Dim object tracking in cluttered image sequences

Kaveh Ahmadi
University of Toledo
A Dissertation

Entitled

Dim Object Tracking in Cluttered Image Sequences

By

Kaveh Ahmadi

Submitted to the Graduate Faculty as partial fulfillment of the requirements for the

Doctor of Philosophy Degree in Engineering

________________________________________
Dr. Ezzatollah Salari, Committee Chair

________________________________________
Dr. Kim, Junghwan, Committee Member

________________________________________
Dr. Jamali, Mohsin, Committee Member

________________________________________
Dr. Carvalho, Jackson, Committee Member

________________________________________
Dr. Eddie Yein Juin Chou, Committee Member

________________________________________
Dr. Amanda C. Bryant-Friedrich, Dean
College of Graduate Studies

The University of Toledo

August, 2016
An Abstract of

Dim Object Tracking in Cluttered Image Sequences

By

Kaveh Ahmadi

Submitted to the Graduate Faculty as partial fulfillment of the requirements for the Doctor of Philosophy Degree in Engineering

The University of Toledo
April, 2016

This research is aimed at developing efficient dim object tracking techniques in cluttered image sequences. In this dissertation, a number of new techniques are presented for image enhancement, super resolution (SR), dim object tracking, and multi-sensor object tracking. Cluttered images are impaired by noise. To deal with a mixed Gaussian and impulse noise in the image, a novel sparse coding super resolution is developed. The sparse coding has recently become a widely used tool in signal and image processing. The sparse linear combination of elements from an appropriately chosen over-complete dictionary can represent many signal patches. The proposed SR is composed of a Genetic Algorithm (GA) search step to find the optimum match from low resolution dictionary. By using GA, the proposed SR is capable of efficiently up-sampling the low resolution images while preserving the image details.

Dim object tracking in a heavy clutter environment is a theoretical and technological challenge in the field of image processing. For a small dim object, conventional tracking methods fail for the lack of geometrical information. Multiple Hypotheses Testing (MHT) is one of the generally accepted methods in target tracking
systems. However, processing a tree structure with a significant number of branches in MHT has been a challenging issue. Tracking high-speed objects with traditional MHT requires some presumptions which limit the capabilities of these methods. In this dissertation, a hierarchal tracking system in two levels is presented to solve this problem. For each point in the lower-level, a Multi Objective Particle Swarm Optimization (MOPSO) technique is applied to a group of consecutive frames in order to reduce the number of branches in each tracking tree. Thus, an optimum track for each moving object is obtained in a group of frames. In the upper-level, an iterative process is used to connect the matching optimum tracks of the consecutive frames based on the spatial information and fitness values.

Another problem of dim object tracking is background subtraction which is difficult due to noisy environment. This dissertation presents a novel algorithm for detecting and tracking small dim targets in Infrared (IR) image sequences with low Signal to Noise Ratio (SNR) based on the frequency and spatial domain information. Using a Dual-Tree Complex Wavelet Transform (DT-CWT), a Constant False Alarm Rate (CFAR) detector is applied in the frequency domain to find potential positions of objects in a frame. Following this step, a Support Vector Machine (SVM) classification is applied to accept or reject each potential point based on the spatial domain information of the frame. The combination of the frequency and spatial domain information demonstrates the high efficiency and accuracy of the proposed method which is supported by the experimental results.

One of the important tools applied in this dissertation is Particle Filter (PF). The PF, a nonparametric implementation of the Bayes filter, is commonly used to estimate the
state of a dynamic non-linear non-Gaussian system. Despite PF’s successful applications, it suffers from sample impoverishment in real world applications. Most of the recent PF based techniques try to improve the functionality of the PF through evolutionary algorithms in the cases of unexpected changes in the system states. However, they have not addressed the discontinuity of observation which is unpreventable in the real world. This dissertation incorporates a recently developed social-spider optimization technique into PF to overcome the drawback of previous methods in these cases.

The problem of object tracking using multi-sensor data is a theoretical and technological challenge in the field of image processing which is presented as the final algorithm in this dissertation. Most of the conventional multi-sensor methods fail to track small dim objects in a cluttered background due to the lack of geometrical target information and unexpected large discontinuities in the measurement data. In this dissertation, a multi-sensor Swarm Intelligence Particle Filter (SIPF) is proposed in an environment covered by a set of multiple calibrated sensors with overlapping field of views. The proposed hierarchical method is divided into two levels. In the lower-level, SIPF is applied to locate the targets in each sensor based on the prior information. Each sensor reports the target position and its related fitness value to a dynamically selected central sensor. In the upper-level, the central sensor finds the best of the reported position for each target and broadcasts its position to all sensors at the lower level as the actual position of the target. Experimental results show this method is able to utilize multi-sensor data to produce highly accurate tracks in noisy datasets even in the case of large jumps or discontinuous observations well beyond the conventional tracking methods.
Dedicated to my wonderful wife Samira, my little daughter Ronica, and my caring parents Masoud and Shohreh.

I used to wonder what I might accomplish – but since Ronica entered my life, I want only to be the best Papa in the world.
Acknowledgements

I could not have completed this work without the mentoring of my PhD advisor Dr. Ezzatollah Salari. I would like to take this opportunity to express my deep gratitude to him for his valuable guidance, constructive suggestions, and continuous encouragement throughout these years. He has not only been an advisor but a family to me during our tough times. I would like to specially thank to all other committee members, Dr. Junghwan Kim, Dr. Mohsin Jamali, Dr. Jackson Carvalho and Dr. Sun, Weiqing, for their valuable suggestions.

My sincere appreciation goes to all the faculty members at the University of Toledo from whom I have learned a lot during the course of my education. I would also like to thank all the staff member of the college of engineering. They often went out of their way to help me.

Also, I am deeply indebted to my wife for her patient, my little daughter for her smile, and my parents for their encouragement. I believe my wife owns a big part of my success.
# Table of Contents

Abstract ........................................................................................................................................ iii

Acknowledgements.................................................................................................................. vii

Table of Contents .................................................................................................................... viii

List of Tables .......................................................................................................................... xii

List of Figures ......................................................................................................................... xiv

1 Introduction .......................................................................................................................... 1

2 Literature Review ................................................................................................................ 6

2.1 Dim Object Tracking Literature Survey ........................................................................ 6

2.2 Dual-Tree Complex Wavelet transform (DT-CWT) ..................................................... 9

2.3 Constant False Alarm Rate (CFAR) detector .................................................................. 10

2.4 Potential targets refinement using SVM .......................................................................... 11

2.5 Generic Particle Filter (PF) ........................................................................................... 12

2.6 Evolutionary Methods .................................................................................................... 14

2.6.1 Object Tracking Using Genetic Algorithm (GA) .................................................... 15

2.6.2 Particle Swarm Optimization (PSO) Tracking ......................................................... 15
6.2.1 Sampling ........................................................................................................... 94
6.2.2 Importance Weighting ......................................................................................... 96
6.2.3 Resampling ........................................................................................................ 97
6.3 Experimental Results .......................................................................................... 98
6.4 Conclusion ........................................................................................................... 106

7 Multi-Sensor Swarm Intelligence Particle Filter (SIPF) for Tracking of Dim Objects with Discontinuous Measurement Data ........................................................................ 108

7.1 The Proposed Multi-Sensor SIPF Method ............................................................. 111
7.1.1 Step1: Finding the best positions of targets in each sensor ......................... 112
7.1.2 Step2: Finding the actual positions of targets in the central sensor ............ 117
7.1.3 Step3: Center Selection ...................................................................................... 120

7.2 Experimental Results .......................................................................................... 120
7.2.1 Example 7.1: A Soldier walking across the bridge ........................................ 122
7.2.2 Example 7.2: Three drones moving in the sky .............................................. 126

7.3 Conclusion ........................................................................................................... 130

8 Conclusion and Future Research ........................................................................... 132

8.1 Summary ............................................................................................................. 132
8.2 Future Research ................................................................................................ 134

References ............................................................................................................. 136
List of Tables

Chapter 3
Table 3.1: Average SSIM of ten runs for different number of initial population .......... 39
Table 3.2 Average SSIM of ten runs for different dictionary size (up-sampling factor is 2) ................................................................. 40
Table 3.3 Comparison of PSNR/SSIM values of different example based SR methods (up-sampling factor 2) .............................................................................................................. 42
Table 3.4 Comparison of PSNR/SSIM values of different example based SR methods (up-sampling factor 3) .............................................................................................................. 42

Chapter 4
Table 4.1: Properties of six image sequences .............................................................. 61
Table 4.2 Average EPF for image sequences S1-S6 ...................................................... 66
Table 4.3 TCR of different methods over S1-S6 .......................................................... 67
Table 4.4 Processing time (in Second) of different methods over 10 frames ............... 68

Chapter 5
Table 5.1: Properties of 50 training image sequences ................................................. 79
Table 5.2: Properties of ten testing image sequences ................................................. 83
Table 5.3: The proposed method achieves a 95% correct target detection rate ........ 84
Table 5.4: Required time (in Second) for tracking 30 frames ..................................... 86
Chapter 6

Table 6.1: Average MSE values for different time stamps .............................................. 100

Table 6.2: Performance of the SOPF for different population sizes ..................................... 101

Table 6.3: Performance of the SOPF for different number of iterations ................................. 101

Table 6.4: Average MSE values for non-Maneuvering Target in 10 different runs ............... 104

Table 6.5: Average MSE values for Maneuvering Target in 10 different runs .................... 106
List of Figures

Chapter 3

Figure 3-1: The block diagram of image degradation process ............................................. 27
Figure 3-2: Sparse coding based super resolution ................................................................. 29
Figure 3-3: The proposed chromosome consists of non-zero elements of sparse coefficient $\alpha$ ........................................................................................................................................ 32
Figure 3-4: One-point crossover operation ............................................................................ 34
Figure 3-5: Average value of SSIM for ten runs with different number iterations .......... 40
Figure 3-6: Visual comparison of Parthenon (up-sampling factor 2) ............................... 43
Figure 3-7: Visual comparison of Parrots (up-sampling factor 3) ....................................... 44
Figure 3-8: Visual comparison of Zebra (up-sampling factor 2) ......................................... 44

Chapter 4

Figure 4-1: Illustrating PD and PFA under arbitrary distribution........................................ 53
Figure 4-2: A comparison of the common dim object tracking methods with the proposed scheme, (A) is the first frame of original sequence, (B) Max-Mean , (C) Bae’s Method, (D) Dong’s Method, (E) Lisha’s method, (F) Blostein’s method, (G) proposed method... 63
Figure 4-3: Error Per Frame (EPF) of different methods over S1-S6............................... 65

Chapter 5

Figure 5-1: Flowchart of the proposed model................................................................. 73
Figure 5-2: (a) The original image, (b) The accumulation of all high frequency sub-bands
.................................................................................................................................................. 74
Figure 5-3: (a, b, c, d, e, f) High frequency sub-bands in six different directions........... 74
Figure 5-4: (a) Original Image, (b) Variance values of sliding window, (c) Results after
CFAR Detection, (d) Results after SVM, (e) Target trajectories ................................. 81
Figure 5-5: Total average values of EPF: For Morphological method = 3.46, For Wavelet
based method = 3.58, For Bilateral method = 2.80, For HVS method = 3.40, For the
Proposed method = 2.10 ........................................................................................................ 84
Chapter 6
Figure 6-1: Proposed SOPF obtains the more accurate estimates even in the case of
significant increase................................................................................................................. 100
Figure 6-2: Estimated tracks of various PF methods (\(\textbf{x}_t\)) ......................................... 103
Figure 6-3: Estimated tracks of various PF methods (\(\textbf{y}_t\)) ............................................. 103
Figure 6-4: Estimated tracks of various PF methods for a maneuvering target (\(\textbf{x}_t\))...... 105
Figure 6-5: Estimated tracks of various PF methods for a maneuvering target (\(\textbf{y}_t\))...... 105
Chapter 7
Figure 7-1: Flowchart of the proposed method................................................................. 119
Figure 7-2: Images from IR and visible light cameras ...................................................... 124
Figure 7-3: EPF of the three tracking methods over 500 frames................................. 125
Figure 7-4: TCR of all the methods are same in Example 7.1................................. 126
Figure 7-5: Positions of three drones from four different views in frame 120........ 128
Figure 7-6: Accumulation of CWT high frequency sub-bands. Small objects are located
in high frequency areas ......................................................................................................... 129
Figure 7-7: EPF of the three tracking methods over 300 frames ................................. 129

Figure 7-8: TCR of the methods in Example 7.2 shows inefficiency of centralized particle filter in finding lost objects .......................................................... 130
Chapter 1

Introduction

Computer vision research started in the 1960’s as an artificial intelligence problem, where the goal was to understand images by detecting objects and defining the relationship between them [1]. Object tracking is an important and challenging task in many computer vision applications such as surveillance, vehicle navigation, and autonomous robot navigation. Recently, small moving object tracking in cluttered image sequences has attracted a great deal of research interests [1-10], mainly because of its application in surveillance systems such as radar, sonar and optical sensors. The dim object tracking appears in many fields including aerospace applications. The dim object tracking in particular is usually concerned with the aerospace Infrared (IR) noisy image sequences. The dim and small target appears to be very close to a point source (several pixels) because of aero-optic disturbances and a significant amount of noises from air turbulence. In addition, clouds in the sky, the atmosphere temperature, and a cluttered background making the image to have a low Signal to Noise Ratio (SNR) usually less than 3 db. The target has unknown intensity and is moving at an unknown velocity. For a small dim object, conventional tracking methods fail for the lack of geometrical information.
An early approach to object tracking is to perform frame differencing followed by thresholding [11]. In the case of dim moving objects where the noise level is high, a systematic procedure to decide whether or not an object is present is needed. Generally speaking, dim object tracking algorithms can be classified into two groups: Track-Before-Detect (TBD) and Detect-Before-Track (DBT). TBD algorithms use motion information and can potentially follow many target points in consecutive frames. Some of these potential tracks could survive based on their characteristics. Popular TBD algorithms are mainly based on Hough transform [12-13], Dynamic Programming (DP)[8], and Multiple Hypothesis Testing (MHT) [14]. Common techniques used in the second category (DBT) detect the objects in each frame based on spatial or frequency domain information. After declaring the presence of the target at each image, tracking phase connects relative target of interest. Each of these two categories has its own advantages, shortcomings, and applications.

Hough transform algorithms mainly combine the image sequence, and try to detect streaks in the sequence. For instance Rauch and Firschein preform a prescreening by 3x3 window operation. Dynamic Programming (DP) was first introduced by Barniv [8] for the detection and tracking of low-observable objects. The tracking performance of the DP based algorithms has been found to be poor even on images with average SNR values [13-14]. The detection of small moving objects of unknown position and speed using Multiple Hypothesis Tracking (MHT) was initially introduced by Blostein [10]. In this method, a large number of candidate trajectories are organized into a tree structure. To prune the tree structure of the candidate trajectory, a truncated sequential probability test is applied at each pixel. In fact, summation of each trajectory pixel values is sequentially
compared against two thresholds. The hypothesis that the trajectory contains a target would be accepted when the summations are greater than the maximum threshold, and rejected for summations less than the minimum threshold. When the test statistic falls between the two thresholds, this decision is deferred and the trajectory state is stored in a list. The main advantages of MHT based algorithms are their independence of target velocity, target direction, and the SNR values of the image sequence. These algorithms have a good performance for very low SNR images, however, the computational requirements for processing large tree structures of tracks could make it inapplicable in some cases.

The second category DBT algorithms can be applied either in spatial or frequency domain. Common spatial techniques detect the objects in each frame based on spatial information of that frame. Kalman filters [15], Particle Filter (PF) [16], Genetic Algorithm (GA) [17-18], Human Visual System (HVS) [19], Multiple Hypothesis Testing (MHT) [20-21] and Support Vector Machine (SVM) [22-24] are common techniques that use spatial features of the shapeless targets in their tracking systems. Due to the lack of object features in small point-sized targets, some other methods attempt to track dim objects by exploring frequency domain information. Morphological object tracking methods [3,25], and wavelet based tracking methods [26], are approaches for dim object tracking utilizing frequency domain information. The basic concept behind these methods is the difference between the frequency of the target and the background. Targets are considered to have higher frequency contents than the background. As a result, a high frequency filter can be used to segment an image into target and background categories.
This dissertation focuses on dim object tracking in noisy cluttered image sequence, and introduces new algorithms to improve the accuracy of applications in this field. The organization of this dissertation is as follows.

Chapter 2 presents the literature review which includes some common object tracking methods and fundamental background information. This chapter starts with the history of object tracking algorithms and illustrates some preliminary tools of the dim object tracking algorithms.

Chapter 3 introduces the prescreening method that enhances the image sequence quality. This chapter explains a novel super resolution (SR) technique based on sparse coding. The proposed SR technique uses Genetic Algorithm (GA) to speed up the enhancement process. The proposed method could be used as a prescreening to dim object tracking methods in cluttered image sequences.

Chapter 4 discusses the proposed algorithm for dim object tracking using evolutionary based MHT technique. This chapter presents a hierarchal tracking system in two levels. For each point in the lower-level, a multi objective particle swarm optimization technique is applied to a group of consecutive frames to reduce the number of branches in each tracking tree. Thus, an optimum track for each moving object is obtained in a group of frames. In the upper-level, an iterative process is used to connect the matching optimum tracks of the consecutive frames based on the spatial information and fitness values.

Chapter 5 presents a novel tracking technique by using Dual Tree Complex Wavelet Transform (DT-CWT) and Support Vector Machine (SVM). The advantage of this
technique is the combination of frequency and spatial domain that empower this algorithm to track small objects in noisy image sequences.

Chapter 6 explains the case of discontinuity in the system observation. The discontinuity could be result of the noise or cause of an obstacle that hide objects for a period of time. A new Particle Filter (PF) method is introduced in this chapter. The proposed PF method is based on spider-social optimization. Diversity of female and male particles in spider-social could search a wider area for a lost object and track targets in the case of large changes or discontinuity in the system observation.

Chapter 7 introduces the multi-sensor object tracking models. A multi-sensor observation system could provide a more robust tracking of objects. In this chapter a new method for fusing the data receiving from different sensors is described. The new method is based on evolutionary search techniques and could locate the target even if it is lost in a few numbers of the sensors.

Finally, Chapter 8 is devoted to conclusion of this work and presents some suggestions for future research extensions.
Chapter 2

Literature Review

The purpose of small dim object tracking algorithms is to detect, track and identify targets from sequences of noisy observations provided by one or more sensors. In the target tracking literature, it is assumed that each target moves independently of all others. In this chapter, some necessary background information for dim object tracking is provided and current state-of-the-art analytical and numerical approaches that are available for dim object tracking are briefly reviewed. Moreover, optimization tools that are applied in this study through different algorithms will be explained.

2.1 Dim Object Tracking Literature Survey

Several researchers have focused on object tracking problems with emphasis on motion-based approaches. The goal of a motion-based approach is extracting information such as target behavior from the image sequences. For a better understanding of the scene, compared to a single image, a sequence of images introduces a new dimension: time, and a new constraint: temporal coherence [1]. The most early object tracking applications were driven preliminary by aerospace application such as radar, sonar, guidance, navigation, and air traffic control. Kalman filter was the mostly used tool for those aerospace applications [29]. However, object tracking applications are extended to
other non-aerospace arenas these days. The applications of object tracking systems can be found easily in autonomous vehicles [30-31], robotics [32-33], biomedical research [34-35], and remote sensing [36-37].

Various background subtraction techniques are available in the literature of moving object detection and tracking. Background subtraction involves the difference of the current image and the reference updated background which is calculated over a period of time. Most important issues of a background subtraction technique could be summarized as the problem of varying illumination condition, background clutter, shadows, camouflage, bootstrapping. Also, motion estimation of the foreground should be done at the same time [38]. However, due to limitation of small dim object tracking algorithms background subtraction techniques are not applicable in this area.

One of the earliest basic solution approaches for the problem of small object tracking was Nearest-Neighbor Kalman Filter (NNKF). In this approach the state at time $k$ is estimated based on observation data at time $k$ and the previous state estimation at time $k - 1$. However, only the closest statistical distance measurements are used to update the state [39]. Later Pulford [40] proposed the strongest neighbor filter, a variant of this algorithm, and used SNR for the statistical measurement. Another common approach is Probabilistic Data Association (PDA) [41] that used all the measurements that are close to the predicted location in a Bayesian update.

The drawback of the preceding approaches is considering the data associations based on one scan at a time. These methods summarized data by a single set of track observations and estimated the next object location based on the single track. One of the more complex latter approaches is Multiple Hypothesis Tracking (MHT). MHT consider
data association across multiple scans and allow limited branching in the hypothesis tree [29]. In 1991, Blostein [10] introduced MHT for tracking of dim moving objects of unknown position and speed. In this method, a large number of candidate trajectories are organized into a tree structure. To prune the tree structure of the candidate trajectory, Wald’s truncated sequential probability test [42] is applied at each pixel. In fact, summation of the pixel values in each trajectory is sequentially compared against two thresholds. The hypothesis that the trajectory contains a target would be accepted when the summations are greater than the maximum threshold, and rejected for summations less than the minimum threshold. When the test statistic falls between the two thresholds, the decision is deferred and the trajectory state is stored in a list. Blackman discusses these issues in a tutorial context [9] and points to some areas for further study.

Small dim object tracking algorithms can be categorized into two broad classes: detect-before-track (DBT), suggesting detection through target intensity first and target motion hypothesis used after, and track-before-detect (TBD), suggesting target motion information is used first and target intensity detection after that. Both DBT and TBD algorithms assume background clutter and noise follow Gaussian distributions, whereas this does not apply to the most of real world systems [3]. Recent techniques such as particle filter based method attempt to tackle this limitation by assuming nonlinear and non-Gaussian distributions for noise and targets.

In this dissertation, more realistic algorithms are proposed which are easily adapted to real world systems. In the following parts of this chapter a few number of image processing tools which are used in this project are explained.
2.2 Dual-Tree Complex Wavelet transform (DT-CWT)

The wavelet transform has been widely used with great success in many signal processing applications [43-45]. However, discrete wavelet transforms suffer from some serious disadvantages, including shift sensitivity and poor directionality. DT-CWT is a relatively recent enhancement to the discrete wavelet transform (DWT) which has some significant added properties: namely, it is shift invariant with added directional selectivity in two or more dimensions. Small object tracking requires a feature that remains invariant by translation and rotation because different video frames may contain a translated and rotated version of a moving object. Therefore, DT-CWT is an ideal candidate for dim object tracking [46-47].

Given a filter bank of a complex-valued band-pass wavelet \( \psi_c(t) \) and a complex-valued low-pass scaling function \( \varphi_c(t) \), the complex wavelet coefficients \( d_c(j, n) \) and the complex scaling coefficients \( c_c(n) \) for any 1-D function \( x(t) \) are computed via the following inner product equations,

\[
\begin{align*}
    d_c(j, n) &= \int_{-\infty}^{+\infty} x(t) \cdot \psi_c(2^j t - n) dt \\
    c_c(n) &= \int_{-\infty}^{+\infty} x(t) \cdot \varphi_c(t - n) dt,
\end{align*}
\]  

(2.1)

The above equation is a time-frequency analysis of the signal by measuring its frequency content (controlled by the scale factor \( j \)) at different times (controlled by the time shift \( n \)) [43].

Considering both the real and imaginary parts of \( \psi_c = \psi_{real} + i \psi_{imaginary} \) and \( \varphi_c = \varphi_{real} + i \varphi_{imaginary} \), where \( i = \sqrt{-1} \), the wavelet and scaling coefficients can be expressed as,
\( d_c(j, n) = d_{\text{real}}(j, n) + i \ d_{\text{imaginary}}(j, n) \)

\( c_c(n) = c_{\text{real}}(n) + i \ c_{\text{imaginary}}(n) \)  

(2.2)

In contrast to the real-valued wavelet transform of a 2-D signal which has three high frequency sub-bands (HL, LH, HH) in three different directions, the DT-CWT of an image has six high frequency sub-bands in six different directions. The use of DT-CWT in the proposed algorithm provides high frequency sub-bands in six directions for each frame in the image sequence. In this way, the method is more directionally sensitive to moving small targets in different directions.

### 2.3 Constant False Alarm Rate (CFAR) detector

A CFAR detector is used to prescreen the image in order to localize the possible targets [48]. Existing CFAR target detection systems are commonly implemented by using the sliding window technique. Using the data available in a block, a threshold \( T \) can be found, leading to a decision criterion for each point of the image \( F(x, y) \) as follows,

\[
\begin{align*}
\{ \text{Potential Target,} & \quad F(x, y) \geq T \\
\{ \text{Background,} & \quad F(x, y) < T
\end{align*}
\]

(2.3)

Determining the value of the threshold \( T \) is an issue in CFAR detection systems. Increasing the value of \( T \) could result in decreasing the false alarm rate as well as the detection rate. Additionally, decreasing \( T \) will increase the probability of detection \( (P_D) \), as well as the false alarms \( (P_{FA}) \). Some earlier CFAR systems simply find the average of the sliding window and multiply it by a factor \( \lambda \) to find the adaptive threshold. The value of \( \lambda \) is determined by calculations according to the desired \( P_{FA} \) [49]. Further research enhanced the value of the threshold by adding a multiplicative factor \( (\lambda) \) of the image.
variance to the mean value of the entire image. Assuming $\mu$ is the mean and $\sigma^2$ is the variance of the entire image, the threshold is then expressed as $[3, 50],
\[ T = \mu + \lambda \cdot \sigma^2, \] (2.4)

### 2.4 Potential targets refinement using SVM

A target block is defined to be a block containing the potential target. Any detection module such as CFAR will detect the possible candidate targets in the current frame. To find the real target position, candidate targets are passed to the SVM module for classification, which uses other spatial or frequency features for final decision.

SVM is a common data classification technique which was proposed by Vapnik and his group at AT&T BELL Laboratories. SVM is a classification and regression prediction tool maximizing predictive accuracy by applying machine learning theory. It has been widely used in human face recognition, vehicle license plate recognition, and handwriting recognition with superior results in most cases [51].

The main concept of the SVM is to construct a hyper-plane or a set of hyper-planes in a high or infinite dimensional space to classify the data. Among the possible hyper-planes, SVM selects the one where the distance, namely, the margin of the hyper-plane from the closest data point is as large as possible. Let $(w, b)$ represent the parameters of the hyper-plane and $\phi(x)$ be a transformation function. Then the value of $f(x) = w \cdot \phi(x) + b$ which is either -1 or +1 determines the class of data $x$. The weight vector $w$ and bias $b$ are obtained through a learning process by training data samples $S$. In an $n$-dimensional hyperspace, the SVM classifier can be trained from the training data samples $S = \{(x_i, y_i) | x_i \in \mathbb{R}^n, y_i \in \{-1, 1\}, i \in 1, \ldots, N\}$ by maximizing the margin $2/\|w\|$ or minimizing $\|w\|^2/2$ such that,
\( y_i(w \cdot \phi(x) + b) \geq 1, \ i \in 1, \ldots, N \), \hspace{1cm} (2.5)

The solution to this problem is given by the saddle point of the Lagrange function,

\[
L(w, b, \alpha) = \frac{1}{2} \|w\|^2 - \sum_{i=1}^{N} \alpha_i \{y_i [w \cdot \phi(x_i) + b] - 1\}, \hspace{1cm} (2.6)
\]

where \( \alpha_i \)'s are the Lagrange multipliers. Classical Lagrangian duality enables the primal problem to be transformed to its dual form, which is easier to solve. The dual form is given by,

\[
\max_{\alpha} \bar{L}(\alpha) = \max \{ \min_{w, b} L(w, b, \alpha) \}, \hspace{1cm} (2.7)
\]

The solution to the dual problem is given by,

\[
\bar{L}(\alpha) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{N} \alpha_i \alpha_j y_i y_j K(\phi(x_i), \phi(x_j)), \hspace{1cm} (2.8)
\]

where \( 0 \leq \alpha_i \leq C \), \( \sum_{i=1}^{N} \alpha_i y_i = 0 \), and \( K(\phi(x_i), \phi(x_j)) \) is the kernel function. For example, linear kernel could be expresses as \( K(\phi(x_i), \phi(x_j)) = \phi(x_i)^T \cdot \phi(x_j) \).

The optimal weight vector is obtained as follows,

\[
w = \sum_{i=1}^{N} \alpha_i y_i \phi(x_i), \hspace{1cm} (2.9)
\]

Therefore the decision function for vector \( x \) would be,

\[
g(x) = \text{sign}(\sum_{i=1}^{N} \alpha_i y_i \phi(x_i) \cdot \phi(x)^T + b), \hspace{1cm} (2.10)
\]

where \( x_i \)'s are the support vectors used in the training phase.

### 2.5 Generic Particle Filter (PF)

Earlier research indicates that the tracking problem can be modeled as a dynamic system involving state estimation. Approximating the current state of a dynamic system has been the common point of several filtering methods. By using the probability theory, the Bayes filter provides a rigorous state estimation framework. The Bayesian approach constructs the PDF of the state based on all of the available information. Among the
various Bayes filter realizations, the Kalman Filter (KF) is the most common technique. The KF is capable of achieving optimal solutions to estimation problems of the linear systems with Gaussian noise [52-55]. For the nonlinear dynamic problems, the Extended Kalman Filter (EKF), and the Unscented Kalman Filter (UKF) have been introduced [53]. These algorithms attempt to linearize the predicted state model so that the KFs can be applied. However, the required PDF is still Gaussian, which may be a gross distortion of the true underlying structure. Two decades ago, Gorden et. al. [55] introduced Particle Filter (PF) as a methodology for sequential signal processing in non-linear and non-Gaussian systems. A finite number of particle samples are used by particle filter to approximate a distribution. Hence, PF can be applied to any class of application regardless of the linear and Gaussian constraints [54], including localizing in robotics, video object tracking, geophysical systems, finance, and wireless communications.

A particle filter is a nonparametric implementation of the Bayes filter and is used to estimate the state of a dynamic system frequently. The key idea is to represent the distribution at each point in time by a set of particles. The particle filter technique can be summarized with the following three steps:

1) **Sampling:** In this step, the samples are drawn to initialize the particles for the next time stamp. The true probability distribution to sample particles from is not in a suitable form for sampling or even is not known.

2) **Importance weighting:** The PF algorithm computes an importance weight for each particle in this step. The weight computations are highly related to the problem definition. This weight is used to resample particles in the next step.
3) **Resampling**: In this step, a predefined number of sampled are drawn from the particle set. Thereby, the likelihood to draw a particle is proportional to its weight. The selected particles are the new set for the next time stamp. This step, replaces particles with low weights by high weight particles.

PF methods use a finite number of samples to approximate the target distribution for next step, therefore, resampling is needed to increase the average particles weights. PF would lose the target track after a number of time stamps without resampling step. On the other hand removing samples can also cause a problem. In the case of removing particles which are close to the target, the problem which is called particle depletion is happened. This case is possible according to the experimental results in the literature of PF [56]. Low-variance resampling is the technique to decrease the chance of particle depletion. In this technique, the particles are not drawn independent of each other in the resampling step. A single random number is generated to choose the first particle. The other particles are drawn based on the previous drawn and a probability proportional to the individual weight. A detailed explanation on PF techniques can be found in [55-62].

### 2.6 Evolutionary Methods

Conventional small dim object tracking algorithms are not capable of solving real-world problems due to incomplete or noisy data. The bio-inspired computing paradigms look like a suitable replacement in this area. These patterns involve simple elements that can solve complicated problems of the real world when working together.

The task of robust object tracking is very challenging regarding illumination variation, background clutter, fast motion, occlusion, structural deformation, real-time restriction, etc. In order to solve these problems, a variety of methods have been proposed.
in the field of image processing. An enormous number of these algorithms are based on evolutionary search methods. These methods provide an optimum search area around the last object position depending on current observation, so they are able to improve the tracking methods to a certain extent. In the following sections the common evolutionary methods which are used in this dissertation are briefly described.

2.6.1 Object Tracking Using Genetic Algorithm (GA)

Genetic algorithm (GA) is a general purpose optimization heuristic inspired by Darwin’s principle of natural evolution. GA can effectively be used to detect and track targets in image sequences. Generally speaking, the time of convergence and computational cost of a genetic algorithm highly depends on the encoding scheme. According to the GA terminology which is affiliated to biology, population is the set of candidate solution, generation is each iteration of the system, parents are selected from the best solutions and used to generate offspring, crossover is the recombinant of the two parent’s information, and mutation is random modification to a solution. In each generation, the better solutions are selected as parent and new candidates are generated by crossover and mutation. In this way, the population evolves and the solutions become better and better adapted to the optimization problem, same as what happen in the nature.

2.6.2 Particle Swarm Optimization (PSO) Tracking

Particle Swarm Optimization (PSO) is a parallel evolutionary algorithm developed by Kennedy and Eberhart based on social swarm behavior [63]. They considered the behavior of swarms in the nature and developed the PSO algorithm. PSO combines self-experiences with social experiences in a particle driven communication system. Same as
other evolutionary algorithms, a candidate solution is presented as a particle. PSO uses the swarm intelligence concept which is the property of a system, whereby the collective behaviors of unsophisticated agents that are interacting locally with their environment, create coherent global functional patterns. In PSO, a particle is moving through the problem hyperspace via the following equation.

\[ X_t = X_{t-1} + V_t \]  \hfill (2.11)

The direction and amplitude of the velocity \( V_t \) is updated in each iteration toward the best position of the history \( L_t \) (local best) and the best particle, leader, of the swarm \( G_t \) (global best) as follows,

\[ V_t = V_{t-1} + \varphi_1 \cdot rand_1 \cdot (L_t - X_{t-1}) + \varphi_2 \cdot rand_2 \cdot (G_t - X_{t-1}) \]  \hfill (2.12)

where \( \varphi_1 \) and \( \varphi_2 \) are the acceleration constants which control the movement of each particle towards its individual and global best position, respectively. Most researches in this field report the best results when \( \varphi_1 + \varphi_2 = 4 \) [64-66]. Also, \( rand_1 \) and \( rand_2 \) are assumed to be two uniform random variables in \([0,1]\).

The velocity of a particle \( V_t \) is a stochastic variable that makes the particle follow wider cycles in the problem hyperspace [64, 67]. In order to moderate these oscillations, \( V_t \) can be restricted in \([-V_{max}, +V_{max}]\) range. According to the characteristics of the problem, the value of \( V_{max} \) can be selected empirically.

PSO is a solid evolutionary algorithm that is applied in many problems including object tracking. However, different versions of PSO have been developed and used in different areas. Some of these versions have the capabilities of other evolutionary computation techniques, such as hybrid versions of PSO. Others incorporate the adaptation of PSO parameters for a better performance such as adaptive PSO [64].
**Hybrid PSO:** Although PSO is efficient in finding the global optimum, PSO may be trapped in local optima. In return, GA is capable of finding the local optima and can avoid trapping in them, however, it is weak in finding the global optimum. Therefore, the hybrid of these two evolutionary algorithms can cover their drawbacks [68]. Many researches have proposed incorporating selection, mutation and crossover into the PSO algorithm to increase the diversity of the population.

In hybrid GA-PSO [69-70], PSO is applied to change the current searching area by considering best global and local values. Also, GA incorporates to enable particles to jump from one area to another by the selection mechanism. It results in accelerating the convergence speed of the whole particles.

**Adaptive PSO:** In some cases, a number of particles would be inactive and they do not have any contribution on local or global best values. These particles do not change their positions a lot, so their velocity is nearly reached to zero. One solution is to adaptively replace the current inactive particles with fresh particles while the existing relationships among the particles are preserved [70]. Other researches have proposed other parameters adjustment for PSO algorithms. Adding a random parameter to velocity formula, applying Fuzzy logic, using a secondary evolutionary search to find optimal parameters, and keep updating the parameters are other adaptive PSO techniques which are proposed to solve this problem [64].

### 2.6.3 Social Spider Optimization (SSO)

Following the development of PSO, several swarm algorithms have been introduced by a combination of deterministic rules and randomness, mimicking the behavior of
insect or animal groups in nature. The SSO algorithm proposed by Cuvas et al. [72] is based on the simulation of biological cooperative behavior of social-spiders.

Social spider society is a self-organize complex cooperative system. Each spider colony acts as an integrated unit and possesses the ability to operate at a distributed manner. Also they are capable of undertaking enormous construction of the global projects. The global order in social insects is a result of internal interactions among members.

Colony members and the communal web are two fundamental components of a spider colony [72]. Members are divided into two different categories: males and females, and communal web is using for different purposes including spiders’ communication. Spider colonies are the highly female-biased. It means that the number of female spiders usually reaches more than 70% of the total colony members. A member cooperates in different activities such as building and maintaining the communal web, or mating and social contact, based on its gender. Also, members’ interactions are divided into direct or indirect. The body contact or the exchange of fluids such as mating is considered as direct interaction. Communications through communal web are indirect interactions which convey important information that is available to each colony member. The spiders’ communications are encoded as small vibrations among members [73].

According to the vibrations emitted over the communal web, the female spiders present an attraction or dislike over other male or female spiders. Many biological researches show that the vibrations presented by heavier spiders have a stronger effect of attraction or dislike to the other spiders. Despite of the weight, the distance between spiders has a negative effect on the vibration, therefore, stronger vibrations are produced
either by heavy spiders or neighboring members. In each situation, the final decision of attraction or dislike is taken according to an internal state influenced by several factors such as reproduction cycle, curiosity and other random phenomena [72].

In the other hand, male spiders are divided into two groups of dominant and non-dominant spiders. This classification is based on the male spider weights. Usually, heavier male spiders are dominant of the colony. While the closest female spiders attract dominant males in the communal web, non-dominant male spiders tend to concentrate in the center of the male population to take advantage of the resources wasted by dominant males [74].

In the SSO algorithm [72], search particles (spiders) population is divided into two different types: males and females which are conducted by a set of different evolutionary operators and mimic different cooperative behaviors. This fact allows SSO to incorporate computational mechanisms to avoid premature convergence which is inescapable in some case of PSO algorithm.

### 2.7 Multi-Sensor Object Tracking methods

Multi-sensor object tracking are important due to three reasons. 1) The need for depth information in 3D object tracking arises. 2) Multi-sensor networks can track objects in a wider field-of-view in compare to single sensor. 3) More accurate information is revealed by combining data from multiple sensors rather than using a single sensor [75, 79]. One of the important issues in using multi-sensor object tracking systems is the relationship between the different sensors which can be manually or automatically.
Despite of the benefits of multi-sensor object tracking systems, high cost of computational resources needed in the conventional algorithms was a barrier to apply it in commercial applications. Researchers attempt to introduce new multi-sensor object tracking algorithms for video surveillance, pedestrian tracking, people behavior tracking, 3D gesture tracking, and autonomous robotics. They have been used different PF approaches and evolutionary techniques to improve their algorithms performance [80-84].

The aforementioned multi-sensor tracking methods assume stationary cameras. However, Kang et al. [85] proposed a continuous tracking across a combination of stationary and pan-tilt-zoom cameras with overlapping views. Other researches have been done to track objects in multiple non-overlapping cameras [86]. These algorithms use a silhouette of the individuals or sparse representation of object observations to track the objects in different non-overlapping views. Also, they have to make some assumptions about the object speed and the path in order to obtain the correspondences across cameras. Therefore, these algorithms highly depend on how much the objects follow the established paths [87].

Traditionally, multi-sensor object tracking systems have been implemented in a centralized architecture. However, distributed architectures are more robust as there is no longer a single point of failure. Also, distributed schemes do not require a single centralized database and have lighter processing and communication load. Three main issues are needed to be addressed for distributed schemes [79]:

1) Architecture: It describes how sensors are connected together and how information is transmitted between them.
2) Sensor Management: It is related to the placement of the sensors for maximizing the coverage of an area.

3) Algorithms: It is the way of processing data between sensors.

Moreover, there are two main problems in distributed multi-sensor object tracking systems. 1) A raw measurement could be used multiple times by different sensors. 2) The sequences of receiving data and processing data may not be same. The emphases of these issues are more highlighted when the tracking object is small and the background is cluttered.

Multi-sensor small and dim object tracking is a novel technique proposed in this dissertation. Multi-sensor empowers tracking algorithms with more information taking from the scene using more than one sensor. However, conventional algorithms perform image registration by comparing object features such as texture and colors which are unavailable for dim objects. Hence, image registration and data fusion are two important issues which should be solved for dim object datasets. The proposed multi-sensor object tracking algorithm not only could observe a wider field of view, but also could use information coming from different types of sensors such as IR-camera, CMOS camera, or temperature sensors.
Chapter 3

Preprocessing: Image Resolution Enhancement via Sparse Coding

Higher resolution images are always desirable to reveal more information of a monitored scene. In many applications of multimedia such as object tracking, remote sensing and navigation, medical imaging, video surveillance, etc., image resolution is restricted to the imaging device capabilities [88-104]. In order to overcome this problem, image super resolution (SR) techniques have become popular to estimate a desired high-resolution (HR) image from one or more low-resolution (LR) images. Moreover, SR is currently an active area of image processing, as it offers the promise of overcoming some of the inherent resolution limitations of low-cost imaging sensors such as cell phones or surveillance cameras. Tsai and Huang [90] were the first to propose an algorithmic technique for SR problems. Conventional SR approaches have multiple LR observations from the same scene, which are aligned with sub-pixel accuracy. Prior knowledge about the observations and reasonable assumptions allow mapping the LR images into a HR grid, and construct the HR image. Unknown registration and blurring operator in the process of image acquisition turn SR image reconstruction into a severely ill-posed problem, therefore, the solution of reconstruction is not unique.
SR techniques can be divided into two categories: reconstruction based methods, and learning based SR. The first category obtains HR images by simulating the image formation process. HR image pixels are computed from LR image pixels based on a random Markov field model. On the other hand, the learning based methods compute a HR image based on a pre-defined dictionary which describes the relationships between small patches of LR and HR images.

In the first category, various regularization methods have been proposed providing much better image quality in comparison with the traditional interpolation methods. The common point of these methods is obtaining a stable solution using a priori knowledge represented by a regularization term in order to obtain a stable solution. A common drawback of this approach is smoothing the edges of the image while reducing the noise. In an effort to remedy this problem, Farsiu et. al. [91] proposed a bilateral total variation (BTV) operator as a regularization term measured by $L_1$ norm to preserve the edge features of the image. However, BTV is not locally adaptive and fails to consider the partial smoothness of an image. To overcome this drawback, locally adaptive based methods were proposed [92-93]. However, these methods require a vast amount of computations and their performance degrades rapidly when the desired magnification factor is large [94].

In the learning based category, machine learning techniques are used to capture the co-occurrence of prior knowledge between LR and HR image patches in order to obtain better quality HR images. Freeman et al. [95] propose a LR to HR patch prediction which explores learning via Markov random field (MRF) leading to a solution by belief propagation. Sun et al. [96] extend this method using primal sketch priors to enhance
blurred edges, ridges and corners. A large database of HR-LR patch pairs makes these methods computationally very intensive. Roweis and Saul [97] adopts the philosophy of Locally Linear Embedding (LLE) from manifold learning, assuming similarity between the two manifolds in the HR-LR patch pairs. Later, Yang et al. [94] apply a compact representation of LR patch to capture the co-occurrence prior, and utilize sparse coding to obtain a HR image from a LR observation. The execution of the sparse coding SR has been significantly improved by Zeyde et al. [98]. They used different training approaches for the dictionary pairs and Orthogonal Matching Pursuit (OMP) [104] for sparse coding. Timofte et al. proposed two fast example-based SR methods called Anchored Neighbor Regression (ANR) [99] and Adjusted Anchored Neighbor Regression (A+) [100]. Their methods maps the LR patches onto HR domain using the projections learned from the neighborhoods. Recently, Kato et al. [101] proposed a multi-frame SR method based on sparse coding. To make reconstructed patches compatible to their surroundings, the neighboring patches overlap with each other. In addition, they maintain the global consistency in the final step in order to reduce the artifacts associated with the border of the patches and make the image more natural. These algorithms can preserve the edges while smoothing the low frequency areas and reducing the noise. From the literature, it appears that the learning based methods provide a better visual quality of the reconstructed HR images in comparison with the other SR methods. However, the reconstruction step in these algorithms is very time consuming. Therefore, these algorithms are inapplicable for time sensitive applications [102].

In an attempt to find better matches of dictionary pairs in a reasonable time, this chapter focuses on sparse coding search step in order to improve the accuracy of the
model. The investigation of the sparse coding algorithms shows that the emphasis has been on the construction of an over-complete dictionary from natural image patches. Wang et al. [103] propose a particle swarm optimization (PSO) based dictionary learning algorithm that improves the accuracy of the online learning dictionary for datasets dealing with the remote sensing images. However, this method is not applicable in SR methods for natural images. The main time consuming step of sparse coding algorithms is due to searching the dictionary for each image patch. Most sparse coding based algorithms apply quadratic programming to search the over-complete dictionary and find the ideal matches. Other projecting methods such as ANR and A+ avoid a full search of the dictionary by mapping LR patches onto HR domain using the projections learned from the neighborhoods. However, this chapter enhances the accuracy and proposes an adaptive GA based approach to perform a full search of the dictionary in order to find the best optimum HR patch for the corresponding LR patch in a reasonable time. The proposed evolutionary based sparse coding (ESC) is part of the SR algorithm which leads to an excellent HR image quality.

The rest of this chapter is organized as follows: the background of the sparse coding as applied to SR problem is briefly reviewed in section 3.1. The proposed SR method with the reconstruction step and the details of ESC are explained in section 3.2. Experimental results are presented in Section 3.3 followed by the conclusion in the last section.

3.1 Preliminary Fundamental

Given a LR image \( Y \), the single-image SR algorithm should be able to recover a HR image \( X \) of the same scene. Using the sparse coding approach, two constraints are
modeled to solve this ill-posed problem [94]. 1) Reconstruction constraint: it means the recovered \( X \) should be consistent with the input image \( Y \) considering the image observation model; 2) Sparse representation: an appropriately chosen over-complete dictionary could assign a HR patch to the observed low resolution patch through sparse representation.

3.1.1 Reconstruction Constraint

In the process of recording a digital image, there is a natural loss of spatial resolution caused by the optical distortions, motion blur due to limited shutter speed, noise that occurs within the sensor or during transmission, and insufficient sensor density. Understanding the reconstruction constraint requires more information about the model of the image degradation that happens in this process. Continuous point spread function \( H(x, y) \) blurs the observed scene through the camera lens. In addition, the image is always warped as a result of relative distortion \( M(\cdot) \) between the scene and the camera. The observed scene will then be converted to a digitized noisy LR image \( Y \) through the CCD image sensors. Therefore, the degradation model can be represented as following:

\[
Y = S H M X + N
\]

(3.1)

where \( S, H, M \), and \( N \) are the down-sampling, blurring, distortion warp, and noise functions, respectively. Figure 3.1 illustrates the above equation.
In general, the spatial distortion $M(\cdot)$ includes translation, rotation, deformation, and other possible distortions in the image acquisition process. Following the conventional formulation of super resolution, the HR and LR image matrices are converted to the vector form, $X: (rn, rm) \rightarrow (r^2nm, 1)$ and $Y: (n, m) \rightarrow (nm, 1)$, where $r$ is the up-sampling scale factor, $n$ and $m$ represents the horizontal and vertical sizes of the LR image, respectively. To keep the consistency in Equation (3.1), the sizes of $H$ and $M$ matrices are $(r^2nm, r^2nm)$ and the size of the down-sampling matrix $S$ is $(nm, r^2nm)$. For a given LR input image $Y$, there are an infinite number of HR images $X$ satisfying the above reconstruction constraint. Thus, SR algorithms are considered to be an ill-posed problem.

3.1.2 Sparse Representation

Sparse coding is concerned with the problem of finding the sparse representation of signals with respect to an over-complete dictionary $D \in \mathbb{R}^{m \times n}$, which includes $m$ sparse representation of signal $x_i \in \mathbb{R}^m$. The dictionary is usually learned from a set of training examples $X = \{x_1, x_2, ..., x_n\}$. Each sparse representation is represented by a coefficient $\alpha \in \mathbb{R}^n$. Given the sparse representation $\alpha$ and a dictionary $D$, the relevant
signal $x$ can be approximated by: $x \approx Da$ in which $\|x - Da\|_2^2 \leq \varepsilon$. Generally, sparse coding is an optimization problem demonstrated in the following form [103]:

$$\arg\min_{D, \alpha} \left\{ \frac{1}{2} \|x - Da\|_2^2 + \lambda \|\alpha\|_1 \right\}$$  \hspace{1cm} (3.2)

where $\lambda$ is the regularization parameter that improves the stability of the solution and increases the rate of convergence.

The above equation is not jointly convex in two variables $\alpha$ and $D$, however, it becomes convex with respect to one variable by keeping the other variable fixed. Therefore, sparse coding algorithms can be approached in two ways: keeping $\alpha$ fixed and finding $D$ or keeping $D$ fixed and finding $\alpha$. Both of these methods find optimal solutions in an iterative loop. The first method is known as dictionary learning. As it was mentioned earlier, the time consuming dictionary learning process is performed only once for SR applications, therefore, it does not play a crucial part in the running time of the algorithm. With this in mind, this chapter focuses on the second step which is the main part of the algorithm for each run, i.e., searching the dictionary for each image patch. In general, the sparse coding algorithms use a variety of techniques such as quadratic programming [94] or orthogonal matching pursuit (OMP) [104] to find the optimum value for $\alpha$ as following:

$$\alpha = \arg\min_{\alpha} \left\{ \frac{1}{2} \|x - Da\|_2^2 + \lambda \|\alpha\|_1 \right\}$$  \hspace{1cm} (3.3)

Applying sparse coding algorithms to solve an ill-posed SR problem requires two pre-assumptions:

1) the LR patch can be represented as a sparse linear combination of an over-complete LR dictionary ($D_l$);
2) the HR patch can be represented by the same sparse representation vector of a joint HR dictionary \( (D_h) \) under a controlled error.

These two assumptions are introduced and used in almost all of SR algorithms that use sparse coding technique [87,104]. Therefore, for each small patch \( y \) of input LR image \( Y \), there is a sparse representation \( \alpha \) with respect to dictionary \( D_l \) such that:

\[
y \approx D_l \alpha
\]  (3.4)

Given \( \alpha \), the high resolution patch \( x \) of HR image \( X \) can be found by \( x = D_h \alpha \).

On this basis, the HR image can be reconstructed using joint dictionary \( D_h \) and \( D_l \). Figure 3.2 illustrates the core process of sparse representation in SR algorithms.

3.2 The Proposed SR Algorithm via ESC

The goal of the proposed SR method is to obtain a high visual quality HR image from a single LR input image in a reasonable time. The proposed method requires two HR and LR over-complete dictionaries \( D_h \) and \( D_l \). Given a large set of HR training images \( X_h = \{x_1, x_2, \ldots, x_n\} \), HR dictionary could be obtained through the following equation:
\[ D_h = \arg \min_{D_h} \|X_h - D_h\alpha\|_2^2 + \lambda \|\alpha\|_1 \] (3.5)

This particular formulation has been extensively studied in the literature [105-107].

The SR algorithm requires a LR dictionary in which the sparse representation \( \alpha \) of the HR patch is the same as the sparse representation of the corresponding LR patch. To achieve this goal, assume HR patch \( x_i \) is represented by \( x_i = D_h\alpha_i \). According to image degradation model in Equation (3.1), the following equation will display the relationship of LR and HR patches.

\[ y_i = S H M x_i = S H M D_h\alpha_i \approx D_l\alpha_i \] (3.6)

Given that \( D_h \) is prepared in advance and taking into account the image degradation process, \( D_l \) can be generated from the HR dictionary using \( D_l = S H M D_h \). Constructing the dictionaries which is performed once is not the emphasis of this chapter and was adopted from ref. [106]. Therefore, our emphasis would be to devise a fast method to search the dictionary which is a time consuming step in the reconstruction of SR images.

The proposed SR algorithm consists of four steps. It starts by processing every point of the input LR image \( Y \) in raster scan order. For each point a small patch is extracted (note, selected patches have overlap). Each patch is then processed by the ESC algorithm to find its sparse representation \( \alpha \) which is used in the reconstruction step. In the third step, the related HR patch is found using the HR dictionary \( D_h \). In the final step, a global consistency is maintained by applying a back projection scheme to the reconstructed image. The overall proposed method can be summarized as following:

- **Raster scan of the input LR image \( Y \):** In the first step, for every point or every other point of the input LR image a patch \( y_i \) is extracted. Overlapping areas of the adjacent patches increases the consistency of the reconstructed image in edge areas.
• **ESC search procedure**: A adaptive GA based algorithm is devised to search LR dictionary to find the related sparse representation $\alpha_i$ for each patch. Sparse representation $\alpha_i$ is used in equation $x_i = D_i \alpha_i$ to find the corresponding HR patch.

• **Reconstruction of HR image using HR image patches**: The corresponding HR patch is placed into the final HR grid. Overlapped patches result in more than one point value for each HR grid point. Therefore, their average would be the final value.

• **Global consistency using back projection**: Due to the patch-wise reconstruction of HR images a back projection procedure is applied to keep the global consistency.

The detail explanation of ESC and the back projection are provided in the following:

### 3.2.1 Evolutionary Sparse Coding (ESC)

This chapter considers an evolutionary based dynamic system optimization to find the sparse representation of an image. The proposed ESC incorporates the use of an adaptive variation of GA into the sparse coding process. A number of complex optimization problems have been solved using the adaptive GA optimization tool via maintaining a population of candidate solutions. Adaptive GA population evolves by selection, crossover, and mutation operators. Similar to the other GA based algorithms, ESC consists of four steps: Defining the chromosome structure, initializing the population, assigning fitness function to each chromosome, reproduction through cross over and mutation operations. These steps are to be designed in the best way in order to find the minimum value of the sparse representation $\alpha$ while the condition $\|y_l - D_l \alpha\|_2^2 \leq \varepsilon$ is satisfied.
3.2.1.1 Defining the Chromosome Structure

An obvious choice for defining the chromosome structure in the proposed GA would be the sparse vector $\alpha$. However, $\alpha$ is a sparse vector and most of its elements are zero. Considering the time constraint, the proposed method defines a chromosome based on only the non-zero elements. To proceed, sparse vector $\alpha$ is initially set to approximate $y_i \approx D_l \alpha$. Consider a new error vector $err = y_i - D_l \alpha$ and a user defined threshold $\varepsilon \ll 1$. To satisfy the condition $\|err\| \leq \varepsilon$, it is required that all elements of vector $err$ to be less than or equal to $\varepsilon$. The details can be described in the following Algorithm 3.1.

Algorithm 3.1: Obtaining non-zero elements of $\alpha$

Input: The corresponding patch $y_i$, dictionary $D_l$, a user defined threshold $\varepsilon < 1$.

1. Initialize $\alpha$ to a vector of all zero elements as the same size as $y_i$.
2. Calculate error vector $err = y_i - D_l \alpha$.
3. For all $j$ which $\alpha(j) > \varepsilon$.
   Update $j^{th}$ element of vector $\alpha$ using $(j, j)^{th}$ element of matrix $D_l$ as following:
   \[ \alpha(j) = \frac{\text{sign}(err(j)) \varepsilon - err(j)}{D_l(j,j)} \]
4. Obtained $\alpha$ is the initial sparse coefficient ($init \_ \alpha$) with optimum non-zero elements in which the corresponding $err$ has a magnitude of less than $\varepsilon$.

Output: Non-zero elements of $init \_ \alpha$ and their positions.

Figure 3.3 shows the proposed chromosome structure that contains only the non-zero elements of initial $\alpha$ in which all its elements are less than or equal $\varepsilon$.

![Diagram of chromosome structure](image)

Figure 3-3: The proposed chromosome consists of non-zero elements of sparse coefficient $\alpha$
3.2.1.2 Initializing the Population

The \texttt{init} vector determined in Algorithm 1 serves as an initial chromosome satisfying \( \| y_l - D_l \cdot \texttt{init} \|_2^2 \leq \varepsilon \), however, it is not the optimum \( \alpha \) satisfying Equation (3.3). Indeed, the elements of the optimal vector \( \alpha \) should have values between zero and the corresponding non-zero elements of \texttt{init}. Thus, elements of the initial population are chosen in this range. The size of initial population will be analyzed later in the experimental results.

3.2.1.3 Fitness Function

The selection of an appropriate fitness function is essential to the design and success of a GA. The goal of the proposed ESC is to minimize Equation (3.3), hence, the fitness function is defined as:

\[
\frac{1}{2} \| y_l - D_l \alpha \|_2^2 + \lambda \| \alpha \|_1
\]

(3.7)

3.2.1.4 Reproduction through Crossover and Mutation

The crossover operation pairs two individuals and produces their offspring. In each iteration of the evolution process, two chromosomes with the best fitness function values are selected as parents for crossover operation. Experimental results show among a variety of crossover operations the level of disruption provided by a simple one-point crossover is enough in this case. This standard form of crossover swaps two sides from two parents with a probability of \( \pi_c \). The one-point crossover works by considering three steps [109]:

1) Alignment of the parent strings
2) Random selection of a crossover point ($P_c$)

3) Swap of the right-end sides of the strings to obtain two offspring. Figure 3.4 depicts details of the crossover step for two chromosomes $A$ and $B$.

![Diagram of crossover operation]

**Figure 3-4: One-point crossover operation**

Mutation is another operation that randomly alters an element of the selected individual. The mutation operator is carried out on two new offspring based on the mutation probability ($\pi_m$). The mutation operator adds a random number to a random element of the chromosome to increase the range of a coefficient.

The values of $\pi_c$ and $\pi_m$ play an important role in the controlling of GA convergence speed. The proposed ESC chooses these values adaptively based on the fitness values. The $\pi_c$ controls the rate of crossover, hence, its higher values introduce new solutions in next generation faster. The mutation probability ($\pi_m$) is a secondary operator to restore genetic material, therefore, large values of $\pi_m$ make the GA a purely random search, while some mutations would prevent the premature convergence of the GA on local optimum solutions [110]. The proposed method adapts the values of $\pi_c$ and $\pi_m$ based on the fitness values of the population in each generation. The adaptive strategy
for updating these probability values for chromosome $i$ and $j$ is expressed in the following form:

\[
\pi_c = \begin{cases} 
    \frac{c_1(f_{max} - \max(f_i, f_j))}{f_{max} - \bar{f}}, & \max(f_i, f_j) > \bar{f} \\
    c_2, & \max(f_i, f_j) \leq \bar{f}
\end{cases}
\]

\[
\pi_m = \begin{cases} 
    \frac{m_1(f_{max} - f_i)}{f_{max} - \bar{f}}, & f_i > \bar{f} \\
    m_2, & f_i \leq \bar{f}
\end{cases}
\]

where \( f_{max} \) is the maximum fitness value, \( \bar{f} \) represents the average fitness value of the whole population, \( f_i \) and \( f_j \) are the fitness values of the chromosomes \( i \) and \( j \), and \( c_1 \), \( c_2 \), \( m_1 \), \( m_2 \) are four random numbers in the internal \([0,1]\). It has been shown in the literature \([110]\) that the difference between the best and the average fitness values, \( f_{max} - \bar{f} \), is an indicator of the population status. If the difference is high the population is scattered in the solution space, while low difference values show that the population has converged to an optimum solution.

The detail of the proposed ESC is summarized in algorithm 3.2.

Algorithm 3.2: Proposed ESC

Input: The corresponding LR patch \( y_i \), dictionaries \( D_l \) and \( D_h \).

1. Obtain the non-zero elements of the corresponding initial sparse vector (\textit{init.}a) and save their positions using Algorithm 3.1.
2. Form the first chromosome structure from \textit{init.}a.
3. Initialize GA population.
4. Set \textit{counter} = 0.
5. While \textit{counter} < \textit{MaxIteration} or termination condition is not achieved:
   - Find two best chromosomes with the best fitness values.
   - Update the crossover and mutation probabilities based on Equations (3.8) and (3.9)
   - Run crossover and mutation operations based on their probability values.
   - Replace two worst chromosomes with two new offspring, with better fitness values.
6. Find the best chromosome with best fitness value.
7. Replace the non-zero elements of \textit{init.}a with elements of the best chromosome.
8. Find HR patch from HR dictionary.
\[ x_i = D_h \cdot \text{init} \cdot \alpha \]
Output: Corresponding HR patch \( x_i \)

### 3.2.2 Back Projection

With the limitation of sparse coding, even an optimal match of \( \alpha_i \) will not be able to generate an exact patch of \( x_i \). Moreover, the patch-wise processing of the method coupled with the unknown noise in the image could reduce the quality of the reconstructed HR images. For these reasons, the initial HR image \( X_0 \) obtained by the above sparse representation approach need to be further enhanced. Thus, the back projection is used to maintain the global consistency by projecting \( X_0 \) onto the space solution of \( Y = SHMX \). Hence, the optimum final HR image \( X^* \) will be obtained from the following equation:

\[
X^* = \argmin_X \{ \| Y - SHMX \|_2^2 + \theta \psi(X) \} \tag{3.10}
\]

In the above equation \( \psi(X) \) is the regularization cost function and \( \theta \in [0,1] \) is a regularization parameter to properly weigh the first term (similarity cost) against the second term (regularization cost) [111]. One of the most widely used regularization term is the following Tikhonov cost function with the \( L_2 \) norm [112-113]:

\[
\psi(X) = \| CX \|_2^2 \tag{3.11}
\]

where \( C \) could be a derivative type of operator (e.g. Laplacian) or even an identity matrix. This regularization method tries to limit the total energy of the image or forcing the spatial smoothness. Since the noise and the edge pixels both contain a high level of energy, they will be treated equally in the regularization process and the resulting image will not contain sharp edges. Total variation (TV) model [114] using the \( L_1 \) norm of the
gradient flow and bilateral total variation (BTV) [91] are other efficient techniques that are introduced to solve this shortcoming.

The proposed SR method in this chapter applies the $L_2$ norm of TV as the regularization function to maintain smoothness, suppress noise, and reduce patch-wise processing effect. However, to preserve the details in the reconstruction process, it is further enhanced by adding high frequency layer of the HR image, namely $E_{HR}$, to $\psi(X)$. A variety of approaches can be used to find the high frequency layer. The proposed method simply obtains the high frequency layer of LR input image through the application of a discrete wavelet transform (DWT). The high frequency component is then expanded to the size of the HR image by a Bicubic interpolation. Therefore, Equation (3.10) is modified as,

$$X^* = \min_{X} \{ ||Y - S\mathbf{H}X||_2^2 + \vartheta (||\nabla X||_2^2 + E_{HR}) \}$$  (3.12)

The following steepest descent optimization technique has been applied to find the solution to the above minimization problem.

$$X_n = X_{n-1} + \delta \left\{ A^T(Y - AX) + \vartheta \sum_{all\ x} \sum_{all\ y} \sqrt{\nabla x^2 + \nabla y^2} + \vartheta E_{HR} \right\}$$  (3.13)

where $A = S \times H \times M$ is the degrading matrix, and $\delta$ is a scalar defining the step size in the direction of the gradient. The optimized $X$ found in Equation (3.13) is regarded as the final HR image with the global consistency.

### 3.3 Experimental Results

In this section, we provide several examples to demonstrate the merits of the proposed SR algorithm in comparison with some other reconstruction or learning based SR techniques. The proposed ESC and SR algorithms are implemented in MATLAB and
the results of experiments are presented. We use a 5×5 LR patch size with 2 pixels overlap of adjacent patches and a scaling factor of 2 and 3 which are all commonly used in the SR literature. Note that for color images, the proposed method can be applied in three different color channels. However, as human eyes are more sensitive to luminance, considering the luminance component will be sufficient. In our experiments, seven different images with two dictionaries for HR and LR patches are considered. Dictionaries are trained from 100,000 patch pairs randomly sampled from natural images.

Two measures are used to evaluate the performance of the proposed method. The first one is the Peak Signal to Noise Ratio (PSNR) metric which is a range invariant distance measure commonly used in image processing. The second measure is the Structural Similarity Index (SSIM) [115] which is a measure more consistent with the human visual system as defined in the following expression,

$$SSIM(a, b) = \frac{(2\mu_a\mu_b)(2\sigma_{ab} + C_2)}{\left(\mu_a^2 + \mu_b^2 + C_1\right)\left(\sigma_a^2 + \sigma_b^2 + C_2\right)}$$  \hfill (3.14)

Note $\mu_a$, $\mu_b$, $\sigma_a$ and $\sigma_b$ are the means and standard deviation values of images $a$ and $b$, respectively. $C_1$ and $C_2$ are both constants and $\sigma_{ab}$ is the covariance of $a$ and $b$.

### 3.3.1 Effects of GA Parameters

To assess the performance of the proposed method, we investigate the effects of the initial population, the maximum number of iterations, and the dictionary size on the reconstructed image quality. There is a small difference between different runs of the algorithm due to the random probability measures used in the adaptive GA. For a more reliable measurement, we run the proposed SR algorithm ten times and present the average of the results. Table 3.1 shows the average SSIM values for various initial population sizes (up-sampling factor is 2).
It can be seen from Table 3.1 that the results are relatively insensitive with the increase in the number of initial population. It appears that after the initial population size of 20, the computational resource does not provide a significant improvement.

**Table 3.1: Average SSIM of ten runs for different number of initial population**

<table>
<thead>
<tr>
<th>Initial Population</th>
<th>Barbara</th>
<th>Cameraman</th>
<th>Butterfly</th>
<th>Parthenon</th>
<th>Parrots</th>
<th>Baby</th>
<th>Zebra</th>
<th>Peppers</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>0.8812</td>
<td>0.8798</td>
<td>0.8204</td>
<td>0.8157</td>
<td>0.8312</td>
<td>0.8745</td>
<td>0.8145</td>
<td>0.8512</td>
</tr>
<tr>
<td>10</td>
<td>0.9145</td>
<td>0.9278</td>
<td>0.8564</td>
<td>0.9130</td>
<td>0.8745</td>
<td>0.9245</td>
<td>0.8415</td>
<td>0.8611</td>
</tr>
<tr>
<td>15</td>
<td>0.9236</td>
<td>0.9616</td>
<td>0.9411</td>
<td>0.9234</td>
<td>0.9245</td>
<td>0.9545</td>
<td>0.8515</td>
<td>0.8912</td>
</tr>
<tr>
<td>20</td>
<td>0.9344</td>
<td>0.9745</td>
<td>0.9587</td>
<td>0.9312</td>
<td>0.9624</td>
<td>0.9594</td>
<td>0.8712</td>
<td>0.9247</td>
</tr>
<tr>
<td>25</td>
<td>0.9389</td>
<td>0.9767</td>
<td>0.9645</td>
<td>0.9501</td>
<td>0.9643</td>
<td>0.9647</td>
<td>0.8757</td>
<td>0.9347</td>
</tr>
<tr>
<td>30</td>
<td>0.9457</td>
<td>0.9775</td>
<td>0.9645</td>
<td>0.9512</td>
<td>0.9778</td>
<td>0.9714</td>
<td>0.8787</td>
<td>0.9471</td>
</tr>
</tbody>
</table>

The adaptive GA algorithm will iterate until the termination condition or the maximum number of iterations is achieved. Figure 3.5 shows the average SSIM of ten runs for the two images as a function of the number of iterations. There is no significant change in SSIM value after iteration 15, therefore, the maximum number of iterations is set to 15 in our experimental results.
The proposed method constructs the dictionary using the training set from Ref. [106]. Table 3.2 depicts the effect of the dictionary size on the performance of the proposed method. As expected, the SSIM improves as the size of the dictionary increases.

**Table 3.2 Average SSIM of ten runs for different dictionary size (up-sampling factor is 2)**

<table>
<thead>
<tr>
<th>Dictionary Size</th>
<th>Barbara</th>
<th>Cameraman</th>
<th>Butterfly</th>
<th>Parthenon</th>
<th>Parrots</th>
<th>Baby</th>
<th>Zebra</th>
<th>Peppers</th>
</tr>
</thead>
<tbody>
<tr>
<td>64</td>
<td>0.6748</td>
<td>0.7581</td>
<td>0.6512</td>
<td>0.8132</td>
<td>0.6812</td>
<td>0.8485</td>
<td>0.7105</td>
<td>0.8158</td>
</tr>
<tr>
<td>128</td>
<td>0.6874</td>
<td>0.8722</td>
<td>0.7588</td>
<td>0.9258</td>
<td>0.8211</td>
<td>0.9155</td>
<td>0.7458</td>
<td>0.8745</td>
</tr>
<tr>
<td>256</td>
<td>0.7941</td>
<td>0.9522</td>
<td>0.8975</td>
<td>0.9275</td>
<td>0.9188</td>
<td>0.9312</td>
<td>0.8014</td>
<td>0.8188</td>
</tr>
<tr>
<td>512</td>
<td>0.8612</td>
<td>0.9511</td>
<td>0.9412</td>
<td>0.9287</td>
<td>0.9524</td>
<td>0.9412</td>
<td>0.8890</td>
<td>0.9058</td>
</tr>
<tr>
<td>1024</td>
<td>0.8745</td>
<td>0.9745</td>
<td>0.9587</td>
<td>0.9312</td>
<td>0.9624</td>
<td>0.9594</td>
<td>0.9395</td>
<td>0.9247</td>
</tr>
<tr>
<td>2048</td>
<td>0.8912</td>
<td>0.9782</td>
<td>0.9630</td>
<td>0.9578</td>
<td>0.9877</td>
<td>0.9701</td>
<td>0.9475</td>
<td>0.9355</td>
</tr>
</tbody>
</table>
3.3.2 Comparison with Other Methods

To further evaluate the efficiency of the proposed method, it is compared with some other related SR techniques. To obtain image quality measure PSNR/SSIM, both original and reconstructed images are required. The experiment starts with seven original HR images from which the artificially degraded LR images are obtained according to the generative model in Equation (3.1). Blurring, down-sampling, and addition of noise are three degrading factors.

The proposed method is compared with six conventional algorithms. The first one is the Bicubic interpolation which is a simple baseline method for almost all of SR algorithms. Five more example based SR methods are also selected for the purpose of comparison. A single image SR based on sparse coding (SRSC) proposed by Yang et al. [94] and a multi-image SR method based on sparse coding (MSRSC) proposed by Kato et al. [101] are two example based algorithms which perform full dictionary search. In addition, three other example based methods using projection learned from the neighbors, namely the Anchored Neighborhood Regression (ANR) [99], Single Image Scale-Up (SISU) [98], and Jointly Optimized Regressors (JOR) [116], are considered to assess the performance of the proposed method. MATLAB codes of all these methods are available on their websites. Table 3.3 (up-sampling factor 2) and Table 3.4 (up-sampling factor 3) show the PSNR/SSIM values of these methods for a number of images. Note, Cameraman is a grayscale image, while the rest are color images.
### Table 3.3 Comparison of PSNR/SSIM values of different example based SR methods (up-sampling factor 2)

<table>
<thead>
<tr>
<th>Method</th>
<th>Barbar</th>
<th>Camera</th>
<th>Butterfly</th>
<th>Parthenon</th>
<th>Parrots</th>
<th>Baby</th>
<th>Zebra</th>
<th>Peppers</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>a</td>
<td>n</td>
<td>y</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bicubic</td>
<td>PSNR</td>
<td>27.07</td>
<td>27.03</td>
<td>25.40</td>
<td>25.00</td>
<td>29.25</td>
<td>35.19</td>
<td>28.16</td>
<td>33.54</td>
</tr>
<tr>
<td></td>
<td>SSIM</td>
<td>0.7998</td>
<td>0.7205</td>
<td>0.8778</td>
<td>0.7570</td>
<td>0.8587</td>
<td>0.9301</td>
<td>0.858</td>
<td>0.8910</td>
</tr>
<tr>
<td>SRSC</td>
<td>PSNR</td>
<td>28.30</td>
<td>28.25</td>
<td>25.60</td>
<td>29.59</td>
<td>31.50</td>
<td>35.85</td>
<td>29.89</td>
<td>33.65</td>
</tr>
<tr>
<td></td>
<td>SSIM</td>
<td>0.8513</td>
<td>0.8578</td>
<td>0.8971</td>
<td>0.8914</td>
<td>0.8874</td>
<td>0.9402</td>
<td>0.892</td>
<td>0.8993</td>
</tr>
<tr>
<td>MSRSC</td>
<td>PSNR</td>
<td>29.96</td>
<td>30.19</td>
<td>26.15</td>
<td>25.41</td>
<td>30.25</td>
<td>35.12</td>
<td>29.50</td>
<td>35.60</td>
</tr>
<tr>
<td></td>
<td>SSIM</td>
<td>0.8841</td>
<td>0.9213</td>
<td>0.9045</td>
<td>0.7914</td>
<td>0.8658</td>
<td>0.9298</td>
<td>0.878</td>
<td>0.9078</td>
</tr>
<tr>
<td>ANR</td>
<td>PSNR</td>
<td>28.36</td>
<td>32.45</td>
<td>29.97</td>
<td>28.69</td>
<td>32.64</td>
<td>37.97</td>
<td>32.58</td>
<td>36.20</td>
</tr>
<tr>
<td></td>
<td>SSIM</td>
<td>0.8611</td>
<td>0.9641</td>
<td>0.9466</td>
<td>0.8296</td>
<td>0.9458</td>
<td>0.9591</td>
<td>0.933</td>
<td>0.9141</td>
</tr>
<tr>
<td>SISU</td>
<td>PSNR</td>
<td>28.45</td>
<td>31.15</td>
<td>29.80</td>
<td>28.71</td>
<td>32.29</td>
<td>37.70</td>
<td>32.54</td>
<td>36.20</td>
</tr>
<tr>
<td></td>
<td>SSIM</td>
<td>0.8629</td>
<td>0.9511</td>
<td>0.9453</td>
<td>0.8274</td>
<td>0.9435</td>
<td>0.9570</td>
<td>0.929</td>
<td>0.9132</td>
</tr>
<tr>
<td>JOR</td>
<td>PSNR</td>
<td>28.47</td>
<td>32.89</td>
<td>31.20</td>
<td>29.15</td>
<td>33.50</td>
<td>38.07</td>
<td>32.95</td>
<td>36.75</td>
</tr>
<tr>
<td></td>
<td>SSIM</td>
<td>0.8676</td>
<td>0.9678</td>
<td>0.9579</td>
<td>0.8408</td>
<td>0.9512</td>
<td>0.9598</td>
<td>0.937</td>
<td>0.9170</td>
</tr>
<tr>
<td>Proposed</td>
<td>PSNR</td>
<td>28.93</td>
<td>33.11</td>
<td>31.41</td>
<td>30.24</td>
<td>33.88</td>
<td>38.02</td>
<td>33.05</td>
<td>36.97</td>
</tr>
<tr>
<td></td>
<td>SSIM</td>
<td>0.8745</td>
<td>0.9745</td>
<td>0.9587</td>
<td>0.9312</td>
<td>0.9624</td>
<td>0.9594</td>
<td>0.939</td>
<td>0.9247</td>
</tr>
</tbody>
</table>

### Table 3.4 Comparison of PSNR/SSIM values of different example based SR methods (up-sampling factor 3)

<table>
<thead>
<tr>
<th>Method</th>
<th>Barbar</th>
<th>Camera</th>
<th>Butterfly</th>
<th>Parthenon</th>
<th>Parrot</th>
<th>Baby</th>
<th>Zebra</th>
<th>Pepper</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>a</td>
<td>n</td>
<td>y</td>
<td>n</td>
<td>s</td>
<td></td>
<td></td>
<td>s</td>
<td></td>
</tr>
<tr>
<td>Bicubic</td>
<td>PSNR</td>
<td>26.02</td>
<td>26.52</td>
<td>23.81</td>
<td>25.75</td>
<td>27.50</td>
<td>33.51</td>
<td>26.08</td>
<td>32.14</td>
</tr>
<tr>
<td></td>
<td>SSIM</td>
<td>0.7476</td>
<td>0.7205</td>
<td>0.8222</td>
<td>0.6908</td>
<td>0.8731</td>
<td>0.9002</td>
<td>0.7867</td>
<td>0.8683</td>
</tr>
<tr>
<td>SRSC</td>
<td>PSNR</td>
<td>27.73</td>
<td>27.55</td>
<td>24.57</td>
<td>26.51</td>
<td>28.20</td>
<td>33.14</td>
<td>27.29</td>
<td>32.85</td>
</tr>
<tr>
<td></td>
<td>SSIM</td>
<td>0.8213</td>
<td>0.8578</td>
<td>0.8758</td>
<td>0.7310</td>
<td>0.8922</td>
<td>0.8876</td>
<td>0.8134</td>
<td>0.8704</td>
</tr>
<tr>
<td>MSRSC</td>
<td>PSNR</td>
<td>27.98</td>
<td>29.69</td>
<td>24.18</td>
<td>26.21</td>
<td>28.25</td>
<td>33.41</td>
<td>27.60</td>
<td>34.15</td>
</tr>
<tr>
<td></td>
<td>SSIM</td>
<td>0.8541</td>
<td>0.9213</td>
<td>0.8712</td>
<td>0.7113</td>
<td>0.8958</td>
<td>0.8972</td>
<td>0.8193</td>
<td>0.8862</td>
</tr>
<tr>
<td>ANR</td>
<td>PSNR</td>
<td>26.50</td>
<td>31.15</td>
<td>24.74</td>
<td>26.34</td>
<td>29.10</td>
<td>34.89</td>
<td>28.03</td>
<td>33.70</td>
</tr>
<tr>
<td></td>
<td>SSIM</td>
<td>0.7786</td>
<td>0.9614</td>
<td>0.8803</td>
<td>0.7242</td>
<td>0.9025</td>
<td>0.9204</td>
<td>0.8377</td>
<td>0.8848</td>
</tr>
<tr>
<td>SISU</td>
<td>PSNR</td>
<td>26.56</td>
<td>30.65</td>
<td>26.01</td>
<td>26.40</td>
<td>28.93</td>
<td>34.84</td>
<td>28.11</td>
<td>33.89</td>
</tr>
</tbody>
</table>

42
The average values in the last column of Table 3.3 and 3.4 indicate the improvement of the proposed method in PSNR/SSIM values. Note, the ESC based SR provides a full search of the dictionary, therefore, it outperforms the neighbor embedding approaches such as ANR and JOR in terms of visual quality. Figures 6 to 8 illustrate the visual quality and the superiority of the proposed method on various images.

![Figure 3-6: Visual comparison of Parthenon (up-sampling factor 2)](image-url)
3.4 Conclusion

This chapter presents a novel method for a SR image reconstruction using an evolutionary sparse coding technique. Sparse coding algorithms generally consume considerable resources to find the sparse representation $\alpha$ for each image patch. The proposed ESC method uses an adaptive variation of GA to reduce the total time of searching the dictionary while improving the accuracy. Experimental results show that
the proposed method is able to outperform other SR methods. It should be noted that ESC is a powerful learning based method which can be used in other areas such as multi-frame SR, remote sensing applications, or vector quantization. Expansion of the ESC approach in other fields will be the focus of future research.
Chapter 4

Small Dim Object Tracking Using Multi Objective Particle Swarm Optimization (MOPSO) Technique

Small moving object tracking in cluttered image sequences has attracted a great deal of research interest in recent years [2-10], mainly because of its application in surveillance systems such as radar, sonar and optical sensors. The dim object tracking appears in many fields including aerospace applications. An early approach to object detection is to perform frame differencing followed by thresholding [10]. In the case of dim moving objects where the noise level is high, a systematic procedure to decide whether or not an object is present is needed. For example, Rauch and Firschein performed a prescreening using 3x3 window morphological operation followed by a track assembler combining the images in the sequence. Dynamic Programming (DP) was first introduced by Barniv [8] for the detection and tracking of low-observable objects. The tracking performance of the DP based algorithms has been found to be poor even on images with moderate SNR values [13].

The detection of small moving objects of unknown position and speed using Multiple Hypothesis Testing (MHT) was initially introduced by Blostein [10]. In this method, a large number of candidate trajectories are organized into a tree structure. To prune the tree structure of the candidate trajectory, Wald’s truncated sequential probability test [42] is applied at each pixel. In fact, summation of the pixel values in
each trajectory is sequentially compared against two thresholds. The hypothesis that the trajectory contains a target would be accepted when the summations are greater than the maximum threshold, and rejected for summations less than the minimum threshold. When the test statistic falls between the two thresholds, this decision is deferred and the trajectory state is stored in a list. The main advantage of MHT based algorithms is they are independent of the target velocity, target direction, and the SNR values of the image sequence. These algorithms have good performance for very low SNR images. However, the computational requirements for processing large tree structures of tracks could make it inapplicable in some cases.

In a large scale search problem, optimization is an appropriate approach used to reduce the search space. The proposed method in this chapter applies Multi Objective Particle Swarm Optimization (MOPSO), a variant of Particle Swarm Optimization (PSO), which consists of several objectives that need to be achieved simultaneously. MOPSO is a computational intelligence-based technique that is not largely affected by the complexity and non-linearity of the problem. In the proposed method, MOPSO is used along with MHT to confine the tree search domain and reduce the computations without any assumption about the target movement. Moreover, a hierarchical optimization approach is introduced, in which all the tracks are first optimized by MHT algorithm followed by an iterative optimization technique to connect the best consecutive tracks found in two groups of image frames.

The organization of this chapter is as follows: Section 4.1 will begin with a brief review of the MHT based algorithms, the basic concepts, and thresholds. In section 4.2, the proposed method will be explained. Describing MOPSO and steps of the algorithm
showing how optimization technique and MHT together can be used to find the most accurate trajectories of multiple moving objects in low SNR image sequences. Section 4.3 presents the experimental results and compares the proposed method with some other dim object tracking methods. Conclusion is presented in the final section.

4.1 Basic Concepts of Multiple Hypothesis Testing

In the following discussion, the mean target intensity is assumed to be higher than the background as well as the noise intensity value. Generally the MHT based approach proceeds with the selection of a predefined number of points with maximum intensity value as the roots of the track trees. From each root, points of its neighbor in the next frame will be selected to construct a tree of tracks. MHT offers two thresholds, $T_1$ and $T_2$, and each point of the track will be compared with these thresholds. If the new point is greater than $T_2$, the track will be recorded and the algorithm continues with the next frame. However, if it is less than $T_1$, the track is rejected. If it is between $T_1$ and $T_2$, the decision defers to the next frame. Following this step, the tree would be pruned and reduced to a number of desired tracks. However, for high speed small targets, the search domain will increase exponentially, making the current MHT algorithms computationally impractical for a moving target with a speed of more than 1 pixel/frame. If the search area increases from 1 to 2 pixel/frame, the growth rate of the search tree would increase from 9 ($3 \times 3$) to 25 ($5 \times 5$) pixels. To resolve the problem of tracking high speed targets, the proposed method applies MOPSO to search for the best track of each root. The proposed algorithm initializes a track instead of a tree for each root which is obtained from the maximum of the neighboring points in the next frame. In each iteration of the process, MOPSO changes the position of all the points in the track toward the best global
and best local to look into a larger search area and find new tracks with better fitness value.

Before explaining the proposed algorithm in details, two thresholds $T_1$ and $T_2$ will be explained through mathematical calculations. In the following sub-section, a threshold $T$ will be declared for binary hypotheses testing and then it will be extended to two thresholds $T_1$ and $T_2$ for multiple hypotheses testing.

### 4.1.1 Binary Hypotheses Testing

For infrared detection and tracking (IRDT) applications, the target is close to a point source (several pixel size) [119] that concentrates itself in a relatively small region with uniform amplitude, and could be described by the following point spread function (PSF) [2, 9].

\[
r(x, y, k) = z(k) \cdot \exp \left\{ -\frac{1}{2} \left[ \left( \frac{x}{h(k)} \right)^2 + \left( \frac{y}{v(k)} \right)^2 \right] \right\}
\]  

(4.1)

In the above equation $r$ represents the target intensity in frame $k$ at spatial coordinates $(x, y)$, $z(k)$ is the target intensity amplitude, $h(k)$ and $v(k)$ are horizontal and vertical extent parameters. Because of aero-optic disturbances and air turbulence in an Infrared (IR) noisy image, the observed value $a(x, y, k)$ can be modeled as,

\[
a(x, y, k) = r(x, y, k) + n(x, y, k)
\]  

(4.2)

where $n$ refers to the noise. The broad objective is to determine the optimum value of $r$, given the observed value $a$. There are a variety of solutions to this problem, depending on the nature of a given image sequence and the characteristics of the noise. The amplitude of a pixel may be modeled as a random variable $R$ with a known or unknown Probabilistic Density Function (PDF) in which $r$ is a specific value that $R$ assumes in a
particular observation of the probabilistic model. Similarly, the noise and the observed value can be modeled as random variables $N$ and $A$, respectively. Therefore,

$$A = R + N$$  \hspace{1cm} (4.3)

In binary hypothesis testing, it is required to choose between two alternatives or hypotheses. The two hypotheses are $H_0$ and $H_1$ and are explained as follows:

$H_0$: the random variable $R$ takes the value $r_0$, the average value of the background and no target is present, i.e., $A = r_0 + N$

$H_1$: the random variable $R$ takes the value $r_1$, the average value of the target, therefore, $A = r_1 + N$

The binary hypothesis testing determines which of the two hypotheses, $H_0$ or $H_1$, is responsible for the observed value $a$ of a random variable $A$. It is assumed that a priori probability of occurrence of $H_0$ is $P(H_0) = h_0$, and a priori probability of occurrence of $H_1$ is $P(H_1) = h_1$. Discussion of hypothesis testing focuses on the criterion of minimum probability of error from choosing $H_0$ or $H_1$ when intensity value $A = a$ is observed in the image. It is an intuitively reasonable conclusion that choosing the hypothesis with maximum conditional probability leads to the minimum probability of error [120] as expressed in the following.

$$P(H_1 | A = a) > P(H_0 | A = a)$$

$$\frac{H_1}{H_0}$$  \hspace{1cm} (4.4)
Let $f_A(a)$ represent the PDF of the observed value of a random variable $A$ with $f_{A|H}(a|H_0)$ and $f_{A|H}(a|H_1)$ representing the conditional density functions with respect to the two hypotheses. Using Bayes’ rule, the above inequality can be rewritten as,

$$\frac{h_1 f_{A|H}(a|H_1)}{f_A(a)} > \frac{h_0 f_{A|H}(a|H_0)}{f_A(a)}$$

(4.5)

Since $f_A(a) > 0$, the above can be simplified as,

$$\frac{h_1 f_{A|H}(a|H_1)}{h_0 f_{A|H}(a|H_0)} < \frac{f_A(a|H_1)}{f_A(a|H_0)}$$

(4.6)

The minimum probability decision rule can be rewritten as a likelihood ratio,

$$A(a) = \frac{f_{A|H}(a|H_1)}{f_{A|H}(a|H_0)} > \frac{h_0}{h_1} = \eta$$

(4.7)

where $A(a)$ refers to the likelihood ratio and $\eta$ refers to a decision point.

Considering a Gaussian distribution for the observed value, i.e.,

$$f_{A|H}(a|H_0) = \frac{1}{\sqrt{2\pi}\sigma} \exp \left[-\frac{1}{2} \left(\frac{a-m_0}{\sigma}\right)^2 \right]$$

(4.8)

$$f_{A|H}(a|H_1) = \frac{1}{\sqrt{2\pi}\sigma} \exp \left[-\frac{1}{2} \left(\frac{a-m_1}{\sigma}\right)^2 \right]$$

(4.9)

where $m_0$ and $m_1$ are the mean intensity values of the background and the target, respectively. The noise is usually considered as a white Gaussian noise with zero mean independent of $m_0$ and $m_1$. Substituting (4.8) and (4.9) into the likelihood ratio in (4.7) will result in,

$$A(a) = \exp \left[\frac{1}{2\sigma^2} (2a(m_1 - m_0) + m_0^2 - m_1^2) \right]$$

(4.10)

Substituting equation (4.10) into (4.7) and simplifying leads to a threshold value for the observed pixel intensity.
\[ H_1 \]

\[ 2a(m_1 - m_0) + m_0^2 - m_1^2 > 2\sigma^2 \ln(\eta) \]
\[ H_0 \]

\[ a > \frac{1}{2(m_1 - m_0)} \left[ 2\sigma^2 \ln(\eta) - m_0^2 + m_1^2 \right] \]  \hspace{1cm} (4.12)

\[ a > \frac{\sigma^2 \ln(\eta)}{m_1 - m_0} + \frac{1}{2}(m_1 + m_0) = T \] \hspace{1cm} (4.13)

Threshold \( T \) obtained above can be used to choose between the binary hypotheses \( H_0 \) and \( H_1 \) with minimum probability of error.

### 4.1.2 Multiple Hypotheses Testing using the Neyman-Pearson Detection Scheme

The traditional concern in multiple hypothesis testing has been about controlling the probability of erroneously rejecting any of the true hypotheses [121]. An alternative approach, referred to as Neyman-Pearson detection scheme, tries to maximize the probability of detection \( P_D \) while keeping the probability of false alarm \( P_{FA} \) below a specified tolerable value. These conditional probabilities are defined by the following integrals under the specific decision region, \( D_1 \).

\[ P_D = P(H_1|H_1) = \int_{D_1} f_{A|H}(a|H_1) \ da \]  \hspace{1cm} (4.14)

\[ P_{FA} = P(H_1|H_0) = \int_{D_1} f_{A|H}(a|H_0) \ da \]  \hspace{1cm} (4.15)

\( P_D \) and \( P_{FA} \) can be obtained from the shaded area under two arbitrary conditional PDF in Figure 4.1.
Region $D_1$ is an interval of the observed value which causes selecting hypothesis $H_1$. In Figure 1, the areas under $f(a|H_1)$ and $f(a|H_0)$ in the interval $D_1$ represents the probability of detecting target ($P_D$) and the probability of false alarm ($P_{FA}$), respectively. In a binary hypothesis testing, threshold $T$ of equation (4.13) leads to an optimum decision region for $D_1$. Increasing the value of $T$ could result in decreasing the false alarm rate as well as the detection rate while decreasing $T$ will increase $P_D$ as well as $P_{FA}$. This characteristic of $T$ could be applied to define two thresholds $T_1$ and $T_2$ for multiple hypotheses testing. Hypothesis indicated by being greater than $T_2$ is referred to as the target pixels while less than $T_1$ indicates non-target pixels, i.e., noise or background. The pixels with values between $T_1$ and $T_2$ are considered to be uncertain and the decision is deferred to the next frame. Experience shows that the best values for $T_1$ and $T_2$ are defined as:

$$T_2 = T + \sigma$$  \hspace{1cm} (4.16)
$$T_1 = T - \sigma$$  \hspace{1cm} (4.17)

where $\sigma$ is the standard deviation of the image. These two thresholds are used in the proposed method in the following section.
4.2 Proposed Method

The proposed hierarchal tracking model consists of two levels. In the lower-level, a predefined number of consecutive frames are considered as a group. Multiple Hypotheses Testing (MHT) and the two thresholds defined in equations (4.16) and (4.17) are used to determine the candidate track for each moving object, and then Multi Objective Particle Swarm Optimization (MOPSO) is applied to refine the initial tracks in each group. The only information used at this level is the pixel intensity value. In upper-level, a second iterative optimization process is applied to connect the matching tracks of two consecutive groups. The overall proposed method can be summarized as following:

Upper-level process:
- Find initial tracks using MHT.
- Refine and adjust the tracks using MOPSO.
- Merge the tracks corresponding to a single object.

Lower-level process:
- Link the tracks of neighboring sequences using continuity

4.2.1 Finding Initial Tracks Using MHT

The continuity theory of image trajectory is two-fold. It states that if a constant point source is moving in 3D space along a continuous and smooth trajectory, (1) the spatial coordinates of its image trajectory are continuous and smooth (up to its first derivative); (2) the intensity of its image trajectory is also continuous [122]. Based on this theory, continuous high intensity points in consecutive frames show a moving object. The proposed method begins with a set of predefined points with maximum intensity values from a set of initial frames in each group. This process starts with the first frame and continues with the following frames until to reach with a
set of predefined maximum points. These set of points will serve as the initial points for each track. This issue will be further discussed in the experimental results section.

For each initial point, a track is constructed by searching in a 3X3 window at the same position in the next frame. The 3X3 window indicates a 1-pixel motion per frame. However, a larger size of window can be used for motion of more than one pixel.

It is assumed that two consecutive pixels on a track can have intensity values between $T_1$ and $T_2$, which is controlled by auxiliary variable $U$. Similarly, only one pixel on a track can have an intensity value less than $T_1$ that is controlled by another auxiliary variable $V$. For each tree root, $U$ and $V$ are initially set to zero. Variable $P$ in the following pseudo code represents the neighboring point in the next frame with maximum value. MHT controls the construction of each track by $T_1$ and $T_2$, as follows:

1) **if** $P > T_2$ **then** record $P$; set $U=0$; go to (5);  
2) **if** $T_2 > P > T_1$ **&&** $U < 2$ **then** record $P$; increment $U$; go to (5);  
3) **if** $P < T_1$ **&&** $V < 1$ **Then** record $P$; set $V=1$; go to (5);  
4) **else** Remove the Track; go to the next root;  
5) For each recorded point, **find** a new $P$ as the maximum neighbor in the next frame; go to (1);

The total number of all the tracks will be less than or equal to the number of initial points because there is a possibility those tracks which did not satisfy the conditions can be removed by the code in line number (4) above.

**4.2.2 Refining and Adjusting the Tracks Using MOPSO**

The initial tracks found in the previous step are subject to improvement by searching in the vicinity of each point to find a track with a better fitness. This chapter aims to confine the search domain of MHT by applying MOPSO which is a variant of
PSO. PSO uses the swarm intelligence concept which is the property of a system, whereby the collective behaviors of unsophisticated agents that are interacting locally with their environment, create coherent global functional patterns. PSO exhibits each solution as a particle moving through the problem hyperspace. Each track, which consists of a set of points represented by a vector $\vec{x}_i$ with corresponding velocity represented by, $\vec{v}_i$, will be considered as a particle. The position of each particle and its motion are related by the following equations [63-64].

$$\vec{x}_i(t) = \vec{x}_i(t - 1) + \vec{v}_i(t) \quad (4.18)$$

$$\vec{v}_i(t) = \vec{v}_i(t - 1) + \varphi_1 \cdot rand_1 \cdot (\vec{L}_i - \vec{x}_i(t - 1)) + \varphi_2 \cdot rand_2 \cdot (\vec{G}_i - \vec{x}_i(t - 1)) \quad (4.19)$$

where, $\varphi_1$ and $\varphi_2$ are two positive numbers and $rand_1$ and $rand_2$ are two random numbers with uniform distribution in the range of $[0.0, 1.0]$. Moreover, $\vec{L}_i$ and $\vec{G}_i$ refer to the local best and the global best particle of the whole swarm, respectively. Best global ($\vec{G}_i$) and best local ($\vec{L}_i$) which are selected based on the fitness values of the particles in each iteration of the process display the particle with the highest fitness value relative to all other values and relative to its own past values, respectively. Equation (4.19) shows that the movement of a particle is based on its own experience and the knowledge of the performance of other individuals in its neighborhood. Since the relative importance of these two factors can vary from one decision to another, it is reasonable to apply random weights to each part.

MOPSO consists of several objectives that need to be achieved simultaneously. The main approach to multi-objective problems is combining all of the objectives into one objective function with weights that can be fixed or dynamically changed during the optimization process [123].
To further explain this method, each track is considered as a particle of the swarm and a point at coordinates \((x,y)\) on the track is an element of the particle. The fitness value of each track is determined based on the accumulation of the difference between each point’s intensity and its ideal value from the Gaussian distribution. The ideal object intensity \(GN\) follows a Normal distribution with mean \(m\) and standard deviation \(\sigma\), i.e.,

\[
GN \sim \text{Normal}(m, \sigma)
\]  

The average deviation of partial summation of all pixels’ intensity in a track from the accumulative Normal distribution is the fitness of the track.

\[
\text{Fitness} = \frac{\sum_{i=1}^{N}\left(\sum_{k=1}^{I}(a(x_k,y_k,k) - GN)\right)}{N}
\]  

Note, \(a(x_k,y_k,k)\) is the pixel intensity of a track point in frame \(k\), and \(N\) is the number of points on the track.

During the iterative process of the MOPSO, the fitness values of the tracks will change. The track with the best fitness value will be the best global particle of the search domain. However, for each track there is also a best local particle which is relative to its own past values and refers to a track with the best fitness value during all the previous iterations.

After initializing the tracks, defining the fitness values, finding the best global, and the best local, the following process is run for a predefined number of iterations (epochs) to find the best optimum tracks for each root-tree. In each epoch, all of the tracks should be considered. For each point \(a(x,y,k)\) on a \(Track_i\), the following steps will be followed iteratively:

a) Select two integer numbers \(i, j \in [1,3]\) randomly.
b) Update each point \( a \) on the Track, by moving \( i \) points in the direction of the best global point followed by moving \( j \) points in the direction of the best local point of Track, in frame \( k \). In this way, a new track is reconstructed according to previous pseudo code in III-A.

c) If a new track is successfully found, Track\( _i \) is then replaced with this new track.

d) Considering the newly reconstructed track, update the best global and the best local particle.

After a predefined number of epochs, a few tracks (less than its initial number) will remain. Best locals are the desired tracks for each root.

4.2.3 Merging Tracks Corresponding to a Single Object

As this chapter does not have any restriction on the size of the moving objects, a moving object larger than a pixel-size is possible. Therefore, it could be possible to select more than one root for a moving object which results in more than one track in the final step for the same object. To solve this problem, the proposed method unifies all the tracks which belong to the same object based on their proximity and finds their average as a representative track.

\( Track_a \) and \( Track_b \) belong to the same object if the following is true:

\[
\sum_{i=1}^{N} \| Q_i - Q'_i \| \leq D \cdot N \tag{4.22}
\]

where \( N \) is the number of points on the track and \( D \) is the maximum allowable distance (20 in our case). \( Q_i \) and \( Q'_i \) are two points on \( Track_a \) and \( Track_b \), respectively.

The coordinates of the representative track is then obtained by the average value of the points, i.e.,

\[
X(j) = \frac{\sum_{k=1}^{N} x_{k}^{(j)}}{N} \quad , \quad Y(j) = \frac{\sum_{k=1}^{N} y_{k}^{(j)}}{N} \quad , \quad \forall \ j \in (1, N) \tag{4.23}
\]
where \((x_k^{(j)}, y_k^{(j)})\) are the coordinates of a point in frame \(k\) of \(Track_j\), and \((X^{(j)}, Y^{(j)})\) are the coordinates of the corresponding point on the representative track.

### 4.2.4 Extending Neighboring Tracks Using Continuity property

In the lower-level, a representative track is found for each moving object in each group of consecutive frames. In the upper-level, the tracks of two consecutive group of frames are matched together to form a larger trajectory. Note that the pixel intensity was the only characteristic used in order to find the best track in the lower-level. However, for the upper-level, the fitness of each track and their proximity are two characteristics that can lead to the extension of the best matching tracks. This hierarchical approach could provide a more reliable trajectory for a moving object.

To connect two tracks \(i\) and \(j\) from two consecutive groups of frames, two characteristics are used, the fitness values and their proximity. Consider \(n\) tracks from the first group and \(m\) tracks from the next group, the following two matrices \(P\) and \(M\) are defined,

\[
P = \begin{bmatrix} P_{1,1} & \cdots & P_{1,m} \\ \vdots & \ddots & \vdots \\ P_{n,1} & \cdots & P_{n,m} \end{bmatrix}, \quad M = \begin{bmatrix} M_{1,1} & \cdots & M_{1,m} \\ \vdots & \ddots & \vdots \\ M_{n,1} & \cdots & M_{n,m} \end{bmatrix}
\]

where,

\[
p_{i,j}^{(0)} = \frac{f_{\text{min}}}{f_{\text{max}}}, \quad M_{i,j} = (|x_i - x_j|, |y_i - y_j|)
\]

Note, \(f_{\text{min}}\) and \(f_{\text{max}}\) are the minimum and maximum fitness values of \(Track_i\) and \(Track_j\). Also, \((x_i, y_i)\) and \((x_j, y_j)\) are the coordinates of the end point of \(Track_i\) and starting point of \(Track_j\), respectively. When the fitness values of two tracks are close to each other, \(p_{i,j}^{(0)}\) approaches to one.
Let \((k,l)\) denote the connection between two tracks \(k\) from the first group and \(l\) from the neighboring group. Two connections \((k,l)\) and \((i,j)\) are defined to be neighbor, if 
\[ \|M_{i,j} - M_{k,l}\| \leq MAX \] (note, \(MAX\) is a predefined constant). The following iterative process based on distance and fitness values is devised to find the best matching tracks between two consecutive groups of frames.

a) \(\forall (k,l)\) as a neighbor of \((i,j)\), let’s define 
\[ G_{i,j}^{n-1} = \sum_{all (k,l)} P_{k,l}^{(n-1)} \] to be the merit of connecting \(Track_i\) and \(Track_j\).

b) Calculate 
\[ P_{i,j}^{(n)} = A \cdot P_{i,j}^{(n-1)} + B \cdot G_{i,j}^{(n-1)} \], where \(A\) and \(B\) are two user-defined constants.

c) 
\[ P_{i,j}^{(n)} = \frac{P_{i,j}^{(n)}}{\sum_{q=1}^{m} P_{i,q}^{(n)}} \], where \(m\) is the number of tracks that match \(Track_i\). Find a new value for the probability matrix; increment \(n\); go to step (a).

The above process will iterate until all the elements of the probability matrix converge either to 0 or 1. Two tracks \(i\) and \(j\) are matching tracks when 
\[ P_{i,j}^{(n)} \approx 1. \]

### 4.3 Experimental Results

The proposed algorithm has been implemented in MATLAB. Its performance and simulation results are presented in this section. To evaluate the performance of the proposed method relative to some other methods, six image sequences, S1, …, S6 with different characteristics as shown in Table 4.1 are selected. Figure 4.2 shows the results of simulations of the proposed method relative to the other techniques, such as, Max-Mean [7], Bae’s Method [124], Dong’s Method [125], and two other MHT based methods, namely, Lisha’s method [126] and Blostein’s method [10]. The speed of the target in image sequence S1 and S2 is between [0,1] pixel/frame in which all of the above
algorithms are applicable. However, high speed moving objects in image sequence S3, …, S6 have a displacement of more than 1 pixel/frame, therefore, Lisha and Blostein’s methods are not applicable on these image sequences.

**Table 4.1: Properties of six image sequences**

<table>
<thead>
<tr>
<th>Image Sequence</th>
<th>Description</th>
<th>Background</th>
<th>Target</th>
<th>Target size (in Pixel)</th>
<th>Image Size (in Pixel)</th>
<th>Target Speed (in pixel/frame)</th>
</tr>
</thead>
<tbody>
<tr>
<td>S1</td>
<td>A small number of motorcycle and cars are moving along the street</td>
<td>The road and some unknown objects on the side of the road</td>
<td>Motorcycles and Cars – Multiple Moving Objects</td>
<td>Different window size from 3 X 3 to 12 X 14</td>
<td>700 X 600</td>
<td>0 1</td>
</tr>
<tr>
<td>S2</td>
<td>A bird moving in clear sky</td>
<td>Clear sky without clouds</td>
<td>A Bird</td>
<td>4 X 5</td>
<td>400 X 320</td>
<td>0 1</td>
</tr>
<tr>
<td>S3</td>
<td>An airplane is flying in cluttered noisy sky</td>
<td>Salt and Pepper noise added to the cluttered background</td>
<td>An Airplane</td>
<td>2 X 5</td>
<td>400 X 320</td>
<td>1 2</td>
</tr>
<tr>
<td>S4</td>
<td>An airplane is flying in the clouds</td>
<td>Clouds in sky</td>
<td>An Airplane</td>
<td>9 X 9</td>
<td>400 X 320</td>
<td>1 2</td>
</tr>
<tr>
<td>S5</td>
<td>Six airplanes are flying in clear sky</td>
<td>Clear sky with a tree in left corner of image</td>
<td>Six Airplanes</td>
<td>One: 11 X 5 Five: 3 X 3</td>
<td>360 X 240</td>
<td>1 3</td>
</tr>
<tr>
<td>S6</td>
<td>An airplane is flying in cloudy sky</td>
<td>A cloudy sky with a city view of tall buildings in background</td>
<td>An Airplane</td>
<td>2 X 3</td>
<td>400 X 320</td>
<td>1 4</td>
</tr>
</tbody>
</table>
Note that S3 contains a significant amount of noise which causes problems for some algorithms. In sequence S5 and S6, the background noise leads to miss detection of the target in some algorithms.

In our implementation, the algorithm considers a set of 100 points with high intensity level as the initial points. Since the number of objects in the sequence (i.e., the number of tracks) is less than 6 in the above data sets, therefore, this should be more than what is needed in order to find all the tracks successfully. In fact, choosing more initial points have no improvements on these data sets.

The number of frames in each group is an issue of concern at the lower-level. In our experimental results, the initial points are selected from the first half of the total frames in a group. Therefore, new objects appearing in the first half of the frames are tracked by this method. Other objects appearing in the second half of the group are picked up in the next group. In our experiments, the number of frames in a group is set to ten, therefore, in regular 30 frames/second videos, the time that a track is lost will be less than \((5/30) = 0.16\) seconds which is relatively negligible. In general, the lost time \((LT)\), indicating the absence of the object on a track, can be measured as, \(LT = nf/(2\times30)\), where \(nf\) is the number of frames in a group.
A comparison of the common dim object tracking methods with the proposed scheme, (A) is the first frame of original sequence, (B) Max-Mean, (C) Bae’s Method, (D) Dong’s Method, (E) Lisha’s method, (F) Blostein’s method, (G) proposed method

To further evaluate the performance of these algorithms, an objective measure, namely Error-rate Per Frame (EPF) is used. EPF is a measure of the average distance between the detected points and the real target points as following,

$$EPF = \frac{\sum_{\text{for all } DT} ||P_{\text{real}} - P_{\text{detected}}||}{DT}$$

(4.26)

where, $P_{\text{real}}$ is the real position of the target, $P_{\text{detected}}$ is the detected position of the target, and DT is the number of detected targets. EPF is calculated for all algorithms over six image sequences S1, ..., S6, and the results are shown in Figure 4.3 in the form of a graph. Moreover, Table 4.2 compares the average EPF values of various methods over
the image sequence S1, …, S6. Both Figure 4.3 and Table 4.2 show the advantages of the proposed method in comparison with the common dim object tracking algorithms.
Figure 4-3: Error Per Frame (EPF) of different methods over S1-S6
Table 4.2 Average EPF for image sequences S1-S6

<table>
<thead>
<tr>
<th>Method</th>
<th>S1</th>
<th>S2</th>
<th>S3</th>
<th>S4</th>
<th>S5</th>
<th>S6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Max-Mean</td>
<td>6.1</td>
<td>1.6</td>
<td>11.7</td>
<td>8.7</td>
<td>3.9</td>
<td>1.7</td>
</tr>
<tr>
<td>Bae's Method</td>
<td>2.3</td>
<td>1.2</td>
<td>2.4</td>
<td>2.3</td>
<td>1.4</td>
<td>4.1</td>
</tr>
<tr>
<td>Dong's Method</td>
<td>3</td>
<td>1.6</td>
<td>2.5</td>
<td>2</td>
<td>1.9</td>
<td>3.5</td>
</tr>
<tr>
<td>Lisha's Method</td>
<td>3.9</td>
<td>2.6</td>
<td>8.9</td>
<td>7.4</td>
<td>8.7</td>
<td>7.8</td>
</tr>
<tr>
<td>Blostein's Method</td>
<td>4.1</td>
<td>1.6</td>
<td>8.3</td>
<td>6.1</td>
<td>9.1</td>
<td>7.9</td>
</tr>
<tr>
<td>Proposed Method</td>
<td>1.2</td>
<td>0</td>
<td>1</td>
<td>0.5</td>
<td>0.5</td>
<td>0.6</td>
</tr>
</tbody>
</table>

It is, however, insufficient to justify the effectiveness of an algorithm simply by measuring the EPF, because there are other critical parameters associated with a tracking algorithm in addition to the detected target points. Critical parameters include detected targets (DT), missed targets (MT), and false alarm detection (FA) in a frame. Therefore, a more comprehensive measure, namely Target to Clutter Ratio (TCR) is defined for comparing different tracking algorithms as expressed below.

\[
TCR = \frac{DT}{DT + MT + FA}
\]  

(4.27)

Table 4.3 shows a comparison of TCR values for various algorithms over image sequences S1,…, S6.
Table 4.3 TCR of different methods over S1-S6

<table>
<thead>
<tr>
<th></th>
<th>S1</th>
<th>S2</th>
<th>S3</th>
<th>S4</th>
<th>S5</th>
<th>S6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Max-Mean</td>
<td>0.222</td>
<td>1.000</td>
<td>0.083</td>
<td>0.143</td>
<td>0.545</td>
<td>0.077</td>
</tr>
<tr>
<td>Bae's Method</td>
<td>0.400</td>
<td>1.000</td>
<td>1.000</td>
<td>1.000</td>
<td>0.650</td>
<td>0.125</td>
</tr>
<tr>
<td>Dong's Method</td>
<td>0.400</td>
<td>1.000</td>
<td>0.997</td>
<td>1.000</td>
<td>0.778</td>
<td>0.112</td>
</tr>
<tr>
<td>Lisha's Method</td>
<td>0.364</td>
<td>1.000</td>
<td>0.111</td>
<td>0.500</td>
<td>0.429</td>
<td>0.111</td>
</tr>
<tr>
<td>Blostein's Method</td>
<td>0.500</td>
<td>1.000</td>
<td>1.000</td>
<td>1.000</td>
<td>0.500</td>
<td>1.000</td>
</tr>
<tr>
<td>Proposed Method</td>
<td>0.833</td>
<td>1.000</td>
<td>1.000</td>
<td>1.000</td>
<td>1.000</td>
<td>1.000</td>
</tr>
</tbody>
</table>

Table 4.3 also indicates that the proposed method has the least number of missed targets or false alarm detections. It can be seen that the detection rate for Blostein’s algorithm is reasonably good. However, this method cannot follow fast moving objects in sequences S3, ..., S6 as indicated in Table 4.2.

The complexity of the proposed method is an issue of concern. Considering an $n \times n$ size video, $k$ frames in a group, and $p$ predefined maximum points. The proposed method begins with selecting the initial maximum points which is in the order of $O(n^2)$. For each initial point, 9 points in the next frame are compared to find maximum point and construct a track of $k$ points. This comparison has a complexity of $O(k.p)$. To refine these tracks by MOPSO, $i$ iterations should be done. In the worst case, for each iteration the position of all $k.p$ points are changed. A change in the position of a point changes the entire track, and required $9.k$ points to be compared in order to reconstruct a new track. Therefore, PSO algorithm has a complexity of $O(i.k^2.p)$. To merge the tracks, each track should be compared to other tracks which leads to a complexity of $O(p^2.k)$. Considering
k,p, and i to be constant, the overall complexity of the proposed algorithm is estimated to be $O(n^2)$.

Table 4.4 shows the time complexity of the proposed method in comparison with others. The processing time of Max-mean filter is superior in terms of its computational time requirement. However, it has inferior detection rates, EPF, and TCR as shown in the above Figures and Tables.

**Table 4.4 Processing time (in Second) of different methods over 10 frames**

<table>
<thead>
<tr>
<th>Method</th>
<th>S1</th>
<th>S2</th>
<th>S3</th>
<th>S4</th>
<th>S5</th>
<th>S6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Max-Mean</td>
<td>6.50</td>
<td>6.89</td>
<td>4.12</td>
<td>5.60</td>
<td>8.11</td>
<td>5.77</td>
</tr>
<tr>
<td>Dong's Method</td>
<td>7.45</td>
<td>6.18</td>
<td>10.52</td>
<td>8.76</td>
<td>9.88</td>
<td>10.91</td>
</tr>
<tr>
<td>Lisha's Method</td>
<td>17.12</td>
<td>17.15</td>
<td>16.50</td>
<td>18.15</td>
<td>19.17</td>
<td>13.44</td>
</tr>
<tr>
<td>Blostein's Method</td>
<td>19.04</td>
<td>18.41</td>
<td>19.75</td>
<td>17.49</td>
<td>20.45</td>
<td>16.59</td>
</tr>
<tr>
<td>Proposed Method</td>
<td>8.56</td>
<td>12.02</td>
<td>8.78</td>
<td>9.37</td>
<td>13.08</td>
<td>7.14</td>
</tr>
</tbody>
</table>

### 4.4 Conclusion

This chapter presents a novel technique for object detection and tracking in low SNR image sequences with unknown background clutter and noise distribution. The algorithm is also insensitive to the speed mismatch or a target maneuver. The proposed method uses the concept of MHT tracking followed by a variation of particle swarm optimization approach to search for the best track instead of searching a large tree. The
proposed multiple objective particle swarm optimization (MOPSO) could reduce the large search area of MHT for each root-tree and find the best track for each object.

The simulation results show that the proposed algorithm is capable of detecting and tracking targets in image sequences with low SNR values (less than 2dB) and high speed moving objects. Comparing with some other common algorithms, the proposed method in this chapter is more effective with relatively simple implementation. In addition, it avoids false track detection when the target signal-to-noise ratio is low.
Chapter 5

Small Dim Object Tracking Using Frequency and
Spatial Domain Information

The detection and tracking of small dim targets in Infrared (IR) images plays a key role in video surveillance and military image guidance. The development of small dim object tracking is of interest in situations where the objects being tracked are a great distance from the imaging equipment causing the objects to appear small and dim in the image plane. In addition, IR images have a low Signal to Noise Ratio (SNR), usually less than 3 dB, which causes the background of these images to be noisy, complicated and chaotic. The development of a novel detection and tracking methodology that can more efficiently identify small dim objects is the interest of this chapter.

Most spatial based Infrared Search and Track (IRST) systems assume a Gaussian distribution for the target region. In addition, they usually assume that the target intensity is either higher or lower than the background. These two assumptions restrict the functionality of tracking systems in the real world [26]. Also, clutter in the background can cause unexpected changes in intensities. In order to deal with this issue, a temporal decomposition wavelet that can detect multiple targets in the presence of clutter using a change detection map is applied. The output at this stage in the tracking process is to
provide a list of candidate targets. The wavelet-based temporal detection approach provides a practical solution but only for the detection of single-pixel targets. In the real world, the target size may greatly vary from one point to several points in an image sequence. In order to overcome this limitation, morphological operations that use frequency domain information are widely used. These methods suppress the background clutter by applying a mathematical morphology which enhances target intensity values. High frequency areas commonly provide features used in these algorithms, however, there is a risk of background edge misdetection. Therefore, the detection of a point target requires the integration of target intensity values over multiple frames.

In this chapter, a Dual-Tree Complex Wavelet Transform (DT-CWT) is applied to each frame to obtain high frequency sub-bands in six different directions. The Constant False Alarm Rate (CFAR) detection is then applied to the high frequency sub-bands to identify potential target areas. The high frequency areas that surpass the threshold are identified as potential blocks that could be small targets (hit) or the edge points of the background (false alarm). Therefore, these potential blocks will pass through a SVM classifier for a final decision. Through the use of this method, the SVM classifier will separate the targets from the non-targets based on features in the spatial domain. The main contribution of this chapter is the integration of both spatial and frequency domain features in order to more accurately locate the position of the targets without any specific assumption.

The organization of this chapter is as follows: Section 5.1 describes the proposed method along with multi-frame trajectory construction method overviews. In Section 5.2, an analysis of the model and its parameters is illustrated through relevant graphs and
tables. Moreover, a comparison of the proposed method with other related dim object tracking algorithms is shown in this section. Finally, the conclusion with a summary of the proposed method and its advantages is presented in the last section.

5.1 Proposed Tracking Method Using DT-CWT and SVM

The proposed method is designed to detect small and dim objects in a cluttered background. According to [2], the small dim object is defined to be close to a point source (or several pixel sizes) that concentrates itself in a relatively small region with higher density at the center. The proposed method consists of three steps. First, a DT-CWT of each frame in the image sequence is obtained. The targets (hit) and edge area of the background (false alarm) are identified in six high frequency sub-bands. In the second step, the variance of a small window in the neighborhood of each coefficient is calculated in high-frequency sub-bands. The CFAR detection module selects points with higher variance values. In the third step, a small block for each of the selected points in the previous step is provided to a SVM classifier. The SVM module will then decide whether a given block belongs to a target or to a background region. Note, CFAR is applied to the frequency domain, while SVM classifies potential areas based on the spatial domain information in order to achieve a consolidation of the frequency and spatial domain information. Figure 5.1 shows the flowchart of the proposed model.
In summary, the proposed method is expressed as follows:

- **Find high frequency sub-bands using DT-CWT.**

- **Detect potential targets in high-frequency sub-bands using the CFAR detection module.**

- **Refine potential targets using SVM in the spatial domain.**

### 5.1.1 Find high frequency sub-bands using DT-CWT

Figure 5.2 (a, b) shows an image with different lines in different directions and the accumulated high frequency sub-bands, respectively. The individual six DT-CWT high frequency sub-bands of this image are shown in Figure 5.3 (a - f). As shown in Figure 5.3, each line in the frequency domain is represented in one or more sub-bands according to its direction. To capture every target regardless of its direction, the accumulation of all
six sub-bands is then necessary. The accumulated sub-bands, shown in Figure 5.2 (b), will serve as an input to the CFAR detection module.

Figure 5-2: (a) The original image, (b) The accumulation of all high frequency sub-bands

Figure 5-3: (a, b, c, d, e, f) High frequency sub-bands in six different directions
5.1.2 CFAR detector

In this step, the CFAR detection is applied in the spatial domain in which the separation of the background pixels from the target pixels is rather difficult. However, the proposed method exploits the use of CFAR detection in high frequency sub-bands, allowing the system to capture areas with large changes in the coefficients in comparison to their neighbors. The proposed method calculates the variance $V(x, y)$ of a sliding window for each coefficient, and then CFAR detection is performed on these variance values. The variances of the coefficients related to the background are very close to zero leading to its mean value to be relatively small. Therefore, the threshold value $T$ of CFAR equation (2.4) in chapter 2 is considerably low. In our implementation, the threshold value $T$ of CFAR equation (2.4) was further enhanced by first calculating the mean variance, $\bar{V}$, i.e., the mean of all the $V(x, y)$ values. Then, $\mu$ and $\sigma^2$ of those $V(x, y)$ which are greater than $\bar{V}$ are calculated. Therefore, the proposed threshold is a combination of the mean and the variance of windows with higher variance values.

Parameter $\lambda$ in CFAR equation (2.4) is determined by a desired $P_{FA}$. According to the central limit theorem, $P_{FA}$ is obtained as follows [3],

$$P_{FA} = 0.5 - \frac{1}{\sqrt{2}} \int_{0}^{\lambda} \sqrt{2 \pi} e^{-x^2} dx,$$

(5.1)

Let $x = z/\sqrt{2}$, equation (5.1) can be rewritten in the following Gaussian form,

$$\frac{1}{\sqrt{2}} \int_{-\infty}^{\lambda} e^{-\frac{z^2}{2}} dz = 1 - P_{FA},$$

(5.2)

Having a desired value for $P_{FA}$, parameter $\lambda$ can then be found from the above equation by a look-up Gaussian Table.
To avoid detecting noise as a target, an additional step is incorporated into equation (2.3). To select \( F(x, y) \) as a potential target, it is assumed that not only \( F(x, y) \), but also the mean of a small block around \((x, y)\) should be greater than \( T \). Therefore, the proposed thresholding operation is expressed as in equation (5.3),

\[
\begin{cases}
\text{Potential Target,} & F(x, y) \geq T \text{ and } E(\text{block}(x, y)) \geq T \\
\text{Background,} & \text{otherwise}
\end{cases}
\] (5.3)

where, \( E(\text{block}(x, y)) \) is the mean of a small block around the point \((x, y)\), called the target block. A target block is defined to be a block containing the potential target. The CFAR detection module will detect the possible candidate targets in the current frame and pass these candidates to the SVM module for classification, which uses spatial domain features for final decision.

5.1.3 Potential targets refinement using SVM

The use of SVM in the proposed method requires an initial training phase with spatial features of the dim objects. Feature selection is important for object tracking systems. Using more features can increase system complexity, yet it may not always lead to higher detection accuracy. Also, complex SVM models may be sensitive to the noise in the training data set and its performance on unseen test data may be poor [127]. In addition, according to a study in Lincoln Lab [48], adding extra features can potentially degrade the performance of a tracking system.

In our experiments, we found that features that depend on the size of objects are not effective in cases where the targets are partially occluded or hidden. Similarly, features which simply count the number of pixels that exceed a threshold value are also ineffective. A set of five prominent spatial features have been selected to train the SVM
classifier. The first two features are the mean and the standard deviation of all the pixels of each target block. High variance blocks can indicate the presence of targets. The third feature, namely, the Hausdorff dimension is a measure of spatial diffusion of the $M$ sprinkles in a target block. In order to compute the value of feature 3, $M$ higher intensity points in each block are selected as sprinkles. The minimum number of 1 x 1-pixel boxes ($d_1 = 1$) that cover all $M$ sprinkles is denoted by $m_1$. This number should be equal to $M$. The minimum number of 2 x 2 pixel boxes ($d_2 = 2$) that cover all $M$ sprinkles is represented by $m_2$. This number is less than or equal to $M$. The Hausdorff dimension for a target block is then calculated as,

$$H_d = \frac{\log m_1 - \log m_2}{\log d_2 - \log d_1}, \quad (5.4)$$

If the spatial distribution of the sprinkles is highly diffused, the value of $m_2$ will be close to $m_1$, and $H_d$ will be close to zero. Sprinkles in this case may be related to noise rather than a target. On the other hand, if the sprinkles are spatially close to each other, the value of $m_2$ will be considerably less than $m_1$, leading to a high value for $H_d$ and, as such, it is more likely to be a target rather than noise or a false alarm.

Feature 4 is the weighted-rank fill-ratio. This feature measures a power-related property of the target block. The power of the $M$ highest points of each block, same as sprinkles in feature 3, should be selected. This power is then normalized by the power obtained from all the pixels in the block. This feature attempts to exploit the fact that power returned from most targets tends to be concentrated in a few bright pixels, whereas power returned from natural-clutter or false alarms tend to be more diffused.
The fifth feature used for the SVM training is based on the average-distance of the foreground pixels in each block relative to its centroid. The centroid of each target block \((\bar{x}, \bar{y})\) can be calculated through the following equations:

\[
\bar{x} = \frac{\sum_{i} \sum_{j} x_i \cdot f(x_i, y_j)}{\sum_{i} \sum_{j} f(x_i, y_j)}, \tag{5.5}
\]

\[
\bar{y} = \frac{\sum_{i} \sum_{j} y_j \cdot f(x_i, y_j)}{\sum_{i} \sum_{j} f(x_i, y_j)}, \tag{5.6}
\]

where, \(f(x_i, y_j)\) is the intensity value of the point at coordinates \((x_i, y_j)\) in a block.

Distances of the entire foreground pixels in each block to the centroid are computed. The foreground pixels are those pixels whose variance passed the CFAR detection threshold in step 2 of the proposed algorithm. Feature 5 is yet another type of spatial distribution measurement. Lower values are more likely representative of the target region because the targets show more dense bright pixels. Conversely, higher values show scattered noises in the block.

The SVM provides the location of each target in each frame. To find the target trajectory, the individual frames need to be combined. The trajectory of a target is continuous and consistent, while that of a non-target (noise) is not. Therefore, multi-frame accumulation can be performed to find the target trajectories. Note, standstill targets are present in more than one frame at the same location. Hence, the accumulated values related to standstill targets have higher values in comparison to noise.

In summary, the CFAR detection scheme is used to detect possible candidate blocks in the frequency domain and the SVM module is applied to finalize the decision using the spatial domain information. Five features are calculated for each candidate block, and the sign of the decision function in equation (5.3) determines whether it is a real target or a false alarm. Accumulation of multiple-frame provides the target trajectory. Alternatively,
only target points in a block can be taken into account in order to display the target trajectory.

5.2 Experimental Results

The proposed method has been implemented in MATLAB and its performance is presented in this section. To evaluate the performance of the proposed method relative to others, the results of simulations are compared with other tracking systems implemented both in frequency as well as spatial domains. It is shown that the combination of the frequency and spatial domain information used in our method can provide superior results.

In the training phase of SVM, we used a collection of 50 different Infrared data sets from the public domain, including MSTAR (www.sdms.afrl.af.mil), OTCBVS benchmark (www.vcipl.okstate.edu/otcbvs/bench), and BU-TIV video benchmark [128] to train the SVM classifier for a vast range of small dim objects. The selected collection includes different image resolutions, all at 30 frames per second. Additionally, some of the images are very noisy with a SNR value lower than 2 dB. Table 5.1 illustrates the characteristics of the 50 training data sets.

| Table 5.1: Properties of 50 training image sequences |
|---------------------------------|-----------------|-----------------|-------------------|
| Moving Target | Background | Target Size | Selected Frames |
| 10 Data sets | Bird | Cluttered cloudy sky | Less than 10x10 | 30 (1 Sec.) |
| 15 Data sets | Airplane | Cluttered cloudy sky | Less than 10x10 | 30 (1 Sec.) |
| 5 Data sets | Car | Street view | Less than 10x10 | 30 (1 Sec.) |
| 15 Data sets | Drone | Cluttered sky + Street view | Less than 10x10 | 30 (1 Sec.) |
| 5 Data sets | Football Players | Playground | Less than 10x10 | 30 (1 Sec.) |
In this implementation, the size of the sliding variance window is set to 7-pixel by 7-pixel, and the target block size is set to 10-pixel by 10-pixel. Therefore, each support vector is a small image of size 10×10 pixels. There are 50 training data sets of a variety of moving objects, as indicated in Table 5.1 with 30 frames in each set. From the training data sets, 100 real targets and 100 non-target (background) blocks are selected manually in the training phase of the SVM. The trained model is constructed base on the 200 target and non-target blocks. The target blocks can train the moving targets as described in Table 5.1 and non-target blocks are selected from the background. We used a homogeneous quadratic polynomial kernel in the form of \( K(x, x_j) = (x^T . x_j)^2 \) to perform the learning phase of SVM. Also, the transformation function is simply set to \( \phi(x) = x \). The trained model is saved on the system for the testing phase.

The SVM is applied to 10 different testing datasets. Each block in the testing data set is classified into either a target or non-target block based on their features and the trained model. Figure 5.4 shows the complete process for five different data sets. Figure 5.4 (b) shows \( V(x, y) \), which is the variance values of accumulated high frequency coefficients of a small window for each point. Figure 5.4 (c) depicts the variance values after thresholding in CFAR detection phase. Figure 5.4 (d) shows the refined targets after the application of the SVM classifier. Finally, Figure 5.4 (e) is the accumulation of the target blocks in consecutive frames, representing various target trajectories.
Figure 5-4: (a) Original Image, (b) Variance values of sliding window, (c) Results after CFAR Detection, (d) Results after SVM, (e) Target trajectories

An objective measure, namely Error-rate Per Frame (EPF), is defined to further evaluate the results. Let $P_{\text{real}}$ be the real position of the target, and $P_{\text{detected}}$ be the position detected by the algorithm; EPF is the average distance between the detected points and the real target points as follows,

$$EPF = \frac{\sum_{\text{for all DT}} \|P_{\text{real}} - P_{\text{detected}}\|}{DT},$$

(5.7)

where, DT represents the number of detected targets.
It is insufficient to justify the effectiveness of an algorithm simply by measuring the EPF, because there are other critical parameters associated with a tracking algorithm in addition to the detected target points. Critical parameters include the number of detected targets (DT), missed targets (MT), and false alarm detection (FA) in a frame. Therefore, a more comprehensive measure called Target to Clutter Ratio (TCR) is defined in order to compare different tracking algorithms as expressed below,

\[
TCR = \frac{DT}{DT + MT + FA},
\]

(5.8)

5.2.1 Comparison with other methods

Morphological based [3, 25], Wavelet based [26], Bilateral filter based [124], and Human Visual System (HVS) [19] based techniques are some of the recent methods which are tested and compared with our proposed algorithm. The first two methods focus on frequency information in the frames, i.e., they try to capture temporal or local high frequency changes. The focus of the last two methods is on the spatial information of the image.

EPF is calculated for all methods over 10 data sets. Both testing and training data sets are 30 frames per second and selected from the same public domain databases. To have a realistic comparison, there is no overlap between training and testing data sets. Different characteristics of testing data sets are shown in Table 5.2. For each method, the average value of the EPF over different data sets in a specific frame is shown in Figure 5.5. It can be seen that the proposed method is superior in comparison with other dim object tracking algorithms which solely focus on either the frequency or spatial domain information. In addition, Table 5.3 shows a comparison of TCR values for various algorithms over same 10 data set image sequences.
Table 5.2: Properties of ten testing image sequences

<table>
<thead>
<tr>
<th>Image Sequences</th>
<th>Description</th>
<th>Background</th>
<th>Target</th>
<th>Target Size</th>
<th>Image Size</th>
<th>Selected Frames</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1)</td>
<td>A person moving from one side of corridor to the other side, in perspective view</td>
<td>The noisy corridor image.</td>
<td>A person</td>
<td>7x7</td>
<td>400x320</td>
<td>30 (1 Sec.)</td>
</tr>
<tr>
<td>(2)</td>
<td>A bat flying in the sky</td>
<td>A static view of a house in a jungle</td>
<td>A Bat</td>
<td>8x8</td>
<td>500x300</td>
<td>30 (1 Sec.)</td>
</tr>
<tr>
<td>(3)</td>
<td>An airplane flying in the cloudy sky</td>
<td>The cloudy sky</td>
<td>An airplane</td>
<td>5x5</td>
<td>400x320</td>
<td>30 (1 Sec.)</td>
</tr>
<tr>
<td>(4)</td>
<td>An airplane flying in the sky. Big objects are added to the background manually</td>
<td>The cloudy sky, and the big objects which are added manually to the background</td>
<td>An airplane</td>
<td>5x5</td>
<td>400x320</td>
<td>30 (1 Sec.)</td>
</tr>
<tr>
<td>(5)</td>
<td>People moving around. Pedestrian tracking system</td>
<td>The upper view of a road and trees</td>
<td>The people</td>
<td>8x8</td>
<td>400x320</td>
<td>30 (1 Sec.)</td>
</tr>
<tr>
<td>(6)</td>
<td>A street view of moving cars and motorcycles</td>
<td>The road and some unknown objects on the road side</td>
<td>Moving cars and motorcycles</td>
<td>Different window size from 3x3 to 8x8</td>
<td>700x600</td>
<td>30 (1 Sec.)</td>
</tr>
<tr>
<td>(7)</td>
<td>A bunch of birds moving in the sky</td>
<td>The cloudy sky</td>
<td>The birds</td>
<td>6x6</td>
<td>400x320</td>
<td>30 (1 Sec.)</td>
</tr>
<tr>
<td>(8)</td>
<td>A bird moving in the cluttered noisy sky.</td>
<td>Manually added noise to sky</td>
<td>A bird</td>
<td>6x6</td>
<td>400x320</td>
<td>30 (1 Sec.)</td>
</tr>
<tr>
<td>(9)</td>
<td>Another view of pedestrian tracking</td>
<td>The upper view of a road</td>
<td>The people</td>
<td>10x10</td>
<td>400x320</td>
<td>30 (1 Sec.)</td>
</tr>
<tr>
<td>(10)</td>
<td>Four drones moving in the sky close to the earth</td>
<td>The cloudy sky, and the road side objects</td>
<td>Four drones</td>
<td>5x5</td>
<td>400x320</td>
<td>30 (1 Sec.)</td>
</tr>
</tbody>
</table>
Figure 5-5: Total average values of EPF: For Morphological method = 3.46, For Wavelet based method = 3.58, For Bilateral method = 2.80, For HVS method = 3.40, For the Proposed method = 2.10

Table 5.3: The proposed method achieves a 95% correct target detection rate

<table>
<thead>
<tr>
<th>Data set</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>Average TCR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Morphological Based</td>
<td>0.80</td>
<td>0.50</td>
<td>0.90</td>
<td>0.70</td>
<td>0.67</td>
<td>0.09</td>
<td>0.75</td>
<td>0.85</td>
<td>0.45</td>
<td>1.0</td>
<td>0.67</td>
</tr>
<tr>
<td>Wavelet Based</td>
<td>1.00</td>
<td>1.00</td>
<td>0.50</td>
<td>0.40</td>
<td>0.54</td>
<td>0.32</td>
<td>0.70</td>
<td>0.80</td>
<td>0.80</td>
<td>0.7</td>
<td>0.68</td>
</tr>
<tr>
<td>Bilateral Filter</td>
<td>1.00</td>
<td>0.90</td>
<td>0.83</td>
<td>0.74</td>
<td>0.85</td>
<td>0.90</td>
<td>0.80</td>
<td>0.90</td>
<td>1.00</td>
<td>1.0</td>
<td>0.89</td>
</tr>
<tr>
<td>HVS</td>
<td>0.60</td>
<td>0.70</td>
<td>0.75</td>
<td>0.85</td>
<td>0.78</td>
<td>1.00</td>
<td>0.74</td>
<td>1.00</td>
<td>0.52</td>
<td>0.8</td>
<td>0.78</td>
</tr>
<tr>
<td>Proposed Method</td>
<td>1.00</td>
<td>1.00</td>
<td>0.85</td>
<td>0.95</td>
<td>1.00</td>
<td>0.90</td>
<td>0.89</td>
<td>0.90</td>
<td>1.00</td>
<td>1.0</td>
<td>0.95</td>
</tr>
</tbody>
</table>
Table 5.3 shows a 95% real target detection rate for the proposed method (last column). It also indicates that the proposed method has the least number of missed targets or false alarm detection rates.

To assess the role of the SVM as a refinement step, a different version of the proposed method is also implemented without using the SVM. The EPF and TCR are also calculated for this new version over the same 10 data sets. The results show that without SVM refinement the proposed method achieved 3.25 for EPF, and 65% correct target detection rate on average. Comparing the results with the EPF and TCR values of the proposed method in Figure 5.5 and Table 5.3 shows that the superior performance of the proposed method is due to the combination of both the frequency and the spatial domain information.

The training phase of the proposed method requires a significant amount of computations over a large number of data sets, however, this is calculated only once. Table 5.4 illustrates the advantage of the proposed method in terms of the required time considering 30 frames (1 Second video) for 10 different data sets.
Table 5.4: Required time (in Second) for tracking 30 frames

<table>
<thead>
<tr>
<th>Data set</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td>Morphological Based</td>
<td>28.50</td>
<td>25.42</td>
<td>25.0</td>
<td>24.2</td>
<td>20.0</td>
<td>18.5</td>
<td>25.0</td>
<td>29.00</td>
<td>21.5</td>
<td>27.0</td>
<td>24.42</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Wavelet Based</td>
<td>29.50</td>
<td>27.50</td>
<td>23.5</td>
<td>24.0</td>
<td>19.0</td>
<td>17.1</td>
<td>24.5</td>
<td>27.45</td>
<td>22.0</td>
<td>27.0</td>
<td>24.16</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Bilateral Filter</td>
<td>30.00</td>
<td>32.50</td>
<td>25.0</td>
<td>27.8</td>
<td>22.5</td>
<td>22.5</td>
<td>28.9</td>
<td>33.00</td>
<td>25.0</td>
<td>30.1</td>
<td>27.73</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>HVS</td>
<td>27.50</td>
<td>24.50</td>
<td>26.8</td>
<td>25.5</td>
<td>21.5</td>
<td>16.1</td>
<td>24.0</td>
<td>26.00</td>
<td>21.0</td>
<td>29.0</td>
<td>24.20</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Proposed Method</td>
<td>26.00</td>
<td>23.00</td>
<td>21.0</td>
<td>23.8</td>
<td>21.0</td>
<td>17.0</td>
<td>22.0</td>
<td>24.30</td>
<td>20.0</td>
<td>24.0</td>
<td>22.21</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

5.3 Conclusion

This chapter presents an integration of the frequency domain and spatial domain information to effectively track dim objects in an image sequence. The first two steps of this method focus on finding the candidate target blocks and exploring frequency domain information. DT-CWT of each frame in an image sequence is obtained in the first step. CFAR detection module is then used to find the candidate target blocks from the variance of a sliding window in high frequency sub-bands. The list of possible candidate targets produced by CFAR is further refined by a SVM classifier. The SVM classifier uses the spatial domain information to determine whether the candidate image block is indeed a target. The combination of the CFAR detection mechanism followed by the SVM validation step provides a more accurate detection module.
Simulation results show that the proposed algorithm is capable of detecting and tracking targets in image sequences with low SNR values (less than 2dB). In comparison with related methods, the proposed method is more accurate. The use of both frequency and spatial domain data prevents false track detection when the target SNR values are low. In addition, the SVM classifier can work with various motion models with a 2D affine transformation and various kernel functions.
Chapter 6

Social-Spider Optimized Particle Filtering (SOPF) For Tracking of Targets with Discontinuous Measurement Data

In non-linear and non-Gaussian models, the PF exhibit better estimation performance over the other Bayesian filters [129]. However, in some cases most particles are concentrated at an improper location prematurely, and lead to a wrong average estimation [57]. Increasing the number of sample particles could be an unsuccessful attempt to resolve this problem, since PF often suffer from a heavy computational cost, especially when the dimension of the model is high [58]. To overcome this drawback, new evolutionary particle filter methods were proposed in recent researches. Sequential Importance Evolutionary (SIE) [59], PF with Immune Genetic Algorithm (IGA) [60], and Evolutionary Particle Filter (EPF) [57] are examples that combine standard particle filtering with genetic algorithms in different ways. In a similar attempt, Particle Swarm Optimization PF (PSOPF) [61] used the Particle Swarm Optimization (PSO) technique to improve functionality of the PF. However, PSO often easily falls into local optimum because the particles could quickly converge with the position of the best particles. Zhang et al. [62] proposed Sequential Particle Swarm Optimization and improved performance of PSO by using adaptive velocity to relocate the target in the case of an improper
premature convergence. In most cases, these evolutionary models are able to track the real target state accurately even when there are small changes in the system. However, none of these methods address the unpreventable discontinuity of observations in a system. In other words, in the case of discontinuous data or large changes in the observation, previous evolutionary models lose the target and cannot follow the track accurately.

To overcome the aforementioned shortcomings, Social-spider Optimized Particle Filter (SOPF) inspired by Social Spider Optimization (SSO), is suggested in this chapter. In SOPF, the spiders or particles are grouped into male and female agents to provide population diversity and searching capabilities. The female and dominant male spiders communicate with each other to find a better position near the real state. In the case of losing the target, the chance of female repulsion increases and particles distribute widely within the system hyperspace to find the new position of the target. The organization of this chapter is as follows. In Section 6.1, the particle filter and SSO are briefly reviewed. The newly developed SOPF is described in section 6.2, and in section 6.3, the experimental results demonstrating the efficiency of the SOPF and its advantages over other evolutionary and simple PF algorithms, are presented. Finally, the last section presents the conclusion of the proposed method.

6.1 Fundamental Background

6.1.1 Generic Particle Filter (PF)

The particle filter is a special Bayes filter which uses Monte–Carlo algorithms to provide a numerical solution for estimation problems. It uses particles as samples to approximate the posterior distribution $p(X_{1:t} | Z_{1:t})$ and the observation distribution
In a $d$-dimensional system at time stamp $t$, a state vector of the target denoted by $X_t \in R^d$ with its history $X_{1:t} = \{X_1, ..., X_t\}$, and the observation vector denoted by $Z_t \in R^d$ with its history $Z_{1:t} = \{Z_1, ..., Z_t\}$ are represented by,

$$X_t = f_t(X_{t-1}) + M_t$$

$$Z_t = h_t(X_t) + N_t$$

where $f_t, h_t : R^d \times R^d \rightarrow R^d$ are system and observation functions, respectively. Also, $M_t, N_t \in R^d$ are non-Gaussian noise sequence independent of past and current states [55].

The ultimate goal of particle filter is estimating $X_t$, the state position of all samples at each time stamp $t$, based on all of the previous observations $Z_{1:t}$, i.e., to construct the PDF $p(X_t|Z_{1:t})$. Knowing the initial value of $p(X_0|Z_1) = p(X_0)$, the PDF may be obtained recursively in two stages: prediction and update phase [55].

In the prediction stage, the prior density of the state at time stamp $t$ given all the observations to this point is obtained through the Chapman–Kolmogorov equation,

$$p(X_t|Z_{1:t-1}) = \int p(X_t|X_{t-1})p(X_{t-1}|Z_{1:t-1})dX_{t-1}$$

where the probability model of the state evolution, $p(X_t|X_{t-1})$, is a Markov model determined by Equation (6.1). Following this step, the observation $Z_t$ becomes available in time stamp $t$, and via Bayes’ rule the posterior density is updated,

$$p(X_t, Z_{1:t}) = \frac{p(Z_t|X_t)p(X_t|Z_{1:t-1})}{p(Z_t|Z_{1:t-1})}$$

where the normalizing constant is given by,

$$p(Z_t|Z_{1:t-1}) = \int p(Z_t|X_t)p(X_t|Z_{1:t-1})$$

Note, the recursive propagation of the posterior density is only a conceptual solution, and it cannot be determined analytically. Implementations of PF apply a Monte
Carlo simulation to estimate the required posterior density function using a set of random samples with associated weights. Large number of samples represents a posterior PDF close to the actual function in which the model approaches the optimal Bayesian estimate.

The basic particle filtering consists of three steps: sampling, importance weighting, and resampling. In the sampling step, particles are sampled according to the posterior probability $p(X_t|Z_{1:t})$. Suppose $X_t = \{X^i_t: i = 1..N\}$ represents $N$ particles in time stamp $t$ with corresponding weights $W_t = \{w^i_t: i = 1..N\}$. The weights are normalized such that $\sum_{i=0}^{N} w^i_t = 1$. Therefore, the posterior density is approximated by,

$$p(X_t|Z_{1:t}) = \sum_{i=1}^{N} w^i_t \delta(X_t - X^i_t) \tag{6.6}$$

where $\delta(.)$ is Dirac Delta function, and $w^i_t$ the weight of the $i$-th sample $X^i_t$ which is found through the following importance weighting function in weighting step.

$$w^i_t \propto w^i_{t-1} \frac{p(Z_t|X^i_t)p(X^i_t|X^i_{t-1})}{q(X^i_t|X^i_{t-1}, Z_t)} \tag{6.7}$$

The function $q(X^i_t|X^i_{t-1}, Z_t)$ is known as the proposal function, and its selection is critical in particle filtering.

Resampling is the last step of PF which involves deleting and multiplying low and high weight particles, respectively. It saves the computational resources used for unlikely particles and improves the estimation performance. While, some PF methods do resampling in every time stamp, some others will perform it when the degeneracy of the particles is severe.

### 6.1.2 Social Spider Optimization (SSO)

The SSO algorithm proposed by Cuvas et al. [72] is based on the simulation of biological cooperative behavior of social-spiders. The search particle (spiders) population
is divided into two different types: males and females which are conducted by a set of different evolutionary operators and mimic different cooperative behaviors. The complete spider population \( S \) composed of \( N \) agent divided into two sub-groups of male \( M \) and female \( F \). Each agent carries the position and weight of a state in the problem hyperspace. To emulate the highly female biased population characteristic of the social spiders, the number of female spiders \( N_f \) is randomly set in the range of 65-90\% of the entire population \( N \) as follows:

\[
S = \{ s^1 = f^1, s^2 = f^2, ..., s^{N_f} = f^{N_f}, s^{N_f+1} = m^1, ..., s^N = m^{N_m} \}
\]

where \( N_m \) is the number of male population, \( f \) and \( m \) represent the female and male, respectively.

Each spider (particle) has a weight \( w_i \) which represents its merit as a solution. The weight of spider \( i \) depends on its fitness value \( fit_i \) and can be detailed as following:

\[
w_i = \frac{fit_i - \text{worst}}{\text{best} - \text{worst}}
\]

where \( \text{worst} \) and \( \text{best} \) are the fitness of the spiders representing the worst and best solutions, respectively.

The colony members communicate with each other through different kinds of vibrations (Vib) in the web. The vibration between spider \( i \) and \( j \) is modeled as, \( V\text{ib}^{i,j} = w_i e^{-d_{i,j}^2} \), where \( d_{i,j} \) is the Euclidean distance between spiders \( i \) and \( j \). There are three types of vibrations perceived by spider \( i \) [73]:

- \( V\text{ib}c^i \) from spider \( c \) who is the nearest spider to \( i \) and possesses a weight higher than \( w^i \).
- \( V\text{ib}b^i \) from spider \( b \) who has the highest weight of the population.
- \( V\text{ib}f^i \) from spider \( f \) who is the nearest female spider to \( i \).
Depending on spider gender, social spiders perform cooperative interaction over other colony members. Female movements consist of attraction or repulsion to other members by the perceived vibrations. In iteration $t$, a random number $rand$ is generated and on that basis a female spider $f_t^i$ then attracts or repulses other spiders. Using the following equations:

$$f_t^i = \begin{cases} 
    f_{t-1}^i + aVibc^i(s^c - f_{t-1}^i) + \beta Vibb^i(s^b - f_{t-1}^i) + \delta \left(\gamma - \frac{1}{2}\right), & \text{if } rand \geq Th \\
    f_{t-1}^i - aVibc^i(s^c - f_{t-1}^i) - \beta Vibb^i(s^b - f_{t-1}^i) + \delta \left(\gamma - \frac{1}{2}\right), & \text{if } rand < Th
\end{cases} \quad (71)$$

Note, $\alpha, \beta, \delta, \text{ and } \gamma$ are four independent random numbers in the range $[0,1]$, and $Th$ is a threshold value.

According to the biological characteristic, the male social-spider population is divided into two types: dominant and non-dominant. Those male spiders that have above the average weight are called dominants. Dominant males are attracted to the nearest female spider. However, non-dominants concentrate in the center of the male population as a strategy to take advantage of resources that are wasted by dominant males. Male spider movement at iteration $t$ can be simulated by the following equations:

$$m_t^i = \begin{cases} 
    m_{t-1}^i + \eta Vibf^i(s^f - m_{t-1}^i) + \delta \left(\gamma - \frac{1}{2}\right), & \text{dominant male} \\
    m_{t-1}^i + \eta \left(\frac{\sum_{j=1}^{N_m} m_{t-1}^j w^{N_f+j}}{\sum_{j=1}^{N_m} w^{N_f+j}}\right), & \text{nondominant male}
\end{cases} \quad (72)$$

where $\eta$ is an independent random number in the range $[0,1]$, $s^f$ is the closest female to the dominant male member $i$, and $\left(\sum_{j=1}^{N_m} m_{t-1}^j w^{N_f+j}/\sum_{j=1}^{N_m} w^{N_f+j}\right)$ represents the weighted mean of the male population.

In the original implementation of SSO, there is a mating step to enhance the total fitness of the population by generating new offspring and replacing worst agents by the
new ones. However, in the proposed SOPF, the mating step is not necessary, because a similar process is performed by resampling step of the particle filter.

6.2 Social-spider Optimized Particle Filter (SOPF)

Following the design of the basic particle filters, the proposed PF in this chapter consists of three steps: Sampling, Importance Weighting, and Resampling. Sample particles are determined based on the prior density of the state. Threshold $T h$ and independent random variable $\alpha$ and $\beta$ as described in the previous section are selected based on the observation of the current time stamp and the SSO optimized position of each state. In the second step, weights of the particles are updated as discussed in the following importance weighting section. Finally, the particles with lower weights are substituted by the multiplication of particles with higher weights.

6.2.1 Sampling

In theory, PF propagates a set of particles that have been randomly sampled from the posterior density $p(X_t|Z_{1:t})$. Practically, basic particle filters start with sampling $N$ particles $\{X_t^i\}_{i=1}^N$ according to the observations $Z_{1:t} = \{Z_t^i\}_{i=1}^N$ and the previous position of the samples $\{X_{t-1}^i\}_{i=1}^N$. Unlike the basic particle filters, the proposed SOPF not only selects samples from the state transition but also undertake the SSO approach to avoid sample impoverishment. To complete the sampling step of SOPF, the following three tasks should be considered.

Task 1) Sampling: Since the target can be located anywhere in the problem hyperspace, the most difficult part of the sampling scheme is the initialization. The proposed method
initializes the position of particles by sampling \( \{X^i_t\}_{i=1,N} \) from the observation likelihood, i.e.,

\[
\text{Draw} \{X^i_t\}_{i=1,N} \sim p(X_t|Z_{t-1})
\]  

(73)

**Task 2)** Selection of SSO parameters: SSO is an iterative process and the initial particles’ positions will change toward the target position in each iteration. If there is a discontinuity in the observation data or the target is lost, all particles should oscillate in the problem hyperspace to relocate the target when it reappears. The proposed method controls the oscillations of the particles through SSO parameters. The value of the threshold \( Th \) in Equation (71) controls the attraction and the repulsion of the female particles. This value is obtained using the following equation:

\[
Th = \left( \frac{\sum_{i=1}^{N} w^i_t}{w^\text{best}_t + \sum_{i=1}^{N} w^i_t} \right)
\]  

(74)

When weight of the best particle \( w^\text{best}_t \) is significantly higher than others, the leader is considered to be close to the target. Therefore, selecting lower value of the threshold attracts more female particles into the position of the best particles (female movement Equation (71)), and the whole population converges. On the other hand, when the difference of the weight of the best particle with others is insignificant, it shows the target is lost. In this case, the value of \( Th \) is increased and more female particles repulse from the position of the best particle to provide scattering and redistribution in the problem hyperspace.

The parameters \( \alpha \) and \( \beta \) control the amplitude and the direction of the female movement toward the nearest and best particle position, respectively. Obtaining these parameters using the following equations assures the female particle movement toward the optimal position.
The proposed method redistributes the female particles when the target is missed due to a significant change or data discontinuity. Note that the dominant males follow female particles and non-dominant male particles try to concentrate in the center of male population.

**Task 3) SSO iterations:** The last task of the sampling step is the actual motion of the particles through SSO iterations. In each iteration, all of the female and male particles’ positions are updated using Eq. (69) and (70). Meanwhile, the parameters $\alpha$ and $\beta$ are also updated according to the change in the weight of the best particle, $w_t^{\text{best}}$. Experimental results show that a small number of iterations are sufficient to find an accurate average estimation of the original track.

### 6.2.2 Importance Weighting

Similar to the generic PF, the second step of SOPF is importance weighting. In this step, the weights of the entire sampled particles are derived in a way that the weighted particles asymptotically approximate the given target distribution. Equation (6.7) relates the importance weighting to the proposal density function denoted by $q(X_t^i|X_{t-1}^i, Z_t)$. However, there is no known proposal density function designed for tracking dim objects in a noisy image sequence. Moreover, based on the weighting importance, generic PF resampling is dependent on large weight particles. Some large weight particles are misleading and leads to an incorrect particles resampling, especially in noisy scenes [130].
The proposed SOPF model moves particles toward optimum position in the sampling step, therefore, to calculate the weight of a particle, a fitness value is required using a fitness function $fit(Z^i_t)$ based on the observation of the $i$-th particle at time stamp $t$. For each particle $X^i_t$, the weight of the $i$-th particle $w^i_t$ is then defined using the following equation:

$$w^i_t = \frac{fit(Z^i_t) - worst_t}{best_t - worst_t}$$  \hspace{1cm} (77)

where $worst_t$ and $best_t$ are defined as below:

$$best_t = \min_{i} \{fit(Z^i_t)\} \text{ and } worst_t = \max_{i} \{fit(Z^i_t)\}$$  \hspace{1cm} (78)

### 6.2.3 Resampling

The last step in a PF model is improving the estimation performance through eliminating samples with low importance weights and multiplying samples with high importance weights. Through SSO iterations, the weights of sampled particles will improve. Therefore, the proposed SOPF applies resampling only in the case of severe diversity. In this way, SOPF can save computational resources in some iterations.

Assume $N_{eff}$, as defined in Equation (80), and $N_{Thr}$, a user defined threshold, to distinguish severe diversity. Resampling is performed only in the case of $N_{eff} < N_{Thr}$. Note, Equation (79) normalizes the true weight of the particles before obtaining $N_{eff}$.

$$WN_t^i = \frac{w^i_t}{\sum_{j=1}^{N} w^j_t}$$  \hspace{1cm} (79)

$$N_{eff} = \frac{1}{\sum_{i=1}^{N} (WN_t^i)^2}$$  \hspace{1cm} (80)

In summary, the proposed SOPF method can be described in the following Algorithm 6.1:
Algorithm 6.1: The proposed SOPF method

Step 1: Sample initial particles $X_0 = \{X_0^i\}_{i=1..N}$ and obtain their related weights $W_0$ according to the fitness values.

Step 2: Update the SSO parameters $T_h$, $\alpha$, and $\beta$ from Eq. (13), (14), (15).

Step 3: Move the particles within the problem hyperspace through SSO iterations using Eq. (10), (11), and find the optimum particles position.

Step 4: Update the weight of each particle applying Eq. (16).

Step 5: Resampling the particles to increase their estimation performance, if needed.

Step 6: If the certain amount of iterations or termination criteria are not achieved, go to step 2.

Step 7: Go to next time stamp $t = t + 1$, calculate new observation likelihood $p(X_t|Z_{t-1})$ based on Eq. (6), and draw new particle samples $X_t = \{X_t^i\}_{i=1..N}$. Go to step 2.

6.3 Experimental Results

To evaluate the performance of the proposed model (SOPF) in comparison with some other evolutionary particle filters, four different PF techniques are chosen. The Unscented Particle Filter (UPF) [131], the Evolutionary Particle Filter (EPF) [57], the Immune Genetic Algorithm based Particle Filter (IGA-PF) [60] and the Particle Swarm Optimization based Particle Filter (PSO-PF) [130] are selected based on their superior performances in their categories. The proposed SOPF and other models have been implemented in MATLAB. To investigate the performance of the proposed method relative to others, three different examples are described. The first example is a one dimensional non-linear model. The second and third examples are related to a non-maneuvering and a maneuvering target, respectively. The comparison between the algorithms is measured based on the Mean Square Error (MSE) of the obtained value relative to the real position.

Example 6.1: One dimensional non-linear model
Consider the following nonlinear Equation (81) and (82) describing the system and the observation equations.

\[
X_t = \frac{x_{t-1}}{2} + \frac{25x_{t-1}}{1+x_{t-2}^2} + 8\cos(1.2(t - 1)) + M_t \tag{81}
\]

\[
Z_t = \frac{x_t^2}{20} + N_t \tag{82}
\]

where \(M_t\) and \(N_t\) are zero-mean Gaussian white noise with variances 10 and 1, respectively. The model is popular in econometrics and is widely used to evaluate the performance of the PF methods in many researches. The initial value of the system is set to 0.1 \((X_t = 0.1)\).

Although in GA based methods, it is claimed that the cross over and the mutation rates are not considered to be highly influential factors for the overall performance of the model [57], both cross over and mutation rates are set to an optimal value of 0.20. Also, the number of iterations is set to 10 for evolutionary based methods. The sample particle size is set to 100 for all methods.

Figure 6.1 shows the performance of the PF methods over 75 time stamps. As seen in Figure 6.1, there is a discontinuity in the flow of data between time stamp 20 and 30 with no observations in between. All of the methods except UPF followed the discontinuity to some extent. However, when the object appeared again with a noticeable discontinuity in time stamp 30, the proposed SOPF is the only method that could follow the track. Other evolutionary methods were able to get on the right track after a few time stamps. In the case of a significant discontinuity as evidenced in time stamp 50, the UPF performed very poor while among other evolutionary methods, EPF had better performance. As is evidenced in Figure 6.1, the proposed SOPF was able to determine the closest estimation to the original track.
Random noise in the system and the observation function causes some changes between different runs of the algorithms. For a fair assessment, 10 independent runs are made and the average of the MSE over the 10 runs is computed. The MSE values are shown in Table 6.1, which indicates the superiority of the proposed SOPF when the target has significant increase or there is discontinuity in the flow of data. The first column of this table is the MSE before the data discontinuity. All evolutionary models have good performance in this period. The second column shows the MSE before the first discontinuity in time stamp 30, and the third column shows the average MSE before the second significant discontinuity in time stamp 50. In addition, Tables 6.2 and 6.3 show the performances of our proposed method for different sample sizes and number of iterations, respectively.

![Figure 6-1: Proposed SOPF obtains the more accurate estimates even in the case of significant increase](image)

<table>
<thead>
<tr>
<th>State Value</th>
<th>Original Track</th>
<th>UPF</th>
<th>EPF</th>
<th>IGA-PF</th>
<th>PSO-PF</th>
<th>Proposed SOPF</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time Stamp</td>
<td>0 5 10 15 20 25 30 35 40 45 50 55 60 65 70 75</td>
<td>0 5 10 15 20 25 30 35 40 45 50 55 60 65 70 75</td>
<td>0 5 10 15 20 25 30 35 40 45 50 55 60 65 70 75</td>
<td>0 5 10 15 20 25 30 35 40 45 50 55 60 65 70 75</td>
<td>0 5 10 15 20 25 30 35 40 45 50 55 60 65 70 75</td>
<td>0 5 10 15 20 25 30 35 40 45 50 55 60 65 70 75</td>
</tr>
</tbody>
</table>

**Table 6.1: Average MSE values for different time stamps**
<table>
<thead>
<tr>
<th>Discontinuity (Time stamp 20)</th>
<th>UPF</th>
<th>EPF</th>
<th>IGA-PF</th>
<th>PSO-PF</th>
<th>Proposed SOPF</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.69907</td>
<td>10.52596</td>
<td>23.28581</td>
<td>52.24</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.752273</td>
<td>2.79269</td>
<td>6.98432</td>
<td>20.3797</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.880031</td>
<td>4.35069</td>
<td>11.81742</td>
<td>35.17</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.600545</td>
<td>4.08543</td>
<td>8.31366</td>
<td>32.37</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.311858</td>
<td>0.324655</td>
<td>1.089428</td>
<td>2.455438</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 6.2 Performance of the SOPF for different population sizes

<table>
<thead>
<tr>
<th>Population Size</th>
<th>MSE before data discontinuity (Time stamp 20)</th>
<th>MSE before time stamp 30</th>
<th>MSE before time stamp 50</th>
<th>Average MSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>0.311858</td>
<td>0.324655</td>
<td>1.089428</td>
<td>2.455438</td>
</tr>
<tr>
<td>200</td>
<td>0.317141</td>
<td>0.335647</td>
<td>0.987874</td>
<td>2.145874</td>
</tr>
<tr>
<td>500</td>
<td>0.310458</td>
<td>0.305789</td>
<td>0.874581</td>
<td>2.054841</td>
</tr>
<tr>
<td>1000</td>
<td>0.310587</td>
<td>0.305471</td>
<td>0.874154</td>
<td>2.012554</td>
</tr>
</tbody>
</table>

Table 6.3 Performance of the SOPF for different number of iterations

<table>
<thead>
<tr>
<th>Number of iterations</th>
<th>MSE before data discontinuity (Time stamp 20)</th>
<th>MSE before time stamp 30</th>
<th>MSE before time stamp 50</th>
<th>Average MSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>0.412245</td>
<td>0.542551</td>
<td>1.458421</td>
<td>2.874210</td>
</tr>
<tr>
<td>10</td>
<td>0.311858</td>
<td>0.324655</td>
<td>1.089428</td>
<td>2.455438</td>
</tr>
<tr>
<td>20</td>
<td>0.306598</td>
<td>0.298541</td>
<td>1.003353</td>
<td>2.410013</td>
</tr>
<tr>
<td>30</td>
<td>0.264585</td>
<td>0.286965</td>
<td>0.902288</td>
<td>2.305380</td>
</tr>
<tr>
<td>100</td>
<td>0.262525</td>
<td>0.281152</td>
<td>0.900142</td>
<td>2.100118</td>
</tr>
</tbody>
</table>

It can be seen from the above tables that SOPF provides better performance regardless of the changes in the values of the SSO parameters such as population size and number of iterations.
Example 6.2: Non-maneuvering target

In this example the target moves in a 2-dimensional plane according to the following second order model,

\[ X_t = AX_{t-1} + BM_t \]  \hspace{1cm} (83)

where \( X_t = [x_t, \dot{x}_t, y_t, \dot{y}_t]^T \) is the state variable, and \( M_t = [Mx_t, My_t]^T \) represents the noise. \((x_t, y_t)\) is the location of the target in the plane, and \((\dot{x}_t, \dot{y}_t)\) denotes the target velocity in two directions. The state transition matrix \( A \), and the noise matrix \( B \) are defined as follows:

\[ A = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad B = \begin{bmatrix} 0.5 & 0 \\ 1 & 0 \\ 0 & 0.5 \\ 0 & 1 \end{bmatrix} \]  \hspace{1cm} (84)

The non-maneuvering target moves straight forward while the relation of two directional velocities \( \dot{x}_t \) and \( \dot{y}_t \) are constant. The object starts moving from the plane origin while all the evolutionary search parameters remain the same as example 6.1.

A fixed observation sensor at the origin of the plane collects the following noisy observations \( Z_t \).

\[ Z_t = \arctan \left( \frac{y_t}{x_t} \right) + N_t \]  \hspace{1cm} (85)

The original track and the estimated tracks found by the different PF methods are shown in Figure 6.2 and 6.3. The target in this example is non-maneuvering, and, there is no change or discontinuity. For this reason, almost all PF methods provide an acceptable performance. As seen in Figure 6.2 and 6.3, the performances of SOPF and PSO-PF are the same. Therefore, it can be concluded that the evolutionary search is not necessary in the case of a non-maneuvering target. However, it is obvious that this example is really rare in the real world.
Figure 6-2: Estimated tracks of various PF methods ($x_t$)

Figure 6-3: Estimated tracks of various PF methods ($y_t$)
The average values of MSE over 10 runs for both \( x_t \) and \( y_t \) positions are compared in Table 6.4. It shows that even in non-maneuvering cases, PSO based methods have a better performance.

### Table 6.4 Average MSE values for non-Maneuvering Target in 10 different runs

<table>
<thead>
<tr>
<th>Method</th>
<th>Average MSE (X)</th>
<th>Average MSE (Y)</th>
</tr>
</thead>
<tbody>
<tr>
<td>UPF</td>
<td>14.24617</td>
<td>14.95599</td>
</tr>
<tr>
<td>EPF</td>
<td>6.810335</td>
<td>7.542674</td>
</tr>
<tr>
<td>IGA-PF</td>
<td>12.01266</td>
<td>9.079742</td>
</tr>
<tr>
<td>PSO-PF</td>
<td>2.372854</td>
<td>2.372854</td>
</tr>
<tr>
<td>Proposed SOPF</td>
<td>2.372854</td>
<td>2.372854</td>
</tr>
</tbody>
</table>

**Example 6.3: Maneuvering target**

This example presents a target that moves via Equations (83) and (84) with the same observation as Equation (85). The only difference is the possibility of a circular motion through matrix \( A \). The angle \( \varphi_t \) is positive for anti-clockwise and negative for clockwise rotation. For the circular motion, the matrix \( A \) is represented in the following form,

\[
A = \begin{bmatrix}
1 & \frac{\sin(\varphi_t)}{\varphi_t} & 0 & -\frac{(1-\cos(\varphi_t))}{\varphi_t} \\
0 & \frac{\cos(\varphi_t)}{\varphi_t} & 0 & \frac{-\sin(\varphi_t)}{\varphi_t} \\
0 & \frac{1-(\cos(\varphi_t))}{\varphi_t} & 1 & \frac{\sin(\varphi_t)}{\varphi_t} \\
0 & \frac{\varphi_t}{\sin(\varphi_t)} & 0 & \frac{\cos(\varphi_t)}{\varphi_t}
\end{bmatrix}
\]  

(86)

The object starts moving from the plane origin and the entire configurations remain the same as example 6.1. Figure 6.4 and 6.5 show the \( x_t \) and \( y_t \) positions of the target over 100 time stamps. A data discontinuity occurs between time stamp 20 and 30. In
addition, there have been two rotated changes for the target trajectory at time stamps 60 and 80 as indicated in the original track.

Figure 6-4: Estimated tracks of various PF methods for a maneuvering target ($x_t$)

Figure 6-5: Estimated tracks of various PF methods for a maneuvering target ($y_t$)
Note that the UPF will not be able to follow the target after a discontinuity in the data flow or a rotation as it is based on the motion model with no abrupt changes. Among other evolutionary methods, the EPF has a better performance, however, there is a number of unnecessary oscillations after the second rotation. These oscillations could be as a result of the crossover and mutation operations used in this method. Note that the deteriorations in the performance of the PSO-PF are due to the presence of changes in the data flow. The track from the proposed SOPF method is the closest to the original target even after the discontinuity in the data flow or rotations.

Table 6.5 shows that the performance of the proposed SOPF is much better than other basic or evolutionary PF methods in tracking maneuvering targets.

**Table 6.5 Average MSE values for Maneuvering Target in 10 different runs**

<table>
<thead>
<tr>
<th></th>
<th>Average MSE (X)</th>
<th>Average MSE (Y)</th>
</tr>
</thead>
<tbody>
<tr>
<td>UPF</td>
<td>33.65476</td>
<td>54.34825424</td>
</tr>
<tr>
<td>EPF</td>
<td>11.22922</td>
<td>7.035331894</td>
</tr>
<tr>
<td>IGA-PF</td>
<td>9.342872</td>
<td>17.27821555</td>
</tr>
<tr>
<td>PSO-PF</td>
<td>12.8566</td>
<td>18.13619735</td>
</tr>
<tr>
<td>Proposed SOPF</td>
<td>3.997904451</td>
<td>4.865612798</td>
</tr>
</tbody>
</table>

6.4 Conclusion

Particle filter is an extension of the Kalman filter to deal with non-Gaussian non-linear systems. However, generic PF suffer from sample impoverishment. Researchers apply evolutionary algorithms to avoid the sample impoverishment in PF. However, our experimental results show that significant changes or discontinuity in the data flow or observations could result in sample impoverishment even in evolutionary based methods. The proposed SOPF method in this chapter applies a social-spider optimization to overcome sample premature impoverishment in critical situations. The population of the
proposed SOPF is divided into two groups, male and female, to provide diversity. When the target is lost due to data discontinuity or a significant change, the overall weights of the particles decrease and the algorithm forces its particles to redistribute in order to locate the target again. When the target is found, the female particles are attracted to the best particle with the best fitness value. This results in the whole population convergence to a unique solution. According to experimental results, the proposed SOPF successfully estimates a system with significant changes and data discontinuities which occur in the real world. The proposed SOPF could also be applied in the low SNR environments for mobile robot localization or dim object tracking problems.
Chapter 7

Multi-Sensor Swarm Intelligence Particle Filter (SIPF) for Tracking of Dim Objects with Discontinuous Measurement Data

Generally speaking, more accurate information is revealed by combining data from multiple sensors rather than using a single sensor [75-78]; this provides either enriched accuracy from identical sensors or the same performance from low resolution or cheaper sensors [79]. A large amount of work on multi-camera surveillance has been performed from different perspectives. Collins et al. [80] found 3D position of the target using calibrated cameras and environmental model. Cai and Aggarwal [81] used geometric and intensity features to track objects in multiple views. Kim and Davis [82] applied particle filter to track people in multiple views. They used background subtraction and classification of the silhouette of the individuals. Davey et al. [83] applied particle filter for a multi-sensor track-before-detect algorithm. In each frame, they used the information of one sensor to find the fitness of particles. Hence, combining the data from all sensors enhanced the accuracy of their model. Recently, Munoz-Salinas et al. [84] used an evidential particle filter using multi-camera as a novel solution to the people tracking problem. In their model, Multiple Evidential Particle Filter (MEPF) was used to track the possible states of the dynamic system. Each sensor was asked for a degree of evidence
and reliability. In a final data fusion step, data collected from all the sensors was fused to provide the best location of people. Most of these particle filter based models perform particle filtering in the central sensor. Usually, the central sensor requests the weights and the reliability of each particle from the other sensors.

In 2009, Liu et al. [132] proposed a distributed particle filter (DPF) to decrease the communication and energy costs in a resource limited network such as sensor network. DPF uses support vector machine (SVM) classifier to choose appropriate data and transmits it over a sensor network. Completely distributed particle filter (CDPF) was proposed by Jiang and Ravindran [133] in 2011. They improved distributed particle filter and minimize the communication cost over sensor networks. Recently, Read et al. [134] proposed a mathematically distributed particle filter in wireless sensor networks. Their method makes all the sensor observations available to every processing node that could have a huge load of communication data over wireless network. A collaborative particle filter structure for people tracking was proposed by Du and Piater [135]. In their method, individual particle filters track the target on individual cameras’ view and data fusion is performed on a shared workplace plane. The fusion results on the ground plane are incorporated by each camera as a boosted proposal function for the next frame. Ni et al. [136] enhanced this people tracking method by generating a more representative set of training examples to update a discriminative classifier (object and background) using an adjusted starting point of the previous frame. However, these distributed particle filter approaches do not receive the benefit of the evolutionary search and are not suitable for tracking small dim objects with no specific shape in noisy image sequences.
There are three issues in multi-sensor particle filter approaches that weaken these methods for dim object tracking applications. 1) Finding a precise probabilistic model that describes the process is a limitation to applying Bayesian filters. In many realistic systems, it may be difficult to obtain complete knowledge of the problem to define prior probability likelihood. 2) An individual set of distributed particles for each sensor is required due to different types of noise and limitations in each sensor. 3) In some cases of small object tracking, most particles are concentrated at a wrong point prematurely, which leads the average estimation to a wrong state.

The above mentioned MEPF method [84] solves the first issue by using evidential particle filter. However, MEPF uses a central particle filter, with no provision to handle the second and third issues. A recent technique developed by Zhu et al. [137] uses Expectation Maximization (EM) algorithm and KF for a joint data association in a distributed particle filter. Their distributed particle filter model considers various kinds of noise and limitation in different sensors. Nevertheless, their model cannot be used to track dim small objects in the case of large jumps or discontinuity in measurement data due to the particles’ premature concentration. A robust vision-based tracking system using multiple features was proposed by Toyama and Hager [138] which provides more information about the object and increases the robustness of the algorithm to occlusions and clutters. In addition, Loza et al. [139] presented a structural similarity-based particle filter for tracking objects in a video sequence which is robust to illumination and contrast changes. In both of these models, different layers cooperate to perform a rapid search for the target and relocate the target when it is lost. However, these robust tracking algorithms fail to relocate the target if it is lost and has disappeared from all possible
views. Moreover, high computational cost decreases their efficiency, especially when several objects should be tracked over a long period of time [140].

To overcome the aforementioned shortcomings, a Multi-Sensor Swarm Intelligence Particle Filter (SIPF) is presented in this chapter to track small dim objects in a noisy environment. The proposed method consists of two hierarchical levels. In the lower level, all sensors use SIPF to find the best position of the targets individually based on the provided prior information, and report it to a dynamically selected center at the upper level. The central sensor at the upper level finds the best of the reported positions for each target and broadcasts it to all the sensors as the actual position of the target. This will be used as the prior probability likelihood in the next frame.

The organization of the remainder of this chapter is as follows: Section 7.1 describes the proposed distributed architecture method along with the hierarchical structure, SIPF, and a process to detect the center sensor dynamically. In section 7.2, the performance of the proposed method and its simulation results are discussed through examples. Moreover, a comparison of the proposed scheme with other multi-sensor models on dim object tracking is presented. The conclusion and summary of the proposed method and its advantages are provided in the final section.

7.1 The Proposed Multi-Sensor SIPF Method

A novel hierarchical method is proposed to track small dim objects in a cluttered low resolution image sequence. The proposed method uses multi-heterogeneous sensors to follow the targets more accurately in the case of large jumps or discontinuities in the measurement data. Since maintaining a complete calibration of a large network of sensors is a significant maintenance task [141], this chapter assumes a calibrated network of
sensors with known sensor bias parameters similar to the most recent models [78, 84]. Each sensor in the proposed method uses SIPF and prior information to locate the best position of the targets in the current frame. The best locations of the targets are then reported to the central sensor at the upper level. In the upper-level, the central sensor selects the best of the reported positions for each target and broadcasts this information to every sensor to be used as the prior information. The central sensor is selected dynamically to cover a wider field of view as described later.

The proposed multi-sensor multi-target tracking is modeled by the following Equations (87) and (88):

\[ x_{k,t} = f_k(x_{k,t-1}) + u_{k,t}, \]  
\[ z_{k,t}^s = h^s(x_{k,t}) + v_{k,t}^s, \]

where \( x_{k,t} \) represents the actual position of target \( k \in [1, m] \) at time frame \( t \), and \( m \) is the maximum number of targets. \( z_{k,t}^s \) is the sensor \( s \) observation of target \( k \) at time frame \( t \).

\( f_k \) is the unknown state transition function of target \( k \) (In real applications, state transition is not a known function). \( h^s \) is the observation function of sensor \( s \). Also, \( u_{k,t} \) and \( v_{k,t}^s \) are non-Gaussian noise sequence independent of the past and current states.

The proposed method proceeds with the execution of the following three steps in each frame of the image sequence:

7.1.1 Step1: Finding the best positions of targets in each sensor

In the lower-level, the proposed SIPF is performed by each sensor individually. It is assumed that all the sensors are aware of the previous frame’s actual target positions \( X_{t-1} \) defined as:

\[ X_{t-1} = [x_{1,t-1}, x_{2,t-1}, ..., x_{m,t-1}]^T \]  

(89)
The input to sensor \( s \) in the lower-level is \( X_{t-1} \), and the output would be the best positions of all targets in the current frame found by sensor \( s \), represented by \( X^s_t \).

\[
X^s_t = [x^s_{1,t}, x^s_{2,t}, ..., x^s_{m,t}]^T
\]  

Note, \( m \) is the maximum number of targets as mentioned earlier.

To find the best position of the target in each sensor, the SIPF procedure which consists of the following three tasks is implemented:

**Task 1) Sampling:** Generic particle filter algorithm samples particles according to the posterior probability \( p(x_{k,t}|Z_{k,1:t}) \), where \( x_{k,t} \) represents the state of \( k\)-th target at time stamp \( t \), and \( Z_{k,1:t} \) is the observations of \( k\)-th particle from the start to the current time stamp.

In SIPF, sampling is simply based on the previous position of the targets. Initially, \( m \) target positions of the previous frame are obtained from \( X_{t-1} \), and \( n \) particles are distributed around each target with \( w^i_{k,t} \) representing the weight of the \( i\)-th particle of target \( k \) at time stamp \( t \).

Therefore, sensor \( s \) maintains a \( n \times m \) matrix of the states of the particles represented by \( P^s_t \). Each column of the matrix represents the position and fitness value of \( n \) particles around one target (\( i \in [1,n] \)). Also, each row of the matrix represents a set of \( m \) random particles which are positioned around \( m \) different targets (\( k \in [1,m] \)).

**Task 2) Weighting:** In generic particle filter, \( w^i_{k,t} \), the weight of the \( i\)-th particle of target \( k \) at time stamp \( t \), is found through the following importance weighting function.

\[
w^i_{k,t} \propto w^i_{k,t-1} \frac{p(z_{k,t}|x^i_{k,t})p(x^i_{k,t}|x^i_{k,t-1})}{q(x^i_{k,t}|x^i_{k,t-1}, z_{k,t})}
\]  

(91)
The function $q(x_{k,t}^i | x_{k,t-1}^i, z_{k,t})$ is known as the proposal function, and its selection is critical in particle filtering. The weights are normalized such that $\sum_{i=0}^{n} w_{k,t}^i = 1$, where $n$ is the number of particles. The posterior density is then approximated by,

$$\sum_{i=1}^{n} w_{k,t}^i \delta(x_{k,t} - x_{k,t}^i)$$

where $\delta(.)$ is the Dirac Delta function.

There is no known proposal density function designed for tracking dim objects in a noisy image sequence. Therefore, to calculate the weight of a particle in sensor $s$, a fitness value is obtained using a fitness function $fit(z_{k,t}^i)$ over observation of the $i$-th particle of target $k$ at time stamp $t$. For each element of the matrix $P_{t}^s$, the weight of the $i$-th particle $w_{k,t}^i$ is then defined using the following equation:

$$w_{k,t}^i = \frac{fit(z_{k,t}^i)}{best_{k,t}^s - worst_{k,t}^s}$$

where $worst_{k,t}^s$ and $best_{k,t}^s$ are defined as below:

$$best_{k,t}^s = \min_{i} \{fit(z_{k,t}^i)\} \text{ and } worst_{k,t}^s = \max_{i} \{fit(z_{k,t}^i)\}$$

The fitness function should be tailored according to the system limitations as explained in the examples in the simulation results section. To reduce the processing cost of the fitness function, many researches focus on developing more efficient target features and similarity measures [142-148]. The use of color value as a feature has been common in color tracking problems, since they allow the global properties of objects to be captured [143]. However, the use of color value alone is not sufficient in the noisy environment of dim object tracking. To enhance the accuracy of dim object tracking in noisy environments, frequency domain information obtained by morphological operations [3, 25] or wavelet transform [26] has been used in the literature. This chapter
suggests a combination of both spatial and frequency domain information for designing the fitness function. In the experimental results, two different fitness functions are defined for different image sequences depending on the reliability of the features.

**Task 3) Swarm Intelligence Resampling:** The last step of a particle filter algorithm is resampling, which involves eliminating particles that have small weights and concentrating on the high weight particles. Some methods in particle filtering do resampling in every time stamp, while others will do it when the degeneracy of the particles is severe. However, resampling is an iterative process in SIPF.

This iterative process is inspired by Particle Swarm Optimization (PSO). Particle Swarm Optimization (PSO) is a parallel evolutionary algorithm developed by Kennedy and Eberhart based on social swarm behavior [63]. PSO uses the swarm intelligence concept which is the property of a system, in which the collective behaviors of unsophisticated agents that are interacting locally with their environment create coherent global functional patterns. PSO exhibits each solution as a particle moving through the problem hyperspace.

In a given iteration \( j \) of the resampling step, the state of each particle in matrix \( P^s_t \) is updated as following,

\[ P^s_j = P^{s,j-1}_t + V^{s,j}_t. \]  

(95)

Similarly, the particles velocity matrix \( V^s_t \) in iteration \( j \) is updated toward the local best matrix \( L^s_t \) and the global best matrix \( G^s_t \).

\[ V^{s,j}_t = V^{s,j-1}_t + rand_1.(L^s_t - P^{s,j-1}_t) \times \Psi 1^s_t + rand_2.(G^s_t - P^{s,j-1}_t) \times \Psi 2^s_t \]  

(96)

Note, the matrices \( V^s_t, L^c_t, \) and \( G^c_t \) are of size \( n \times m \). It is obvious that all the elements in a column of \( G^c_t \) are equal, since it is the value of the global best. The
parameters $rand_1$ and $rand_2$ in the above equation are selected from the uniform
distribution in the interval $[0,1]$. The matrices $\Psi_1^s$ and $\Psi_2^s$ are in diagonal form of
size $m \times m$ that control the acceleration of the movement of each particle towards its
individual and global best positions, respectively.

\[
\Psi_1^s = \begin{bmatrix}
\psi_{1,1,t} & \ldots & 0 \\
\vdots & \psi_{k,t} & \vdots \\
0 & \ldots & \psi_{m,t}
\end{bmatrix}, \quad \Psi_2^s = \begin{bmatrix}
\psi_{1,1,t} & \ldots & 0 \\
\vdots & \psi_{k,t} & \vdots \\
0 & \ldots & \psi_{m,t}
\end{bmatrix}, \quad k \epsilon [1, m], \quad (97)
\]

where the elements $\psi_{1,k,t}$ and $\psi_{2,k,t}$ are explained in Equations (100) and (101). Each
element of the velocity matrix $\mathbf{V}_t^s$ is a stochastic variable that controls the motion of the
particles in the problem hyperspace. To limit the oscillations, the values of the elements
in the $k$-th column of the velocity matrix $\mathbf{V}_t^s$ which are related to target $k$ are restricted in
the range of $[-V_{max}^s_{k,t}, +V_{max}^s_{k,t}]$. Selecting the values of $V_{max}^s_{k,t}$ and the elements
of $\Psi_1^s$ and $\Psi_2^s$ enables the sensor $s$ to track target $k$ in the case of large jumps or
observation discontinuities. The selection of these values is based on the reliability of the
sensor in the previous frame.

The distance between the positions of target $k$ in the previous frame found by
sensor $s$ represented by $\mathbf{x}_{k,t-1}^s$ to its actual position found by the center $(\mathbf{x}_{k,t-1})$ is
obtained by the following Euclidean distance:

\[
d_{k,t}^s = \| \mathbf{x}_{k,t-1}^s - \mathbf{x}_{k,t-1}^s \|, \quad (98)
\]

A high value of $d_{k,t}^s$ shows the degree of unreliability in the previous detection of
target $k$ by sensor $s$. If the distance value exceeds a threshold, the target is lost due to
object occlusion. In these cases, the particles should be redistributed everywhere in the
frame in order to find a new position for the target. Therefore, maximum velocity changes to infinity and the global acceleration $\psi_2^{s_k,t}$ is set to zero to increase the divergence of particles’ positions as following.

\[
V_{max}^{s_k,t} = \begin{cases} 
\infty, & \text{if } d_k^{s,t} > Thr \\
 d_k^{s,t}, & \text{otherwise}
\end{cases}
\]

(99)

\[
\psi_1^{s_k,t} = \begin{cases} 
4, & \text{if } d_k^{s,t} > Thr \\
\frac{d_k^{s,t}}{Thr}, & \text{otherwise}
\end{cases}
\]

(100)

\[
\psi_2^{s_k,t} = \begin{cases} 
0, & \text{if } d_k^{s,t} > Thr \\
\frac{Thr - d_k^{s,t}}{Thr}, & \text{otherwise}
\end{cases}
\]

(101)

where $Thr$ is a user-defined threshold. Most researches in the field of PSO report best results when the summation of global and local acceleration is equal to 4 [64, 66].

When a maximum number of iterations are reached, the best position of all targets found by sensor $s$ is reported to the central sensor.

### 7.1.2 Step 2: Finding the actual positions of targets in the central sensor

The central sensor at the upper level finds the actual position of each target from the information reported by all other sensors. At frame $t$, the input to the central sensor is $X_t^{s}$ from all other sensors and the output is the actual positions of the targets $X_t$ in the corresponding frame. The central sensor finalizes the actual position of each target by choosing the best among the reported positions as provided by different sensors. The following equation describes the actual positions of targets obtained by the central sensor,
\[ X_t = [x_{1,t}, x_{2,t}, \ldots, x_{m,t}]^T \]
\[ \forall k \in [1, m], \ x_{k,t} = \text{best}\{x_{k,s}^s | s = 1..N_s\}. \] (102)

where \( N_s \) is the number of sensors in the system.

The central censor broadcasts \( X_t \) to each sensor to be used as the input for the next frame \( t + 1 \). The proposed method is summarized as a flowchart in the following Figure 7.1.
Figure 7-1: Flowchart of the proposed method
7.1.3 Step3: Center Selection

In the proposed Multi-sensor tracking scheme, the central sensor is selected dynamically. The sensors are calibrated and their positions are known to each other. Choosing a fixed central sensor has a disadvantage of limiting the field of view for an application to one sensor. Therefore, the central sensor is selected dynamically according to the position of the target.

Similar to other sensors at the lower level, the central sensor performs the search algorithm to find its own target positions denoted by $X_{Center}^t$. The central sensor is not eligible to be at the center if the number of visible targets in its field of view is less than half of the total number of targets. In this case, the central sensor will be replaced by a sensor with the most number of targets in the field of view. All sensors should be aware of this selection in order to change their calibration parameters with respect to the new central sensor.

7.2 Experimental Results

The proposed SIPF method has been implemented in MATLAB, and its performance is evaluated in comparison with other related algorithms. The simulation results of two different image sequences are explained in this section. The first sequence is a standard dataset borrowed from Bristol Eden multi-sensor project [149]. The other dataset is obtained by the authors using an array of cameras capturing the movement of three drones in the air.

The performance of the proposed SIPF is compared with a centralized particle filter. The centralized architecture is implemented using a particle filter running in a fixed central sensor which is responsible for the final results. For each particle, all the sensors
are asked for the fitness value of each particle. In the final data fusion step, data collected from all the sensors are fused to provide the best fitness for each particle.

In addition, the proposed SIPF is compared with the results of the structural similarity-based particle filter [139] which is a robust method for object tracking in noisy image sequences. When a target is lost, this method uses different layers of data to perform a rapid search for the target and continues tracking.

An objective measure, namely Error-rate Per Frame (EPF) is used to assess the performances of various methods. EPF as defined in Equation (103) is a measure of the average distance between the detected targets and their ground truth states. The ground truth states correspond to the true positions of the object obtained manually [20].

\[
EPF = \frac{\sum_{\text{for all } DT} \|c_{\text{truth}} - c_{\text{detected}}\|}{DT} \tag{103}
\]

In the above equation, \(c_{\text{truth}}\) represents the coordinates of the ground truth state, \(c_{\text{detected}}\) specifies the coordinates of the detected position of the target, and \(DT\) is the total number of detected targets.

It is, however, insufficient to justify the effectiveness of a dim object tracking algorithm simply by measuring the EPF, because there are other critical parameters associated with a dim object in addition to the detected target positions. The number of detected targets \(DT\), missed targets \(MT\), and false alarm \(FA\) detections in a frame are other critical parameters to be considered in the evaluation. Hence, a more comprehensive measure, namely Target to Clutter Ratio (TCR) as defined in the following is an appropriate measure for comparing different tracking algorithms [20],

\[
TCR = \frac{DT}{DT + MT + FA} \tag{104}
\]
In our implementation, both EPF and TCR measures are used to compare the effectiveness of the various methods. The following two examples will further explain the relative performances of the above mentioned algorithms.

7.2.1 Example 7.1: A Soldier walking across the bridge

This example is a video of a soldier walking across a bridge. The video sequence was filmed by both visible light and infrared (IR) cameras at the Eden Project Biome in Cornwall, UK, and more details are available in [149]. This video is filmed for more than 1 minute, however, the first 500 frames are selected in this example. In this video, the soldier is camouflaged with the branches and leaves. In frame 180, the soldier is partially hidden in leaves for 50 frames, and he is completely hidden behind the leaves in frame 335 for duration of 105 frames. Figure 7.2 shows the two corresponding frames using visible light and IR cameras. Note, both visible light and IR videos are registered. The soldier is relatively close to the sensors and the IR camera is able to locate the target more accurately when the target is hidden behind the leaves. In frame 1, the soldier is visible to both cameras, however, he can be located only by IR camera in frame 180. The soldier is hidden from both cameras in frame 335.

This example uses an IR camera in its sensing network which does not involve any color information. In addition, the moving target is relatively large, and the use of frequency domain information does not seem important. A better fitness function $fit(z_k^{L_S})$ can be defined based on the image intensity values.

In the first frame of both image sequences, a tracking box surrounding a target is defined as shown in Figure 7.2. The upper-left corner point of the tracking box determines the position of the target or particle. The fitness value of each particle as
mentioned in step1-task2 of the proposed algorithm is obtained based on the luminance intensities of the corresponding tracking box pixels. The difference of the luminance intensity values of the particle in the current frame and the target position in the previous frame is calculated for all pixels in the corresponding tracking boxes. The average of these differences is assigned to the fitness value of the particle.
Figure 7-2: Images from IR and visible light cameras
The EPF and TCR measurements of our proposed SIPF method is compared with the centralized and similarity-based particle filter techniques and the results are presented in Figure 7.3 and Figure 7.4, respectively. As can be seen from Figure 7.3 all three methods have the same performance up to frame 150. The target partially disappears after frame 180 and it reappears again in frame 220. There is a small increase of error rate in all three methods at frame 220. Between frames 335 to 440, the target is completely hidden and EPF is zero. However, the evolutionary search used in the SIPF can relocate the target quickly after a large error jump at frame 440.

![Figure 7-3: EPF of the three tracking methods over 500 frames](image)

Example 7.1 deals with a single object, therefore, the TCR value is either zero or one indicating whether the target is detected or not. Figure 7.4 shows the TCR values for the three algorithms in which none of the algorithms could detect the target between frames 335 to 440.
Example 7.2: Three drones moving in the sky

In the second example, a set of four cameras is used to capture three moving drones in the sky. The cameras are of type CMOS taking visible light images and have fixed position with known registration parameters. The targets are far from the cameras and the images are relatively noisy providing a typical dim object tracking example. Similar to example 1, the quality of the initial frames of the video are acceptable and the presence of all the objects can be located in at least one of cameras. Figure 7.5 shows that in frame 120 drone #3 gets close to the car’s headlights in the background, and drone #2 are far from the cameras in a noisy background. Therefore, the tracks of some drones will be lost during the video.

All four cameras in example 7.2 record the color information. Therefore, color and luminance intensities are separated using YCbCr color space. Similar to example 7.1, the average difference of the color intensity values of the current frame particle and the previous frame target position is calculated for all pixels in the corresponding tracking
boxes. The average difference value is one of the features used to define the fitness function $f_{\text{fit}}(z_{k,t}^{ls})$ for this example.

The targets in this example are small; hence, high frequency areas can be used to find the locations of the possible targets. A Complex Wavelet Transform (CWT) is used in this example to capture the high frequency sub-bands information. The accumulation of six CWT high frequency sub-bands is shown in Figure 7.6. The average values of the accumulated high frequency coefficients in the tracking box of a particle is a second feature used in the calculation of the fitness function in this example. The combination of these two features can track small objects more accurately.

Figure 7.7 shows the EPF measurement of different methods over 300 image frames. When one of the targets is lost, the proposed SIPF algorithm using the evolutionary search could relocate it much faster than others. Between frames 160 and 180, drone #1 is out of the views of all cameras. The TCR values on Figure 7.8 show that the proposed method is capable of relocating the target in a reasonable time by using independent evolutionary search in all sensors (cameras). However, the centralized particle filter cannot relocate the target, whereas the similarity-based particle filter finds it after a longer period of time.
Figure 7-5: Positions of three drones from four different views in frame 120
Figure 7-6: Accumulation of CWT high frequency sub-bands. Small objects are located in high frequency areas.

Figure 7-7: EPF of the three tracking methods over 300 frames.
7.3 Conclusion

This chapter presents a novel hierarchical architecture to track dim objects in a multi-sensor environment. To achieve the optimum position of a target in each frame, the proposed method uses a distributed particle filter in a multi-sensor environment. In a hierarchical two-level structure of sensors, an evolutionary search based algorithm is applied in each sensor to locate the targets. Each sensor in the lower level finds the best positions and reports its information to a dynamically selected central sensor at the upper level. The central sensor finds the best among the reported positions for each target and broadcasts it to all other sensors as the actual position.

The experimental results show that the proposed method is capable of detecting and tracking targets in image sequences with low SNR values (less than 2dB) even in the case of large jumps or discontinuities in the observation data. Comparing with some other
methods, the proposed method is more effective with relatively simple implementation. The proposed distributed approach provides more information about the object and increases the robustness of the algorithm to occlusions and clutters. When the target is lost, the sensors cooperate with each other and perform their individual evolutionary search to relocate the target. In addition, the proposed method avoids the premature convergence of the particles and enlarges the search region of particles by using the evolutionary search in a multi-sensor framework. This is especially useful when the target states change abruptly with a large discontinuity in the measurement data.
Chapter 8
Conclusion and Future Research

8.1 Summary

In this dissertation, dim object tracking methods in cluttered image sequences are studied in different situations, and a number of novel techniques are developed. This study begins with a novel SR technique to increase the resolution of the image sequences. A GA based sparse coding framework for obtaining an HR image from the LR image has been introduced. In the sparse coding framework, LR image is processed block-wise in which a block is considered to be a patch. For each LR patch, a sparse representation called $\alpha$ is to be found from an LR dictionary. Multiplication of $\alpha$ and HR dictionary is the reconstructed HR patch. Finding the $\alpha$ representative for each LR patch is a time consuming step in conventional sparse coding techniques. In this dissertation, a GA optimization is applied to decrease the total time of dictionary searching. Experimental results show that the proposed method is able to outperform conventional SR methods image quality while its execution time is less than most of the conventional methods.

For dim object tracking with various speed, a TBD algorithm based on MHT technique is proposed. The algorithm is not based on any assumption of background noise distribution. The proposed MHT method uses MOPSO optimization search to
follow the best track in each group of frames. Therefore, the execution time of the
algorithm is reasonable. The simulation results show that the algorithm is capable of
detecting and tracking high speed objects in cluttered image sequences. Also, the
proposed method involves fewer computations with relatively simple implementation.

To subtract the cluttered background from dim objects a SVM classifier method is
presented that divides object and background blocks into two separate groups. The SVM
classifier decides through spatial domain features. However, a prescreen CFAR detector
selects possible targets through frequency information and pass them to SVM. Therefore,
this method uses a combination of both frequency and spatial domain information to
detect and track targets more accurately in noisy backgrounds. It is shown in the
experimental results that this method outperforms other conventional frequency or spatial
domain object tracking methods.

The discontinuity of data observation causes the loss of tracks in dim object
tracking methods. Particle Filter (PF) is studied as an extension of the Kalman filter to
deal with non-Gaussian non-linear dynamic systems. Sample impoverishment is the flaw
of the PF techniques in the case of observation discontinuity. A new PF based method
called Social-swarm Optimized Particle Filter (SOPF) is introduced to overcome this
flaw of the PF. The proposed SOPF applies spider-social optimization evolutionary
search in the sampling step to overcome the premature impoverishment. The proposed
SOPF divides particles into male and female groups with different moving behavior.
Through this diversity SOPF provides a wide search area in the case of data discontinuity
and relocate the objects when they reappear. The superior performance of the proposed
SOPF over other evolutionary PF techniques in the case of data discontinuity is evidenced in the experimental results.

Finally, a new multi-sensor dim object tracking algorithm is presented in this dissertation. It is clear that the data received from different sources actually will help the reliability of the decision-making process. The proposed algorithm called SIPF combines swarm intelligence and PF to track objects in the network of multi-sensors. The sensors could be any type of CMOS cameras, IR cameras, RADARs, or Sonars. The proposed SIPF is implemented in a hierarchical two levels distributed architecture. The lower level performs an evolutionary PF in each sensor and finds the best possible targets. All the sensors report their obtained positions to the central sensor. The central sensor fuses all the received information and decides the final positions of the targets. The final positions are broadcasted to all the sensors to use as the prior data for next frame processing. According to the experimental results the proposed method is capable of detecting and tracking targets in noisy image sequences in the case of large jumps or discontinuities in the observations.

8.2 Future Research

Although the dim object tracking algorithms proposed in this dissertation are very efficient in many of the real world applications, it can be more robust by eliminating some of the limitations as listed below:

- Additional sources of information such as voice could be used to improve the accuracy of the object tracking in specific cases such as people tracking.
- A multi-sensor object tracking system that begins with non-calibrated cameras and could adjust the cameras continuously would be an enhancement to the proposed
SIPF multi-sensor model. In this system each camera could move toward one object individually.

- The multi-sensor object tracking methods can be extended to track people behaviors and to detect any behavior abnormality in the crowd. It would be a valuable feature for security cameras in the public areas.
References


[90] R. Y. Tsai, and T. S. Huang, "Multiframe image restoration and registration,"


[103] L. Wang, H., Geng, P., Liu, K., Lu, J., Kolodziej, R., Ranjan, and A. Y., Zomaya,


