.69	45.07	94.77

6.2.3 Case Study 3 : Tracking Object Under Motion Blur, Fast Motion and Scale Change

In this case study, the video is taken from [1, 2]. In the video a boy is tracked who is jumping abruptly. In this video, there is motion blur, fast motion and scale change. In this case study, there is no much difference in results among the trackers. Table. 6.3 lists the results of tracking and Fig. 6.9, Fig. 6.10, Fig. 6.11, Fig. 6.12 shows sample sequences of tracking. While selecting the template, some amount of background information is purposefully kept inside the template to test the robustness of the measurement model along with the tracker. Although SIR-PF (Sampling Importance Resampling Particle Filter) works fine, bat algorithm loses track for a few frames when the object moves away and shrinks and background information predominates inside the template. But, due to its global optimization feature, it again tracks back. This video is chosen mainly to see the performance of the proposed motion dynamics model when the object moves abruptly in any direction rather than moving in one direction. This proposed adaptive model moves the mean value of the process noise in one direction. It has been proven mathematically that, when object changes its direction abruptly, still it can track. Also, as usual, this model reduces the degeneracy problem as compared to SIR-PF (Sampling Importance Resampling Particle Filter). When there is no degeneracy, the particles are moved toward some better location using type-2 jump. Due to the fast motion, if object track is lost, type-1 jump searches in the entire state-space and tracks the lost object immediately. There is no clutter in the test case, but yet the template is chosen in such a way so that it contains some background information which may be regarded as clutter. In this case, the proposed algorithm outperforms as usual. The proposed tracker does not get trapped at any local optima, or the chance of such trapping is comparatively lesser in case of the proposed algorithm. But, again, even if it gets trapped at a local optima anyhow, the proposed resampling algorithm helps the tracker come out of the trapped local optima; this is because, when the object gets trapped at the local optima, the weight is obviously still lower than that of the actual location, and so, type-1 jump gets activated and then it searches throughout the entire space. The same thing happens in case of occlusion also. Hence, the algorithm comes out of the trapped or wrong location much faster as compared to SIR-PF (Sampling Importance Resampling Particle Filter) and bat algorithm. As usual, due the proposed particle filter algorithm, maximum error is also quite lesser than the others. Moreover, in this video, scale change and blurring are also happening together. They are solved by the proposed template update algorithm. As the scale of the object changes, the content of the template also changes accordingly. This changed information is captured and updated by the template update algorithm. Hence, the overall performance of the proposed algorithm is much better.

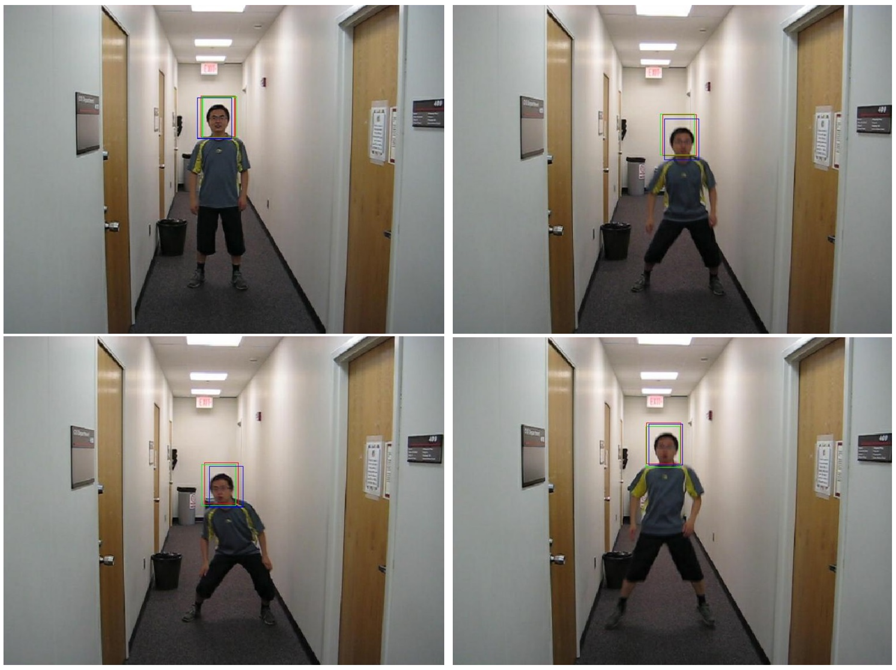

Figure 6.9: Tracking sequence: Frame 25, 80, 88, and 102 are shown (starting from the top left one in clockwise direction). Red box is for SIR-PF, Green box is for bat algorithm and Blue box is for the proposed algorithm for Table. 6.3

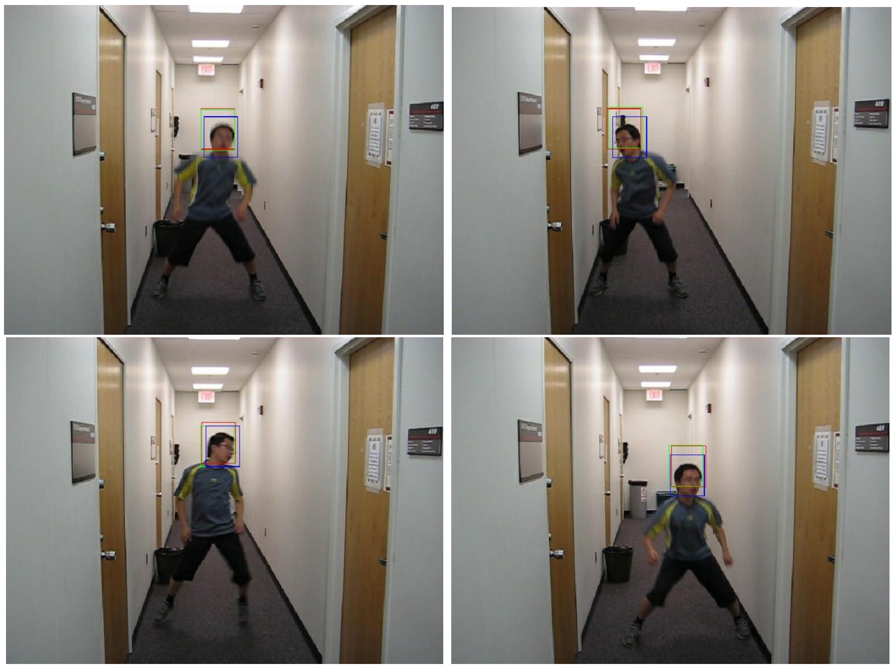

Figure 6.10: Tracking sequence: Frame 105, 121, 130, and 138 are shown (starting from the top left one in clockwise direction). Red box is for SIR-PF, Green box is for bat algorithm and Blue box is for the proposed algorithm for Table. 6.3

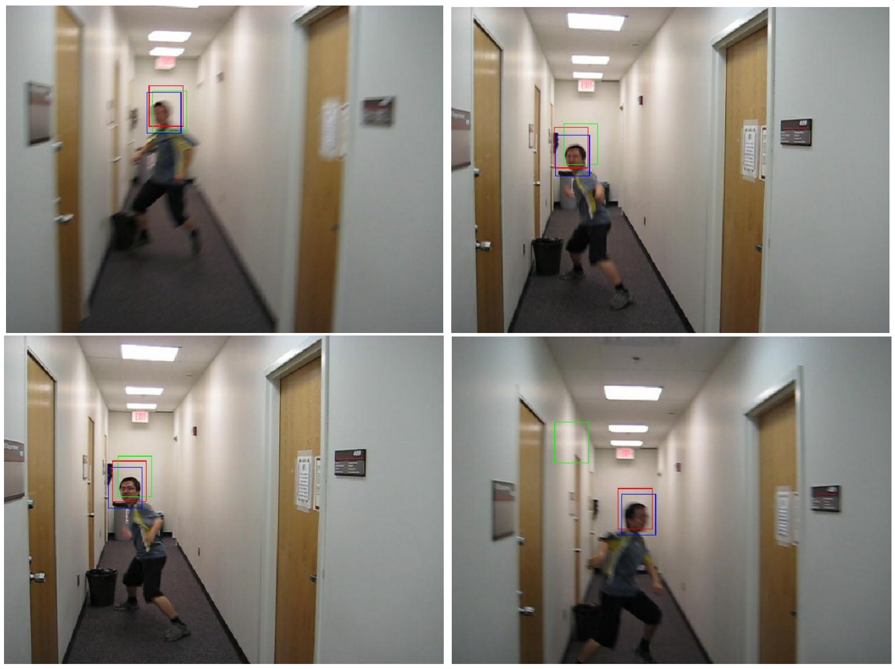

Figure 6.11: Tracking sequence: Frame 266, 272, 273, and 386 are shown (starting from the top left one in clockwise direction). Red box is for SIR-PF, Green box is for bat algorithm and Blue box is for the proposed algorithm for Table. 6.3

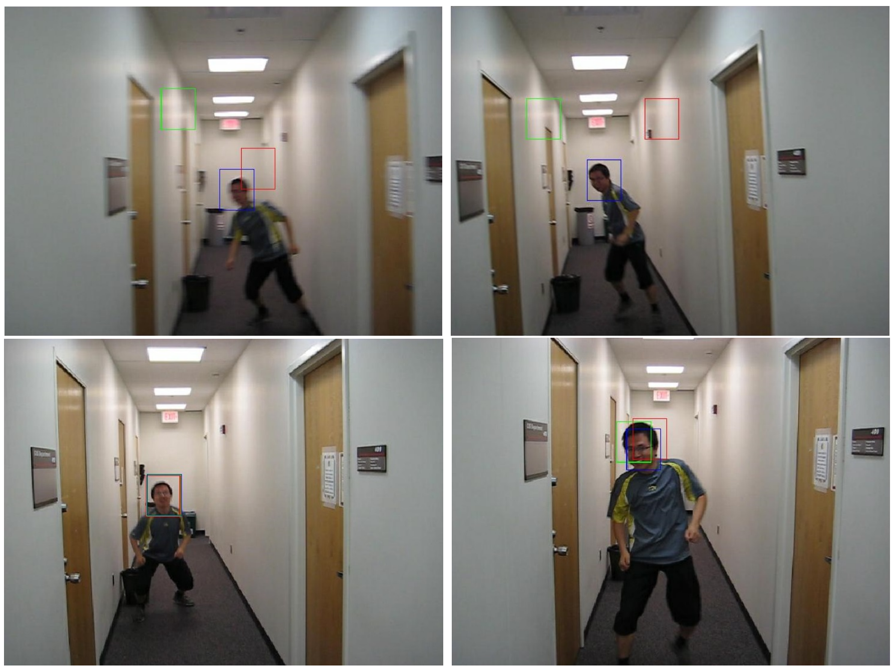

Figure 6.12: Tracking sequence: Frame 401, 409, 556, and 600 are shown (starting from the top left one in clockwise direction). Red box is for SIR-PF, Green box is for bat algorithm and Blue box is for the proposed algorithm for Table. 6.3

Table 6.3: Case Study 3

—	SIR-PF	Bat Algo.	Proposed Algo.
Mean Error	16.08	19.24	10.47
Maximum Error	68.06	139.44	35.79
Std. Deviation	7.11	27.39	6.8
Frames per Second	2.77	1.79	1.73
Success Rate	87.17	85.33	88.67

6.2.4 Case Study 4 : Tracking Object Under Occlusion, Fast Motion and Clutter

In this case study, the video is taken from [1, 2]. In this video, there is mainly motion blur, complete occlusion, background clutter, minute illumination change and scale change. Table. 6.4 lists the results of tracking and Fig. 6.13, Fig. 6.14, Fig. 6.15, Fig. 6.16 shows sample sequences of tracking. SIR-PF (Sampling Importance Resampling Particle Filter) and bat algorithm gets trapped at clutter, while the proposed algorithm can track accurately. When complete occlusion occurs, the track is lost. But the proposed algorithm immediately tracks as soon as the object comes into sight. The proposed algorithm outperforms SIR-PF (Sampling Importance Resampling Particle Filter) and bat algorithm. In this case also the object is moving abruptly. Thus the performance of the proposed motion dynamics model when the object moves abruptly in any direction rather than moving in one direction is observed. This proposed adaptive model moves the mean value of the process noise in one direction. It has been proven mathematically that when object changes its direction abruptly, still it can track. This has been verified in this test case again in presence of occlusion. There is complete occlusion along with clutter. When there is no degeneracy, the particles are moved toward some better location using type-2 jump. Due to occlusion, if object track is lost, type-1 jump searches in the entire state-space and tracks the lost object due to complete occlusion immediately. The clutter problem is solved by the proposed measurement model. Both SIR-PF (Sampling Importance Resampling Particle Filter) and bat algorithm fails in presence of clutter. But the proposed algorithm outperforms as usual. The proposed tracker does not get trapped at any local optima for this reason, or the chance of such trapping is comparatively lesser in case of the proposed algorithm. But, again even if it gets trapped at a local optima anyhow, the proposed resampling algorithm helps the tracker come out of the trapped local optima; this is because, when the object gets trapped at the local optima, the weight is obviously still lower than that of the actual location, and so, type-1 jump gets activated and the it searches throughout the entire space. The same thing happens in case of occlusion also. Hence, the tracker comes out of the trapped or wrong location much faster as compared to SIR-PF (Sampling Importance Resampling Particle Filter) and bat algorithm. In the same way, the loss of track due to fast motion is also solved. As usual, due the proposed particle filter algorithm, maximum error is also quite lesser than the others. The small changes in scale is adapted by the template update algorithm. Hence, the overall performance of the proposed algorithm is much higher.

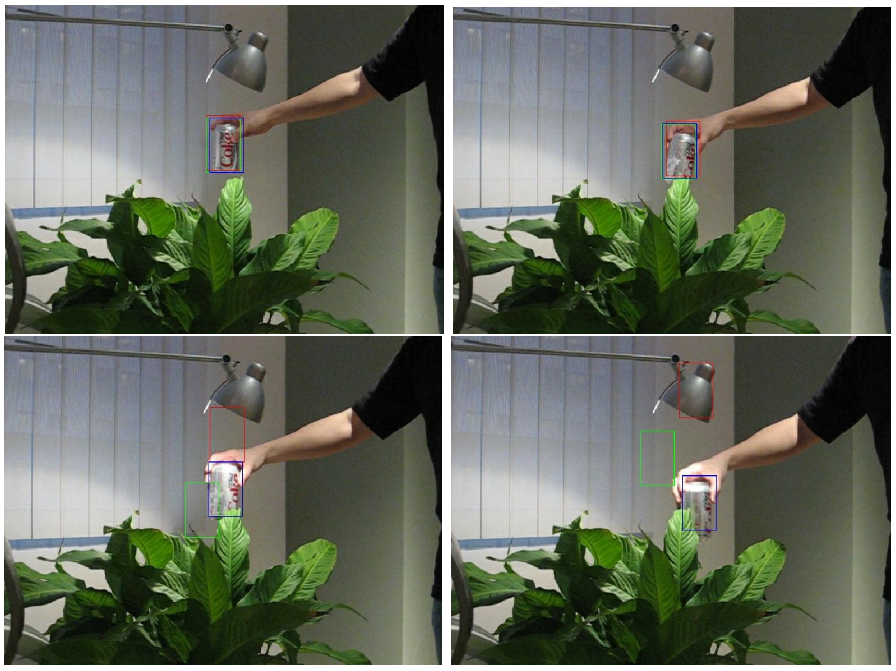

Figure 6.13: Tracking sequence: Frame 5, 13, 21, and 34 are shown (starting from the top left one in clockwise direction). Red box is for SIR-PF, Green box is for bat algorithm and Blue box is for the proposed algorithm for Table. 6.4

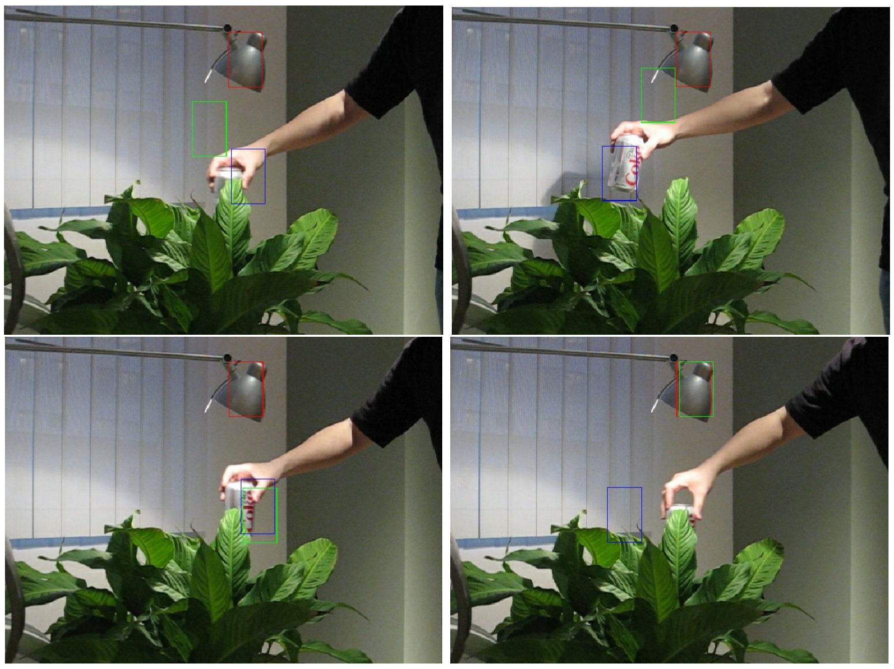

Figure 6.14: Tracking sequence: Frame 40, 48, 74, and 190 are shown (starting from the top left one in clockwise direction). Red box is for SIR-PF, Green box is for bat algorithm and Blue box is for the proposed algorithm for Table. 6.4

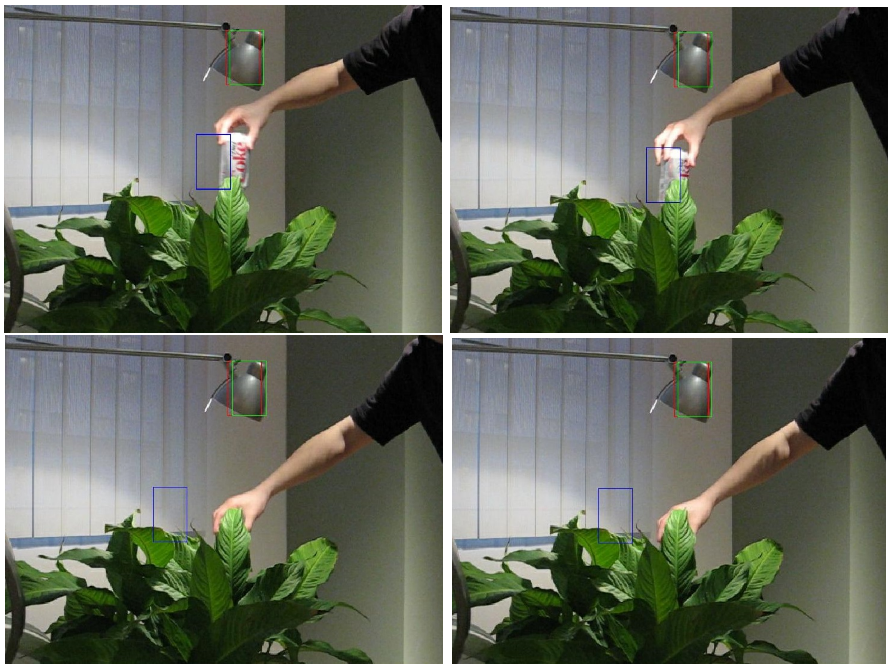
Figure 6.15: Tracking sequence: Frame 200, 222, 257, and 258 are shown (starting from the top left one in clockwise direction). Red box is for SIR-PF, Green box is for bat algorithm and Blue box is for the proposed algorithm for Table. 6.4

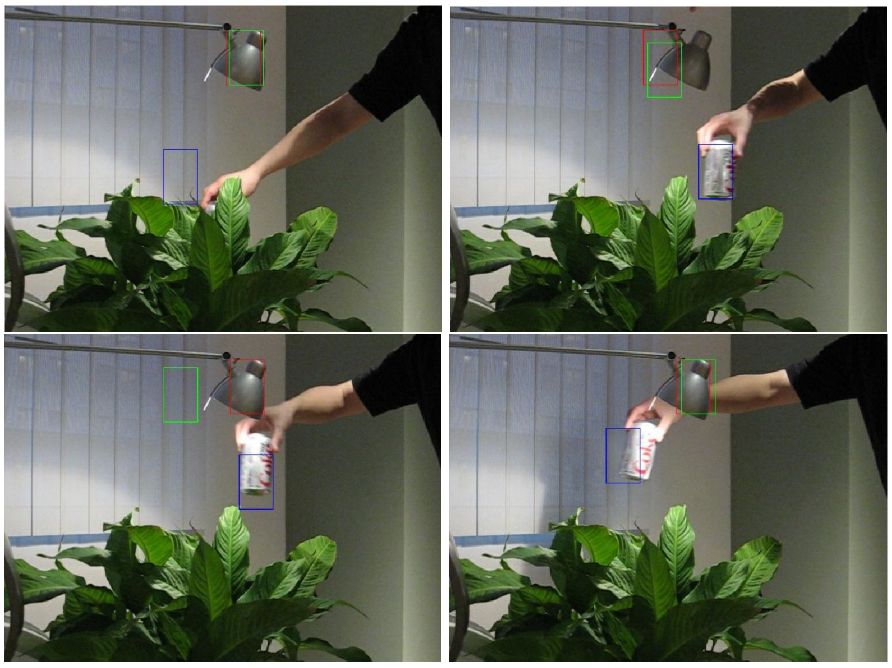
Figure 6.16: Tracking sequence: Frame 265, 278, 285, and 291 are shown (starting from the top left one in clockwise direction). Red box is for SIR-PF, Green box is for bat algorithm and Blue box is for the proposed algorithm for Table. 6.4

Table 6.4: Case Study 4

—	SIR-PF	Bat Algo.	Proposed Algo.
Mean Error	150.79	91.93	81.84
Maximum Error	256.56	257.12	251.96
Std. Deviation	48.11	66.12	70.09
Frames per Second	2.24	1.60	1.38
Success Rate	5.84	7.02	15.12

6.2.5 Case Study 5 : Tracking Object Under Occlusion and Clutter

In this case study, the video is taken from [1, 2]. In the video, a person is tracked who is walking and the person faces short-time occlusion. Table. 6.5 lists the results of tracking and Fig. 6.17, Fig. 6.18, Fig. 6.19, Fig. 6.20 shows sample sequences of tracking. The performance of the proposed algorithm is a bit lesser than that of the SIR-PF(Sampling Importance Resampling Particle Filter) This video is chosen mainly to analyze the performance of all the three algorithms in a simple case. Minor occlusion and clutter are present. The object moves in one direction in most of the time. So the accuracy increases due to the proposed motion dynamics model as usual. It has been proven mathematically that when object moves in one direction, the shifted mean produces better particles. This is tested in this test case mainly. Also, as usual, this model reduces the degeneracy problem as compared to SIR-PF (Sampling Importance Resampling Particle Filter). When there is no degeneracy, the particles are moved toward some better location using type-2 jump. Due to the fast motion, if object is lost, type-1 jump searches in the entire state-space and tracks the lost object due to fast motion immediately. There is no much clutter in the test case, and so none of the trackers gets trapped in any wrong position. In case of occlusion, type-1 jump gets activated and the it searches throughout the entire space. In this test case, all the three algorithms are performing almost same although the proposed algorithm is working little bit better due to the proposed particle filter, adaptive motion model and proposed resampling algorithm. The lesser success rate is due to the template selection. The person is wearing a gray trousers and the background is also grayish. Inside the template, grayish pixels are predominating. So the performance is a bit reduced in case of the proposed algorithm.

Table 6.5: Case Study 5

—	SIR-PF	Bat Algo.	Proposed Algo.
Mean Error	17.67	22.76	16.79
Maximum Error	56.8	67.96	34.22
Std. Deviation	12.34	13.79	7.51
Frames per Second	1.39	1.11	0.97
Success Rate	70.6	55.2	64.8

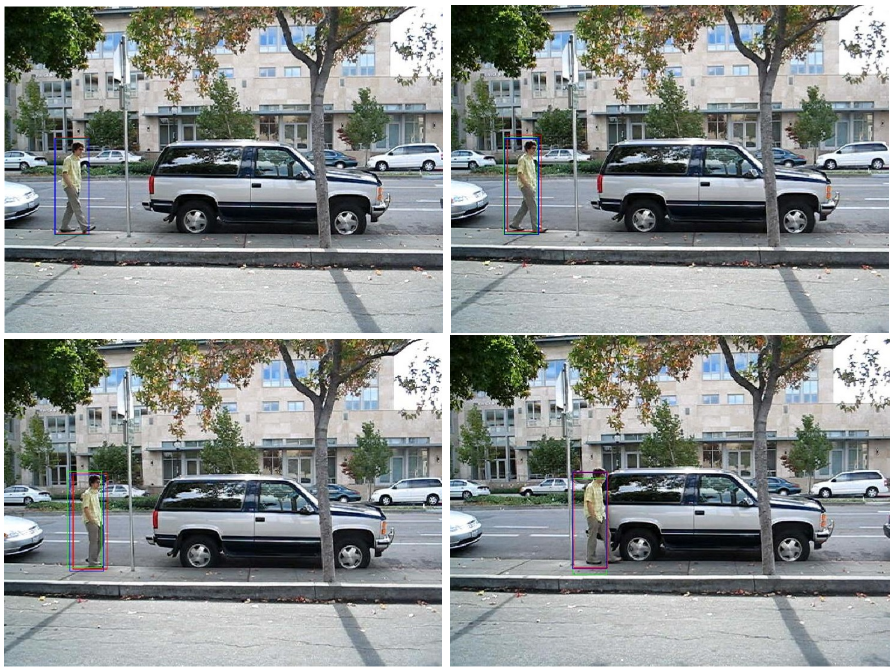

Figure 6.17: Tracking sequence: Frame 3, 6, 11, and 36 are shown (starting from the top left one in clockwise direction). Red box is for SIR-PF, Green box is for bat algorithm and Blue box is for the proposed algorithm for Table. 6.5

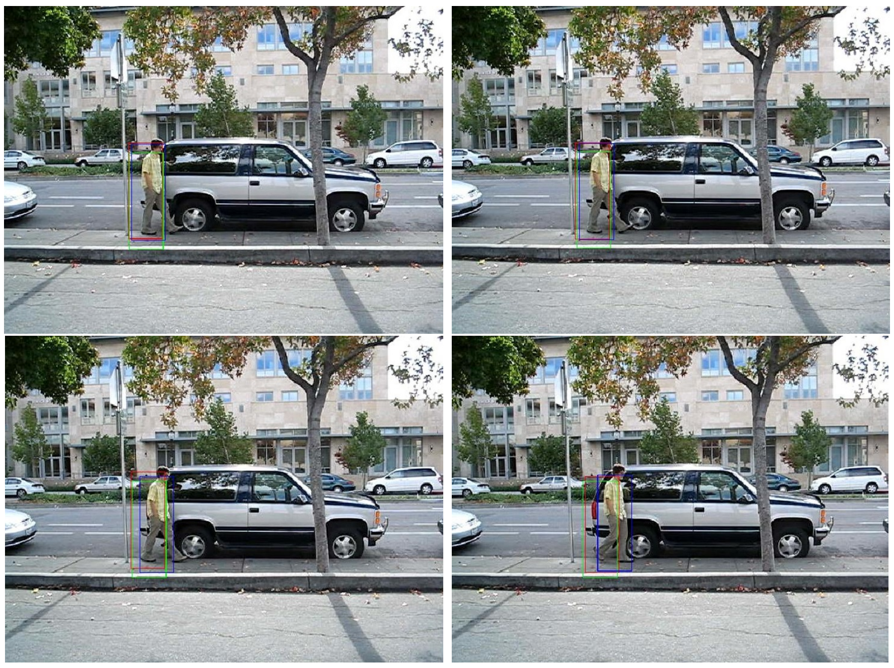

Figure 6.18: Tracking sequence: Frame 37, 38, 39, and 42 are shown (starting from the top left one in clockwise direction). Red box is for SIR-PF, Green box is for bat algorithm and Blue box is for the proposed algorithm for Table. 6.5

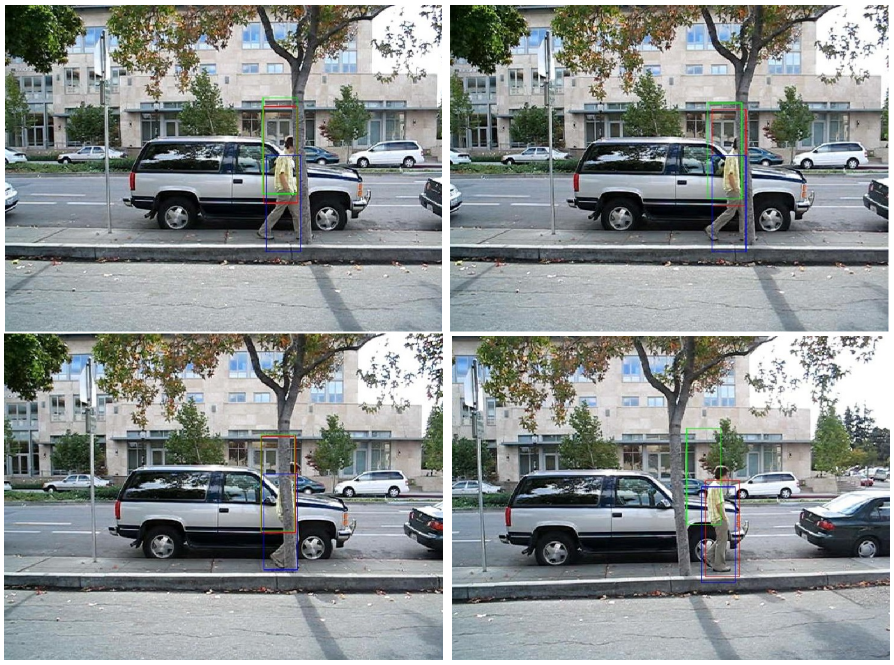
Figure 6.19: Tracking sequence: Frame 81, 82, 85, and 95 are shown (starting from the top left one in clockwise direction). Red box is for SIR-PF, Green box is for bat algorithm and Blue box is for the proposed algorithm for Table. 6.5

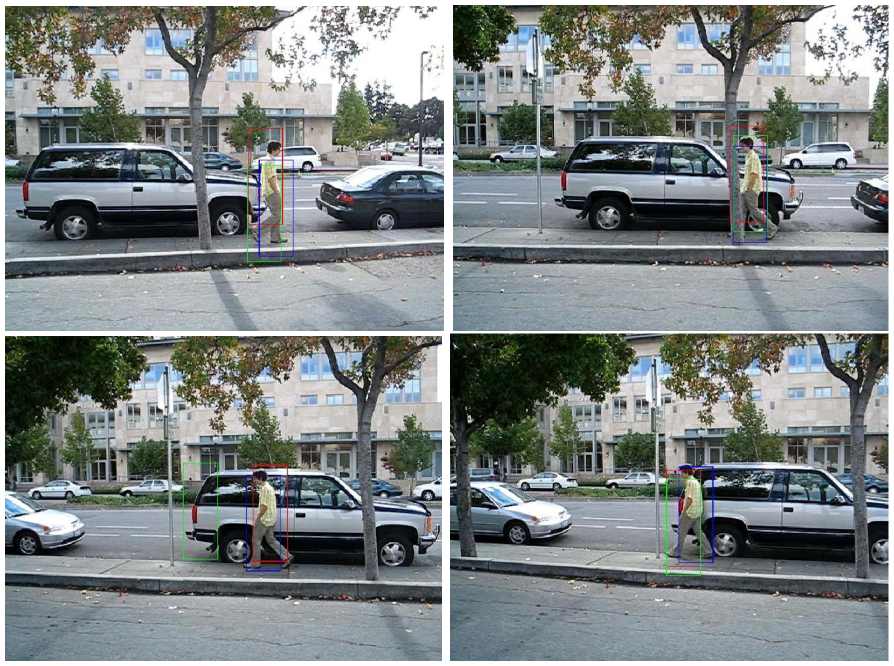
Figure 6.20: Tracking sequence: Frame 104, 162, 184, and 230 are shown (starting from the top left one in clockwise direction). Red box is for SIR-PF, Green box is for bat algorithm and Blue box is for the proposed algorithm for Table. 6.5

6.2.6 Case Study 6 : Tracking Object Under Occlusion, Fast Motion and Scale Change

In this case study, the video is taken from [1, 2]. In the video a person is tracked who is walking and faces occlusion along with scale change. Table. 6.6 lists the results of tracking and Fig. 6.21, Fig. 6.22, Fig. 6.23, Fig. 6.24 shows sample sequences of tracking. In this video also, performances of all the three trackers are almost same, but the proposed algorithm is a bit better than the others two trackers. This video is taken due to three reasons: occlusion, large scale change and less diversity in color. Many of the measurement models fail to process information when diversity of color is less. In this video, on two major colors are present in the object. One of them is black which in large quantity. It has been observed that proposed measurement model works better than others. Occlusion is also present. In this case, the object is moving in one direction in the presence of occlusion. This proposed adaptive model moves the mean value of the process noise in one direction. It has been proven mathematically that the chance of this model to generate comparatively better particles is more. This has been verified in this test case again in presence of occlusion. SIR-PF (Sampling Importance Resampling Particle Filter) and bat algorithm gets deviated slightly due to clutter, while the proposed algorithm can track accurately. When complete occlusion occurs, the track is lost. But the proposed algorithm immediately tracks the object as soon as the object comes into sight. The proposed algorithm outperforms SIR-PF and Bat algorithm. When there is no degeneracy, the particles are moved toward some better location using type-2 jump. Due to occlusion, if object track is lost, type-1 jump searches in the entire state-space and tracks the lost object due to complete occlusion immediately. Clutter is also present in the video. The problem of clutter is solved by the proposed measurement model. Both SIR-PF (Sampling Importance Resampling Particle Filter) and bat algorithm fail in presence of clutter. But the proposed algorithm outperforms as usual. The proposed tracker does not get trapped at any local optima for this reason, or the chance of such trapping is comparatively lesser in case of the proposed algorithm. But, again even if it gets trapped at a local optima anyhow, the proposed resampling algorithm helps the tracker come out of the trapped local optima; this is because, when the object gets trapped at the local optima, the weight is obviously still lower than that of the actual location, and so, type-1 jump gets activated and the it searches throughout the entire space. The same thing happens in case of occlusion also. Hence, the algorithm comes out of the trapped or wrong location much faster as compared to SIR-PF (Sampling Importance Resampling Particle Filter) and bat algorithm. In the same way, the loss of track due to fast motion is also solved. As usual, due the proposed particle filter algorithm, maximum error is also quite lesser than the others. Moreover the scale of the object is also changing in large amount. This change is comparative larger than other videos. The changes in scale is adapted by the template update algorithm. Hence, the overall performance of the proposed algorithm is comparatively much better.

6.2.7 Case Study 7 : Tracking Object Under Illumination Variation and Clutter

In this case study, the video is taken from [1, 2]. In the video, a person is perform in stage where abrupt change in illumination occurs along with background clutter. Table. 6.7 lists the results of tracking and Fig. 6.25, Fig. 6.26, Fig. 6.27, Fig. 6.28 show sample sequences of tracking. This video

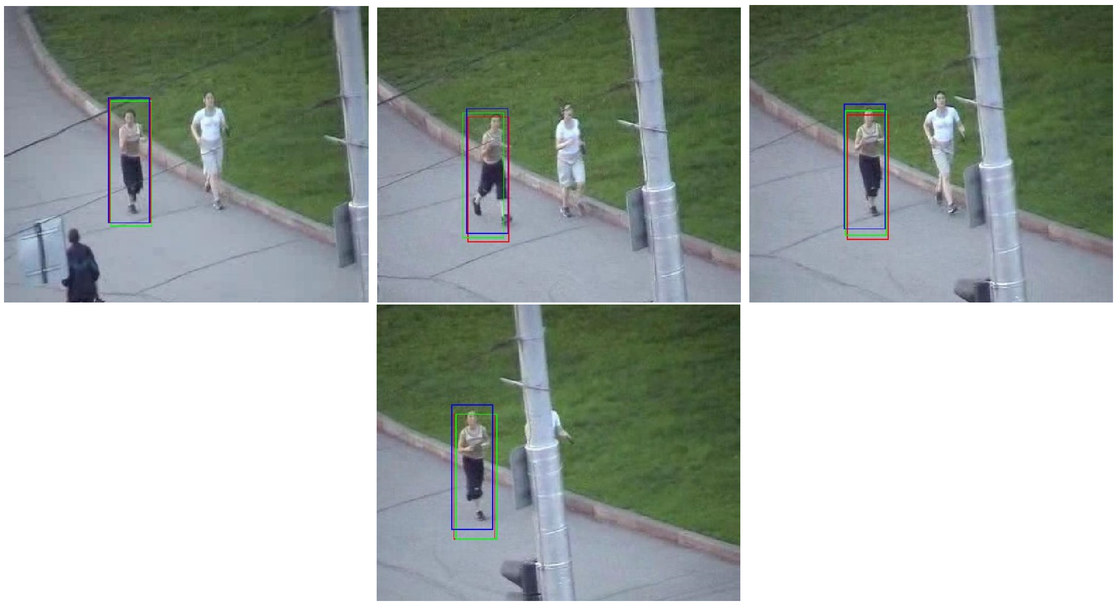

Figure 6.21: Tracking sequence: Frame 3, 26, 38, and 55 are shown (starting from the top left one in clockwise direction). Red box is for SIR-PF, Green box is for bat algorithm and Blue box is for the proposed algorithm for Table. 6.6

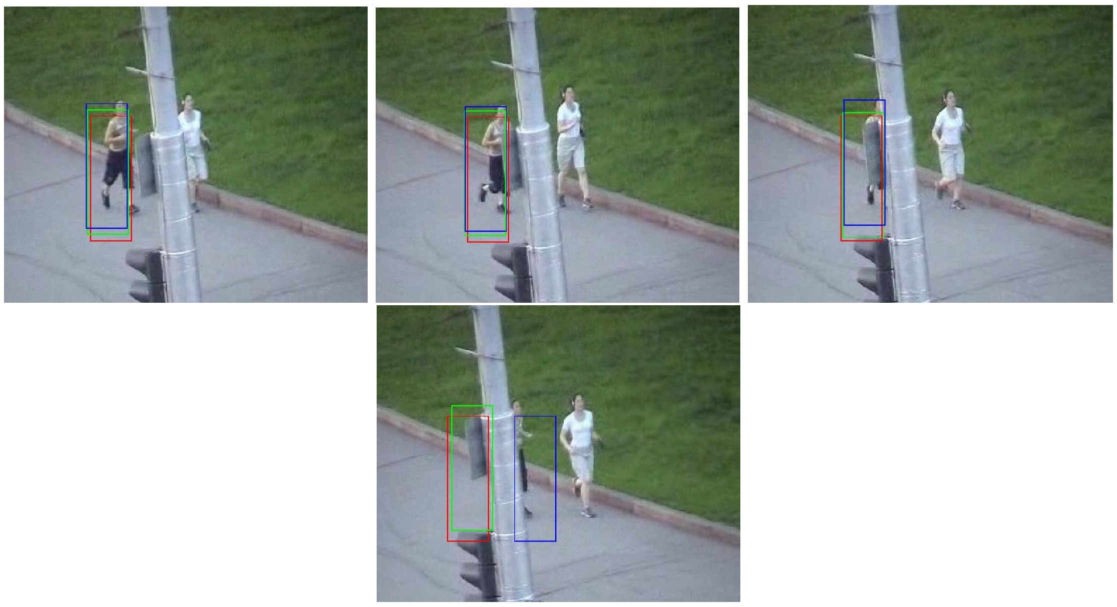

Figure 6.22: Tracking sequence: Frame 61, 65, 70, and 78 are shown (starting from the top left one in clockwise direction). Red box is for SIR-PF, Green box is for bat algorithm and Blue box is for the proposed algorithm for Table. 6.6

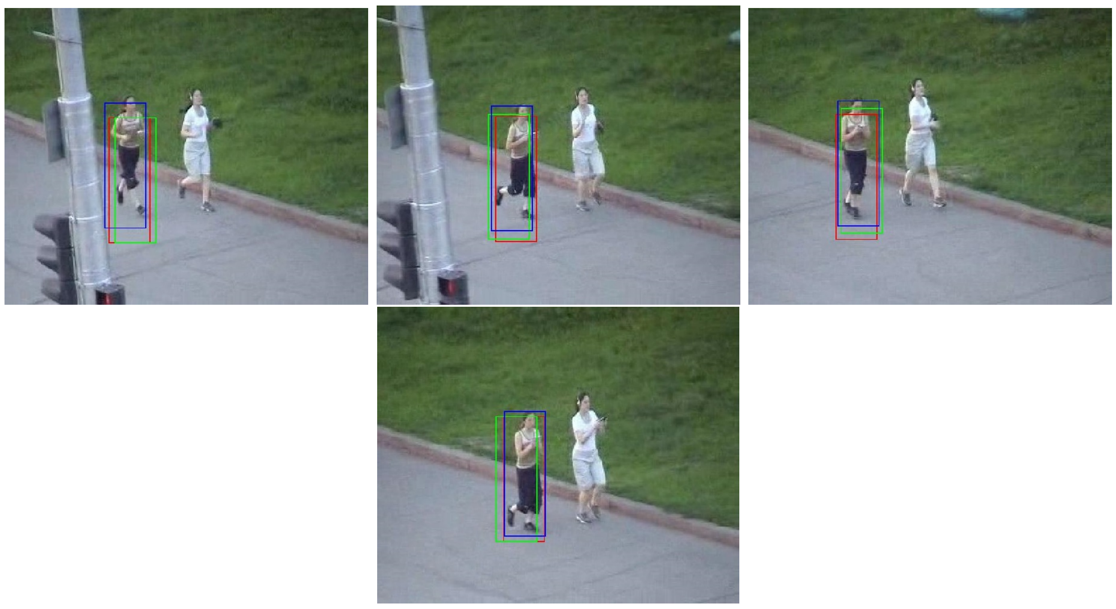

Figure 6.23: Tracking sequence: Frame 88, 100, 121, and 154 are shown (starting from the top left one in clockwise direction). Red box is for SIR-PF, Green box is for bat algorithm and Blue box is for the proposed algorithm for Table. 6.6

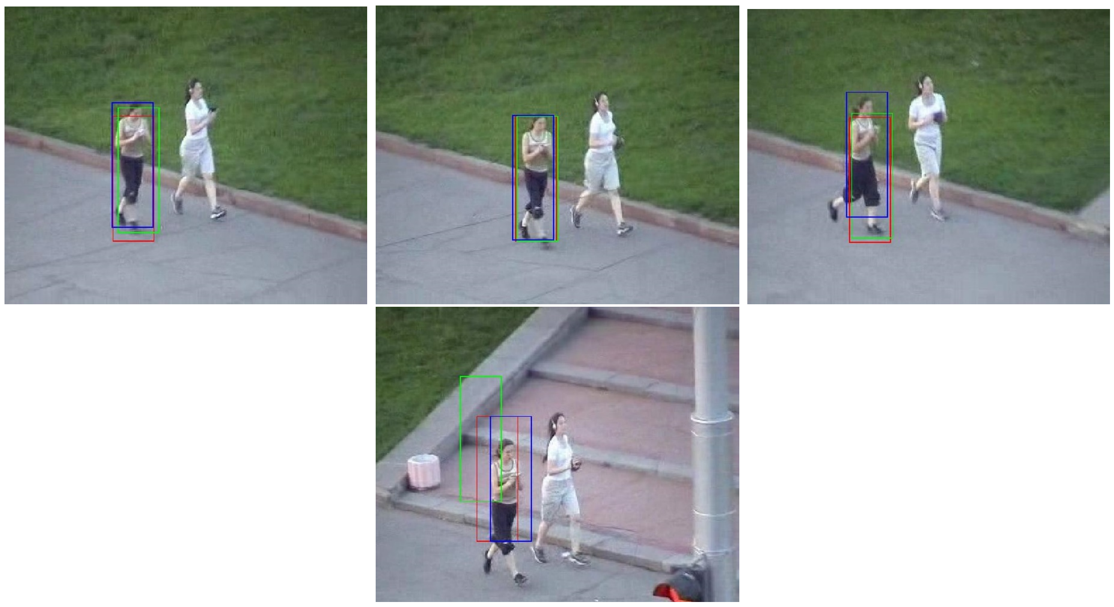

Figure 6.24: Tracking sequence: Frame 192, 210, 239, and 299 are shown (starting from the top left one in clockwise direction). Red box is for SIR-PF, Green box is for bat algorithm and Blue box is for the proposed algorithm for Table. 6.6

Table 6.6: Case Study 6

—	**SIR-PF**	**Bat Algo.**	**Proposed Algo.**
Mean Error	17.32	16.45	13.2
Maximum Error	50.21	74.01	42.57
Std. Deviation	9.70	11.84	7.57
Frames per Second	1.98	1.34	1.37
Success Rate	61.82	72.88	82.95

is a bit different from others. The light intensity in this video is less and flashes of light are also present. Neither severe occlusion nor severe scale change is there. In this video also, the proposed algorithm outperforms the other two trackers. It has been observed that proposed measurement model works better than others. In this case, the object is moving abruptly in the presence of clutter and illumination fluctuation. This proposed adaptive model moves the mean value of the process noise to a better direction. It has been proven mathematically that the chance of this model to generate comparatively better particles is more. This has been verified in this test case again in presence of abrupt motion, clutter and illumination change. SIR-PF (Sampling Importance Resampling Particle Filter) and bat algorithm gets deviated due to clutter, while the proposed algorithm can track accurately. If the track is lost anyhow, the proposed algorithm immediately catches the track. When there is no degeneracy, the particles are moved toward some better location using type-2 jump. And, if object is lost from tracking, type-1 jump searches the entire state-space and tracks the lost object. The problem of clutter is solved by the proposed measurement model. Both SIR-PF (Sampling Importance Resampling Particle Filter) and bat algorithm fails in presence of clutter. But the proposed algorithm outperforms as usual. The proposed tracker does not get trapped at any local optima for this reason, or the chance of such trapping is comparatively lesser in case of the proposed algorithm. But, again even if it gets trapped at a local optima anyhow, the proposed resampling algorithm helps the tracker come out of the trapped local optima; this is because, when the object gets trapped at the local optima, the weight is obviously still lower than that of the actual location, and so, type-1 jump gets activated and the it searches throughout the entire space. Hence, the algorithm comes out of the trapped or wrong location much faster as compared to SIR-PF (Sampling Importance Resampling Particle Filter) and bat algorithm. As usual, due the proposed particle filter algorithm, maximum error is also quite lesser than the others. The illumination problem is solved by the adapted by the template update algorithm. Hence, the overall performance of the proposed algorithm is comparatively much better.

Table 6.7: Case Study 7

—	**SIR-PF**	**Bat Algo.**	**Proposed Algo.**
Mean Error	67.44	90.33	14.2
Maximum Error	260.52	148.55	43.73
Std. Deviation	49.3	78.68	10.5
Frames per Second	2.32	1.72	1.51
Success Rate	10.14	15.62	58.63

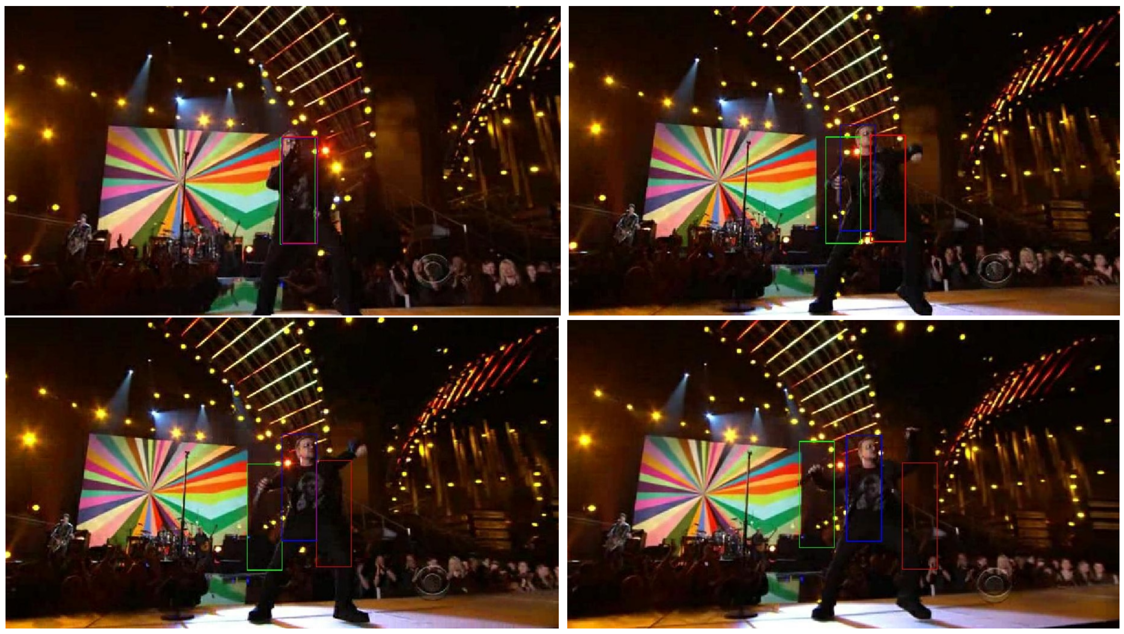

Figure 6.25: Tracking sequence: Frame 7, 20, 23, and 27 are shown (starting from the top left one in clockwise direction). Red box is for SIR-PF, Green box is for bat algorithm and Blue box is for the proposed algorithm for Table. 6.7

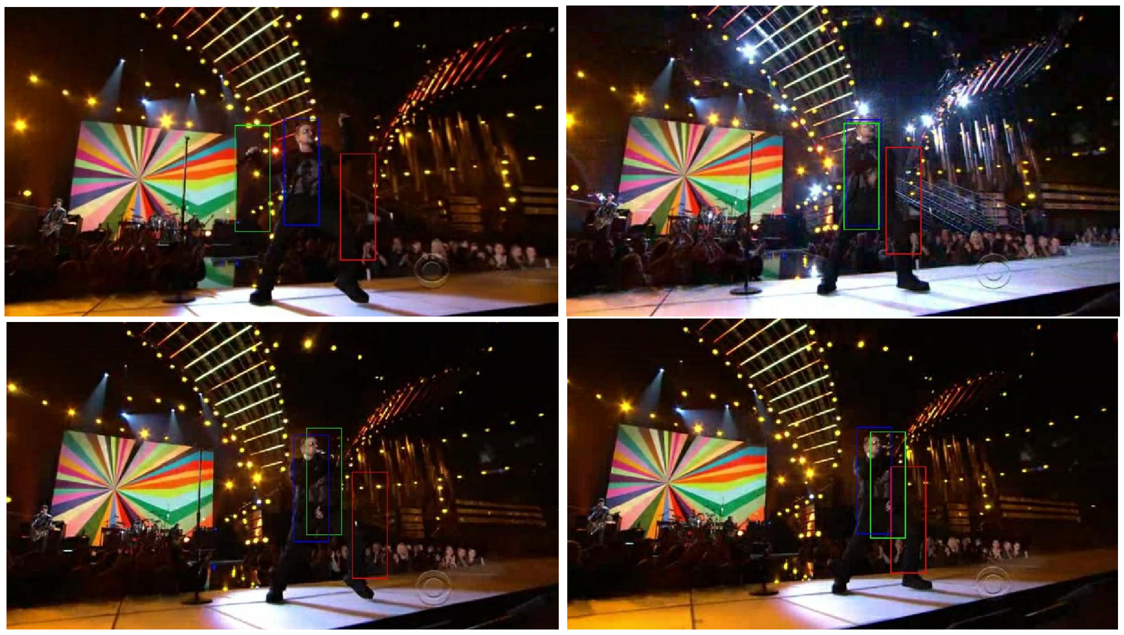

Figure 6.26: Tracking sequence: Frame 28, 36, 43, and 48 are shown (starting from the top left one in clockwise direction). Red box is for SIR-PF, Green box is for bat algorithm and Blue box is for the proposed algorithm for Table. 6.7

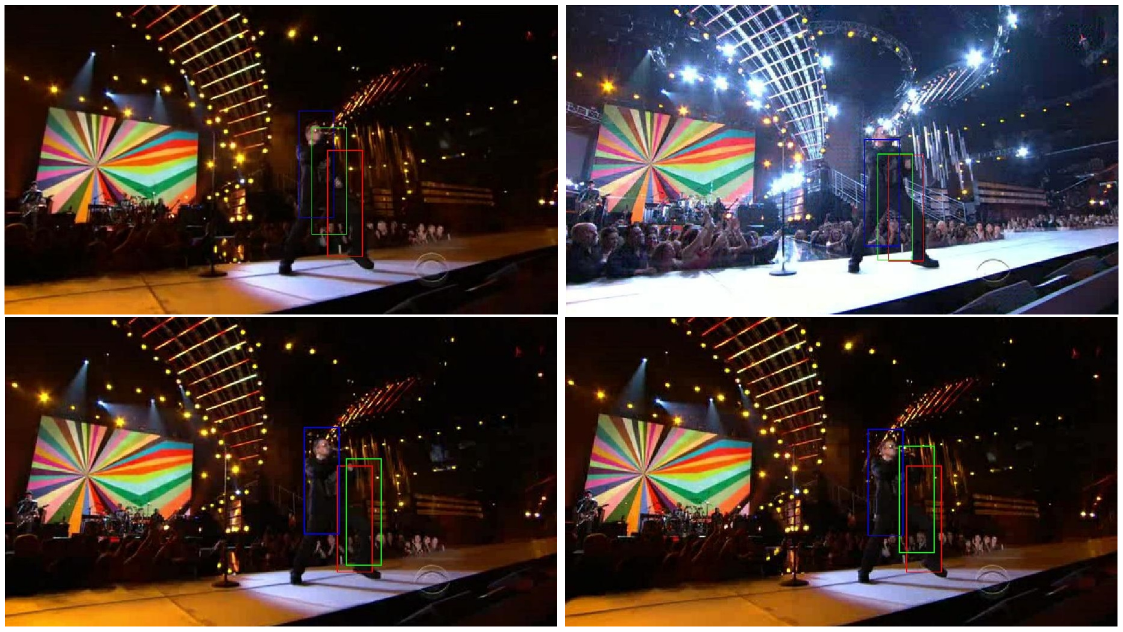

Figure 6.27: Tracking sequence: Frame 55, 60, 62, and 66 are shown (starting from the top left one in clockwise direction). Red box is for SIR-PF, Green box is for bat algorithm and Blue box is for the proposed algorithm for Table. 6.7

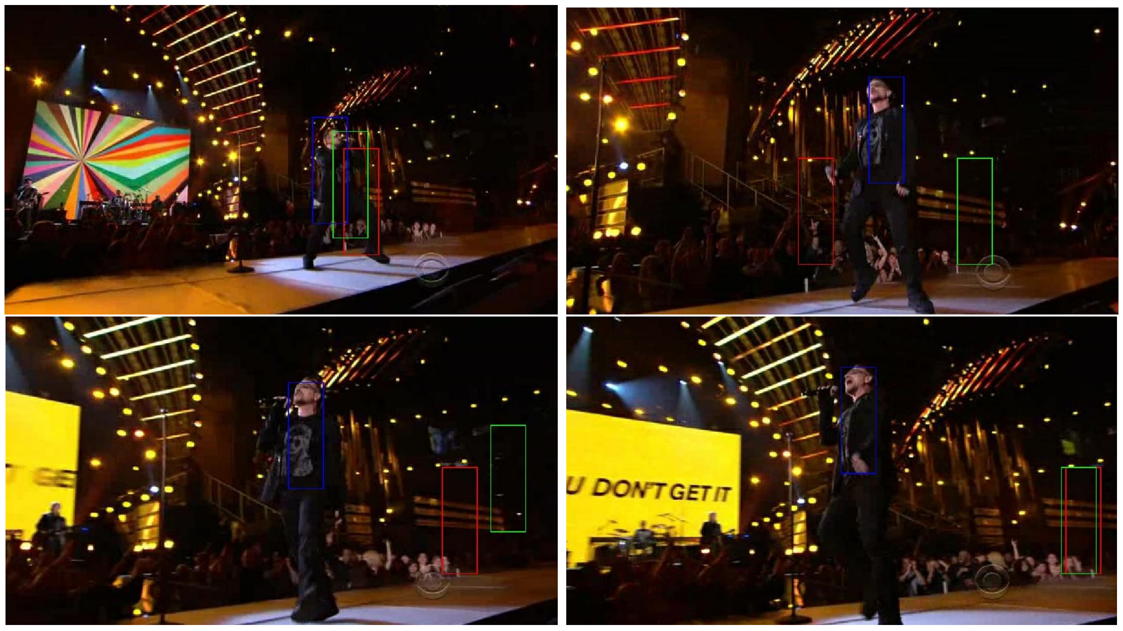

Figure 6.28: Tracking sequence: Frame 69, 322, 352, and 365 are shown (starting from the top left one in clockwise direction). Red box is for SIR-PF, Green box is for bat algorithm and Blue box is for the proposed algorithm for Table. 6.7

6.2.8 Case Study 8 : Tracking Object Under Scale Change, Partial Occlusion and Clutter

In this case study, the video is taken from CAVIAR dataset [3]. In the video, a person is walking whose scale is changing. There is background clutter and short-time partial occlusion in the video also. Table. 6.8 lists the results of tracking and Fig. 6.29, Fig. 6.30, Fig. 6.31, Fig. 6.32 shows sample sequences of tracking. This video is taken because of the presence of clutter, minor scale change and less diversity among colors. The object mainly contains only black and white. The template is chosen so that part of the foreground color continuously remains in the background to evaluate the robustness of the proposed measurement model. There is minute change in light intensity also. Neither severe occlusion nor severe scale change is there. The proposed algorithm outperforms the other two trackers. It has been observed that proposed measurement model works better than others. There is partial occlusion also. In this video, the object is moving in one direction only in the presence of clutter. Thus the proposed adaptive model moves the mean value of the process noise in one direction. It has been proven mathematically that the chance of this model to generate comparatively better particles is higher. This has been verified in this test case again in presence of uni-directional motion, clutter and partial occlusion. SIR-PF (Sampling Importance Resampling Particle Filter) as well as bat algorithm gets deviated due to clutter, while the proposed algorithm can track accurately. If the track is lost anyhow, the proposed algorithm immediately catches the track. When there is no degeneracy, the particles are moved toward some better location using type-2 jump. And, if object is lost from tracking, type-1 jump searches the entire state-space and tracks the lost object. Thus the clutter problem is solved by the proposed measurement model. Both SIR-PF (Sampling Importance Resampling Particle Filter) and bat algorithm fail in presence of clutter. The proposed algorithm outperforms as usual. The proposed tracker does not get trapped at any local optima for this reason, or the chance of such trapping is comparatively lesser in case of the proposed algorithm. But, again even if it gets trapped at a local optima anyhow, the proposed resampling algorithm helps the tracker come out of the trapped local optima; this is because, when the object gets trapped at the local optima, the weight is obviously still lower than that of the actual location, and so, type-1 jump gets activated and the it searches throughout the entire space. Hence, the algorithm helps the tracker come out of the trapped or wrong location much faster as compared to SIR-PF (Sampling Importance Resampling Particle Filter) and bat algorithm. Due to the same reason, the proposed algorithm works better in presence of partial occlusion present in the video frames. As usual, due the proposed particle filter algorithm, maximum error, mean error and standard deviation of error are also quite lesser than the others, as particles are not allowed to move to the comparatively worse location. This is the inherent feature of the proposed selective particle filter. This feature along with the proposed adaptive motion model reduces the chance of generating degeneracy among the particles. Finally, the illumination problem is solved by the template update algorithm. Hence, the overall performance of the proposed algorithm is comparatively much better than that of the SIR-PF (Sampling Importance Resampling Particle Filter) and bat algorithm.

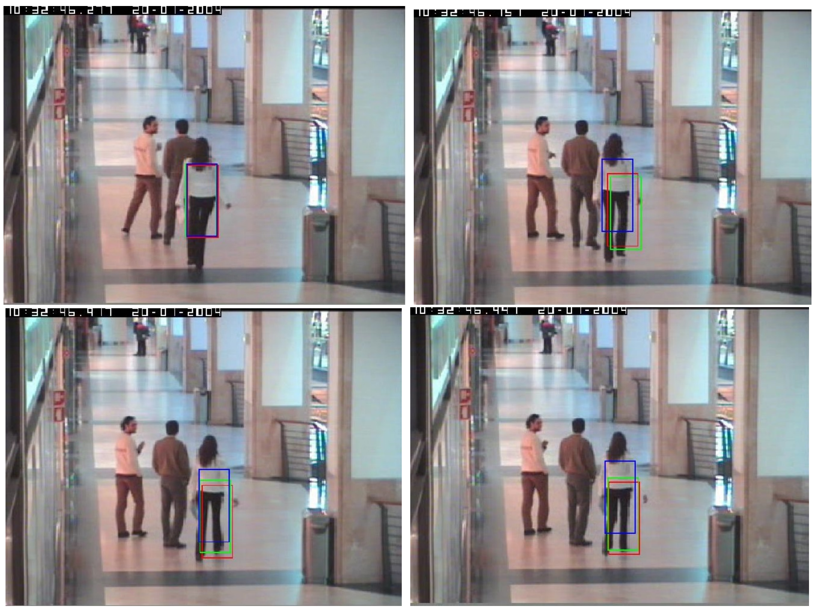

Figure 6.29: Tracking sequence: Frame 352, 364, 368, and 370 are shown (starting from the top left one in clockwise direction). Red box is for SIR-PF, Green box is for bat algorithm and Blue box is for the proposed algorithm for Table. 6.8

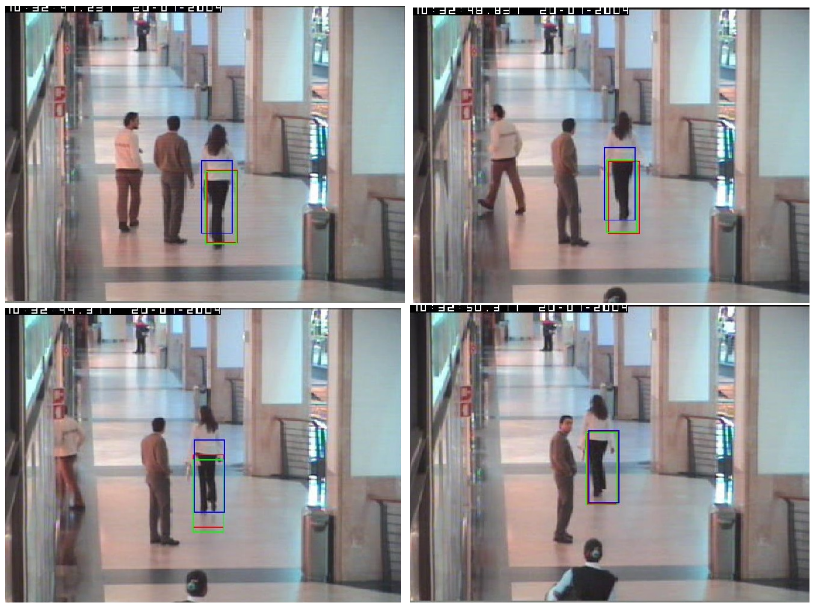

Figure 6.30: Tracking sequence: Frame 376, 419, 428, and 454 are shown (starting from the top left one in clockwise direction). Red box is for SIR-PF, Green box is for bat algorithm and Blue box is for the proposed algorithm for Table. 6.8

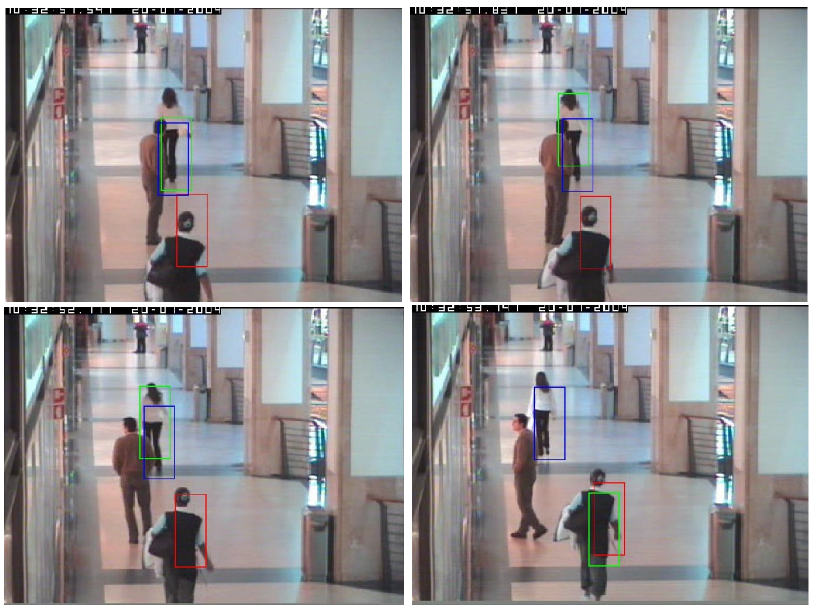
Figure 6.31: Tracking sequence: Frame 485, 491, 513, and 540 are shown (starting from the top left one in clockwise direction). Red box is for SIR-PF, Green box is for bat algorithm and Blue box is for the proposed algorithm for Table. 6.8

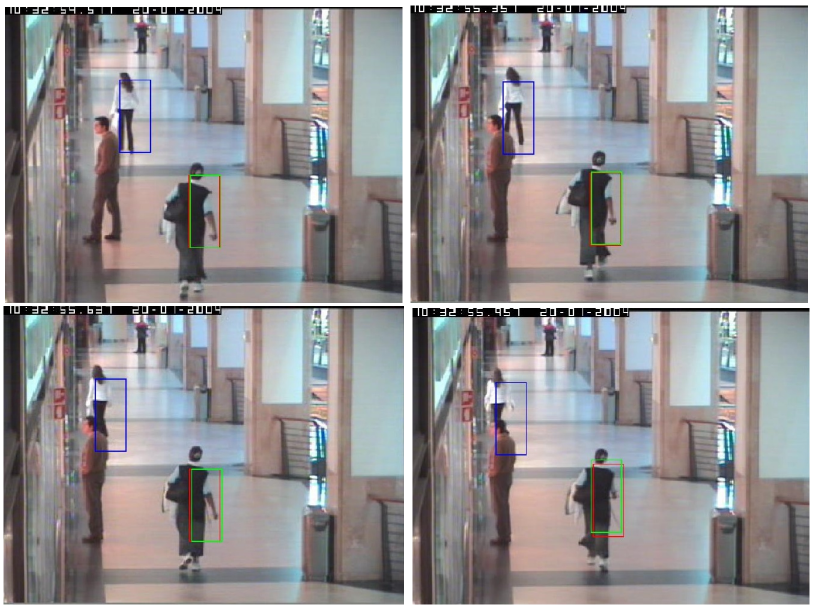
Figure 6.32: Tracking sequence: Frame 558, 579, 586, and 594 are shown (starting from the top left one in clockwise direction). Red box is for SIR-PF, Green box is for bat algorithm and Blue box is for the proposed algorithm for Table. 6.8

Table 6.8: Case Study 8

—	SIR-PF	Bat Algo.	Proposed Algo.
Mean Error	66.11	60.35	16.35
Maximum Error	159.11	150.6	31.62
Std. Deviation	50.84	49.96	7.20
Frames per Second	3.92	3.15	2.89
Success Rate	36.65	39.29	60.82

6.2.9 Case Study 9 : Tracking Object Under Complete Occlusion and Clutter

In this case study, the video is taken from [3]. In the video, a person is walking whose scale is changing. There is background clutter and complete occlusion also. Table. 6.9 lists the results of tracking and Fig. 6.33, Fig. 6.34, Fig. 6.35, Fig. 6.36 shows sample sequences of tracking. This video is considered because of the presence of clutter, minor scale change and complete prolonged occlusion. There is minute change in light intensity too. Severe complete occlusion is present. The proposed algorithm outperforms the other two trackers. It has been observed that proposed measurement model works better than others in presence of clutter and occlusion. In this case too, the object is moving in one direction only in the presence of clutter. This proposed adaptive model moves the mean value of the process noise in one direction. It has been proven mathematically that the chance of this model to generate comparatively better particles is more. This has been verified in this test case again in presence of uni-directional motion, clutter and partial occlusion mainly. SIR-PF (Sampling Importance Resampling Particle Filter) and bat algorithm gets deviated due to clutter, while the proposed algorithm can track accurately. If the track is lost anyhow (due to clutter or complete prolonged occlusion), the proposed algorithm immediately catches the lost object after the occlusion is over. When there is no degeneracy, the particles are moved toward some better location using type-2 jump. And, if object is lost from tracking, type-1 jump searches the entire state-space and tracks the lost object. Thus the clutter problem is solved by the proposed measurement model. Both SIR-PF (Sampling Importance Resampling Particle Filter) and bat algorithm fail in presence of clutter. The proposed algorithm outperforms as usual. The proposed tracker does not get trapped at any local optima for this reason, or the chance of such trapping is comparatively lesser in case of the proposed algorithm. But, again even if it gets trapped at a local optima anyhow, the proposed resampling algorithm helps the tracker come out of the trapped local optima; this is because, when the object gets trapped at the local optima, the weight is obviously still lower than that of the actual location, and so, type-1 jump gets activated and the it searches throughout the entire space. Hence, the algorithm helps the tracker come out of the trapped or wrong location much faster as compared to SIR-PF (Sampling Importance Resampling Particle Filter) and bat algorithm. Due to the same reason, the proposed algorithm works better in presence of both prolonged complete occlusion and clutter. The overall performance of the proposed algorithm is comparatively much better than that of the SIR-PF and Bat algorithm.

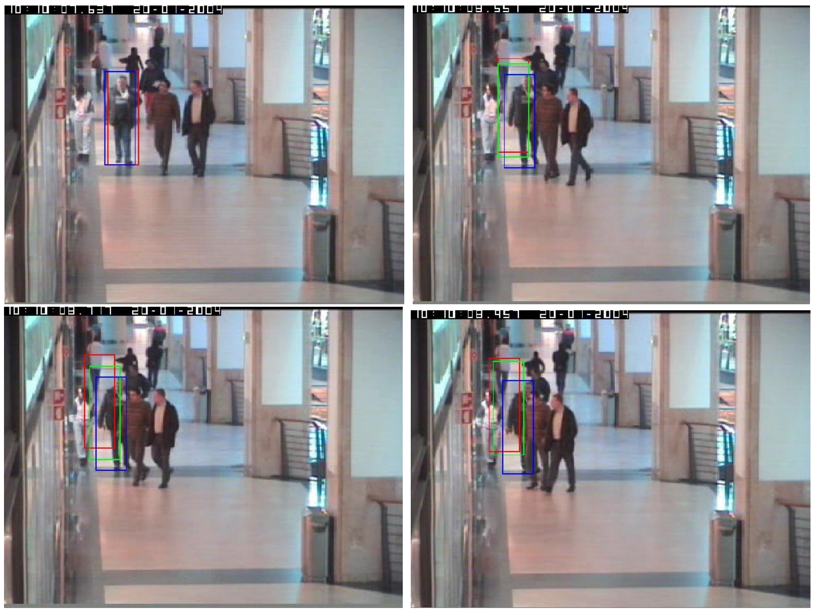
Figure 6.33: Tracking sequence: Frame 192, 214, 218, and 224 are shown (starting from the top left one in clockwise direction). Red box is for SIR-PF, Green box is for bat algorithm and Blue box is for the proposed algorithm for Table. 6.9

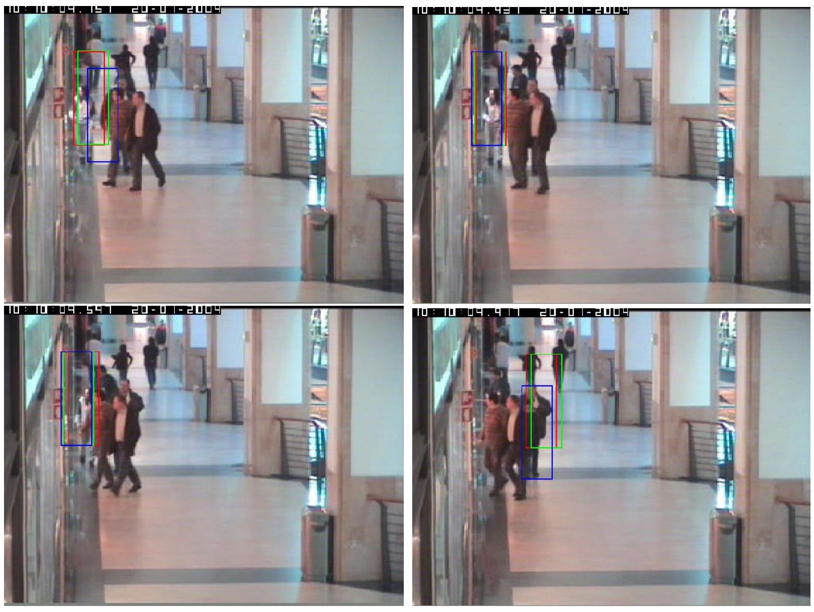
Figure 6.34: Tracking sequence: Frame 229, 237, 240, and 248 are shown (starting from the top left one in clockwise direction). Red box is for SIR-PF, Green box is for bat algorithm and Blue box is for the proposed algorithm for Table. 6.9

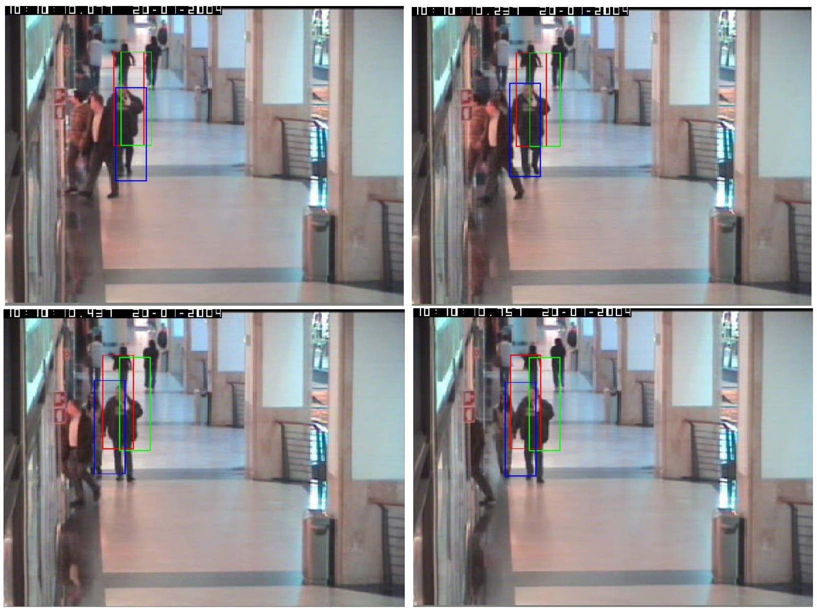

Figure 6.35: Tracking sequence: Frame 252, 256, 262, and 269 are shown (starting from the top left one in clockwise direction). Red box is for SIR-PF, Green box is for bat algorithm and Blue box is for the proposed algorithm for Table. 6.9

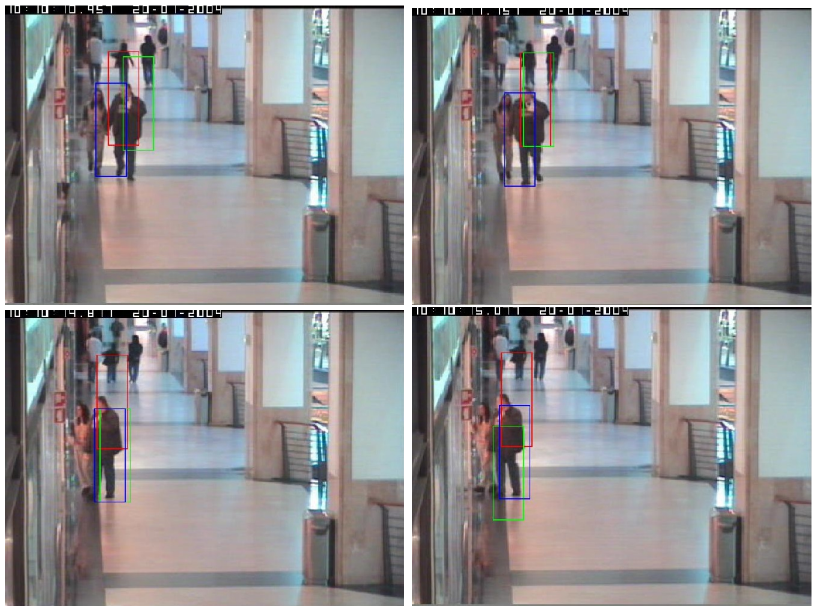

Figure 6.36: Tracking sequence: Frame 275, 294, 373, and 377 are shown (starting from the top left one in clockwise direction). Red box is for SIR-PF, Green box is for bat algorithm and Blue box is for the proposed algorithm for Table. 6.9

Table 6.9: Case Study 9

—	**SIR-PF**	**Bat Algo.**	**Proposed Algo.**
Mean Error	31.94	27.26	13.38
Maximum Error	49.24	52.77	54.58
Std. Deviation	11.06	13.25	11.40
Frames per Second	4.05	3.13	2.80
Success Rate	14.14	32.98	81.68

6.2.10 Case Study 10 : Tracking Object Under Illumination Variation and Motion Blur

This video is taken from [1, 2]. This video is exceptional from others because, in other videos, the pixel-values cover the entire range of the histogram. But, in this case, the pixel-values cover a small part of the entire histogram and eventually increase to towards the maximum value and cover the entire histogram. In the video, a person is moving from extreme dark to the extreme brightness. So, in the beginning, if the full-range (as for example, 0–255 for 8–bit image data) histogram is used, the step-size between the two adjacent bins cannot discriminate the color information properly in the beginning. Hence, a new scheme – adaptive step-sizing – has been proposed for this type of videos. This step is activated by setting: $fullscale = false$. Hence, $fullscale = false$ is set and, according to the Algorithm 7, the step sizes will vary adaptively. Similarly, other changed parameter values for this video only are: $LostTrackTH = 0.4$, $UpdateTH = 0.4$, $simiTH = 0.4$, $\alpha = 1$. For this, the ground truth information is available only after frame number 300. But, to check the robustness of the proposed algorithm under varied illumination change, the tracking has been started from the first frame. So only qualitative results are shown instead of the quantitative result. Fig. 6.37 shows sample sequences of tracking. The proposed tracker tracks better. The proposed tracker tracks better because it can discriminate among the colors much better than other tracking algorithms named SIR-PF (Sampling Importance Resampling Particle Filter) and bat algorithm. Also the template is updated accordingly to catch the adapt the changing histogram. It has been observed that only type-2 jump is being executed as there is no occlusion or chance of missing the target. The problem due to the presence of clutter is solved by the proposed measurement model. Since there is no mechanism for updating the template along with the adaptive step-sizing and the histogram does not carry any spatial information, SIR-PF (Sampling Importance Resampling Particle Filter) and bat algorithm fails to track the object properly as shown the test results.

6.2.11 Analysis

The tests are sun for several times, and the average values are noted. The proposed algorithm is found to be better than SIR-PF (Sampling Importance Resampling Particle Filter) and bat algorithm. All the modules contribute to the better performance of the proposed algorithm. In the next section, the computation cost of the algorithm is analyzed and compared with others.

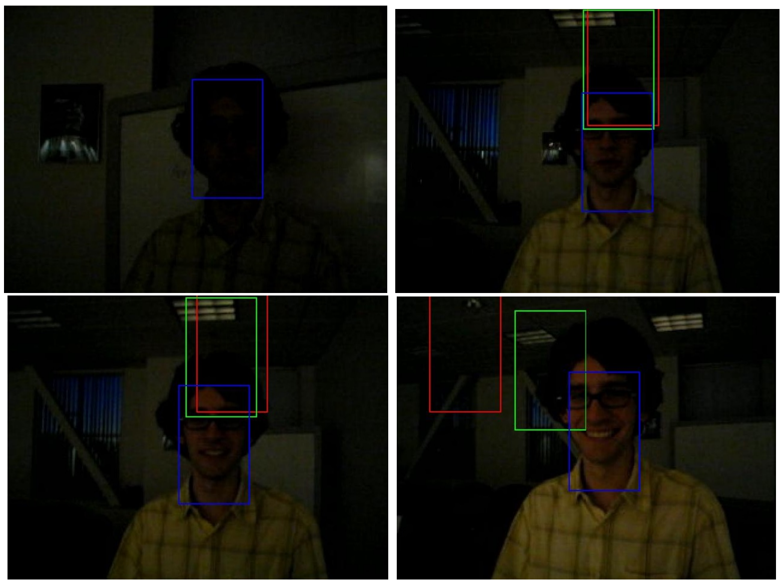

Figure 6.37: Tracking sequence: Frame 1, 2, 124, and 170 are shown (starting from the top left one in clockwise direction). Red box is for SIR-PF, Green box is for bat algorithm and Blue box is for the proposed algorithm Section. 6.2.10

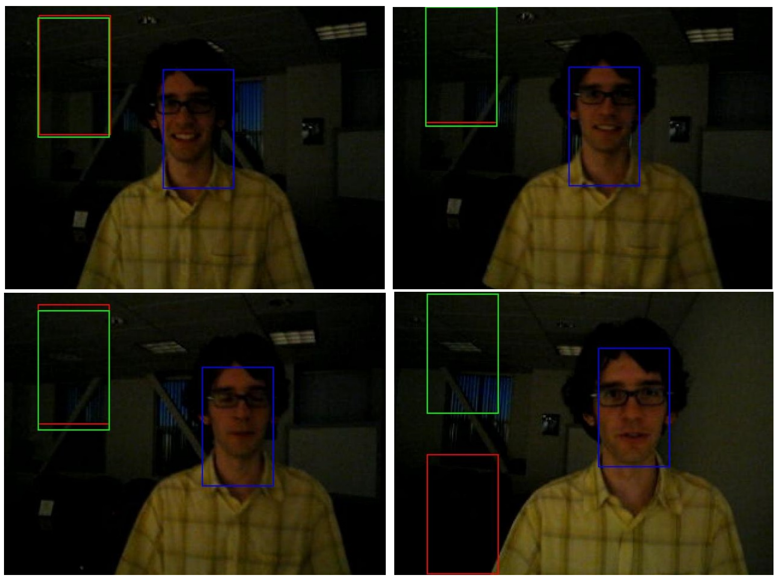

Figure 6.38: Tracking sequence: Frame 190, 204, 212, and 260 are shown (starting from the top left one in clockwise direction). Red box is for SIR-PF, Green box is for bat algorithm and Blue box is for the proposed algorithm for Section. 6.2.10

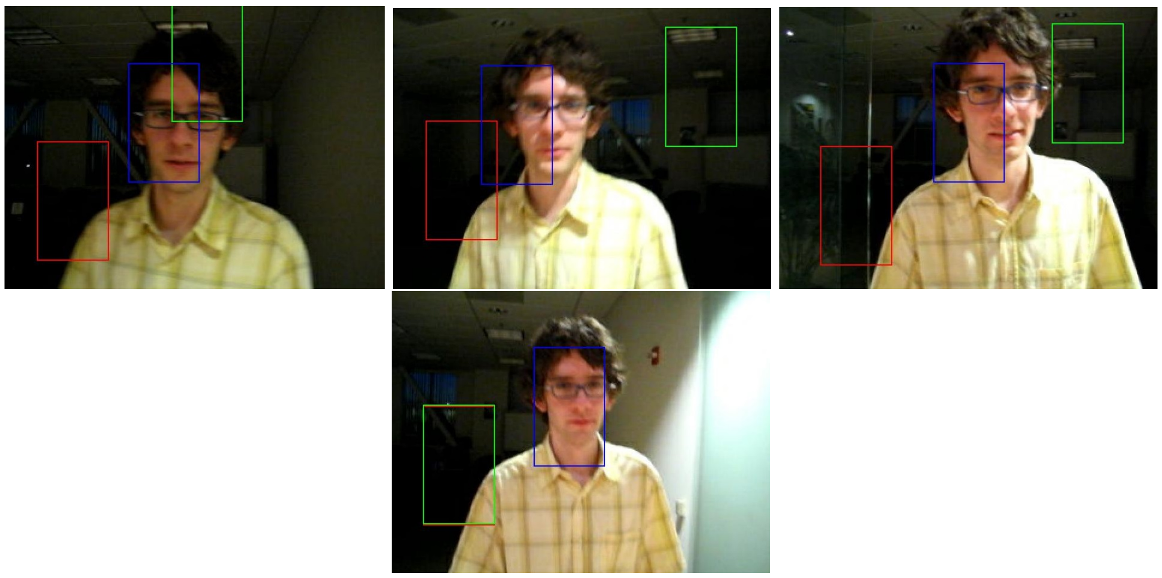

Figure 6.39: Tracking sequence: Frame 302, 309, 329, and 354 are shown (starting from the top left one in clockwise direction). Red box is for SIR-PF, Green box is for bat algorithm and Blue box is for the proposed algorithm for Section. 6.2.10

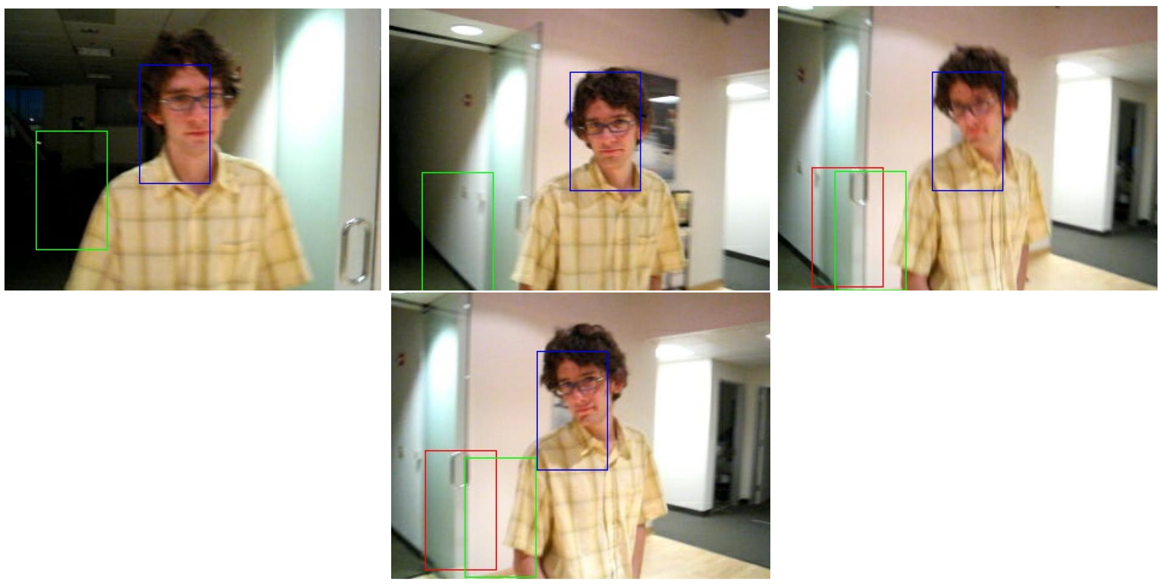

Figure 6.40: Tracking sequence: Frame 372, 385, 423, and 430 are shown (starting from the top left one in clockwise direction). Red box is for SIR-PF, Green box is for bat algorithm and Blue box is for the proposed algorithm for Section. 6.2.10

6.3 Computational Cost Analysis

Tracking time depends on the size of the template. The non-optimized MATLAB implementations of the algorithms are run on a 2.2 GHz 32-bit Intel Core2 Duo processor with 2 GB RAM. The Table. 6.10 lists average computation time (in second) of all the algorithms for randomly selected different template sizes 25×25, 40×120 and 50×140 and for particles or total number of iterations 100, 200, 300 and 400. It has been observed that, under similar number of particles or total number of iterations, the computational cost of the proposed algorithm is higher than that of the SIR-PF (Sampling Importance Resampling Particle Filter) and bat algorithm:

Table 6.10: Study of Computational Cost

—	**25 × 25**	**40 × 120**	**50 × 140**
SIR-PF(100)	0.03	0.09	0.15
Bat(100)	0.03	0.15	0.2
Proposed(100)	0.15	0.67	0.85
SIR-PF(200)	0.04	0.2	0.26
Bat(200)	0.07	0.3	0.38
Proposed(200)	0.22	1.2	1.7
SIR-PF(300)	0.07	0.29	0.4
Bat(300)	0.08	0.4	0.6
Proposed(300)	0.31	1.7	2.6
SIR-PF(400)	0.08	0.4	0.57
Bat(400)	0.15	0.5	0.8
Proposed(400)	0.4	2.3	3.4

Generally number of particles of particle filters are selected based on the complexity of the shape of the posterior distribution and the number of states. The typical range of particles is $100 - 1000$. The tests are done by varying number of particles or bats. Throughout the test cases, it has been observed that 400 particles for SIR-PF (Sampling Importance Resampling Particle Filter), 40 bats with 10 iterations for bat algorithm and 20 particles with 5 iterations for each particle is a good trade-off between speed and accuracy. Fig. 6.41, Fig. 6.42 and Fig. 6.43 show the typical computational cost and accuracy comparison of all the trackers for the Case Study 3, and this trend is approximately followed by all the benchmark videos.

It has been observed from the case studies that, at the cost of some extra computational cost, the proposed algorithm tracks with much higher accuracy.

Further the time complexity of the proposed algorithm is analyzed with three other state-of-the-art algorithms: LOT [55], MTT [56] and SCM [57]. The complexity of the proposed algorithm is compared with these three algorithms because LOT and MTT are simulated using MATLAB, and SCM is simulated using mixed MATLAB and C language, in [1, 2]. The proposed algorithm is also coded in MATLAB. All these four algorithms belong to particle filter based tracking. Hence, the computational time is comparable. All these three algorithms are executed on Intel i7 3770 CPU having 3.4GHz clock speed. On the other hand, the proposed algorithm is executed on Intel 32-bit Core2 Duo processor having 2.2 GHz clock speed. Thus the proposed algorithm is run on a comparatively lower grade platform. Thus, if the code of the proposed is optimized and run on a better configuration, the speed of tracking will increase more. LOT initializes 250 particles while

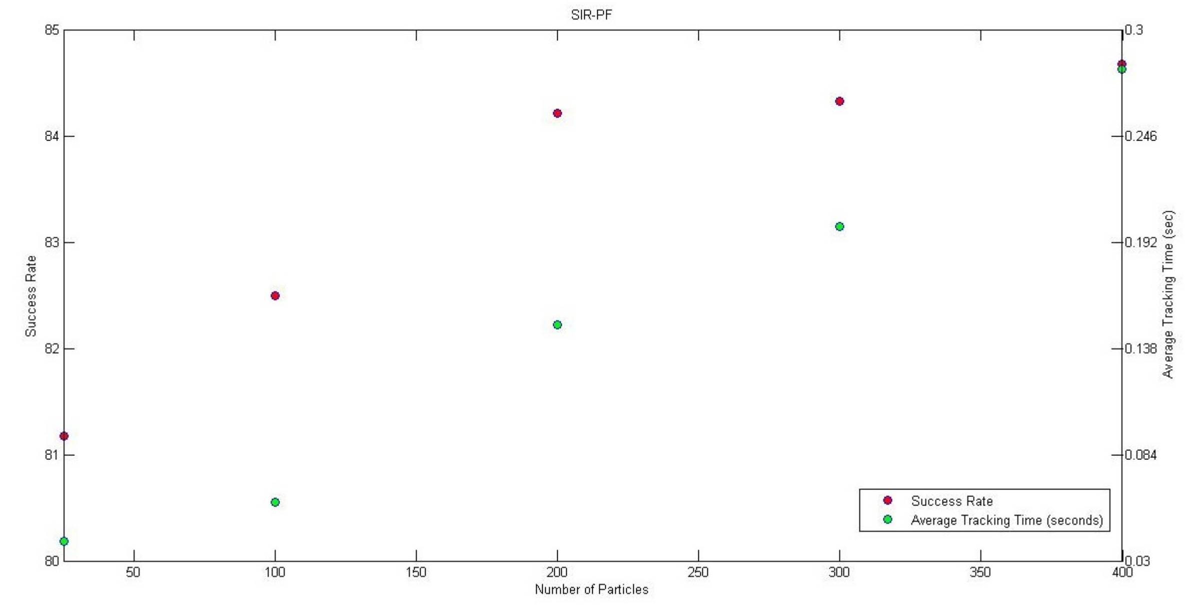

Figure 6.41: Comparison of tracking time and success rate for SIR-PF (Sampling Importance Resampling Particle Filter) with number of particles 25, 100, 200, 300 and 400

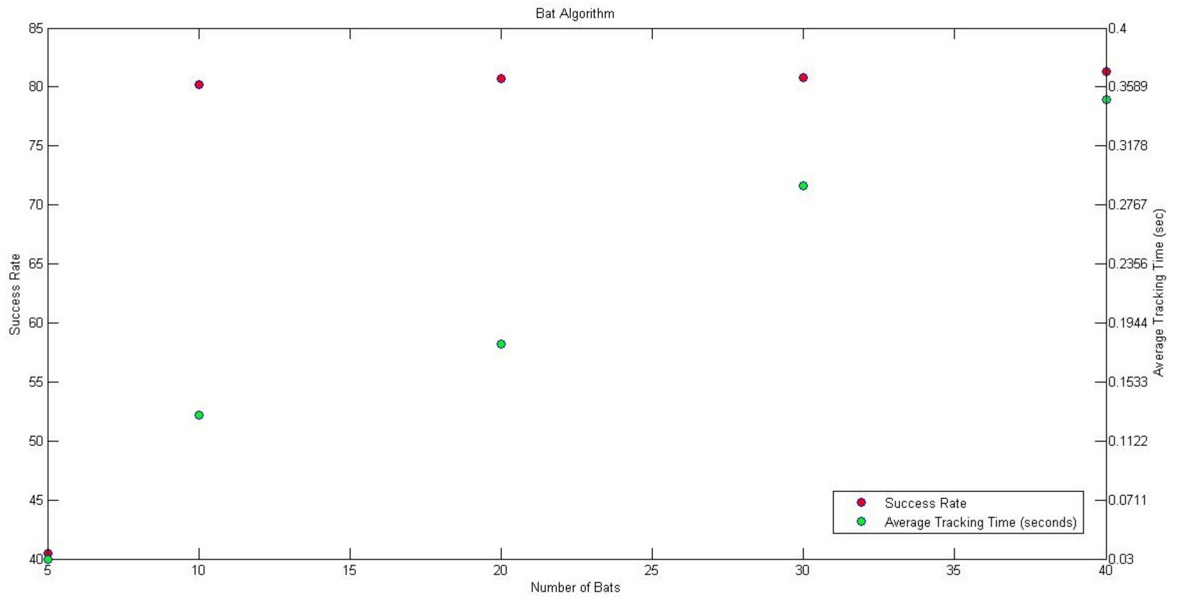

Figure 6.42: Comparison of tracking time and success rate for bat algorithm with number of bats 5, 10, 20, 30 and 40. Iteration for each bat is 10

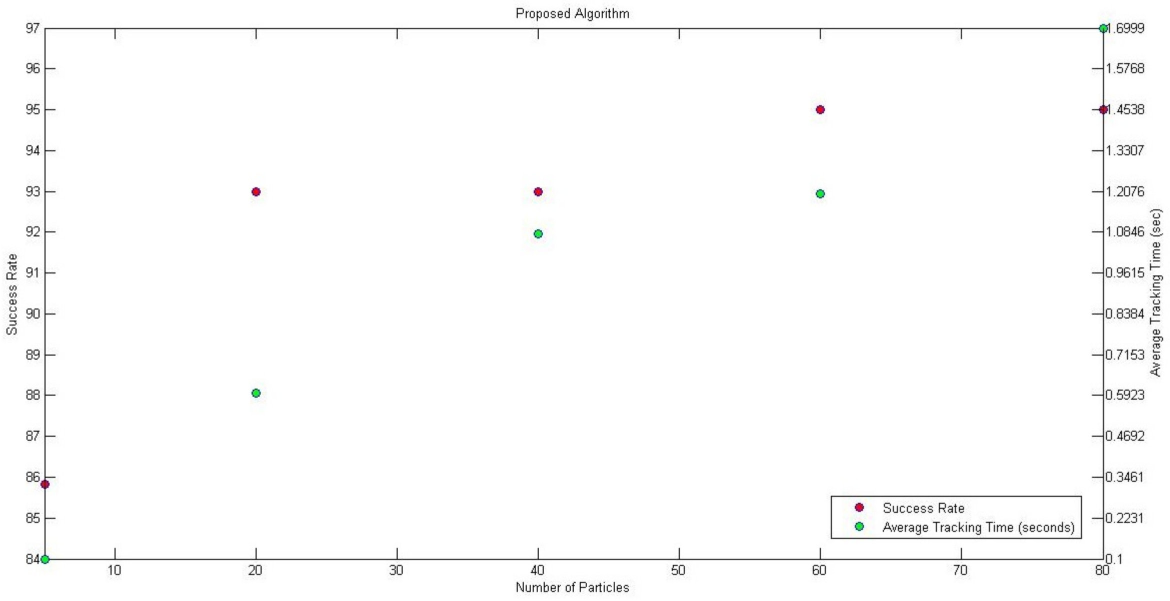

Figure 6.43: Comparison of tracking time and success rate for proposed algorithm with number of particles 5, 20, 40, 60 and 80. Iteration for each particle is 5

MTT has initializes 400 particles. The cost of computation among the four algorithms is compared against minimum and maximum template sizes (because other settings are kept fixed throughout the experiments as mentioned) 122 × 148, 32 × 46, 34 × 82, 26 × 62, 68 × 122 and 30 × 24 for the algorithms in Table. 6.11.

Table 6.11: Comparative Computational Cost in FPS (frame per second)

—	**122 × 148**	**32 × 46**	**34 × 82**	**26 × 62**	**68 × 122**	**30 × 24**
LOT	0.1	2.0	–	–	–	–
MTT	–	–	0.3	1.9	–	–
SCM	–	–	–	–	0.4	0.6
Proposed Algo.	0.21	2.5	1.44	2.22	0.51	2.77

6.4 Comparison in Subtracted Color Space

In this section, some additional test results are listed for the SIR-PF (Sampling Importance Resampling Particle Filter), bat algorithm and the proposed algorithm.. The tests are done on the same video sequences as given above which means that the video sequence used in *Case Study 1* is same as the *Case Study 1* in this section. There are lots of color spaces defined till now. Out of them, some color spaces are suitable for tracking purpose. In visual object tracking, mainly RGB, HSV and rg color spaces are widely used. The main drawback of RGB color space in tracking that the colors or hue cannot be separated from intensity. Again, although HSV can separate hue, saturation and value or intensity, it is not linear and its computational time is large [18]. On the other hand, rg-space is suitable for face-tracking [8]. In this subsection, experiment is done with subtracted color space which is faster in computation as compared to HSV (HSV has more arithmetic and logical operations per pixel as compared to this subtracted color space). Let R, G, B be the three color channels of RGB color space, the R', G', B' is the transformed color space for every pixel l is as follows:

$$R' = |R - G| \tag{6.2}$$

$$G' = |G - B| \tag{6.3}$$

$$B' = |B - R| \tag{6.4}$$

Let f be a function which changes all R, G, B values of l-th pixel by the same amount. Thus, after subtraction, their values will remain same. This makes the subtracted color space robust against influencing functions like f which can be as for example, illumination. But, the major drawback of this subtracted color space is that it cannot distinguish between black $R = 0, G = 0, B = 0$ and white $R = 255, G = 255, B = 255$ or any other color where $R = G = B$ for l-th pixel. Thus, in general, the subtracted color space works good if the object does not contain any such color. This has been proven by experiment in this section. *Proposed Algo. (Sub)* in tables' column heading means the result of the proposed algorithm using subtracted color space.

6.4.1 Case Study 1 : Tracking Object Under Illumination Variation, Occlusion and Clutter

The performance of the algorithm with the subtracted RGB color space is included in Table. 6.12. This color space works because all the RGB values for green, which is the predominant color, are different and also for all the shades of green.

Table 6.12: Case Study 1 : Subtracted Color Space

—	**SIR-PF**	**Bat Algo.**	**Proposed Algo.**	**Proposed Algo. (Sub)**
Mean Error	78.95	93.44	14.03	16.1
Maximum Error	346.69	344.94	161.45	191.1
Std. Deviation	103.19	103.51	18.53	23.9
Success Rate	51.17	41.66	86.82	89.24

6.4.2 Case Study 2 : Tracking Object Under Clutter, Fast Motion and Scale Change

The performance of the algorithm with the subtracted RGB color space is included in Table. 6.13. This color space works because all the RGB values for green and yellow, which are the predominant colors, are different and also for all the shades of green and yellow.

Table 6.13: Case Study 2 : Subtracted Color Space

—	SIR-PF	Bat Algo.	Proposed Algo.	Proposed Algo. (Sub)
Mean Error	54.66	181.81	12.54	9.15
Maximum Error	146.65	412.89	22.67	24.00
Std. Deviation	27.58	105.34	16.42	4.84
Success Rate	49.69	11.69	94.77	98.77

6.4.3 Case Study 3 : Tracking Object Under Motion Blur, Fast Motion and Scale Change

The performance of the algorithm with the subtracted RGB color space is included in Table. 6.14. In this case, the accuracy is less as the object contains black and almost white background in some proportion. Still it tracks because, for the skin color, RGB values do not match for each pixel. Hence, the subtraction is not around (0,0,0).

Table 6.14: Case Study 3 : Subtracted Color Space

—	SIR-PF	Bat Algo.	Proposed Algo.	Proposed Algo. (Sub)
Mean Error	16.08	19.24	10.47	12.4
Maximum Error	68.06	139.44	35.79	81.61
Std. Deviation	7.11	27.39	6.8	7.8
Success Rate	87.17	85.33	88.67	85.00

6.4.4 Case Study 4 : Tracking Object Under Occlusion, Fast Motion and Clutter

The performance of the algorithm with the subtracted RGB color space is included in Table. 6.15.This color subtracted space works because the object contains some red marks which have different RGB values.

Table 6.15: Case Study 4 : Subtracted Color Space

—	SIR-PF	Bat Algo.	Proposed Algo.	Proposed Algo. (Sub)
Mean Error	150.79	91.93	81.84	20.9
Maximum Error	256.56	257.12	251.96	101.19
Std. Deviation	48.11	66.12	70.09	18.36
Success Rate	5.84	7.02	15.12	61.51

6.4.5 Case Study 5 : Tracking Object Under Occlusion and Clutter

The performance of the algorithm with the subtracted RGB color space is included in Table. 6.16. This color space does not work because RGB value of different shades of gray, which is the color of the trousers of the person, have same RGB value (211,211,211; 192,192,192; 169,169,169; 128,128,128).

Thus this color space cannot differentiate between the object from the grayish background and, the tracker, in this case loses its track, but still Bhattacharyya matching coefficient is as large as 0.89 in average. Thus the subtracted color space does not work.

Table 6.16: Case Study 5 : Subtracted Color Space

—	**SIR-PF**	**Bat Algo.**	**Proposed Algo.**	**Proposed Algo. (Sub)**
Mean Error	17.67	22.76	16.79	77.21
Maximum Error	56.8	67.96	34.22	266.97
Std. Deviation	12.34	13.79	7.51	80.19
Success Rate	63.6	55.2	64.8	35.2

6.4.6 Case Study 6 : Tracking Object Under Occlusion, Fast Motion and Scale Change

The performance of the algorithm with the subtracted RGB color space is included in Table. 6.17. This color space does not work because RGB value of different shades of gray, which is the color of the trousers of the person, have same RGB value (211,211,211; 192,192,192; 169,169,169; 128,128,128). Moreover the person is person is waring black trousers. Thus all pixel values are centered around 0. The subtracted RGB color space cannot differentiate between the object from the grayish background and, the tracker, in this case, loses its track; but still Bhattacharyya matching coefficient is as large as 0.93 in average. Thus the subtracted color space does not work.

Table 6.17: Case Study 6 : Subtracted Color Space

—	**SIR-PF**	**Bat Algo.**	**Proposed Algo.**	**Proposed Algo. (Sub)**
Mean Error	17.32	16.45	13.2	71.69
Maximum Error	50.21	74.01	42.57	111.8
Std. Deviation	9.70	11.84	7.57	20.75
Success Rate	61.82	72.88	82.95	4.58

6.4.7 Case Study 7 : Tracking Object Under Illumination Variation and Clutter

The performance of the algorithm with the subtracted RGB color space is included in Table. 6.18. Although majority of color is black, still it tracks, using the subtracted RGB color space, because the face-color has different RGB value and the shade of background is not almost black. But still it is inferior to the RGB color space due to the already discussed reasons.

6.4.8 Case Study 8 : Tracking Object Under Scale Change, Partial Occlusion and Clutter

The performance of the algorithm with the subtracted color space is included in Table. 6.19. Since the object contains only black and white, tracking fails in subtracted RGB color space.

Table 6.18: Case Study 7 : Subtracted Color Space

—	SIR-PF	Bat Algo.	Proposed Algo.	Proposed Algo. (Sub)
Mean Error	67.44	90.33	14.2	35.31
Maximum Error	260.52	148.55	43.73	139.28
Std. Deviation	49.3	78.68	10.5	32.70
Success Rate	10.14	15.62	58.63	47.67

Table 6.19: Case Study 8 : Subtracted Color Space

—	SIR-PF	Bat Algo.	Proposed Algo.	Proposed Algo. (Sub)
Mean Error	66.11	60.35	16.35	97.27
Maximum Error	159.11	150.6	31.62	185.06
Std. Deviation	50.84	49.96	7.20	52.23
Success Rate	24.7	35.29	60.82	12.75

6.4.9 Case Study 9 : Tracking Object Under Complete Occlusion and Clutter

The performance of the algorithm with the subtracted color space is included in Table. 6.20. The subtracted color space cannot distinguish between the grayish object and background, and so, the tracker gets trapped.

Table 6.20: Case Study 9 : Subtracted Color Space

—	SIR-PF	Bat Algo.	Proposed Algo.	Proposed Algo. (Sub)
Mean Error	31.94	27.26	13.38	93.69
Maximum Error	49.24	52.77	54.58	135.97
Std. Deviation	11.06	13.25	11.40	23.02
Success Rate	14.14	32.98	81.68	3.14

6.4.10 Analysis

Thus from then test results it has been observed that, if the object has such a color whose RGB values are all different for the color and all its shades, then the subtracted color space works better than the original RGB color space. Otherwise, RGB color space is far better than the subtracted color space. Further it has also been proved that the tracker works in different color spaces also. Clubbing the tracker with more robust color space can result in more accurate tracking.

6.5 Comparison with Other State-of-the-art Trackers

In this section, the proposed tracker is compared with other trackers CPF [18], KMS [8], Frag [58], SCM [59], LOT [55], VTD [60], MIL [61] and IVT [62]. The testing is done 12 times for each video and each time at different starting frame. For other trackers, the results are available in [1, 2]. Out of them, CPF, LOT, SCM and IVT algorithms are particle filter based trackers, but their measurement models are different.

KMS is MeanShift based and CPF is particle filter based tracking algorithm. Frag uses integral histogram as well as the concept of multiple patches inside the template; it uses EMD (Earth Mover Distance) to measure similarity. Similarly, LOT also uses EMD (Earth Mover Distance) to measure the distance between two image patches which are formulated in terms of joint spatial-appearance form. Color values are considered and pixel positions in the patch is normalized within [0 1]. MIL uses a classifier which uses Haar like feature. The MIL appearance model is divided into two groups: positive and negative group, and this group gets updated every time. SCM is intended for tracking objects where drastic changes in appearance of the object happens. It derives sparsity based features from histograms. VTD is also robust to abrupt changes in appearance in the objects tracked. Thus the performance of the proposed algorithm is compared with other contemporary particle filter based algorithms and others. In this section, all results are tabulated. The tests are done on the same video sequences as given above which means that the video sequence used in *Case Study 1* is same as the *Case Study 1* in this section. But, the reported performance of the proposed algorithm is different in this section as compared to the previous sections. This is because the template sizes are different in these cases and also their initial locations are. Throughout the experiments, salmon box is used CPF, green box is used for Frag, white box is used for IVT, cyan box is used for KMS, magenta box is used for LOT, dark orange box is used for MIL, red box is used for SCM, yellow box is used for VTD and blue box is used for the proposed algorithm. For these other state-of-the-art algorithms, test results are taken from the results folder of [1, 2]. With the same setting, the proposed algorithm is run for each video. Then, their results are compared using the same metric. Results are compared according to *Temporal Robustness Evaluation*[1, 2]. In this case, each time the tracker is initialized at a different frame of the same video. The tracker is run up to the last frame of each video 12 times. With the help of the ground truth data [1, 2], average results are computed for each video i.e. average result for all the 12 experiments for each video which start at 12 different initial frames.

6.5.1 Case Study 1 : Tracking Object Under Illumination Variation, Occlusion and Clutter

The performance comparison among the algorithms is given in Table. 6.21. Fig. 6.44, Fig. 6.45, Fig. 6.46 and Fig. 6.47 show sample sequences of tracking. The proposed algorithm tracks better than all other algorithms except for VTD tracker. VTD basically decomposes the models and so the performance is better to some extent.

6.5.2 Case Study 2 : Tracking Object Under Clutter, Fast Motion and Scale Change

The performance comparison among the algorithms is given in Table. 6.22. Fig. 6.48, Fig. 6.49, Fig. 6.50 and Fig. 6.51 show sample sequences of tracking. In this case, although the proposed algorithm's performance is found to be the best, both CPF and the proposed algorithm works with almost same accuracy.

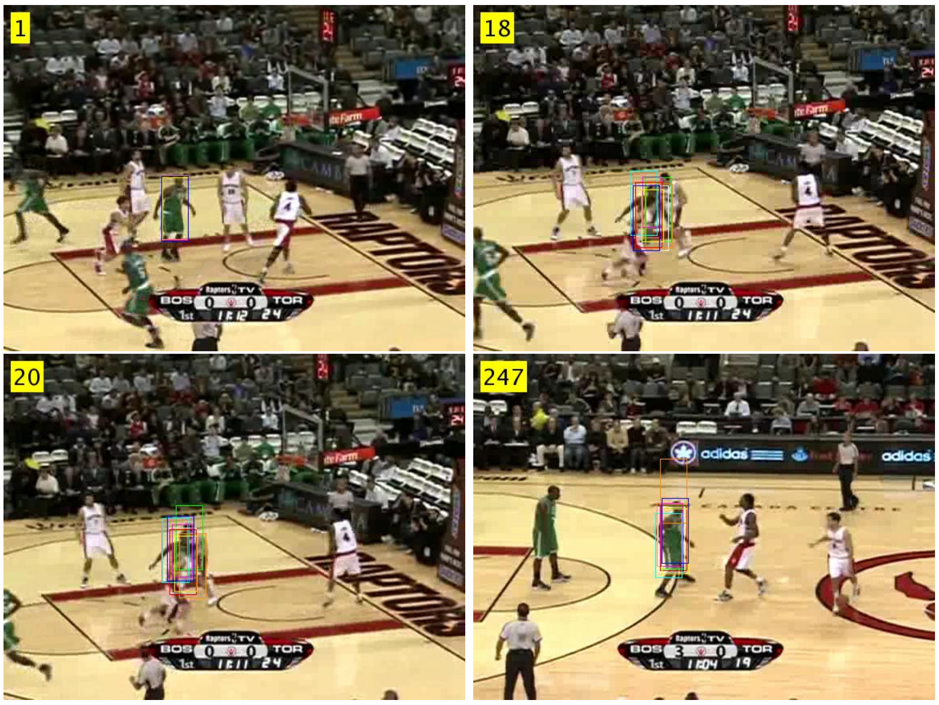

Figure 6.44: Tracking sequences for Table. 6.21

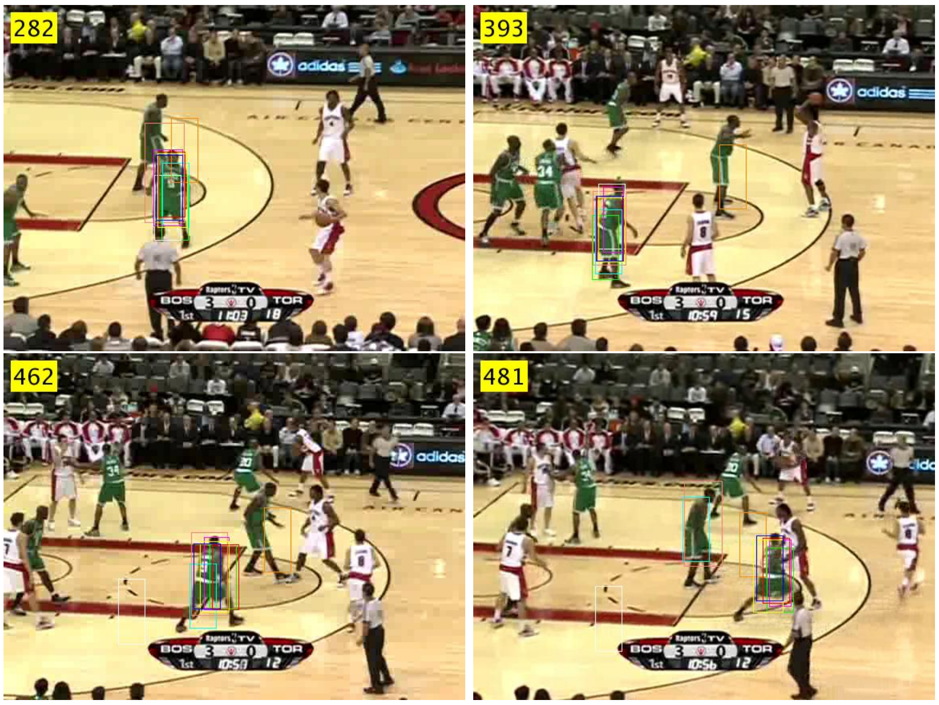

Figure 6.45: Tracking sequences for Table. 6.21

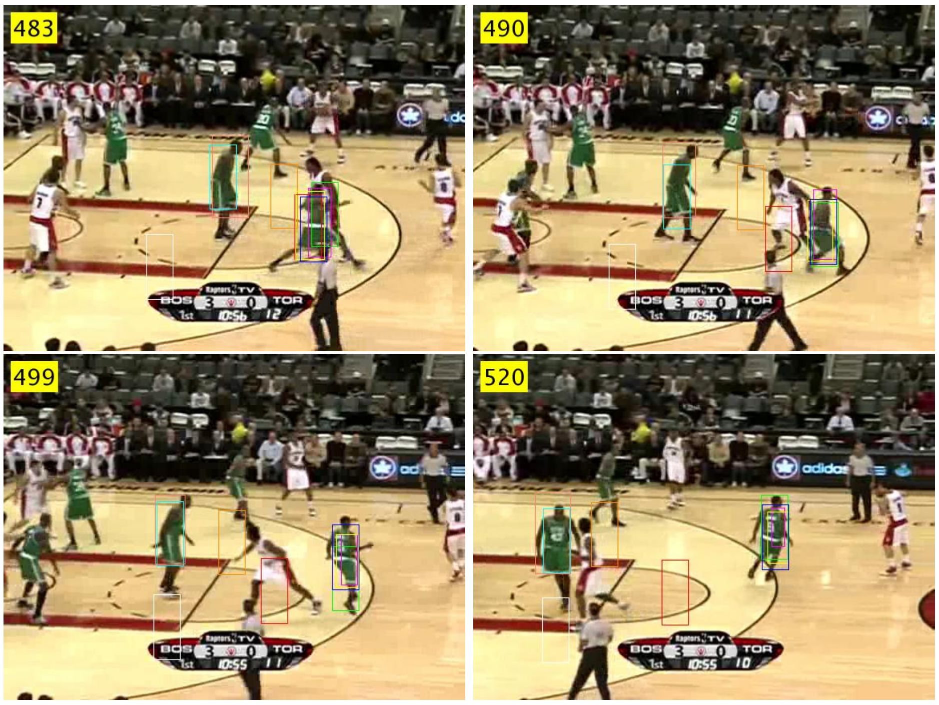

Figure 6.46: Tracking sequences for Table. 6.21

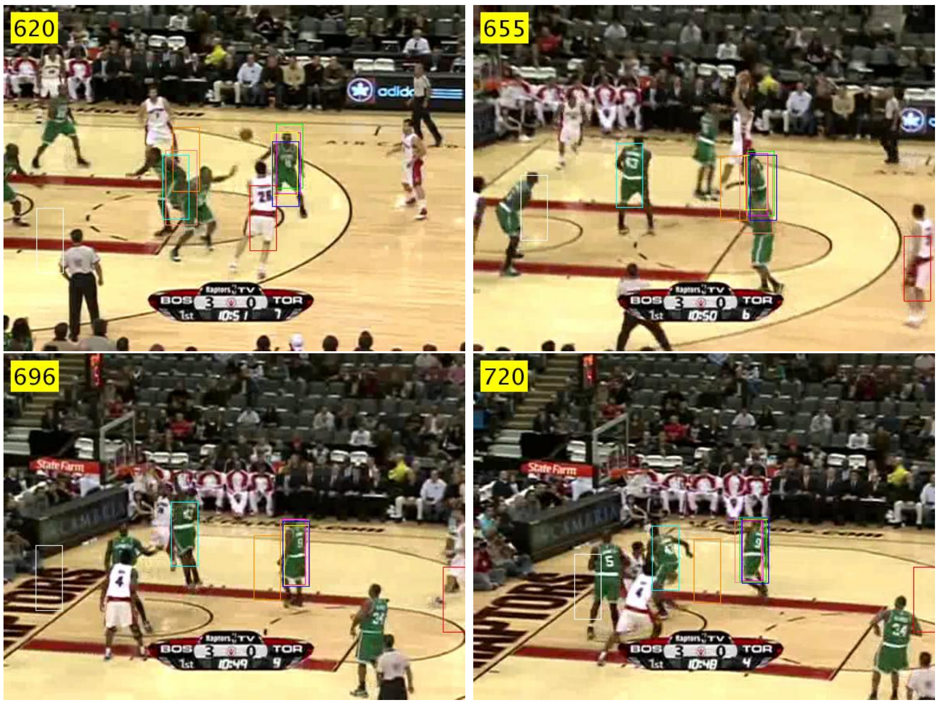

Figure 6.47: Tracking sequences for Table. 6.21

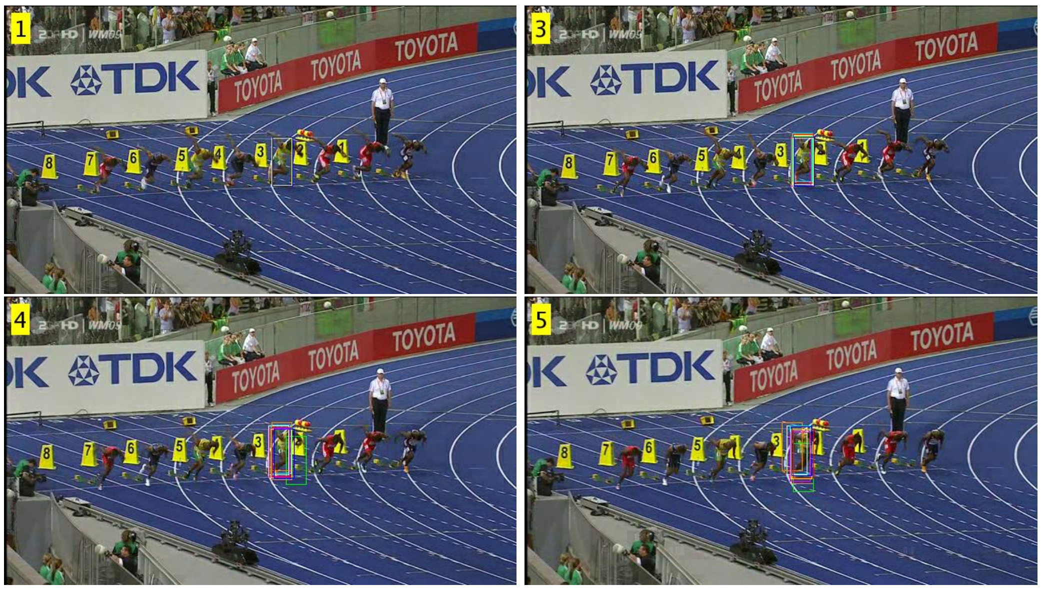

Figure 6.48: Tracking sequences for Table. 6.22

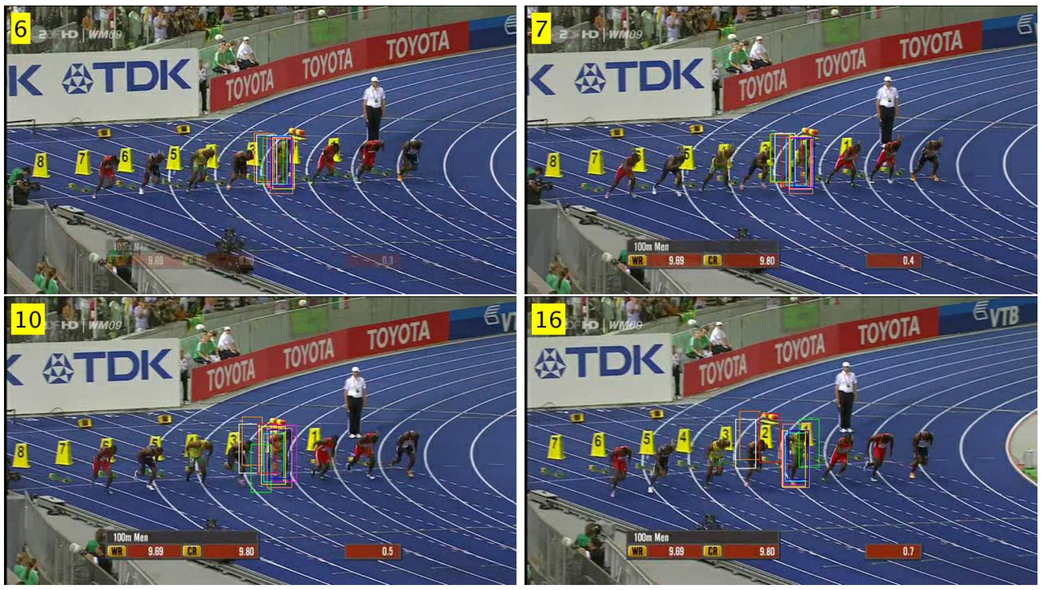

Figure 6.49: Tracking sequences for Table. 6.22

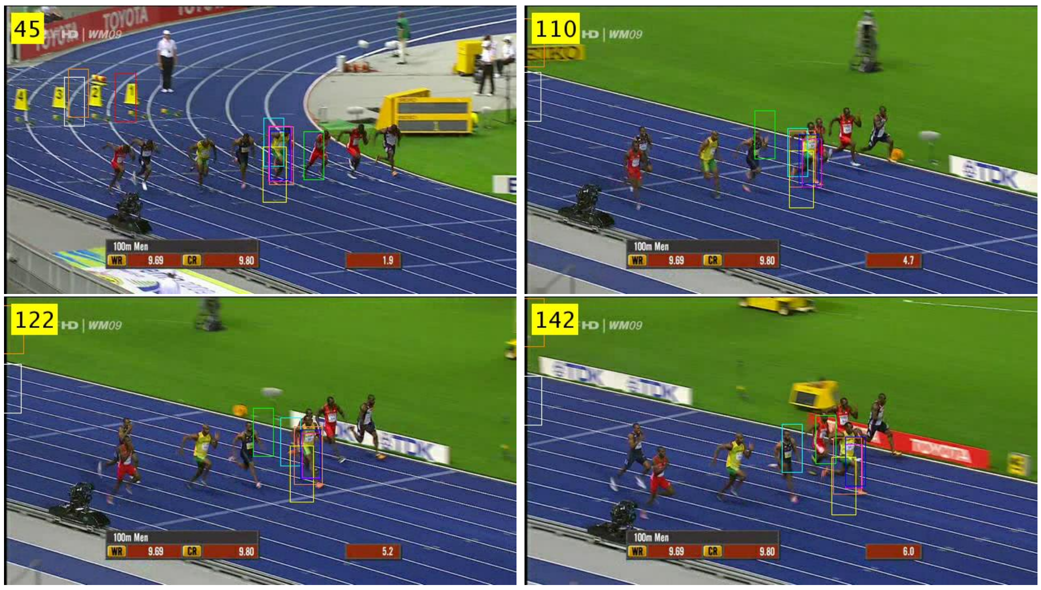

Figure 6.50: Tracking sequences for Table. 6.22

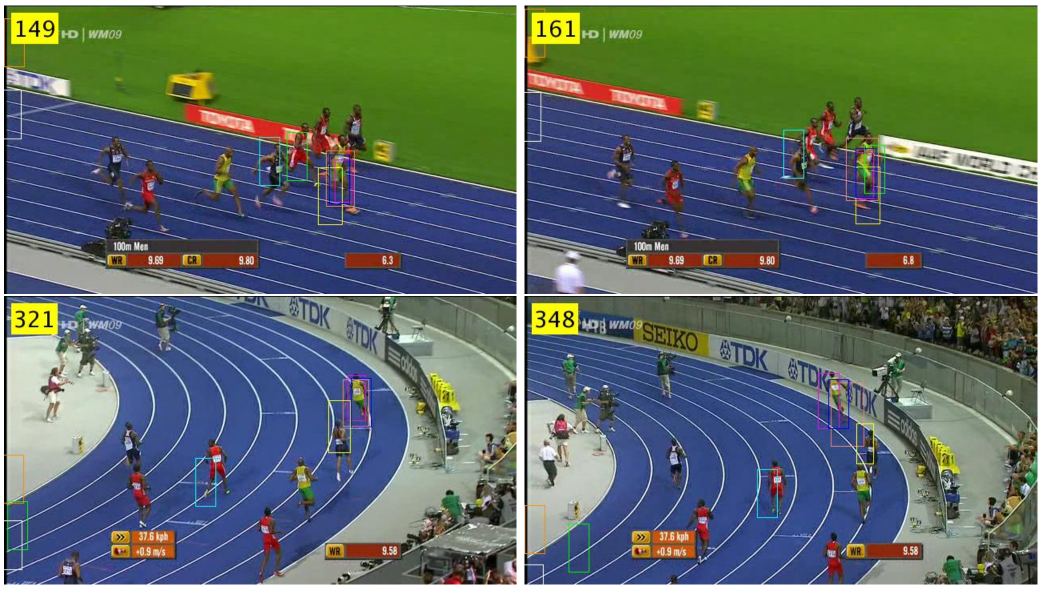

Figure 6.51: Tracking sequences for Table. 6.22

Table 6.21: Case Study 1 : Comparative Study

—	**Mean Error**	**Maximum Error**	**Std. Deviation**	**Success Rate**
CPF	41.47	149.36	43.4	75.71
Frag	47.11	183.84	49.78	67.80
IVT	57.13	186.27	55.15	62.42
KMS	79.42	259.18	79.65	60.19
LOT	41.00	212.02	54.13	78.67
MIL	56.65	172.91	47.12	59.80
SCM	99.58	260.65	80.63	45.80
VTD	8.13	38.15	6.66	**98.35**
Proposed	7.79	98.87	6.70	**97.96**

Table 6.22: Case Study 2 : Comparative Study

—	**Mean Error**	**Maximum Error**	**Std. Deviation**	**Success Rate**
CPF	18.99	46.75	12.56	**92.56**
Frag	160.91	357.01	124.14	18.22
IVT	160.43	285.42	76.59	15.65
KMS	114.63	233.89	78.91	25.63
LOT	70.56	141.72	44.56	80.81
MIL	213.54	418.65	147.83	30.22
SCM	98.54	233.49	73.69	38.41
VTD	39.38	61.94	13.74	84.58
Proposed	14.18	14.18	9.55	**92.92**

6.5.3 Case Study 3 : Tracking Object Under Motion Blur, Fast Motion and Scale Change

The performance comparison among the algorithms is given in Table. 6.23. Fig. 6.52, Fig. 6.53, Fig. 6.54 and Fig. 6.55 show sample sequences of tracking. In this case, CPF performs the best, then VTD and then the proposed algorithm. The reason for the comparatively lower performance of the proposed algorithm is due to the template chosen where a part of the skin color (face portion) remains in the background; so this portion is suppressed. This reduces the performance of the proposed algorithm a bit.

6.5.4 Case Study 4 : Tracking Object Under Occlusion, Fast Motion and Clutter

The performance comparison among the algorithms is given in Table. 6.24. Fig. 6.56, Fig. 6.57, Fig. 6.58 and Fig. 6.59 show sample sequences of tracking. The order of performance (starting from the best) is : CPF, proposed algorithm and IVT. The reason behind the proposed algorithm's comparatively lower performance is the color difference between the background and foreground is not separable and hence, foreground colors are suppressed.

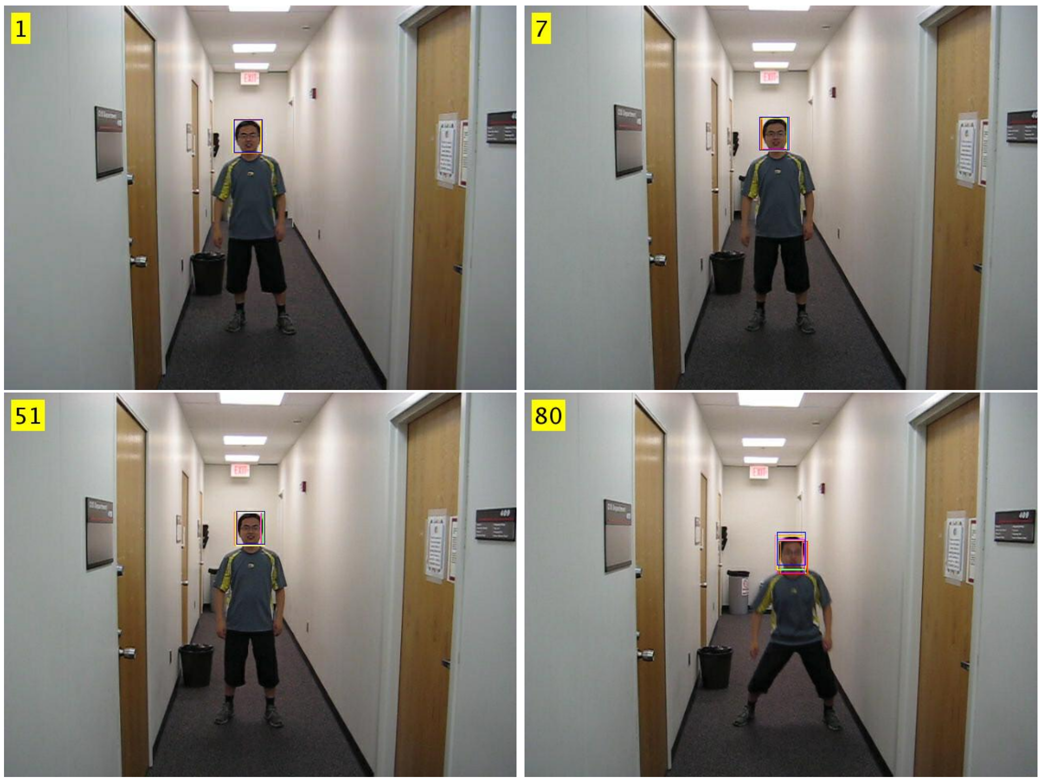

Figure 6.52: Tracking sequences for Table. 6.23

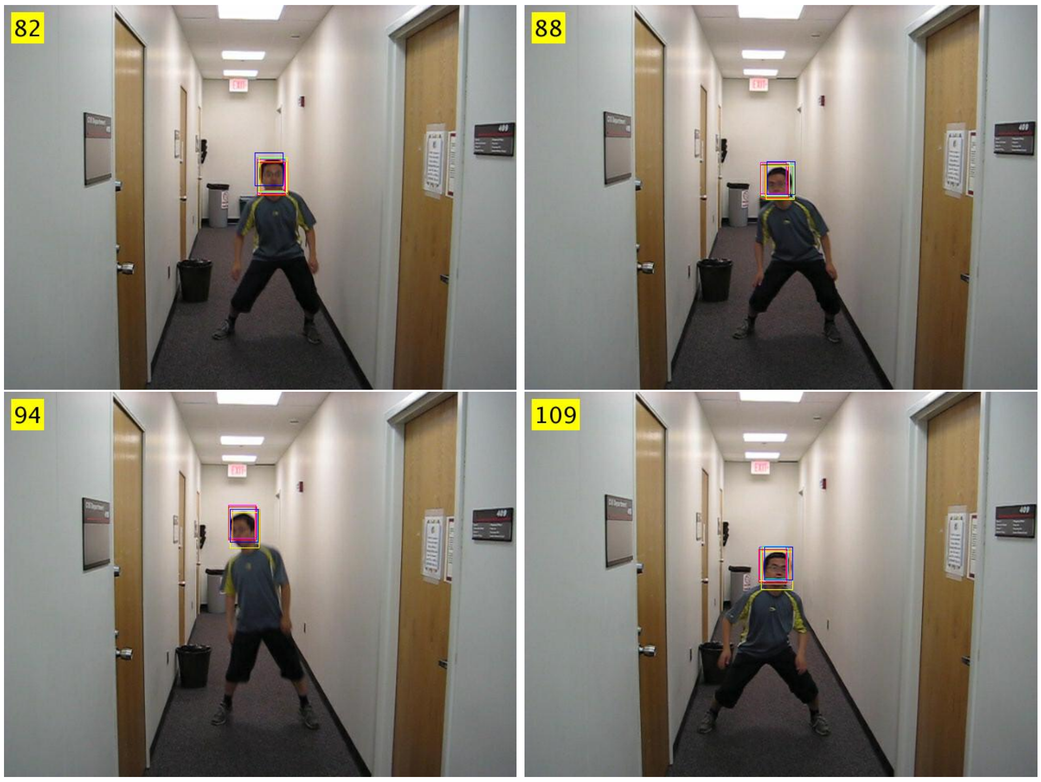

Figure 6.53: Tracking sequences for Table. 6.23

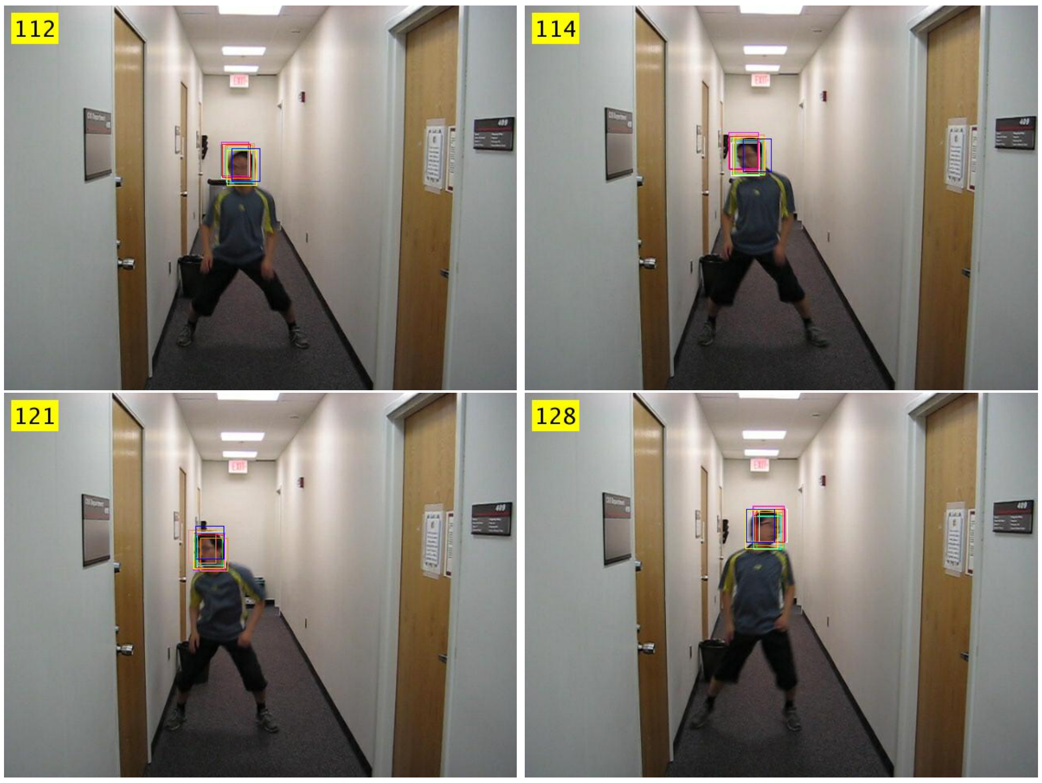

Figure 6.54: Tracking sequences for Table. 6.23

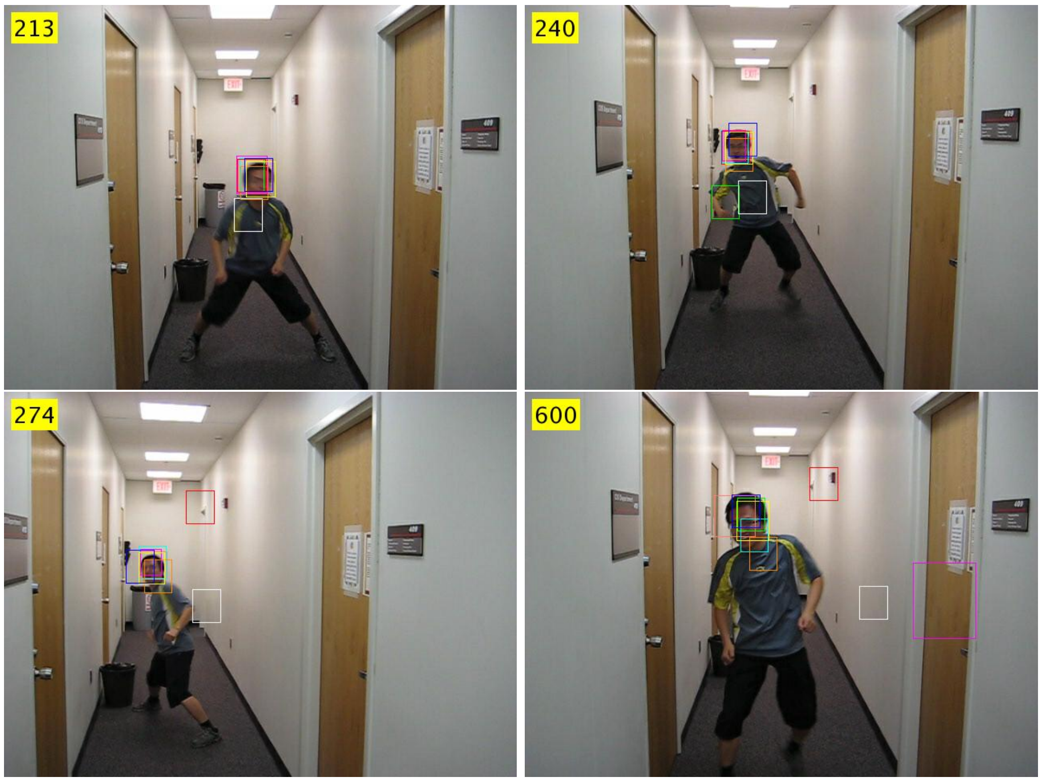

Figure 6.55: Tracking sequences for Table. 6.23

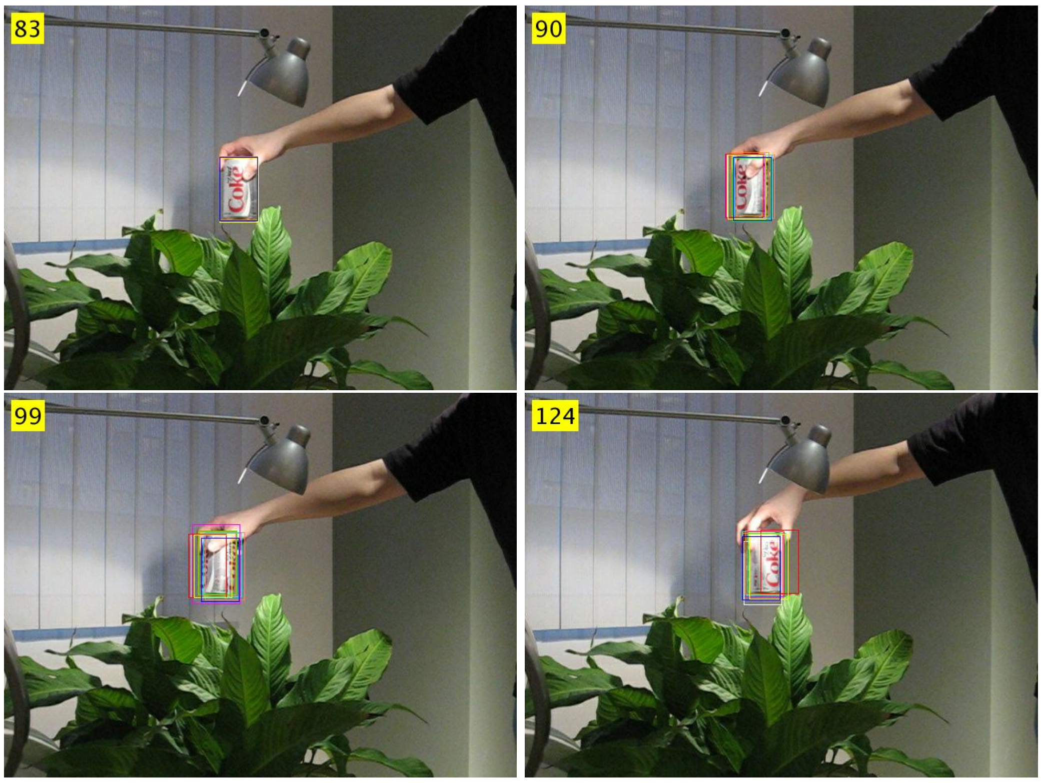

Figure 6.56: Tracking sequences for Table. 6.24

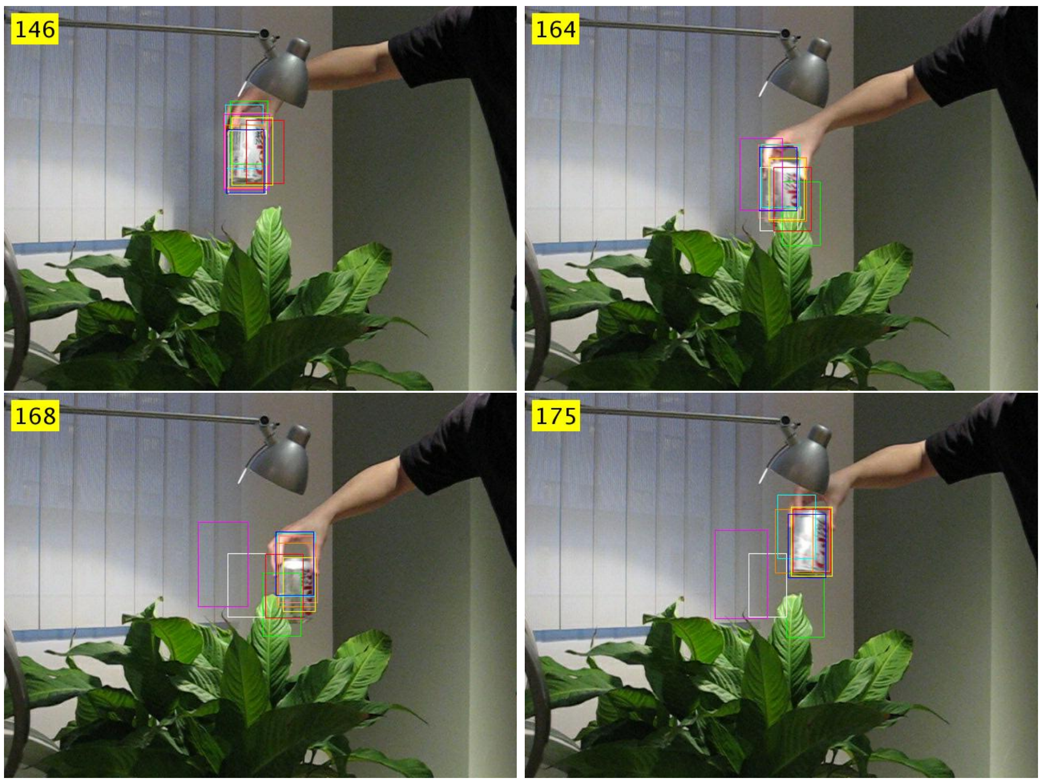

Figure 6.57: Tracking sequences for Table. 6.24

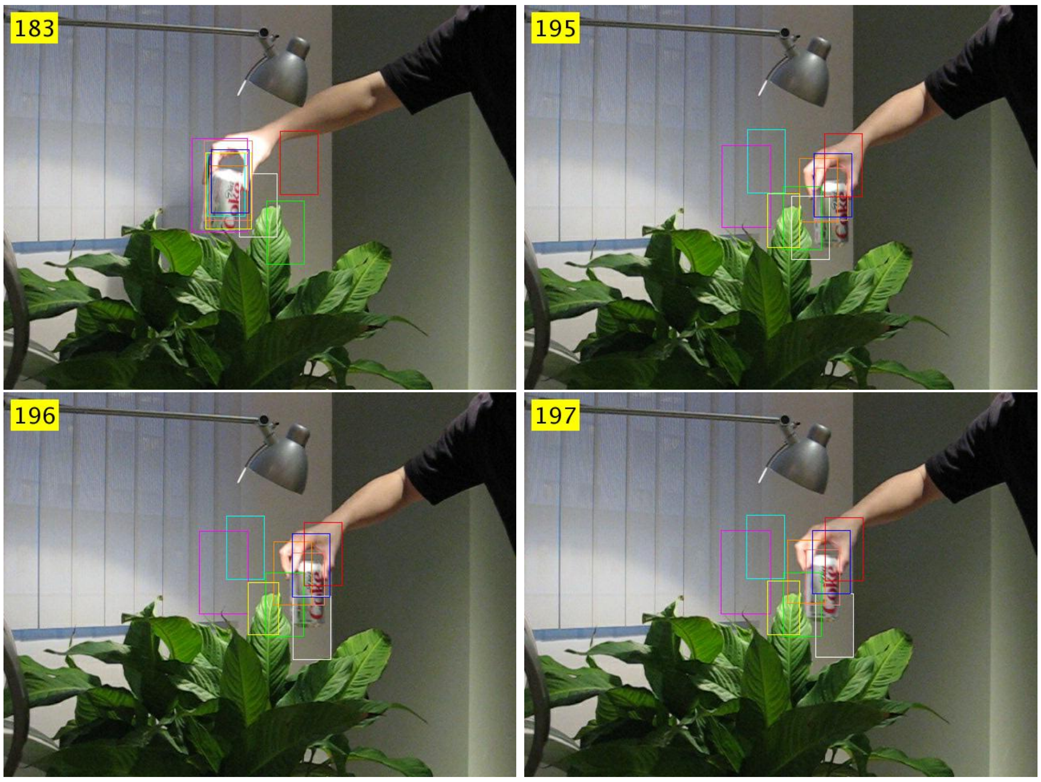

Figure 6.58: Tracking sequences for Table. 6.24

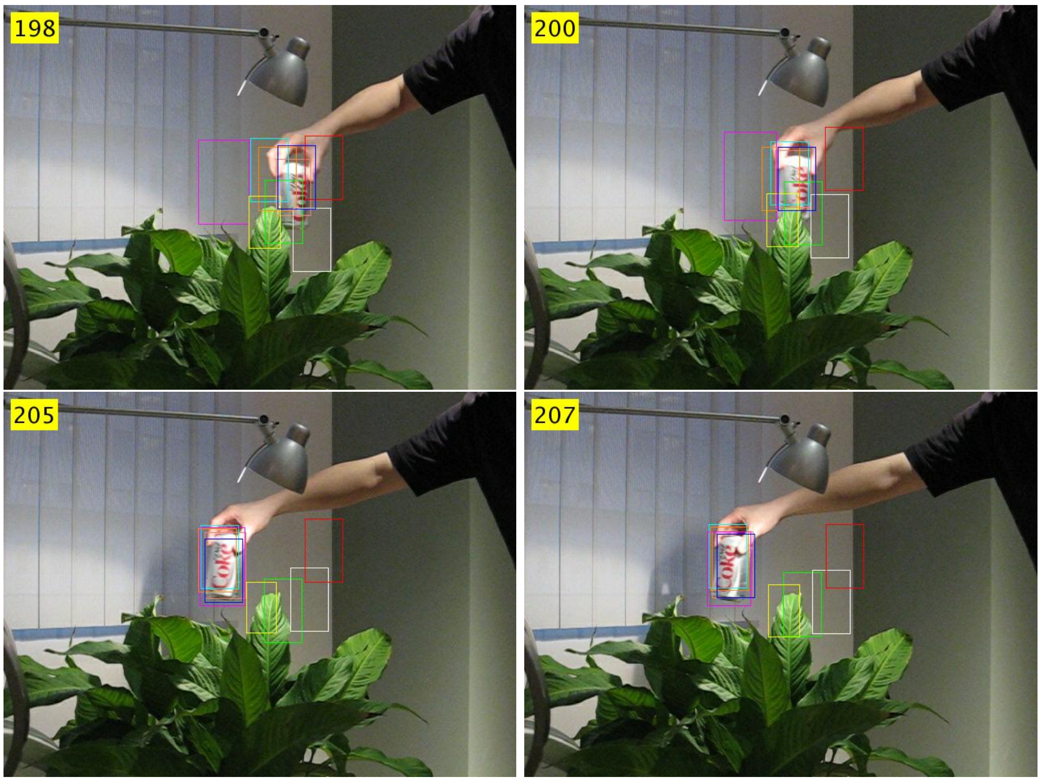

Figure 6.59: Tracking sequences for Table. 6.24

Table 6.23: Case Study 3 : Comparative Study

—	Mean Error	Maximum Error	Std. Deviation	Success Rate
CPF	5.15	17.69	3.12	**99.87**
Frag	47.88	172.78	43.64	50.82
IVT	86.05	236.03	65.58	24.15
KMS	5.14	23.64	3.86	99.18
LOT	41.44	139.77	46.22	75.40
MIL	15.05	62.59	13.88	83.37
SCM	77.98	197.50	59.60	29.35
VTD	8.91	26.78	5.24	**95.89**
Proposed	10.18	34.17	5.27	**95.72**

Table 6.24: Case Study 4 : Comparative Study

—	Mean Error	Maximum Error	Std. Deviation	Success Rate
CPF	31.32	143.25	33.80	**62.55**
Frag	42.64	120.30	26.28	**47.10**
IVT	72.79	171.60	44.15	24.72
KMS	46.49	161.00	40.79	45.28
LOT	94.62	257.05	79.18	39.89
MIL	35.79	87.68	21.29	43.12
SCM	53.60	140.90	35.91	30.62
VTD	41.09	119.20	28.31	39.28
Proposed	49.03	242.4	59.69	**47.42**

6.5.5 Case Study 5 : Tracking Object Under Occlusion and Clutter

The performance comparison among the algorithms is given in Table. 6.25. Fig. 6.60, Fig. 6.61, Fig. 6.62 and Fig. 6.63 show sample sequences of tracking. The proposed algorithm outperforms other algorithms in this case study.

6.5.6 Case Study 6 : Tracking Object Under Occlusion, Fast Motion and Scale Change

The performance comparison among the algorithms is given in Table. 6.26. Fig. 6.64, Fig. 6.65, Fig. 6.66 and Fig. 6.67 show sample sequences of tracking. The order of performance (starting from the best) is : CPF, Frag and proposed algorithm. This reason for this is that when the scale of the object exceeds the tightly fit template, foreground color appears in the background and is suppressed in the proposed algorithm.

6.5.7 Case Study 7 : Tracking Object Under Illumination Variation and Clutter

The performance comparison among the algorithms is given in Table. 6.27. Fig. 6.68, Fig. 6.69, Fig. 6.70 and Fig. 6.71 show sample sequences of tracking. The order of performance is (starting with

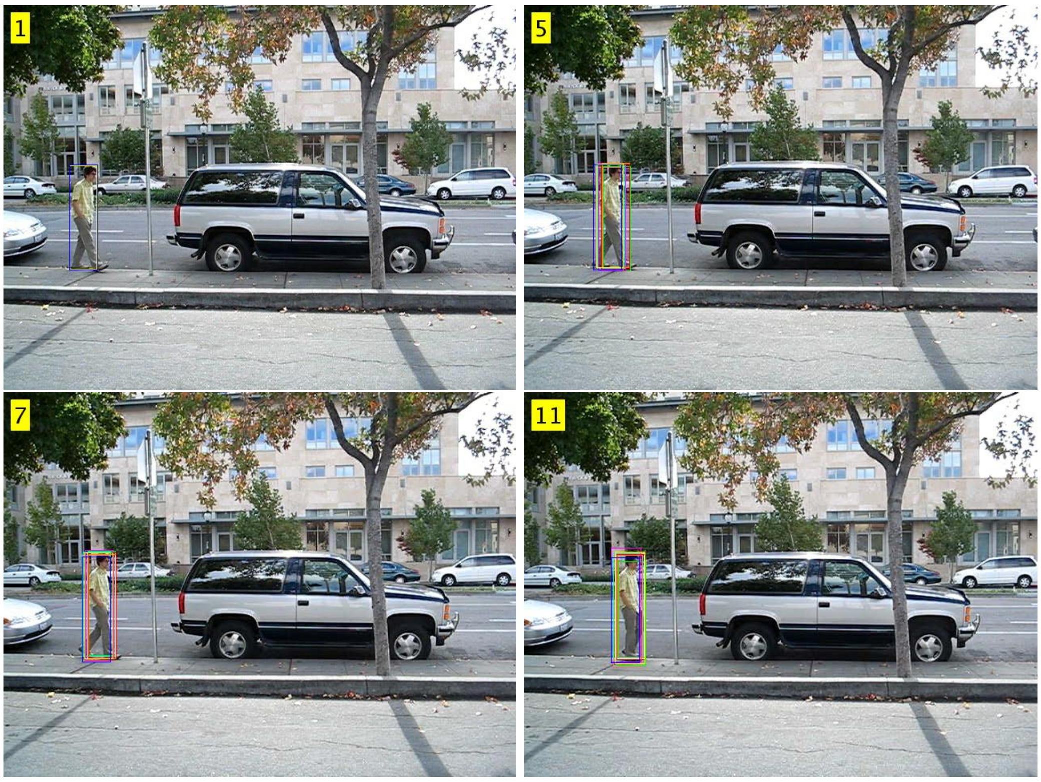

Figure 6.60: Tracking sequences for Table. 6.25

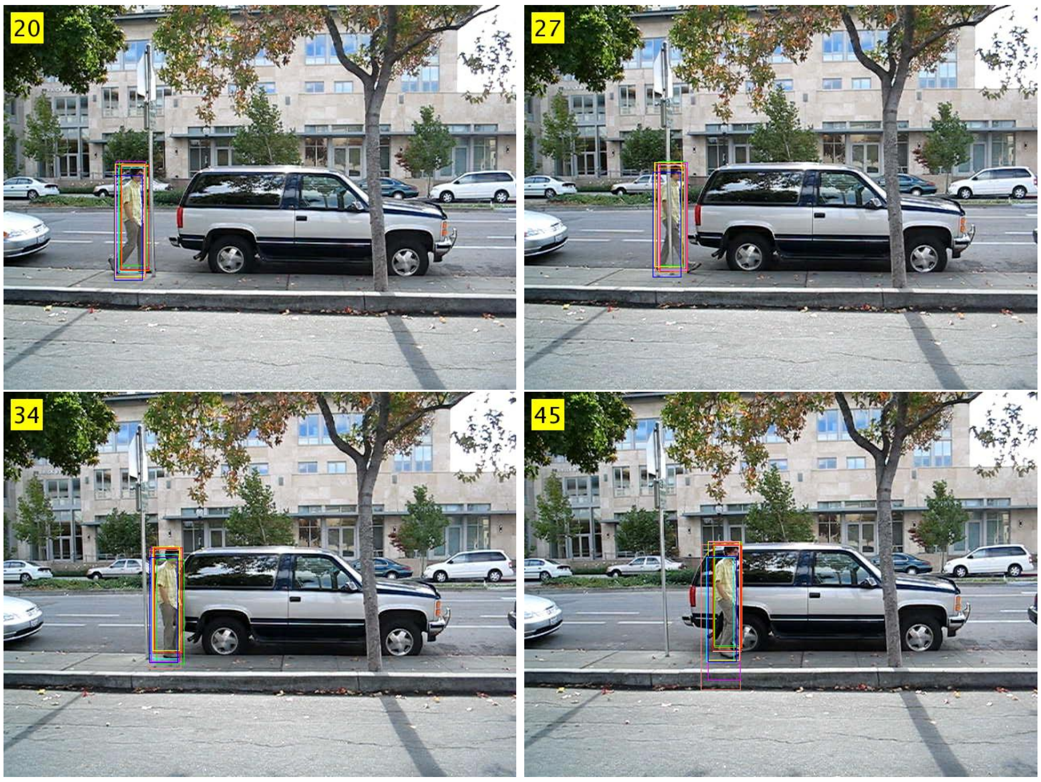

Figure 6.61: Tracking sequences for Table. 6.25

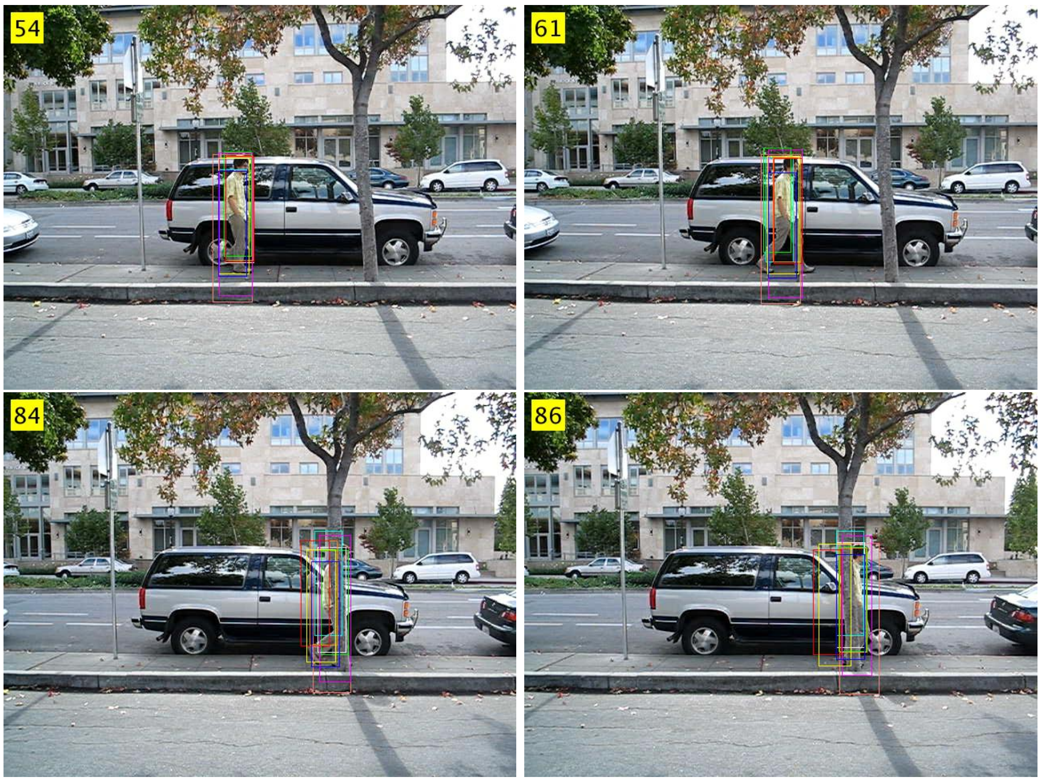

Figure 6.62: Tracking sequences for Table. 6.25

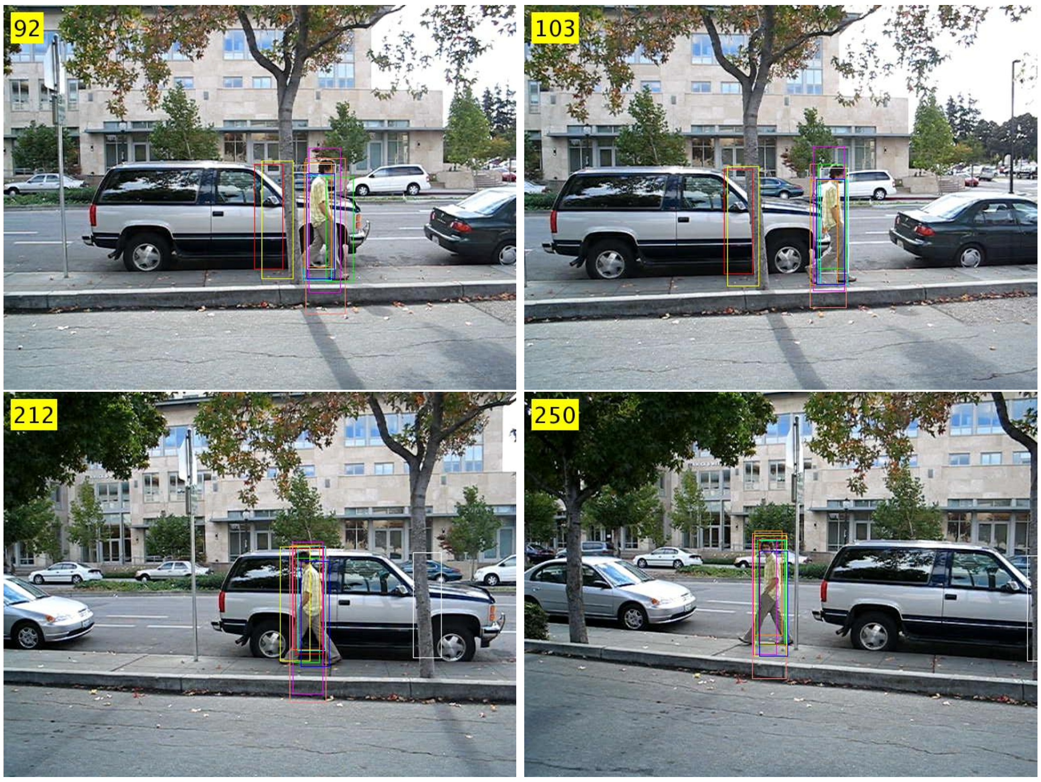

Figure 6.63: Tracking sequences for Table. 6.25

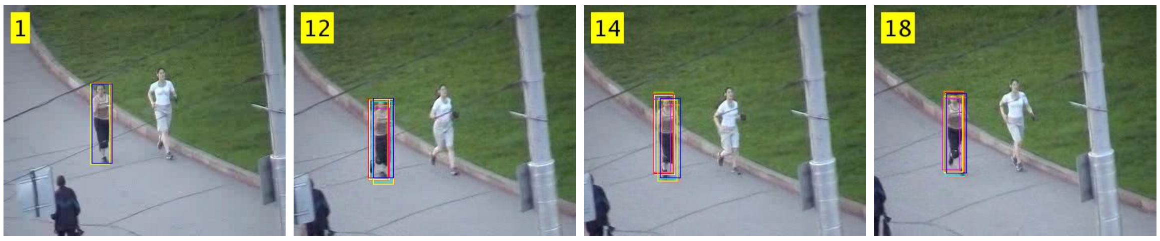

Figure 6.64: Tracking sequences for Table. 6.26

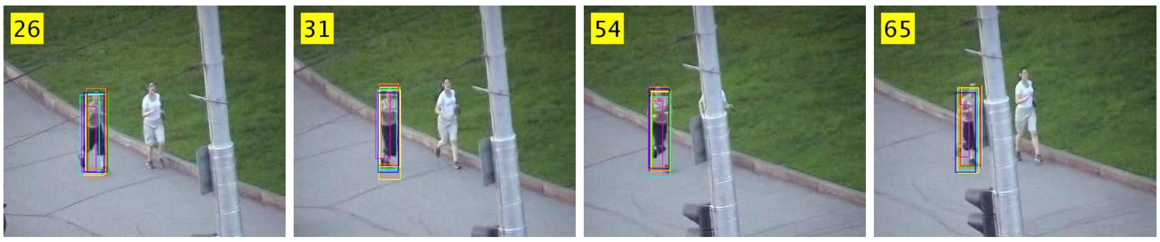

Figure 6.65: Tracking sequences for Table. 6.26

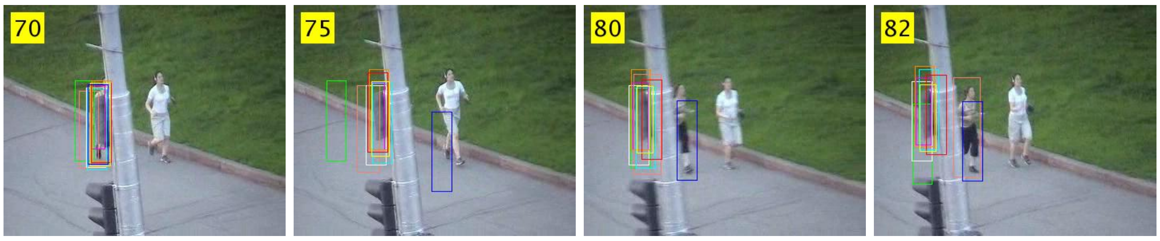

Figure 6.66: Tracking sequences for Table. 6.26

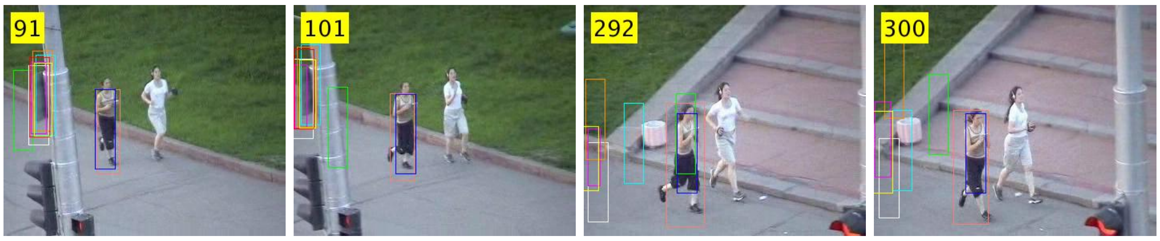

Figure 6.67: Tracking sequences for Table. 6.26

Table 6.25: Case Study 5 : Comparative Study

—	Mean Error	Maximum Error	Std. Deviation	Success Rate
CPF	9.38	21.01	4.30	**92.58**
Frag	83.65	241.46	79.15	48.52
IVT	71.64	288.83	89.61	64.25
KMS	9.55	32.23	5.87	93.56
LOT	9.99	23.44	5.23	91.50
MIL	89.57	261.57	85.04	43.00
SCM	75.45	200.21	65.26	59.15
VTD	83.77	282.76	96.49	50.74
Proposed	13.1	21.78	4.49	**95.57**

Table 6.26: Case Study 6 : Comparative Study

—	Mean Error	Maximum Error	Std. Deviation	Success Rate
CPF	6.45	26.67	4.31	**94.79**
Frag	14.28	68.44	16.53	**86.79**
IVT	42.24	67.52	18.88	64.51
KMS	42.80	72.26	19.08	63.81
LOT	48.46	129.27	31.53	57.10
MIL	54.71	84.68	25.98	64.65
SCM	50.30	84.58	25.86	71.92
VTD	44.35	71.26	20.19	64.56
Proposed	11.96	50.65	8.30	**95.36**

the best one) : VTD, SCM and proposed algorithm. The performance of the proposed algorithm is a bit reduced by the choice of improper template size.

6.5.8 Case Study 11 : Tracking Object Under Deformation, Clutter, Scale Variation and Fast Motion

The performance comparison among the algorithms is given in Table. 6.28. Fig. 6.72, Fig. 6.73, Fig. 6.74 and Fig. 6.75 show sample sequences of tracking. In this case study, the proposed algorithm works better than the CPF algorithm, and both MIL and SCM tracks with equal accuracy in tracking. KMS, which is MeanShift based deterministic tracking algorithm, performs little bit better than the proposed algorithm.

6.5.9 Case Study 12 : Tracking Object Under Clutter, In-plane Rotation and Out-plane Rotation

The performance comparison among the algorithms is given in Table. 6.29. Fig. 6.76, Fig. 6.77, Fig. 6.78 and Fig. 6.79 show sample sequences of tracking. The order of the performance (starting from the best) is : IVT, VTD and proposed algorithm.

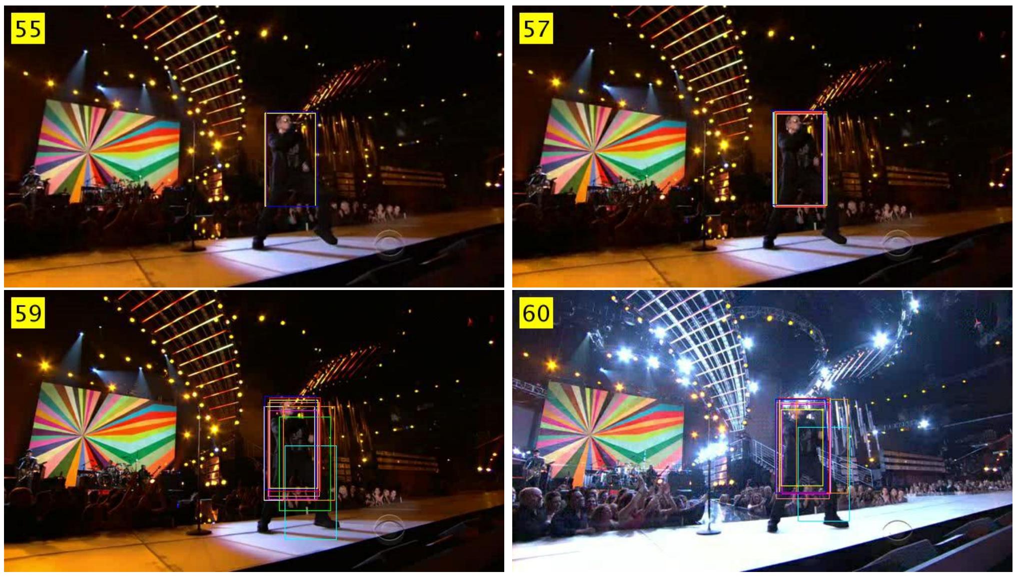

Figure 6.68: Tracking sequences for Table. 6.27

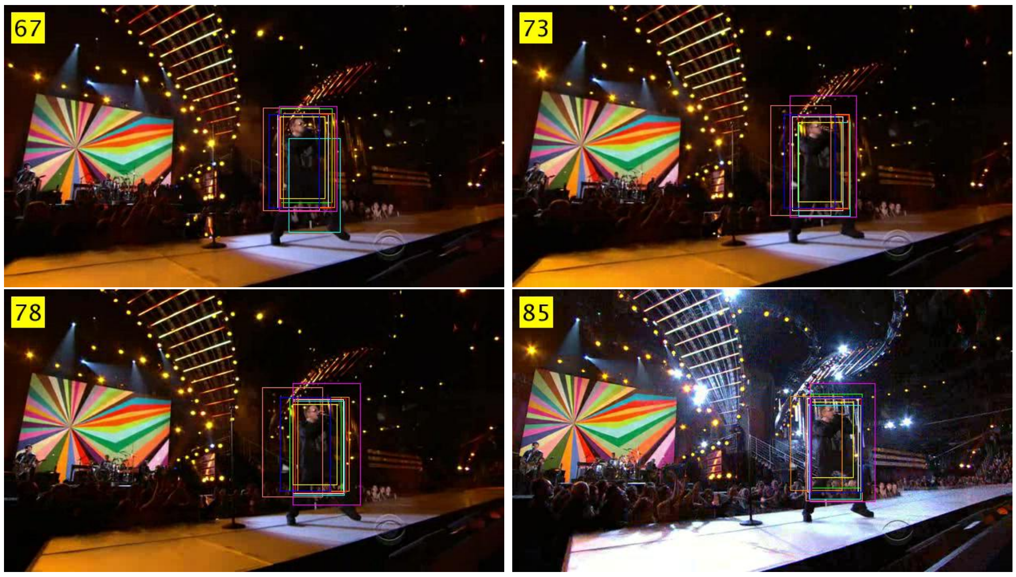

Figure 6.69: Tracking sequences for Table. 6.27

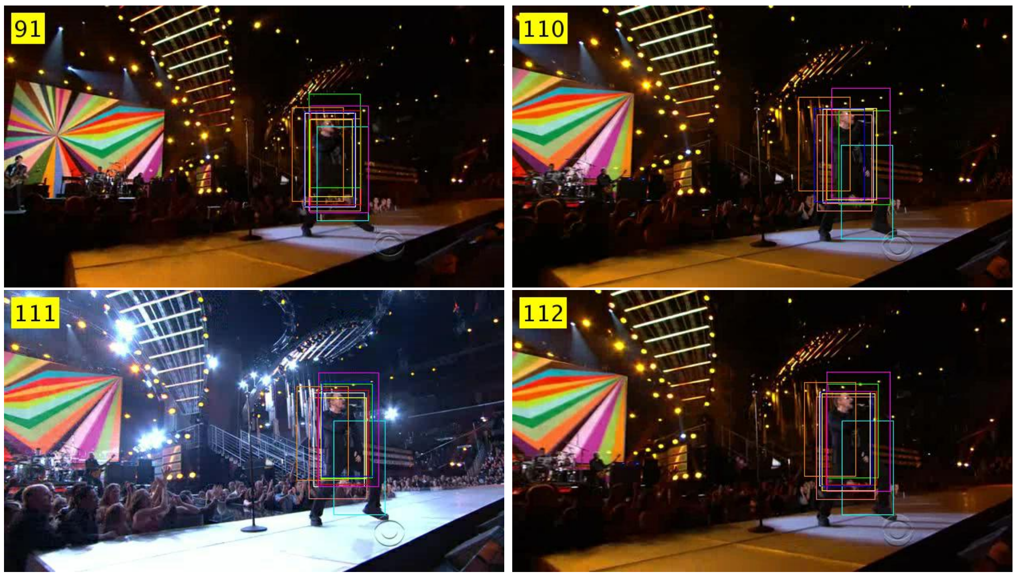

Figure 6.70: Tracking sequences for Table. 6.27

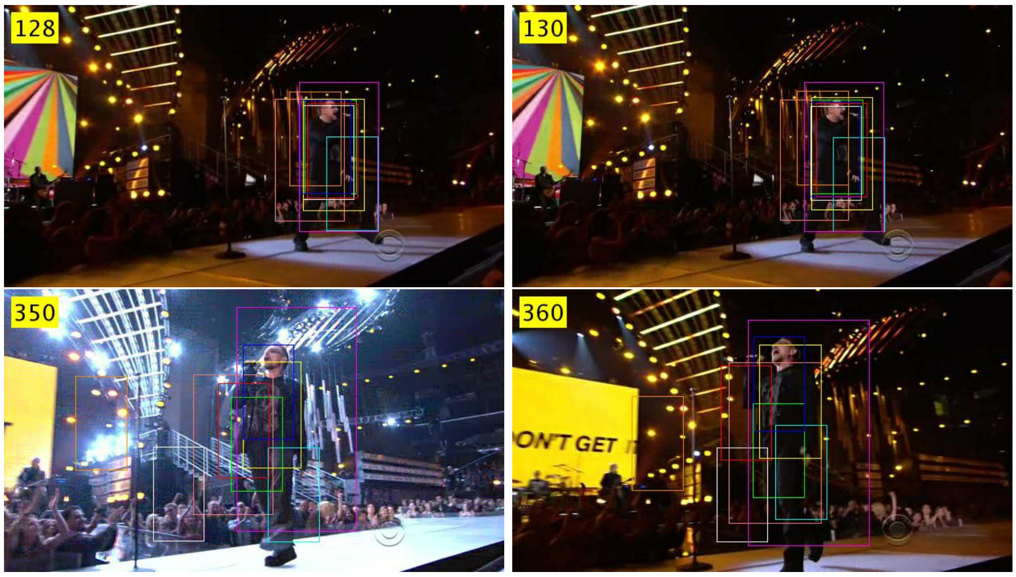

Figure 6.71: Tracking sequences for Table. 6.27

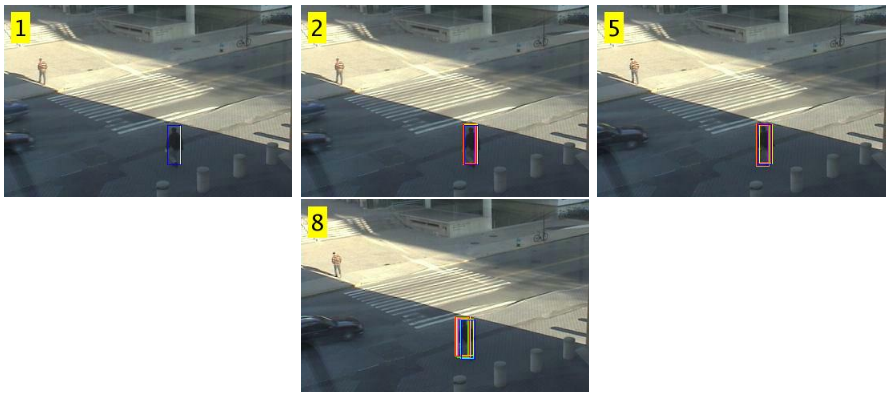

Figure 6.72: Tracking sequences for Table. 6.28

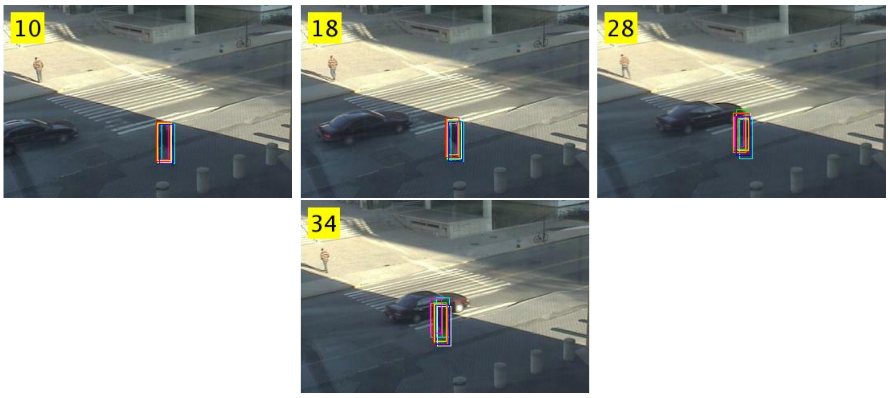

Figure 6.73: Tracking sequences for Table. 6.28

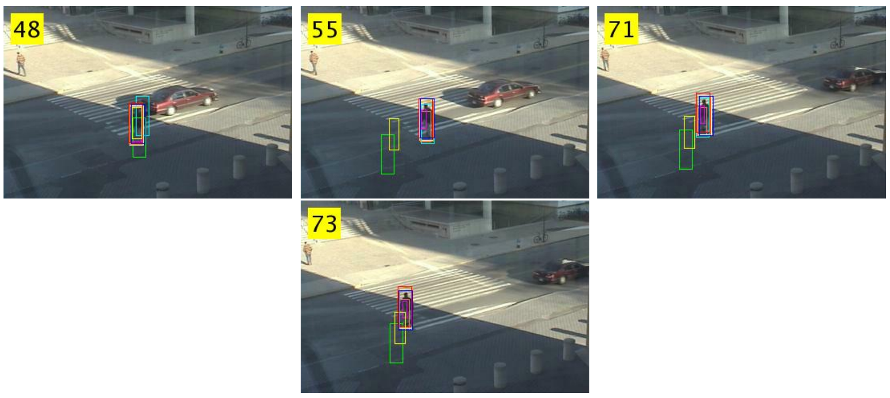

Figure 6.74: Tracking sequences for Table. 6.28

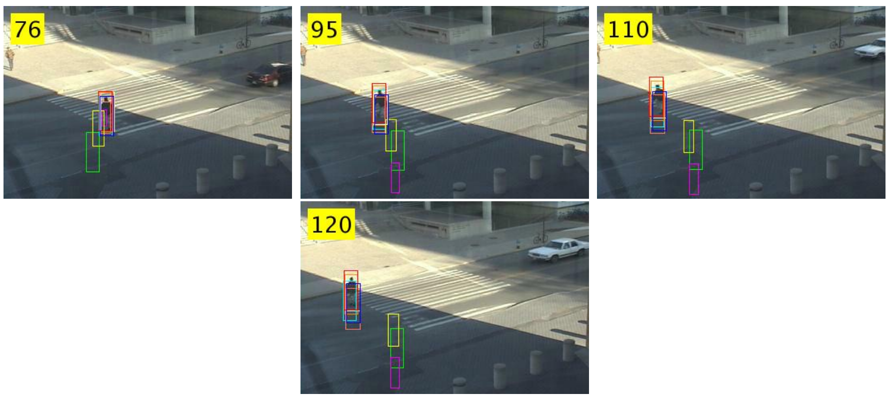

Figure 6.75: Tracking sequences for Table. 6.28

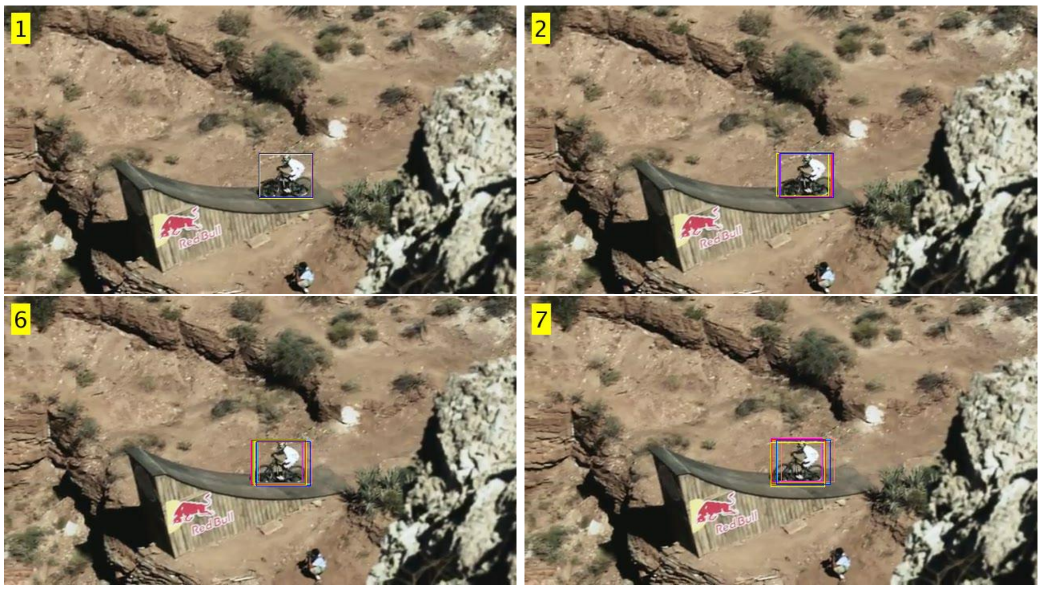

Figure 6.76: Tracking sequences for Table. 6.29

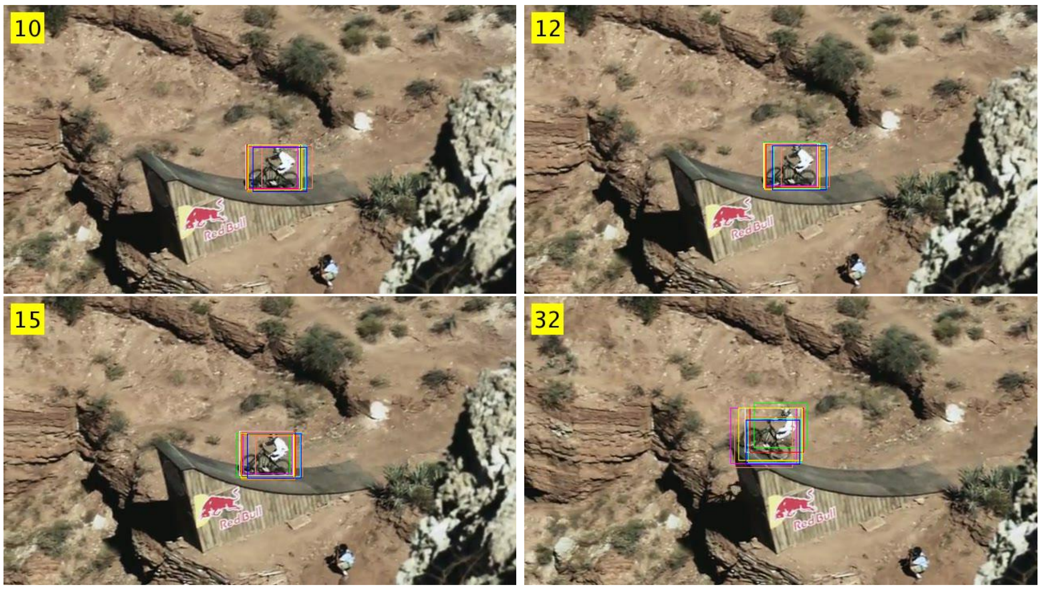

Figure 6.77: Tracking sequences for Table. 6.29

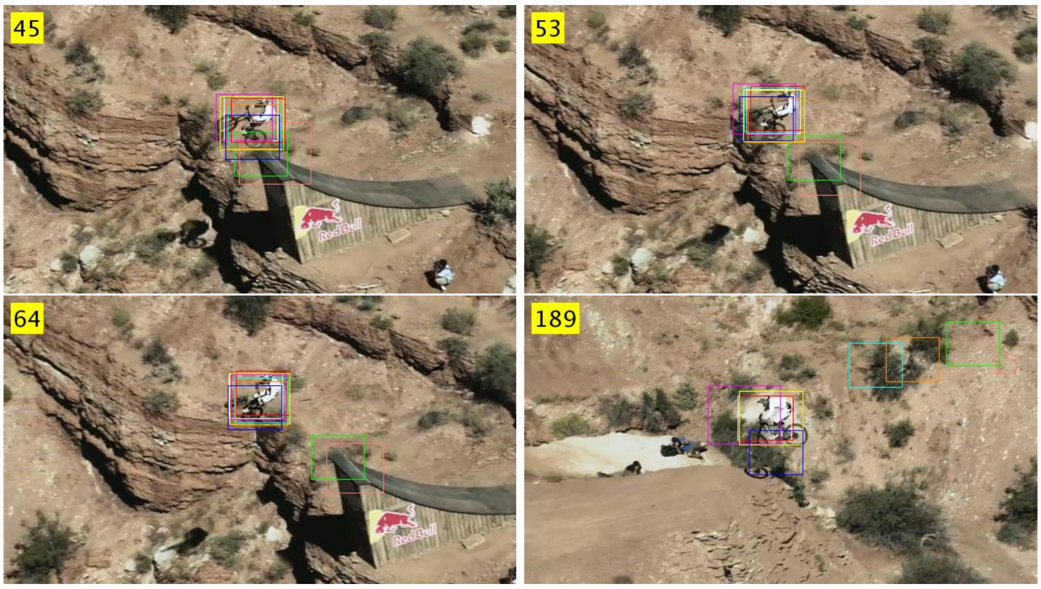

Figure 6.78: Tracking sequences for Table. 6.29

Table 6.27: Case Study 7 : Comparative Study

—	**Mean Error**	**Maximum Error**	**Std. Deviation**	**Success Rate**
CPF	34.46	82.29	18.82	29.36
Frag	76.75	193.82	49.76	18.84
IVT	76.55	201.36	59.09	35.55
KMS	71.05	144.41	39.62	16.27
LOT	36.38	98.40	27.15	50.29
MIL	33.74	106.49	27.35	52.55
SCM	41.92	120.93	35.65	**56.61**
VTD	13.48	38.42	8.27	**88.27**
Proposed	26.30	72.15	17.81	**53.99**

Table 6.28: Case Study 11 : Comparative Study

—	**Mean Error**	**Maximum Error**	**Std. Deviation**	**Success Rate**
CPF	13.75	41.51	10.92	84.70
Frag	76.75	193.82	49.76	50.99
IVT	9.30	18.03	4.77	**93.17**
KMS	9.73	19.60	4.65	**95.96**
LOT	52.81	126.94	43.21	44.20
MIL	3.85	7.85	1.94	**100**
SCM	1.63	4.08	0.84	**100**
VTD	28.51	74.40	23.39	55.97
Proposed	8.4	24.37	6.23	**93.79**

6.5.10 Case Study 13 : Tracking Object Under Clutter, Object Deformation and Occlusion

The performance comparison among the algorithms is given in Table. 6.30. Fig. 6.80, Fig. 6.81, Fig. 6.82 and Fig. 6.83 show sample sequences of tracking. The performance of the proposed algorithm is the best in this case.

6.5.11 Case Study 14 : Tracking Object Under Illumination Variation, Motion Blur, Fast Motion, Background Clutter and Low Resolution

The performance comparison among the algorithms is given in Table. 6.31. Fig. 6.84, Fig. 6.85, Fig. 6.86 and Fig. 6.87 show sample sequences of tracking. The performance of the proposed algorithm is the best in this case. Since, in some frames, the object goes out of bound, those frames cannot be used for initializing the templates. Hence, some frames are selected for initializations: 31, 39, 46, 54, 62, 69, 115, 130, 138 and 145. The performance of the proposed algorithm degrades to some extent because the object goes out of the bound of the frame's search space. The search space is a region inside the frame with a vertical and horizontal margin equal to the vertical and horizontal dimensions of the template. Thus the *centroid* of the template moves within this search space randomly. Objects out of this search space cannot be tracked by the proposed tracker.

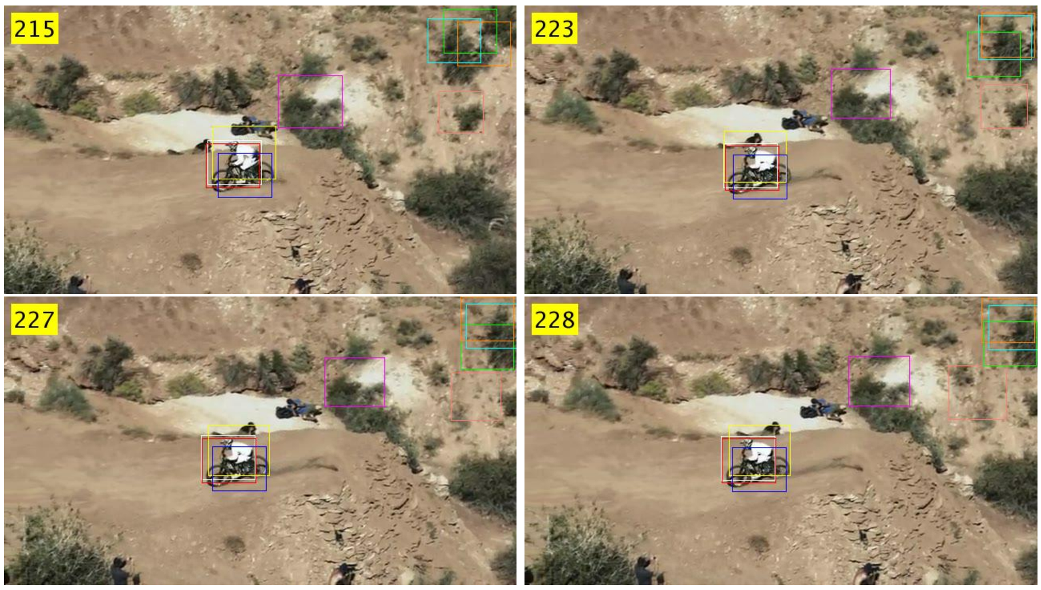

Figure 6.79: Tracking sequences for Table. 6.29

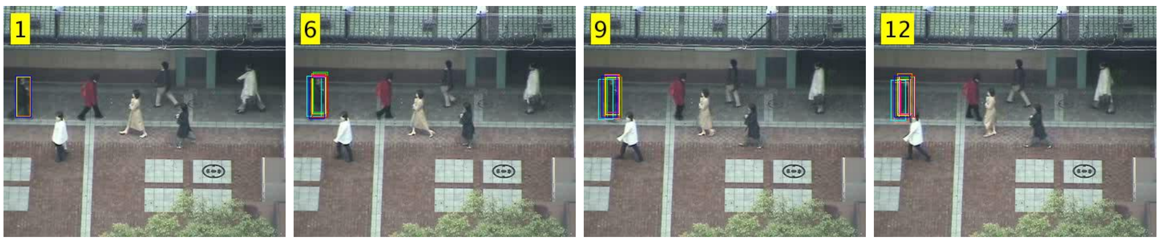

Figure 6.80: Tracking sequences for Table. 6.30

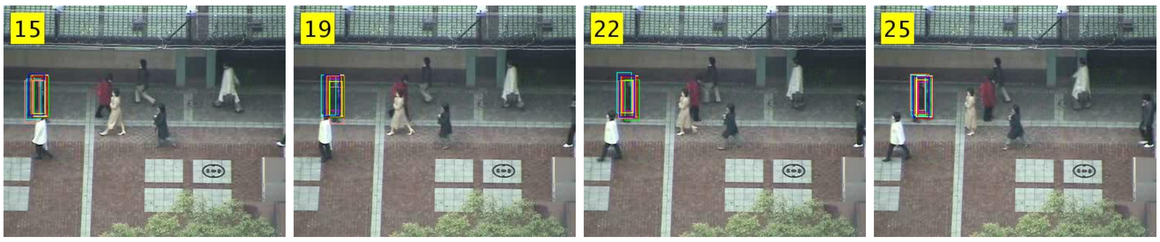

Figure 6.81: Tracking sequences for Table. 6.30

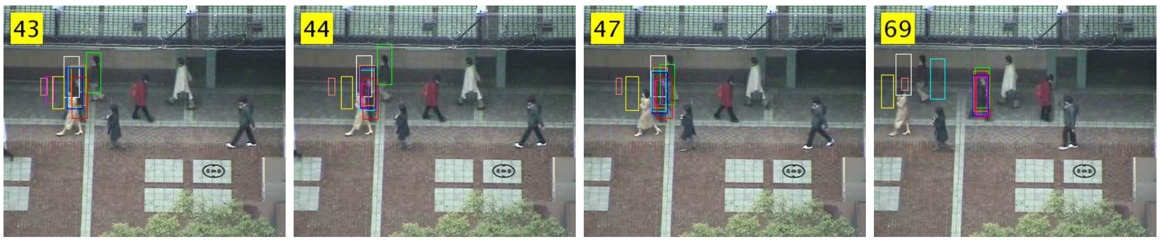

Figure 6.82: Tracking sequences for Table. 6.30

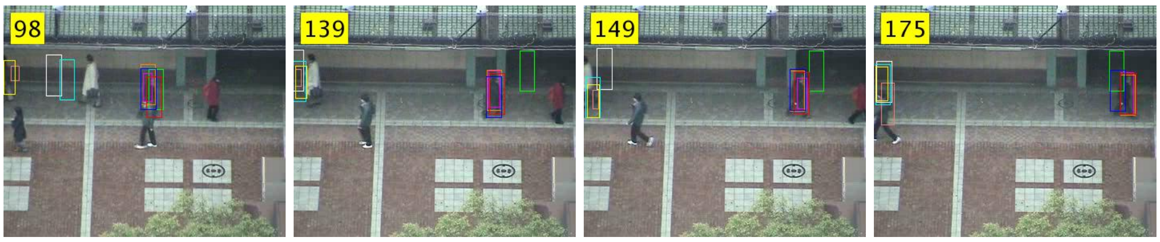

Figure 6.83: Tracking sequences for Table. 6.30

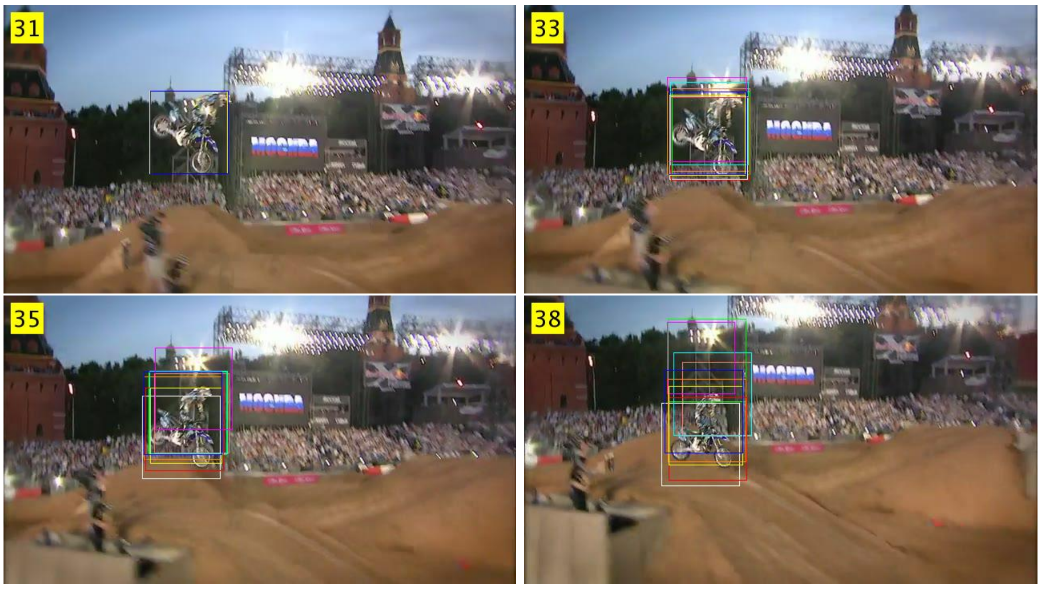

Figure 6.84: Tracking sequences for Table. 6.31

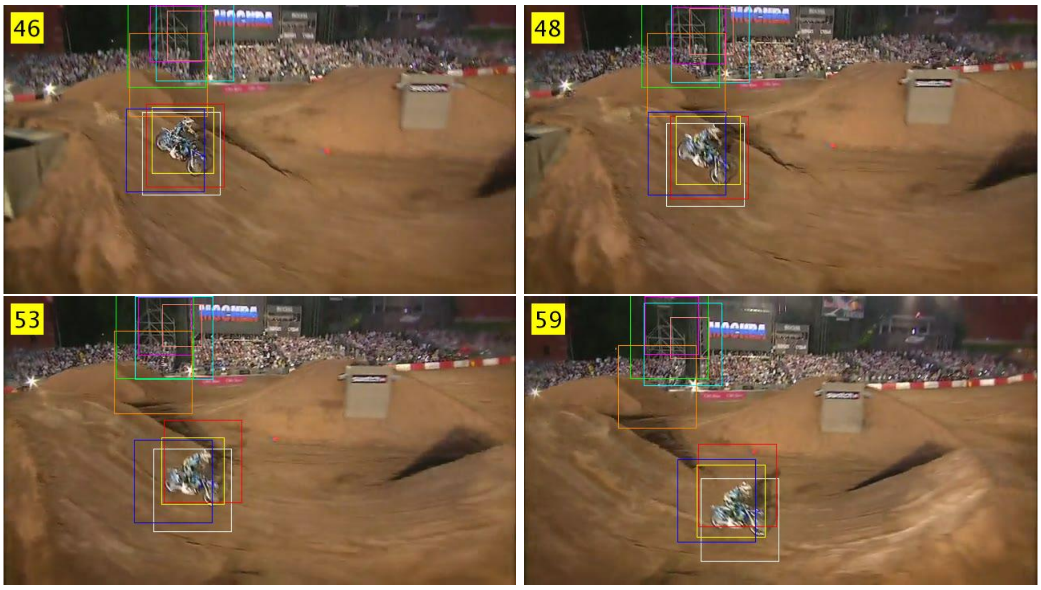

Figure 6.85: Tracking sequences for Table. 6.31

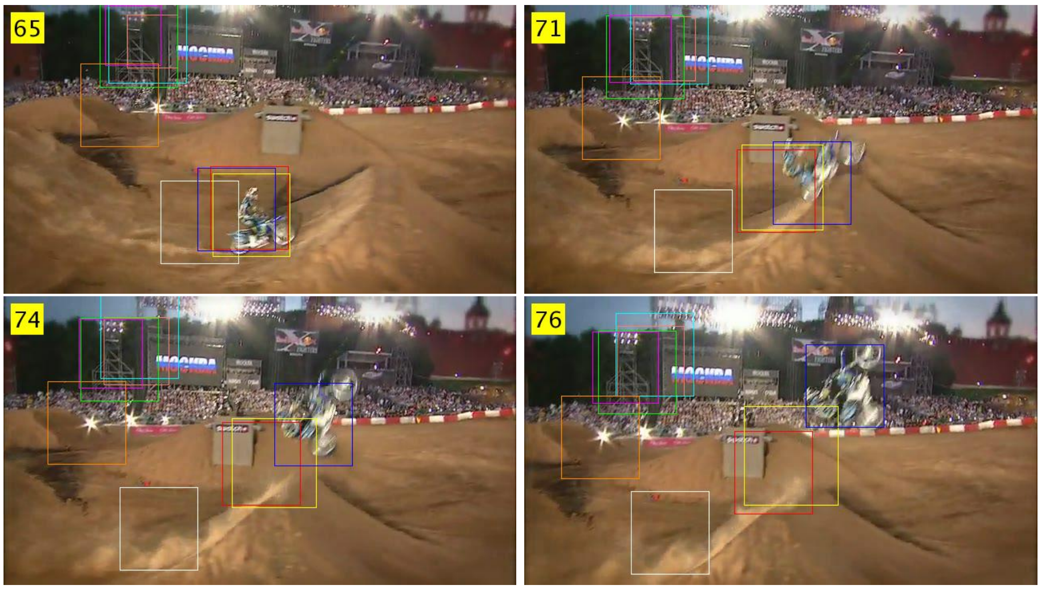

Figure 6.86: Tracking sequences for Table. 6.31

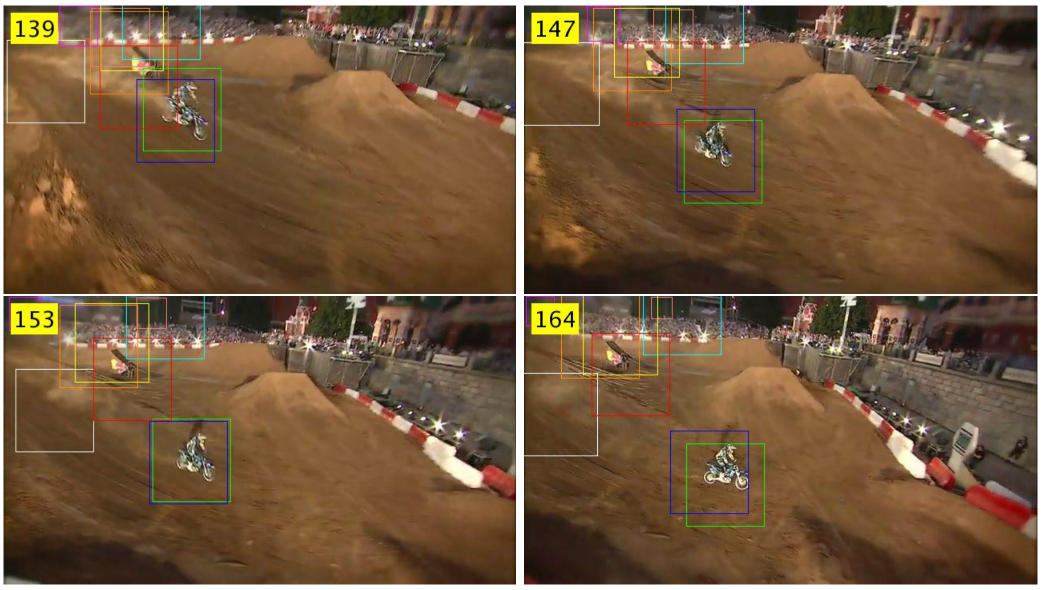

Figure 6.87: Tracking sequences for Table. 6.31

Table 6.29: Case Study 12 : Comparative Study

—	**Mean Error**	**Maximum Error**	**Std. Deviation**	**Success Rate**
CPF	40.32	97.38	26.69	54.91
Frag	119.82	301.89	102.44	43.79
IVT	10.08	21.41	5.68	**95.37**
KMS	43.31	180.11	49.90	53.36
LOT	60.51	219.08	60.61	52.04
MIL	31.19	120.72	37.61	87.46
SCM	25.82	112.92	28.99	77.21
VTD	10.60	33.04	6.91	**94.28**
Proposed	10.89	31.31	6.36	**90.95**

Table 6.30: Case Study 13 : Comparative Study

—	**Mean Error**	**Maximum Error**	**Std. Deviation**	**Success Rate**
CPF	30.82	77.48	23.48	79.63
Frag	8.69	42.70	10.31	86.94
IVT	111.75	235.88	70.39	30.56
KMS	97.97	228.01	73.32	33.62
LOT	70.17	136.38	39.30	63.06
MIL	31.99	66.76	18.04	84.44
SCM	16.37	35.52	11.27	93.49
VTD	95.98	184.93	57.07	43.97
Proposed	6.4	28.90	5.82	**95.54**

6.5.12 Analysis

The overall performance of the proposed algorithm throughout all the videos test cases is satisfactory. *In some cases* only, CPF, VTD or SCM performs better than the proposed algorithm; other algorithms never perform better than the proposed algorithm. Thus the proposed algorithm works better than other contemporary particle filter based algorithms like IVT and LOT. It also works better than other particle filter based algorithms like CPF and SCM in most of the case studies. This proves the robustness of the proposed algorithm.

6.6 Summary

In this chapter, all the test results are explained along with their parameter settings. The parameters are kept constant throughout the entire set of video sequences. But, in practice, if the user knows the object to be tracked and its ambiance beforehand, the user can tune the tracker on a number of test videos under that similar condition. This will result in more accurate tracking. But, here the same set of parameters is used for all the videos to observe the robustness of the proposed algorithm against others. Here the tests are done a number of times to observe their consistency in the tracking result. In this proposed work, the bat algorithm has been modified to reduce its computational complexity. Also the proposed particle filter with the resampling strategy reduces sample degeneracy as well as sample impoverishment problem. The proposed particle filter can generate *good* particles having

Table 6.31: Case Study 14 : Comparative Study

—	**Mean Error**	**Maximum Error**	**Std. Deviation**	**Success Rate**
CPF	216.76	127.75	59.11	29.71
Frag	108.92	242.98	76.17	34.12
IVT	133.06	264.78	71.72	28.61
KMS	89.16	200.87	60.94	44.46
LOT	113.15	212.93	60.54	40.96
MIL	86.28	175.42	46.89	39.79
SCM	78.84	194.07	58.59	42.62
VTD	94.24	189.25	55.41	39.35
Proposed	42.88	146.69	41.84	**57.87**

enough diversity among them. The main advantage of using bat algorithm is that the probability of jumps of the bats can be switched between type-1 jump and type-2 jump by changing tuning parameter called pulse-rate. Hence, in the proposed algorithm, when object is moving slow and there is no occlusion present, movement of the particles according to type-2 jump is given higher priority and, when either there is occlusion or the object is lost, movements of the particles according to type-1 jump is given higher priority by tuning the pulse-rate parameter properly. The proposed adaptive motion model increases the probability of generating the particles towards the moving object rather than away from the object. It works even better in case the object moves in one direction uniformly. But, the parameters are defined in such a way so that, it has been showed that if object is moving haphazardly too, still it can be tracked. Finally the update mechanism along with the dynamic histogram ranging can solve vast illumination change while tracking object. The algorithm has been tested for both fast and slow illumination change. This update mechanism can also resolve scale change and blurring effect. The algorithm has been tested with other videos and has been found to be working satisfactorily. The test results prove the robustness of the proposed algorithm against clutter, illumination fluctuation, scale change, fast object movement, motion blur and complete occlusion.

Chapter 7

Conclusion

In this project, a robust and new variant of particle filter based visual object tracking algorithm has been proposed. The algorithm has been tested intensively with many challenging video datasets [1, 2, 3] and found to be robust against clutter, illumination fluctuation, scale change, fast object movement, motion blur and complete occlusion.

This report mainly contains the detail literature review of visual object tracking algorithms and the proposed work. In Chapter 1, a complete overview of tracking applications with some motivating example is given. Then, in Chapter 2, all leading single object tracking algorithms are introduced and critically analyzed. In Chapter 3, a different approach to multiple object tracking, called data association techniques, have been described. Then, chapter 4 describes multiple camera object tracking algorithms including 3D object tracking. Chapter 5 narrates the proposed work. Different modules are analyzed separately. Chapter 6 shows the test results and their analysis. Both qualitative and quantitative analysis of test results have been done.

As the proposed algorithm has been found to be robust, it can be extended for tracking multiple object tracking and also for fast implementation (real-time). Implementation of algorithms for faster processing is another field of research. The proposed algorithm has been tested in MATLAB. For real-time applications, generally computer vision algorithms are implemented using C++. OpenCV (Open Source Computer Vision) is a popular platform for implementing Computer Vision related algorithms in C++. OpenCV (Open Source Computer Vision) consists of rich library of computer vision functions in C++ or other languages for real-time application development. Another way for faster implementation is hardware implementation which can be done using FPGA (Field Programmable Gate Array). FPGAs (Field Programmable Gate Arrays) have been used popularly for faster computer vision algorithm implementations. Many optimized architecture have been proposed for FPGA (Field Programmable Gate Array) based Computer Vision applications. FPGA (Field Programmable Gate Array) architecture has more parallelism than the CPUs and so FPGAs (Field Programmable Gate Arrays) are much faster than the CPUs. FPGA(Field Programmable Gate Array) is a very effective solution for real-time application development. Thus, as a future work, this proposed algorithm can be implemented in hardware for real-time applications along with multiple object tracking.

References

[1] Y. Wu, J. Lim, and M.-H. Yang. Online Object Tracking: A Benchmark. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* 2411–2418.

[2] Y. Wu, J. Lim, and M.-H. Yang. Tracker Benchmark v1.0. *https://sites.google.com/site/trackerbenchmark/benchmarks/v10* .

[3] E. F. C. project/IST 2001 37540. CAVIAR Test Case Scenarios. *http://homepages.inf.ed.ac.uk/rbf/CAVIARDATA1/* .

[4] J. Krumm, S. Harris, B. Meyers, B. Brumitt, M. Hale, and S. Shafer. Multi-camera Multi-person Tracking for EasyLiving. *Proceedings of Third IEEE International Workshop on Visual Surveillance* 3–10.

[5] X. Wang. Intelligent Multi-camera Video Surveillance: A review. *Pattern Recognition Letters* 34, (2013) 3–19.

[6] A. Yilmaz, O. Javed, and M. Shah. Object Tracking: A Survey. *ACM Computing Surveys* 38.

[7] Y. Wang, R. Zhai, and K. Lu. Challenge of Multi–Camera Trackings. *7th International Congress on Image and Signal Processing* 7, (2014) 32–37.

[8] D. Comaniciu, R. Visvanathan, and P. Meer. Kernel Based Object Tracking. *IEEE Transactions On Pattern Analysis And Machine Intelligence* 25, (2003) 564–577.

[9] G. Phadke and R. Velmurugan. Illumination Invariant Mean-shift tracking. *Proc. Indian Conf on Computer Vision, Graphics, and Image Processing* 407–412.

[10] G. Phadke and R. Velmurugan. Improved Mean Shift for Multi–target Tracking. *IEEE International Workshop on Performance Evaluation of Tracking and Surveillance (PETS)* 37–44.

[11] J. Jeyakar, R. V. Babu, and K. Ramakrishnan. Robust Object Tracking with Background-weighted Local Kernels. *Computer Vision and Image Understanding* 112, (2008) 296–309.

[12] G. R. Bradski. Real Time Face and Object Tracking as a Component of a Perceptual User Interface. *Fourth IEEE Workshop on Applications of Computer Vision, 1998* 214–219.

[13] F. Malik and B. Baharudin. Quantized Histogram Color Features Analysis for Image Retrieval Based on Median and Laplacian Filters in DCT Domain. *International Conference on Innovation Management and Technology Research (ICIMTR)* 624–629.

[14] F. Porikli, O. Tuzel, and P. Meer. Covariance Tracking using Model Update Based on Lie Algebra. *Proc. of IEEE Computer Society Conference on Computer Vision and Pattern Recognition* 728–735.

[15] Y. Wu, J. Cheng, J. Wang, H. Lu, J. Wang, H. Ling, E. Blasch, and L. Bai. Real–Time Probabilistic Covariance Tracking With Efficient Model Update. *IEEE Transactions On Image Processing* 21, (2012) 2824–2837.

[16] H. T. Nguyen and A. W. Smeulders. Fast Occluded Object Tracking by a Robust Appearance Filter. *IEEE Transactions On Pattern Analysis And Machine Intelligence* 26, (2004) 1099–1104.

[17] A. K. Jain, Y. Zhong, and S. Lakshmanan. Object Matching Using Deformable Templates. *IEEE Transactions On Pattern Analysis and Machine Intelligence* 18, (2002) 267–278.

[18] P. Perez, C. Hue, J. Vermaak, and M. Gangnet. Color-Based Probabilistic Tracking. *Springer–Verlag Berlin Heidelberg* 661–675.

[19] M. Arulampalam, S. Maskell, N. Gordon, and T. Clapp. A Tutorial on Particle Filters for Online Nonlinear/Non-Gaussian Bayesian Tracking. *IEEE Transactions On Signal Processing* 50, (2002) 174–188.

[20] P. Prez, J. Vermaak, and A. Blake. Data Fusion for Visual Tracking With Particle Filter. *Proceedings of the IEEE* 92, (2004) 495–513.

[21] C. Hue, L. Cadre, and P. Perez. Tracking Multiple Objects with Particle Filtering. *IEEE Transactions On Aerospace and Electronic Systems* 38, (2002) 791–812.

[22] N. A. Thacker, F. J. Aherne, and P. I. Rockett. The Bhattacharyya Metric as an Absolute Similarity Measure for Frequency Coded Data. *Kybernetika* 34, (1998) 363–368.

[23] Y. Bar-Shalom, F. Daum, and J. Huan. The Probabilistic Data Association Filter. *IEEE Control Systems Magazine* 82–100.

[24] T. E. Fortmann, Y. Bar-Shalo, and S. Molly. Multitarget Tracking Using Joint Probabilistic Data Association. *IEEE Conference on Decision and Control including the Symposium on Adaptive Processes* 19, (1980) 807–812.

[25] M. Liebens, T. Sakiyama, and J. Miura. Visual Tracking of Multiple Persons in a Heavy Occluded Space Using Person Model and Joint Probabilistic Data Association. *IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems* 547–552.

[26] M. Jaward, L. Mihaylova, N. Canagarajah, and D. Bull. Multiple Object Tracking Using Particle Filters. *Aerospace Conference* .

[27] N. T. Pham, K. Leman, M. Wong, and F. Gao. Combining JPDA and Particle Filter for Visual Tracking. *IEEE International Conference on Multimedia and Expo (ICME)* 1044–1049.

[28] Z. Zhang. A Flexible New Technique for Camera Calibration. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 22, (2000) 1330–1334.

[29] J. Heikkila and O. Silven. A Four-step Camera Calibration Procedure with Implicit Image Correction. *IEEE International Conference on Computer Vision and Pattern Recognition* 1106–1112.

[30] M. Shah. Video Lecture 12 : Camera Model. *Video Lecture* .

[31] O. Javed, Z. Rasheed, K. Shafique, and M. Shah. Tracking Across Multiple Cameras With Disjoint Views. *IEEE International Conference on Computer Vision* 2, (2003) 952–957.

[32] S. Bi, T. K. Ahmed, S. Cristian, D. Chong, A. F. Jay, and A. K. Roy-Chowdhury. Tracking and Activity Recognition Through Consensus in Distributed Camera Networks. *IEEE Transactions. On Image Processing* 19, (2010) 2564–2579.

[33] Q. Cai and J. Aggarwal. Tracking Human Motion in Structured Environments Using a Distributed-Camera System. *IEEE Transactions On Pattern Analysis And Machine Intelligence* 21, (2002) 1241–1247.

[34] O. Javed, K. Shafique, and M. Shah. Appearance Modeling for Tracking in Multiple Non-overlapping Cameras. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)* 2, (2005) 26–33.

[35] D. Makris, T. Ellis, and J. Black. Bridging the Gaps between Cameras. *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition* 2, (2004) II–205–II–210.

[36] G. Catalin and S. Nedevschi. Object Tracking from Stereo Sequences using Particle Filter. *4th International Conference on Intelligent Computer Communication and Processing* 279–282.

[37] J. Zhu, Y. F. Zheng, and R. E. Ewing. Measuring Object Speed using Stereo Tracking. *IEEE International Conference on Robotics and Automation (ICRA)* 5949–5954.

[38] A. Wu, M. Shah, and N. D. V. Lobo. A Virtual 3D Blackboard: 3D Finger Tracking using a Single Camera. *Fourth IEEE International Conference on Automatic Face and Gesture Recognition* 536–543.

[39] A. S. Sabb and M. Huber. Particle Filter Based Object Tracking in a Stereo Vision System. *IEEE International Conference on Robotics and Automation* 2409–2415.

[40] L.-W. Zheng, Y.-H. Chang, and Z.-Z. Li. A Study of 3D Feature Tracking and Localization using a Stereo Vision System. *International Computer Symposium (ICS)* 402–407.

[41] G. S. Walia and R. Kapoor. Intelligent Video Target Tracking using an Evolutionary Particle Filter based upon Improved Cuckoo Search. *Conf. 2012 Ninth Conf. Computer and Robot Vision* 41, (2014) 6315–6326.

[42] Y. Rui and Y. Chen. Better Proposal Distributions: Object Tracking Using Unscented Particle Filter. *Conf. 2012 Ninth Conf. Computer and Robot Vision* 2, (2001) II–786–II–793.

[43] C. Shan, Y. Wei, T. Tan, and F. Ojardias. Real Time Hand Tracking by Combining Particle Filtering and Mean Shift. *Proc. Sixth IEEE International Conf. Automatic Face and Gesture Recognition* 669–674.

[44] S. K. Zhou, R. Chellappa, and B. Moghaddam. Visual Tracking and Recognition Using Appearance-Adaptive Models in Particle Filters. *IEEE Trans. on Image Processing* 13, (2004) 1491–1506.

[45] S. Akhtar, A. Ahmad, and E. M. Abdel-Rahman. A Metaheuristic Bat–Inspired Algorithm for Full Body Human Pose Estimation. *Conf. 2012 Ninth Conf. Computer and Robot Vision* 369–375.

[46] S. Y. Chen. Kalman Filter for Robot Vision: A Survey. *IEEE Trans. on Industrial Electronics* 59, (2012) 4409–4420.

[47] C. Shen, J. Kim, and H. Wang. Generalized Kernel-Based Visual Tracking. *IEEE Trans. on Circuits and Systems for Video Technology* 20, (2010) 119–130.

[48] Q. Wang, F. Chen, J. Yang, W. Xu, and M.-H. Yang. Transferring Visual Prior for Online Object Tracking. *IEEE Trans. on Image Processing* 21, (2012) 3296–3305.

[49] P. Li, T. Zhang, and B. Ma. Unscented Kalman Filter for Visual Curve Tracking. *Image and Vision Computing* 22, (2004) 157–164.

[50] X. Yang. Nature-Inspired Optimization Algorithms. *London, UK: Elsevier* .

[51] X. Rong Li and V. P. Jilkov. Survey of Maneuvering Target Tracking. Part I. Dynamic Models. *IEEE Transactions on Aerospace and Electronic Systems* 39, (2004) 1333–1364.

[52] L. J. Latecki and R. Miezianko. Object Tracking with Dynamic Template Update and Occlusion Detection. *18th International Conference on Pattern Recognition (ICPR'06)* 1, (2006) 556–560.

[53] X. Dong and Z. Mao. Adaptive Template Based Object Tracking with Affine Model. *Symposium on Photonics and Optoelectronics* 1, (2010) 1–4.

[54] L. Yuan, J. Gu, Y. Xu, Y. Miao, T. Qiu, and Y. Jin. A Template Update Cam Shift Algorithm Based on LTP Texturel. *International Conference on Computer Science and Mechanical Automation (CSMA)* 170–174.

[55] S. Oron, A. Bar-Hillel, D. Levi, and S. Avidan. Locally Orderless Tracking. *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)* 1940–1947.

[56] T. Zhang, B. Ghanem, S. Liu, and N. Ahuja. Robust Visual Tracking via Multi–task Sparse Learning. *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)* 2042–2049.

[57] W. Zhong, H. Lu, and M.-H. Yang. Robust Object Tracking via Sparsity–based Collaborative Model. *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)* 1838–1845.

[58] A. Adam, E. Rivlin, and I. Shimshoni. Robust Fragments-based Tracking using the Integral Histogram. *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)* 798–805.

[59] W. Zhong, H. Lu, and Y. Yang, ; Ming-Hsuan .

[60] J. Kwon and K. M. Lee. Visual Tracking Decomposition. *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)* 1269–1276.

[61] B. Babenko, M.-H. Yang, and S. Belongie. Robust Object Tracking with Online Multiple Instance Learning. *IEEE Trans. on Pattern Analysis and Machine Intelligence* 33, (2010) 1619–1632.

[62] D. Ross, J. Lim, R.-S. Lin, and M.-H. Yang. Incremental Learning for Robust Visual Tracking. *International Journal of Computer Vision* 77, (2008) 125–141.