By the name of God

Palestine Polytechnic University

College of Administrative Science and Informatics

Information Technology

General Purpose Real Time Object Tracking Using Particle Swarm Optimization Algorithm

Students

Ahlam Hasan Albashiti

Ala' "Mohammad Ahmad" Jabari

Supervisor

Dr. Hashem Tamimi

A final project is submitted in partial fulfillment of the requirement for the degree of B.Sc. in Information Technology

2009

Abstract

Object Tracking is a field of computer vision which concerns with following and locating moving desired objects in video sequences. Previous techniques depend on detecting the entire image, pixel by pixel which is time consuming. Recently new biologically inspired approaches were developed that rely on Meta heuristic algorithms, such as Particle Swarm Optimization (PSO) algorithm, which proved to be robust and very fast.

In this project we study the performance of the PSO algorithm to build a system for general purpose real time object tracking in dynamic environment and we apply it to indoor and outdoor environment.

The PSO algorithm is used in each frame to search for the desired object and then to track the object in the video sequence. Color histogram is used to model the object to be tracked. The Hue color component demonstrates robustness under different illumination.

The experimental results show that applying PSO algorithm to the object tracking field leads to efficient results. The PSO converges to desired object within seven iterations and it keeps tracking the object under different conditions.

Acknowledgment

At this pleasure moment, many special persons come to our mind, to thank them, who have great efforts to complete our project.

We are thankful to our supervisor Dr. Hashem Tamimi for his efforts and intelligent ideas. Special appreciate to him; because he teaches us how to work, the meaning of the work, and how to be creative.

Team

Ahlam & Ala'

Table of Contents

Abstract ..1

Acknowledgment ..2

Table of Contents ..3

Table of Figures ..5

1 Introduction ..7

1.1 Overview ..7

1.2 Project Goal ..7

1.3 Project Objectives ..8

1.4 Project constraints ..9

1.5 The organization of this document ..9

2 Background 11

2.1 Theoretical Background ..11

2.1.1 Standard Particle Swarm Optimization Algorithm 11

2.1.2 Fitness Function 11

2.1.3 Standard PSO Algorithm Flowchart 13

2.1.4 Particle Swarm Optimization Variance 15

2.1.5 Direction of Research 16

2.1.6 PSO Application 16

2.2 Image Processing ..17

2.2.1 Color Model 17

2.2.2 Color Histogram 19

2.2.3 Object Tracking 20

2.3 Literature Review ..20

2.3.1 Object Tracking 21

2.3.2 PSO Algorithm and Object Tracking 21

2.4 Summary ..22

3 Methodology 24

3.1 PSO Algorithm as a solution to the Object Tracking Problem 24
3.1.1 The Search Space Of Particle Swarm Optimization Algorithm 24
3.1.2 Defining the Goal 24
3.1.3 A Swarm of Interacting Individuals 25
3.1.4 Phases of the PSO in Tracking Process 25
3.1.5 Fitness Function 28
3.2 Summary of PSO Algorithm for Object Tracking 29
3.3 Summary 35
4 Experiments and Results 37
4.1 Testing Environment 37
4.1.1 Hardware and Software Specification 37
4.2 Experiments and Results 37
4.2.1 Images Experiments and Results. 37
4.2.2 Particle Swarm Optimization Experiments and Results 38
4.2.3 PSO Tracking Based Algorithm Parameters 40
4.2.4 Experiments and results for tracking process using PSO algorithm 43
4.2.5 Experiments and results for tracking outdoor environment 47
4.2.6 Experiments and Results for Tracking Different Objects either Indoor or Outdoor. 48
4.2.7 Tracking with Neighborhood Topology 52
4.3 Summary 52
5 Conclusion 54
References 55

Table of Figures

Figure 1-1 The tracking process 8
Figure 2-1 PSO Flowchart 14
Figure 2-2 Topology Types 16
Figure 2-3 RGB Color Model 18
Figure 2-4 HSV Color Model 18
Figure 2-5 Color Histogram 19
Figure 3-1 Shape of the goal 25
Figure 3-2 The fitness value for each particle at time t 29
Figure 3-3 flowchart of PSO algorithm for tracking 31
Figure 4-1 Color Models 37
Figure 4-2 Hue component 38
Figure 4-3 the initialization, updating, convergence steps in PSO 39
Figure 4-4 the average fitness values in each iteration 40
Figure 4-5 determining the number of particles in tracking process. 41
Figure 4-6 the relationship between average fitness values and inertia factor 42
Figure 4-7 the effect of decreasing inertial factor in tracking process 43
Figure 4-8 tracking inside indoor environment 45
Figure 4-9 tracking inside indoor environment in running state 46
Figure 4-10 Tracking Outdoor Environment 48
Figure 4-11 Tracking Yellow pen and circular motion. 49
Figure 4-12 Tracking red packet. 49
Figure 4-13 Tracking blue blouse and right, left motion 50
Figure 4-14 tracking human hand with rotation and translation 51
Figure 4-15 Tracking candle with rotation and translation 51
Figure 4-16 neighborhood effect in tracking process 52

Chapter 1
Introduction

1.1 Overview

Object tracking means following and detecting a desired object in video sequences [2], it is a challengeable field of the computer vision science; because of the illumination changing which affect the color intensity, the complexity of the object shape and motion, the object occlusion… etc. To enable tracking, we need an efficient algorithm that satisfies the three constrains. 1) It should be appropriate for real time cases. 2) It should be suitable under object transformation. 3) It should work under dynamic environment, and illumination changes.

PSO algorithm is an optimization algorithm that is inspired by the social behavior of a swarm of birds, fishes etc. to find the optimal solution in a search space [4]. This algorithm is suitable for real time application; because of a set of individual behavioral rules that allow reaching the optimal or near the optimal solution quickly. These rules imitate the behavior of birds in our life such as the communication and interactions between the individuals of this algorithm as we will discuss in Chapter two. On the other hand the parameters of this algorithm are limited which makes it easy to optimize them. This algorithm is heuristic; it depends on fitness values to get the optimal solution.

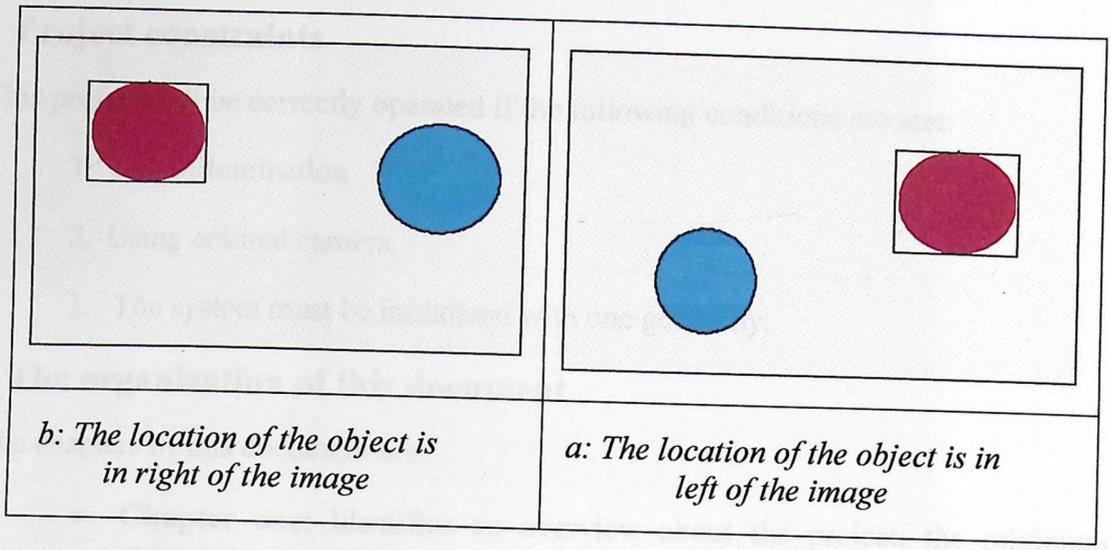
This project is a contribution to the solution of tracking problem inside indoor and outdoor environment using image processing techniques and particle swarm optimization algorithm (PSO); our aim is to build up a system for general purpose object tracking under dynamic environment. This system is for research purposes, it could be enhanced to be any application of object tracking applications, such as giving information of the behavior of the tracked object, surveillance application, and cognitive vision.

In this project, the tracking is done based on color. Other algorithms use texture and shape. But color information is easier to extract which makes the process real time.

1.2 Project Goal

To enable the computer to track any selected object based on its color and regardless of its shape using PSO algorithm while maintaining the real time issues.

Figure 1-1 shows an illustration of this goal where the desired object is the circle and square is the tracker. The algorithm should neglect any other object in the image.



b: The location of the object is in right of the image

a: The location of the object is in left of the image

Figure 1-1 The tracking process

1.3 Project Objectives

The minimum objectives that are expected to be achieved from this project are:

- **General purpose object tracking**: Our project will track any object defined by the user regardless of its shape, color, size, direction of motion.

  Our project is not for specific application but for scientific research. We will build a simulation to track any object and this simulation can be enhanced to suit any tracking application such as surveillance application, system or service robotics or public security system.

- **Real time tracking:** The system will track the desired object taking into account real time issues; the system will process 25 to 30 frames per second.

- **The algorithm is invariant object transformation:** the algorithm will track the object even there is translation, rotation, and scaling.

- **Tracking is done under dynamic environment**: If the environment contains other mobile objects than the desired object, their existence will not affect the tracking process if there color differs from object color.

- **Tracking inside indoor and outdoor environment**: The tracking process will be done inside indoor and outdoor environment.

- **Minimal Hardware requirements**: Only personal computer (PC) with: 256 G RAM, 1.7GH CPU speed, up to 3G free HD space and a webcam connected to the PC and no other accelerating hardware components are used.

1.4 Project constraints

This project will be correctly operated if the following conditions are met:

1. Good illumination
2. Using colored camera
3. The system must be initialized with one goal only.

1.5 The organization of this document

The chapters of this document are:

- **Chapter one**: Identifies an overview about the project, the minimum objectives that will be achieved when this project complete, hypotheses, constraints.

- **Chapter two** gives a general background to the reader. It is organized into two sections, first section is theoretical background that describes all the terms, topics, and methods are needed to understand this document. While the Second section is the literature review, in which we mention previous works related to the object tracking using PSO algorithm and other algorithms in addition to object tracking, we review other works that are done using PSO algorithm.

- **Chapter three** describes all the operations, sequences, diagrams that are needed to implement the system. We describe the PSO algorithm in terms of tracking process, the pseudo code of the object tracking based PSO algorithm.

- **Chapter four**: shows the experiments for the PSO algorithm and the results. The experiments that are done inside indoor and outdoor environments and the results of them.

- **Chapter five**: shows the conclusion and the future work.

Chapter 2
Background

In this chapter, we will discuss the necessary topics, terms, formulas that are used in the project in general in order to understand it. We divide this chapter into two sections. The first one is theoretical background, which is needed to build sufficient background about the project. The second one is previous work of the Particle Swarm Optimization algorithm and object tracking.

2.1 Theoretical Background

In this section, we will talk about the standard PSO algorithm, its phases and variation. Then we will proceed to talk about computer vision and image processing terms, such as object tracking, color model, histograms and finally comparing histograms.

2.1.1 Standard Particle Swarm Optimization Algorithm

Particle swarm optimization (PSO) algorithm is swarm intelligence based algorithm that is inspired by the social behavior of fish schooling or bird flocking to find a solution to an optimization problem in a search space [3]. This algorithm is introduced by Ebhart and Keneddy in the year 1995 [4].

PSO algorithm has two prominent properties. The first feature is the population-based algorithm; it consists of individuals that are used to explore the search space, each individual is called a particle, which represents a candidate solution in the search space. The second feature is the communication and interaction between these particles in order to get the optimal solution as robust and quick as possible [5].

In this algorithm, particles "candidate solution to a given problem" imitates the behavior of a swarm of birds searching food, to reach the food as quickly as possible. All birds follow the nearest bird to that food. In order to do this, each particle in PSO algorithm has a position and velocity in all dimensions. A fitness value is also maintained to evaluate the position for each particle [6, 7].

PSO algorithm starts by randomly initializing particles "candidate solution" in the search space. Then the particles search for the optimal solution by updating their position to become nearer to the optimal solution. To do this, each particle must keep two best values; the first one is the best position in the swarm called global best position. The second one is the best position each particle itself has called local best position [8]. The global and local best values depend on the fitness function explained below.

2.1.2 Fitness Function

The fitness function forms the backbone of the PSO; it measures how far each particle from the solution, in other words, it determines how the current positioning in the search space is close to the goal. Local and global best positions are determined by the highest fitness value each particle itself has, and the highest fitness value in the swarm respectively. The implementation of the fitness function depends on the problem.

The PSO has three phases [9], namely:

1. Initialization.

2. Optimization.
3. Termination.

The following section describes each phase in details.

First, we use the $[.]_i^d(t)$ to denote a vector element i of dimension d at iteration t.

1. Initialization Phase

In this phase, all particles are randomly initialized to cover the search space. The common way to initialize the particles is through the use of the uniform distribution function. The following equations are used for the initialization:

$$x_i^d(t=0) = x_{\min}^d + \left(x_{\max}^d - x_{\min}^d\right) \cdot rand \tag{2.1}$$

Where x_i^d is the initial position of the particle , $x_{\min}^d$, $x_{\max}^d$ are the maximum and minimum value of the search space respectively, these values are used to restrict the particles to be in the search space of the problem. In order to get random initialization the equation is multiplied by random number between [0, 1] *rand* .

The velocity of all particles is set to zero in each direction.

$$v_i^d(t=0) = 0 \tag{2.2}$$

Where v_i^d is the velocity of the particle.

2. Optimization Phase

The optimization phase contains several iterations, in each iteration the particle positions and velocities are updated, in order to reach the optimal solution; this is done using the following velocity equation [10]:

$$v_i^d(t+1) = Inertia + PersonalInfluence + SocialInfluence \tag{2.3}$$

Where:

$$Inertia = w \cdot v_i^d(t) \tag{2.4}$$

Here, w is the inertia weight, which is proposed by Shi and Eberhart to control the impact of the previous velocity on the current velocity [4].

$$PersonalInfluence = c_1 \cdot rand_1 \left(pBestPos_i^d(t) - x_i^d(t)\right) \tag{2.5}$$

The personal influence part measures the distance between the particle's best position $pBestPos_i^d(t)$ at time t and the current position $x_i^d(t)$.

$$SocialInfluence = c_2 \cdot rand_2 \left(gBestPos_i^d(t) - x_i^d(t)\right) \quad (2.6)$$

The social influence measures the distance between the best position in the swarm at time t and the current position ($x_i^d(t)$). c_1 and c_2 are constants, which are used to determine how much *SocialInfluence* and *PersonalInfluence* will affect the particle's movement.

$$v_i^d(t+1) = w \cdot v_i^d(t) + c_1 \cdot rand_1 \left(pBestPos_i^d(t) - x_i^d(t)\right) + \cdots$$
$$c_2 \cdot rand_2 \left(gBestPos_i^d(t) - x_i^d(t)\right) \quad (2.7)$$

The position equation depends on the velocity as follows:

$$x_i^d(t+1) = x_i^d(t) + v_i^d(t+1) \quad (2.8)$$

Where $x_i^d(t+1)$ is the *ith* particle at time $t+1$, and $x_i^d(t), v_i^d(t+1)$ are as explained in Equation 2.3.

3. Termination Phase

In this phase, one or more termination conditions may be used, such as all particles reach the same position, or little or no improvement is observed in particles positions, or when the solution has been found by at least one particle or for a given number of iterations.

2.1.3 Standard PSO Algorithm Flowchart

Figure 2-1 describes the standard PSO algorithm, where the terminating condition is reaching using predefined number of iteration. It starts with random initialization of all the particles. Then Equations 2.7 and 2.8 are used to updates the velocity and the position for each particle. The fitness value is used to determine the local and global best positions.

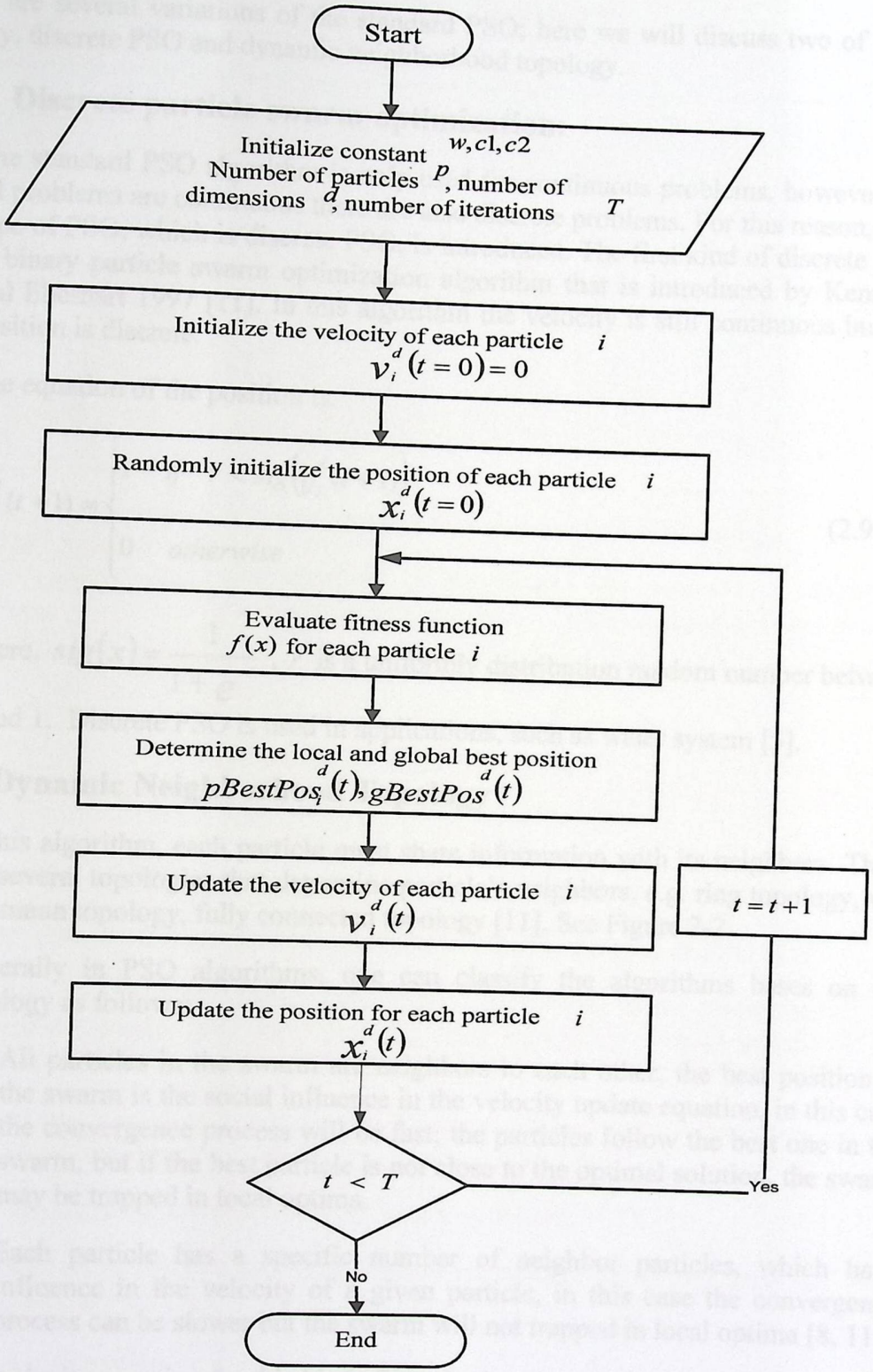


Figure 2-1 PSO Flowchart

2.1.4 Particle Swarm Optimization Variance

There are several variations of the standard PSO; here we will discuss two of them namely, discrete PSO and dynamic neighborhood topology.

- **Discrete particle swarm optimization:**

  The standard PSO algorithm is only used for continuous problems, however not all problems are continuous there are also discrete problems. For this reason, new type of PSO, which is discrete PSO, is introduced. The first kind of discrete PSO is binary particle swarm optimization algorithm that is introduced by Kennedy and Eberhart 1997 [11]. In this algorithm the velocity is still continuous but the position is discrete.

  The equation of the position is:

$$x_i^d(t+1) = \begin{cases} 1 & if \quad r < sig\left(v_i^d(t+1)\right) \\ 0 & otherwise \end{cases} \tag{2.9}$$

  where, $sig(x) = \frac{1}{1+e^{-x}}$, r is a uniformly distribution random number between 0 and 1. Discrete PSO is used in applications, such as water system [5].

- **Dynamic Neighborhood Topology**

  In this algorithm, each particle must share information with its neighbors. There are several topologies that determine particle's neighbors, e.g. ring topology, von Neumann topology, fully connected topology [11]. See Figure 2-2.

  Generally in PSO algorithms, one can classify the algorithms bases on the topology as follows:

  - All particles in the swarm are neighbors to each other, the best position in the swarm is the social influence in the velocity update equation, in this case the convergence process will be fast; the particles follow the best one in the swarm, but if the best particle is not close to the optimal solution, the swarm may be trapped in local optima.
  - Each particle has a specific number of neighbor particles, which have influence in the velocity of a given particle, in this case the convergence process can be slower but the swarm will not trapped in local optima [8, 11].

  The velocity equation for this kind of topologies

$$v_i^d(t+1) = w \cdot v_i^d(t) + c_1 \cdot rand_1\left(pBestPos_i^d(t) - x_i^d(t)\right) + c_2 \cdot rand_2\left(gBestPos_i^d(t) - x_i^d(t)\right) \ldots$$
$$+ c_3 \cdot rand_3\left(nBestPos_i^d(t) - x_i^d(t)\right) \tag{2.14}$$

Where $nBestPos_i^d(t)$ is the best position of the neighbors for the each particle, all other parameters are the same is discussed in Equation 2.7.

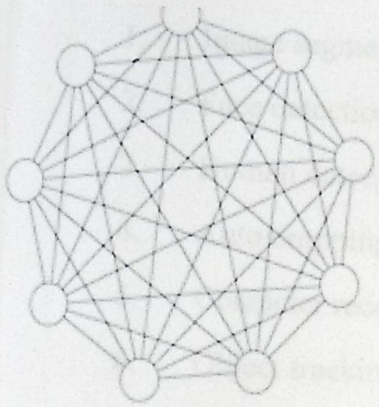
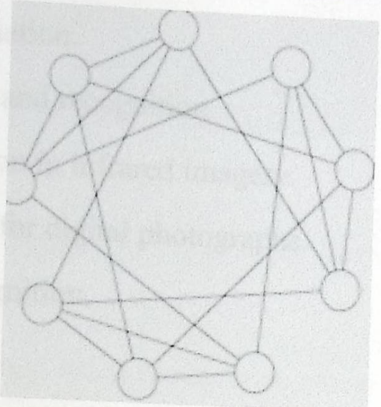
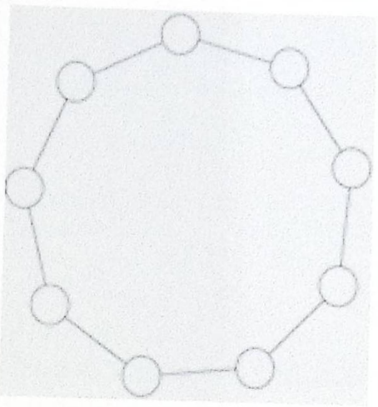
		
a: Fully connected topology. The number of neighbors for each particle is all the swarm.	*b: Von Neumann topology.* *The numbers of neighbors for each particle are four.*	*c: Ring topology.* *The numbers of neighbors for each particle are two.*

Figure 2-2 Topology Types [11]

2.1.5 Direction of Research

A number of research directions are currently pursued in the area of PSO [11]:

- **Comparing PSO algorithm with other algorithms:** Some researches compares between PSO algorithm and other algorithms such as genetic algorithm for human head tracking [6].
- **Parameter selection**: such as how many particles, suitable topology, number of iterations, the value of the inertia factor …etc. [10].
- **New variants**: New variation of PSO such as discrete PSO[5].
- **Application of PSO to different kind of problems:** this is explained in details below:

2.1.6 PSO Application

PSO algorithm has been deployed across numerous kinds of applications, ranging from security and military [17] and biomedical [12], to network [13], clustering and classification [14], to scheduling [15], to robotics [16] …etc.

In a research done by Poli [18], an analysis of about 1100 publications of IEEE database is conducted in an attempts to help researchers to take an overview of what applications have been done up to the year 2007. The paper categorizes the

applications into 26 categories; here will show examples for Image and Video category.

Approximately 9% of the papers are concerned with Image and Video analysis, this include the following topics:

1. Image segmentation.
2. Face detection and recognition.
3. Human detection in infrared imagery.
4. Auto cropping for digital photographs.
5. Character recognition.
6. Object tracking.
7. Body posture tracking.

2.2 Image Processing

Image processing is any form of information processing for which the input is an image such as photographs or frames and the output can either be an image or set of features related to the image. Most image processing technique involves treating the image as a two dimensional array of pixels, each pixel represents a sample of the original image, the intensity of each pixel is variable; in color images each pixel has three components such as red, green, and blue (the RGB model).

The following terms and topics are necessary for the reader in the area of image processing:

2.2.1 Color Model

Color model is "an abstract mathematical model describing the way colors can be represented, typically as three or four values of color components" [21].

There are several types of models; we will discuss two of them RGB and HSV models that we use in our project.

2.2.1.1 RGB Color Model

The main colors that are used in this model are Red, Green, and Blue. These three colors are the basic components; other colors are made by mixing these colors together.

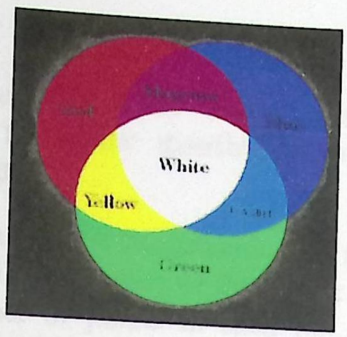


Figure 2-3 RGB Color Model

In our research input images from the camera are in RGB color model. Then these images will be converted into HSV color model, (see next section).

2.2.1.2 HSV Color Model

HSV (Hue, Saturation, and Value) model contains three components [22, 23]:

- **Hue**: what the color is e.g. red, orange, and blue.
- **Saturation:** the purity of the color.
- **Value**: the brightness or the darkness of the color.

HSV color model can be visualized as cone (see Figure 2-4), the base of the cone represents Hue, the distance from the center to the circular cross-section of the cone (radius) represents Saturation, and the distance from the center to end of the cone (height) represents the value [6]. HSV has advantage over RGB color model that is invariant to the illumination [25].

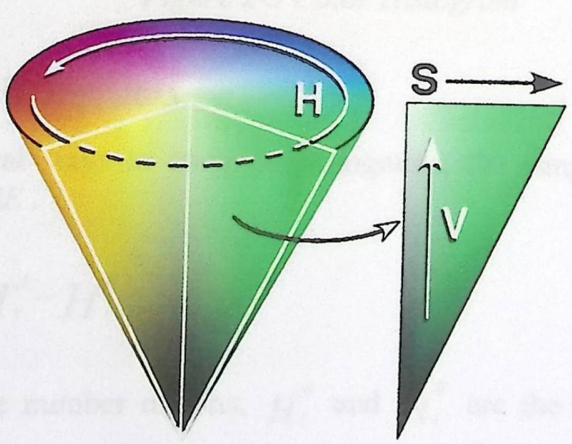


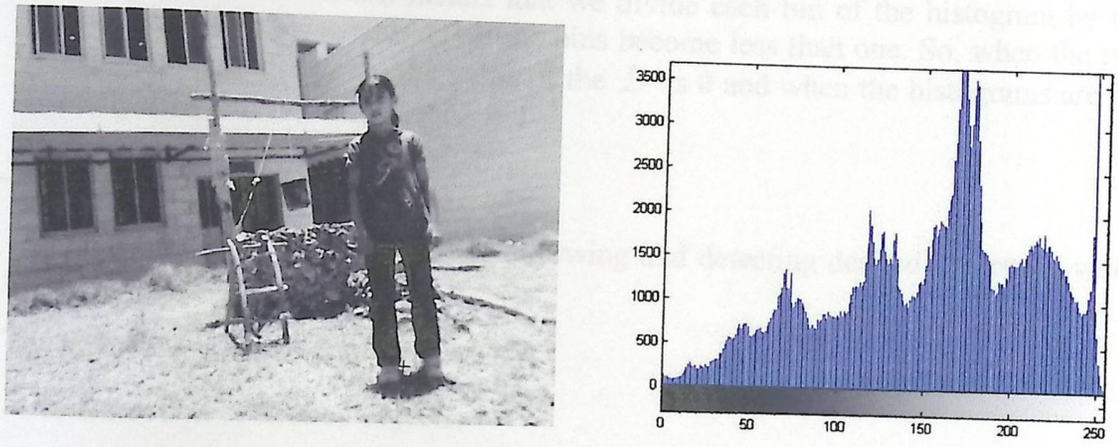
Figure 2-4 HSV Color Model [23]

2.2.2 Color Histogram

Color histogram is a representation of colors distribution in an image. The representation depends on the color distribution of the object being studied, regardless of its shape and texture.

Histograms can be built from images in various color model, RGB, HSV …etc, the distribution is made first by discrimination of the colors in the image into a number of bins, and counting the number of pixels in each bin.. The number of bins in the histogram may be 16, 32, 64 or 256. Figure 2-5(b) represent a 256-bin histogram, the X-axis represents the color intensity (number of the bin), and Y-axis represents the number of occurrences.

Color histogram may be one dimension. This is done using one component of any color model such as red, green, or blue in RGB color model, or it can be two dimensions or three dimensions, using two components or three components respectively.



a: Original image *b: Histogram with 256 bins*

Figure 2-5 Color Histogram

- **Comparing Histograms**

There are several ways to compare histograms; the simplest way is the Mean Square Error *MSE* .

$$MSE = \frac{1}{N}\sum_{i}^{N}\left(H_i^A - H_i^B\right)^2 \qquad (2.10)$$

Where N is the number of bins, H_i^A and H_i^B are the histograms of the two desired images.

Another way to compare histograms is Bhattacharyya Coefficient [10], which we use in our research; it is globally used and has proved to give efficient results than other methods.

Bhattacharyya Coefficient works on normalized histogram with an identical number of bins, the following is its equation:

$$BC\left(H^A \cdot H^B\right) = \sum_{i=1}^{N} \sqrt{H^A \cdot H^B} \tag{2.11}$$

The distance between the histograms is calculated using the following formula:

$$D\left(H^A, H^B\right) = \sqrt{1 - \sum_{i=1}^{N} \left(H^A \cdot H^B\right)} \tag{2.12}$$

Where D is the distances between two images, its ranges is [0, 1], H^A is the histogram of the first image, and H^B is the histogram of the second image.

Histogram normalization means that we divide each bin of the histogram by the summation of all the bins, so all the bins become less than one. So, when the two histograms are the same, the value of the D is 0 and when the histograms are the totally different the value of D is 1

2.2.3 Object Tracking

Tracking process is concerned with following and detecting desired objects in video sequences [19].

The key steps in video analysis are:

1. Detection of desired objects.
2. Tracking of the desired object from frame to frame, where the term frame is "one of the still images that are shown in quick succession in a video" [20].

Object tracking process can be complex process because of [21]:

1. Noise in the image, but some tracking algorithms overcomes this difficulty.
2. Complex object motion.
3. Object occlusions, partial or total occlusion.
4. Complex object shapes.
5. Scene illumination changes.
6. Real time processing requirements.

2.3 Literature Review

PSO algorithm is considered as an efficient optimization technique; this is due to its easy implementation, its low computation cost, and its robustness results, as a result many researchers use this algorithm in their applications. In this section we will

review the previous works that considered particle swarm optimization algorithm and object tracking process.

2.3.1 Object Tracking

Tracking algorithms can be classified into two categories: deterministic and stochastic methods. [26], in deterministic methods an iterative search is applied to find the local maxima of a similarly cost function between the template image and the current image. Mean Shift algorithm and its adaptation CamShift algorithm are types of deterministic methods [24]. It is used to track a human face depending on one dimensional histogram of the hue component of HSV images.

On the other hand, stochastic methods use the state space to model the tracking system. Such as swarm intelligent algorithms that are a population based algorithms, which are inspired by the biological behavior such as insects. Nielsen .E et.al [28] proposed a new approach using swarm intelligence metaphor, this approach depends on a prey-predator manner, where the group of particles is the predator, and a herd of pixels are the prey; each predator stores its position, velocity, speed, and color bank list. In addition to that, the movement of the swarm depends on four rules: color and topography, grouping, alignment and prediction, the result of each rule is a vector of velocity, and the summation of the four weighted victories, this is used to define the particle's final velocity and speed. This new approach is proved to give efficient results.

Particle filter is also another stochastic algorithm, which depends on Bayesian sequential sampling method that includes two steps: predict and update. Yang .C et.al [26] use the particle filter algorithm to track an object in real time, they enhance this algorithm for tracking, by using the integral image to compute the edge orientation histogram. Experimental results show the effectiveness of applying particle filter in tracking single and multiple objects.

2.3.2 PSO Algorithm and Object Tracking

A number of tracking applications are done using PSO algorithm, such as in the comparative research between the genetic algorithm (GA) and PSO algorithm [6]. In this research, the paper compares the performance of GA algorithm and PSO algorithm in tracking a human head by partner robot. The following fitness function equation is used in both algorithms:

$$f_i = C_{skin} + C_{hair} + \eta_1 \cdot C_{skin} \cdot C_{hair} - \eta_2 \cdot C_{others} \qquad (2.13)$$

Where C_{skin}, C_{hair} and C_{others} represent the number of pixels of the color corresponding to skin, hair, and other colors respectively, η_1 and η_2 are coefficients.

Each particle in PSO algorithm or each individual in GA represents image template or sub window, the template is represented with the head position and the size (height and width) of the head, and colors of the skin and the hair are used to detect the head.

The experimental results show that PSO algorithm and GA algorithm success to track the human head. But applying PSO in such problem gives more effective results than applying GA.

Another research is done using PSO is to track an object based on its color, in this research [25] the cascade classifier based on Haar-like features detector is used to detect the object, and PSO algorithm is used to track it, an accumulative histogram is used to build up the fitness function, and each particle is represented by a rectangle of four dimensions: width, height and the center of the rectangle. Particles converge to the goal within seven iterations. The experiments show that applying PSO algorithm in tracking process gives robust and efficient results.

2.4 Summary

In this chapter we have discussed and mention the terms, formulas and topics that are needed to understand this project. We discuss the standard PSO algorithm; the phases of this algorithm, and the variation of the PSO. On the other hand we have discussed the image processing mechanics such as histogram, comparing histograms. Finally the previous works of tracking either PSO algorithm or other algorithms were also reviewed.

Chapter 3
Methodology

After we discussed PSO algorithm as an algorithm for optimization problems, we will discuss it as a solution to the object tracking. We divide this chapter into two sections. Firstly, the PSO based object tracking algorithm is discussed in details. Secondly, the logical view of the system is explained to give sufficient information to build it and test it for the next chapter.

3.1 PSO Algorithm as a solution to the Object Tracking Problem

PSO is an optimization algorithm that assumes the existence of a search space, a goal to reach it, and a swarm consists of interacting individuals. In the following we discuss each of them in terms of object tracking.

3.1.1 The Search Space of Particle Swarm Optimization Algorithm

Search space is the space in which all the candidate solutions (particles) are initialized in order to find the goal. We define the search space in our system as a frame in video sequences. The size of the search space is the width by the height of the frame. Our search space is bounded by a minimum value and maximum value in each dimension (x, y) that doesn't change over the time. These values are used to forbid any candidate solution (particle) to go far away from the search space. The forbidden process is done using the following constraint.

Set $center^d$ the center of the frame in each dimension

If $x_i^d(t+1) >= x_{max}^d$ OR $x_i^d(t+1) <= x_{min}^d$ then

$x_i^d(t+1) = center^d$

End If

Pseudo code 3.1 *Constraint to keep the particle in the search space*

3.1.2 Defining the Goal

In order to track any desired object (goal), we must pre-define it to the system. One of the objectives of this system is general purpose object tracking, which means that object is user defined, this object may be an image of a car, a human head, bacteria … etc, it is free to the user to select it regardless of its shape, color, motion. The user can select the goal from the scene (initial frame) using the mouse defining a rectangle shape over his desired object. This rectangle is called region of interest (ROI). This ROI will be used to determine the part of the image by storing the center (x, y), width, and height of the ROI as shown in Figure 3-1.

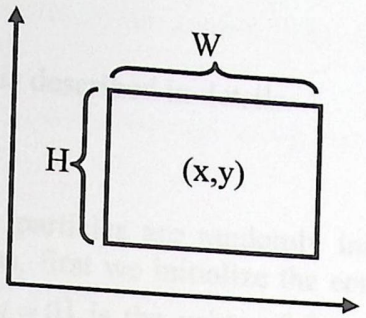


Figure 3-1 Shape of the goal

3.1.3 A Swarm of Interacting Individuals

In order to reach the goal in the search space, PSO algorithm assumes the existence of interacting individuals, which are called particles, each of them represents a candidate solution in the search space.

Since we deal with frames (images) we represent each particle, in the same way as the goal, as a window (rectangle shape) with four dimensions, so each particle has center (X, Y), width, and height. This window looks like the shape of the goal.

Simply, this rectangle is a part of the search space (frame), if we have p particles, $P_i = \{X, Y, w, h\}$, $i \in p$. We will have the same number of rectangles that form different parts of the search space.

The width and height (W, H) of the window are called the size of the window (size of the particle), while the position (X, Y) is called the center of the particle.

One virtue of the PSO algorithm is the set of individual and group behavior rules as we discuss in section 2.1. These rules are achieved by keeping the best particle of the swarm, we call its center as the best center of the swarm $gBestPos_i^d(t)$ at iteration t.

Also each particle keeps its best center $pBestPos_i^d(t)$ with respect to the goal that has it at iteration t.

Another benefit to the PSO algorithm is its suitability for the high dimensional space, our system is four dimensions space these dimensions are: the center and the size of the particle $P_i = \{X, Y, w, h\}$.

3.1.4 Phases of the PSO in Tracking Process

There are four phases of PSO-based tracking algorithm:

1. Initialization phase.
2. Evaluate phase.
3. Predict phase.

4. Resample phase.

In the following, each phase is described in details.

1. Initialization Phase

During this phase all the particles are randomly initialized to cover the search space in all the dimensions, first we initialize the center of all the particles using Equation 2.1. Where $x_i^d(t=0)$ is the value of the center of the particle in each dimension (X,Y), $x_{\max}^d$ is the maximum value in each dimension of the search space, $x_{\min}^d$ is the minimum value in each dimension of the search space. *rand* is the random number between [0,1], it is used to make the initialization random in order to cover the search space. By using Equation 2.1, all the particles are distributed.

The size of the particles is initialized after initializing the center of the particles, by applying the following constraint:

If $x_{\max}^d - x_i^d(t=0)$ less than $x_i^d(t=0) - x_{\min}^d$

Size= rand * $x_{\max}^d - x_i^d(t=0)$

Else

Size =rand* $x_i^d(t=0) - x_{\min}^d$

Pseudo code 3.2 Constraint for initialization the size of the particle

According to the velocity, we assume each particle stars at velocity equal to zero, we initialize the velocity of each particle using equation 2.2. Where $v_i^d(t=0)$ is the velocity of the i^{th} particle in all the dimensions.

After initializing the center and the size of each particle, we cover the search space with particles, each one with four dimensions space (window shape). We assume each one is a candidate solution.

2. Evaluation Phase

Evaluation phase is used to evaluate the current position of each particle with respect to the goal. How far each particle from the goal? What is the best particle of the swarm (global best)? And what is the best position the particle has at the current frame (local best)? All these questions are answered after each evaluation phase. We evaluate in order to know the best particle and keep its position to be

broadcasted to all the particles in the swarm. Also for each particle, its local best position is kept at this time.

The three questions above are evaluated using the fitness function that will be explained in Section 3.1.5.

3. Predict Phase

In order to get closer to the desired object, the velocity of each particle is updated using Equation (2.7)

Where $v_i^d(t+1)$ is the new velocity in all the dimensions, $v_i^d(t)$ is the previous velocity at iteration t, $pBestPos_i^d(t)$ is the best position for the corresponding particle at iteration t in all the dimensions, $gBestPos_i^d(t)$ is the position of the best particle in the swarm, and $x_i^d(t)$ is the last previous position of particle i.

The position of the particle in all the dimensions is updated using Equation 2.8, where $v_i^d(t+1)$ and $x_i^d(t)$ as discussed in previous velocity equation.

The predict phase is done in each iteration t in each frame.

4. Resample Phase

Instead of making initialization phase in each frame (which lead to lose the previous positions of the particles in the frame), we resample the particles in the search space for next frame so as to maintain new position near the previous one.

This is very important in the object tracking because the new poison of the moving object is assumed not far from its previous position. So if we re-initialize the particles we will lose this information that enables us to track the object.

In the resample phase the center of the particles are randomly moved a little bit far away from the previous position, and the velocity of the particle for the next frame starts with zero.

According to the position of the particle, we update the center of each particle using Equation 2.1. After calculating the new position we apply the following constraint:

$$x_{current} = x_i^d(t) = x_{\max}^d + \left(x_{\max}^d - x_{\min}^d\right) \cdot rand$$

$$x_{prev} = x_i^d(t-1)$$

$$\Delta x = x_{current} - x_{prev}$$

$$x_{current} = \begin{cases} x_{prev} - \varepsilon & , \quad \Delta x \leq -\varepsilon \\ x_{prev} + \varepsilon & , \quad \Delta x > \varepsilon \\ x_{current} & , \quad otherwise \end{cases}$$

Pseudo code 3.3 Constraint for the position resample phase

Where $x_{current}$ is the new position of the particle in the current frame, which must be between $[-\varepsilon, \varepsilon]$.

3.1.5 Fitness Function

The fitness function is used to evaluate the current particle's position, the input of this function is a vector with four dimensions and the output is a value representing the distance of the particle's histogram from the goal's histogram.

Each frame in the video is converted from RGB color model into HSV color space. Hue component of the HSV image is used to calculate the histogram for both the desired object and each particle. This histogram is not extracted from the whole frame, but from the region in which the corresponding particle lies on it. In order to calculate the similarity between histograms of the particle and goal, we use Bhattacharya coefficient distance which is defined by Equation (2.12) [10]. Where D is the distances between the goal and each particle, it ranges between [0, 1], H^A represents the particle's histogram, H^B represents the goal's histogram and i represents the number of bins.

Figure 3-2 shows the fitness values D at time t=1. The best particle is 39th one; because its fitness value is the lowest while worst particle is 49th one; because its fitness value is the highest.

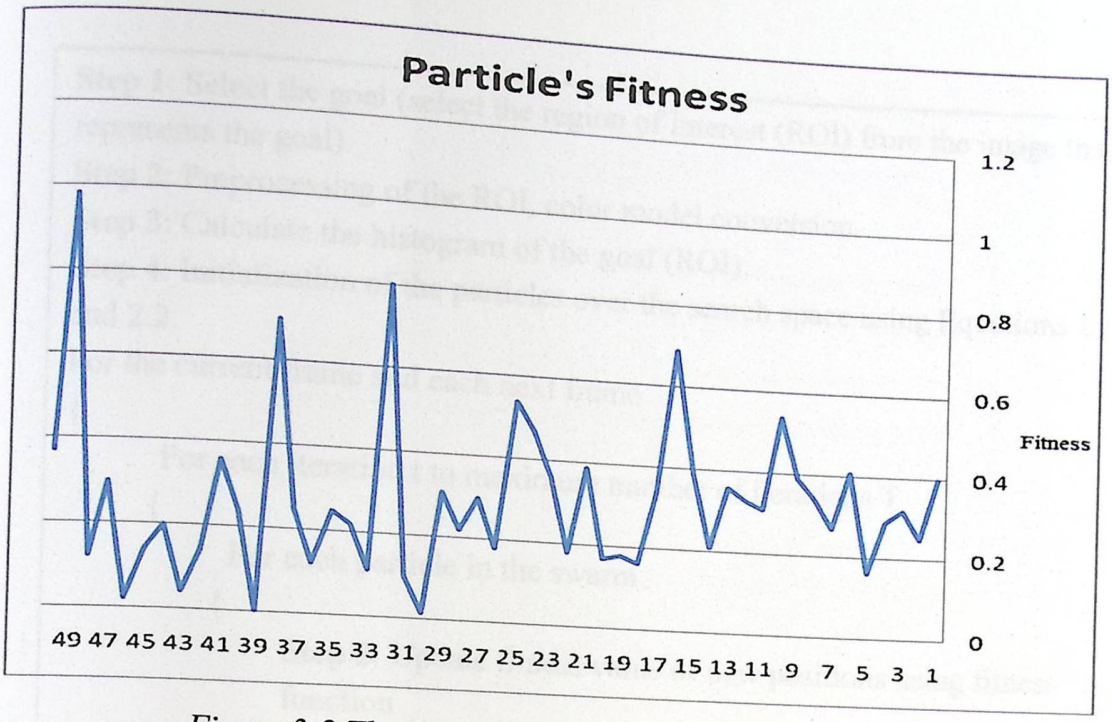


Figure 3-2 The fitness value for each particle at time t

3.2 Summary of PSO Algorithm for Object Tracking

The steps that are required to reach the goal begin with selecting the goal from the frame. The frame is preprocessed by converting it from RGB to HSV color model. The Hue component is maintained. After that the histogram of the goal is calculated, from the Hue component, which is one dimensional histogram.

In order to reach the goal perfectly, all the particles are randomly initialized to cover all the search space using Equation 2.1, the velocity is initialized using Equation 2.2. Afterward, the fitness value is determined for each particle, in each iteration in the same frame, and the best fitness value the particle has will determine the particle's best position $pBestPos_i^d(t)$. The best fitness value in the swarm will determine the global best position $gBestPos^d(t)$.

After that, these best positions are used in the velocity update equation (see Equation (2.7)); the updated velocity is used to update the particle's position as in Equations (2.8). The evaluation and predict process occur in each iteration. Finally, after reaching the predefined number of iterations, the particles are resampled for the next frame; where new position will be randomly shift a little from the previous one. See pseudo code 3.4.

Step 1: Select the goal (select the region of interest (ROI) from the image that represents the goal).
Step 2: Preprocessing of the ROI, color model conversion.
Step 3: Calculate the histogram of the goal (ROI).
Step 4: Initialization of the particles over the search space using Equations 2.1 and 2.2.
For the current frame and each next frame
{
    For each iteration t to maximum number of iterations T
    {
        For each particle in the swarm
        {
            Step 5: Update fitness value of new positions using fitness function

            Step 6: Calculate the velocity and the position in each dimensions using Equations 2.2 and 2.4
        } // end for
    } // end for
Step 7: Resample the particles around the goal for the next frame.
}

Pseudo code 3.4 Algorithm steps

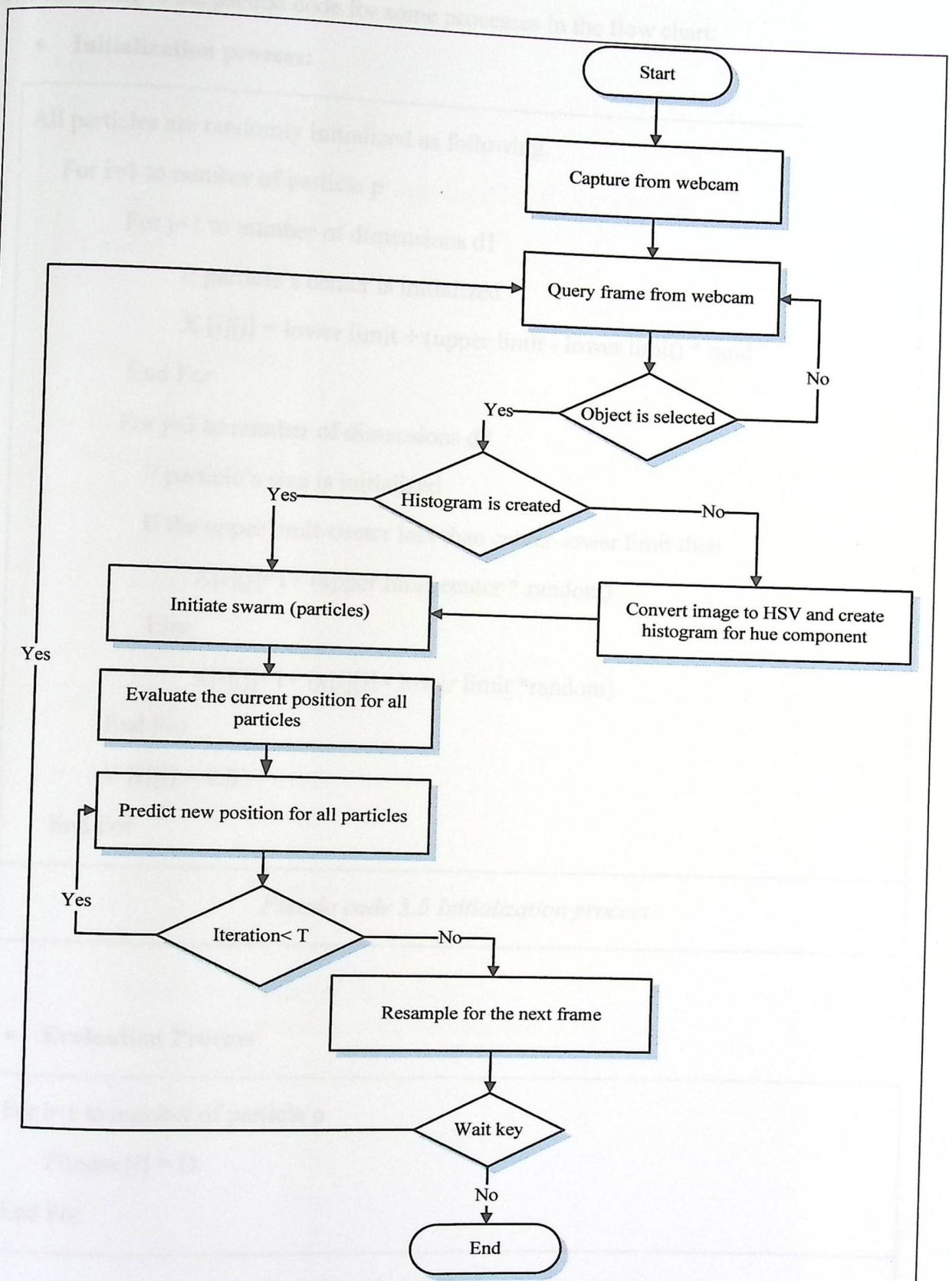


Figure 3-3 General overview of overall the system (flowchart of PSO algorithm for tracking)

The following is the pseudo code for some processes in the flow chart:

- **Initialization process:**

```
All particles are randomly initialized as following.
  For i=1 to number of particle p
      For j=1 to number of dimensions d1
          // particle's center is initialized
          X [i][j] = lower limit + (upper limit - lower limit) * rand
      End For
      For j=3 to number of dimensions d2
        // particle's size is initialized
        If the upper limit-center less than center-lower limit then
            X[i][j]=1+ (upper limit-center * random)
        Else
            X[i][j]=1+ (x[i][j] - lower limit *random)
      End For
      V [i][j] = 0.0
  End For
```

Pseudo code 3.5 Initialization process

- **Evaluation Process**

```
For i=1 to number of particle p
    Fitness [i] = D
End For
```

Pseudo code 3.6 Evaluation process

Where D is the Bhattacharyya coefficient distance as in Equation 2.12, and it is calculated as following:

```
// finding  the Bhattacharyya coefficient value

Set Sum=0
For i=1 to number of particle p
    For n=0 to number of bins
        Sum= Square (Particle Histogram[i]* goal histogram[i])
        BC[i]=BC[i]+ Sum
    End For
End For

//Finding the distances D

For i=1 to number of particle p
    Fitness[i]=Square(1-BC[i])
End For
```

Pseudo code 3.8 Calculating Bhattacharyya coefficient distance D

The following is the pseudo code to determine the global best position in the swarm.

```
//find global best

For i=1 to number of particle p
    If  bestFit < gBestFit[i]

        //keep best fitness in the swarm
        bestFit=  gBestFit[i]

        For j=1 to number of dimensions d
          // store last position as the best position in the swarm
          pBestPos [i][j] -=X [i][j]
        End For

    End If
End for
```

Pseudo code 3.9 Calculating global best fitness value in the swarm

The following is the pseudo code to determine the local best position for each particle.

```
//find local best
For i=1 to number of particle p
    If  bestFit < pBestFit[i]
        //keep best fitness in the swarm
        bestFit=  pBestFit[i]
        For j=1 to number of dimensions d
         // store last position as the best local position
          pBestPos [i][j] =X [i][j]
        End For
    End If
End for
```

Pseudo code 3.10 Calculating local best fitness value for each particle

- **Predict process:**

```
For i=1 to number of particle p
    For j=1 to number of dimensions d
        V [i][j] = w * V [i][j]
                + c1 * rand1 * ( pBestPos [i][j] - X [i][j] )
                + c2 * rand2 * (gBestPos [j]-X [i][j])
        X[i][j] = X[i][j] + V[i][j]
    End For
End For
```

Pseudo code 3.7 Predict process

- **Resample process:**

```
For i=1 to number of particle p
    // center resampling
    For j=1 to number of dimensions d1
        New Center = lower limit
```

```
                + (upper limit - lower limit) * random
    // New Center constraint
    If (New Center - x[i][j]) not between predetermined range then
        X[i][j]=x[i][j] + predetermined value
     Else
        X[i][j]=New Center
End For
 // size resampling
For j=3 to number of dimensions d2
    If the upper limit-center less than center-lower limit then
        New Size =1+ (upper limit-center * random)
    Else
        New Size =1+ (x[i][j] - lower limit *random)
          // perform the same constraints for the New Size as in the center
          constraints.
End For
End For
```

Pseudo code 3.7 Resample process

3.3 Summary

In this chapter we discussed the PSO algorithm in terms of object tracking, we talked about the fitness function, the formulas that were used in the proposed algorithm. We depicted the flowchart of overall the system to understand the project when it becomes real.

Chapter 4
Experiments and Results

In this chapter we introduce the experiments that we conducted, and the results for each experiment, in order to show how the algorithm is sufficient to perform real time object tracking and to prove the objectives of our project that are mentioned in Chapter 1. We divide this chapter in to two sections. After we test the environment and the platform of the project in the first section, in the second section we will test the captured images, the conversion of the color models, and the histogram of the images. Then we will test the PSO algorithm; the initialization, optimization, convergence of the particles to the goal, and finally we will test the tracking process using PSO algorithm either inside indoor environment or outside environment under different conditions.

4.1 Testing Environment

The following is the experiments for the hardware and software.

4.1.1 Hardware and Software Specification

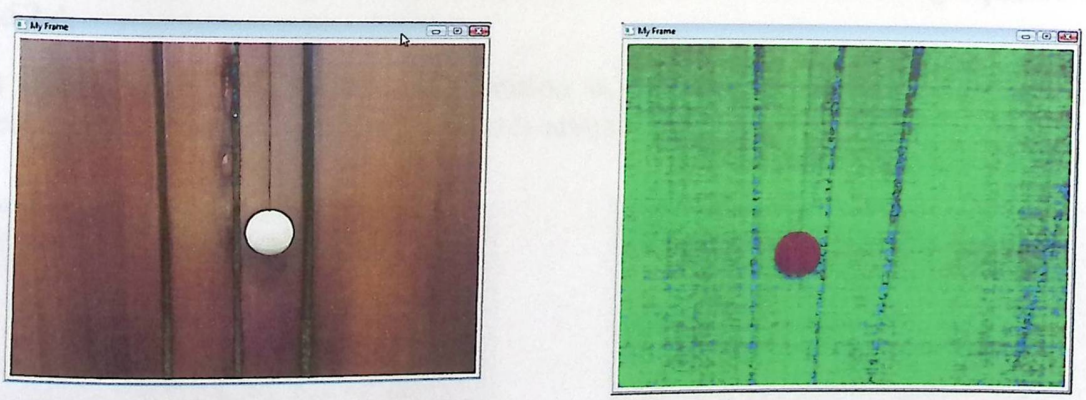
The system is written using VS C++ SDK v 2005 and open source computer vision library (OpenCV v 1.1pre1a). It runs under windows platform. A webcam with 30 frames per second and 640 by 480 pixels is used.

4.2 Experiments and Results

The following sections are the experiments and results for images, PSO algorithm and tracking process.

4.2.1 Image Experiments and Results.

The object is modeled based on its color. Firstly the RGB frame is converted into HSV color model. The hue component of the HSV frame is used to calculate the histogram; it is sufficient due to its invariance to the illumination [6].

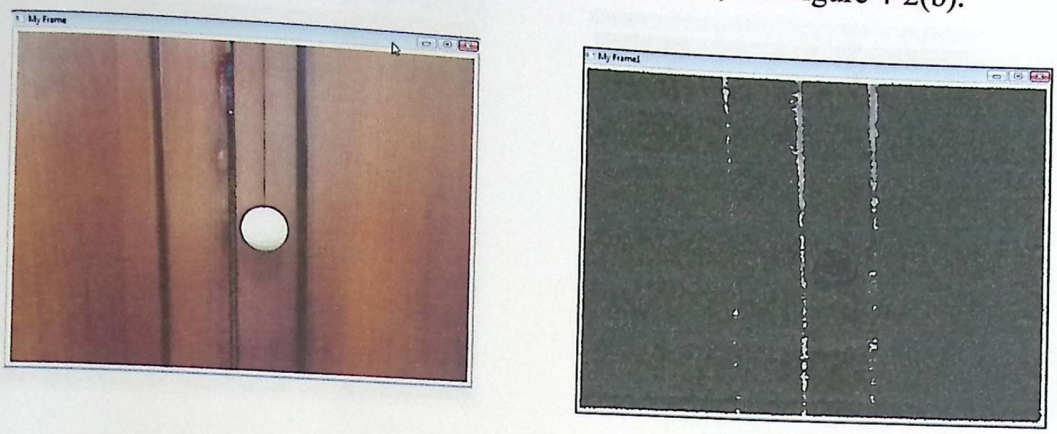


a: RGB Image *b:HSV Image*

Figure 4-1 Color Models

The image in Figure 4-1(a) is the captured image from the webcam; it is in the RGB color model. While the image in Figure 4-1(b) is the captured image after it is

converted to the HSV color model. This is done to calculate the histogram of the selected goal and each particle using the hue component, see Figure 4-2(b).



a: RGB Image

b: *Hue component*

Figure 4-2 Hue component

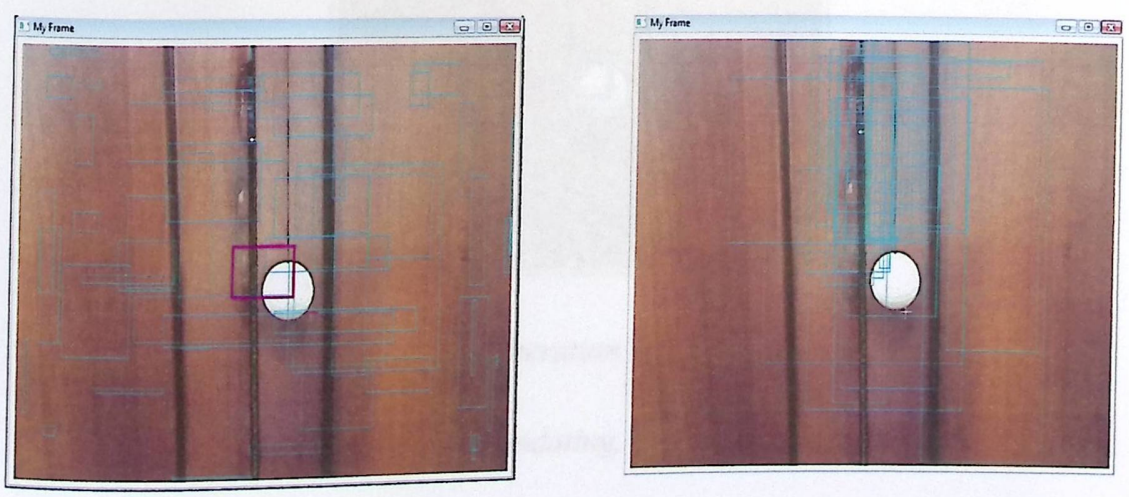
Figure 4-2(b) shows the hue component of the image in the Figure 4-1(b).

After we test the conversion of the images and the images preprocessing, the next section is test for the PSO algorithm and the best values of its parameters.

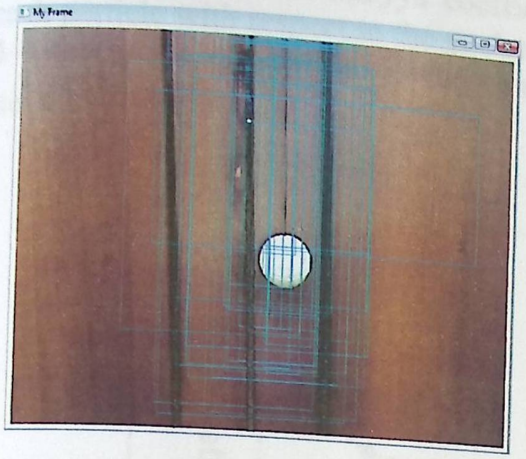
4.2.2 Particle Swarm Optimization Experiments and Results

Initially all the particles are randomly initialized to cover the search space, then all the particles will move until they converge. This is done within 7 iterations. In tracking process 5 to 7 is sufficient [6], in our project 7 iterations is sufficient in each frame to converge to the goal. The Figure 4-3 shows the PSO phases, the random initialization is done in iteration zero using Equation 2. 1 and 2. 2, from iteration 1 to iteration 7 the particles positions are updated in order to reach the using Equation 2.3 and 2.4.

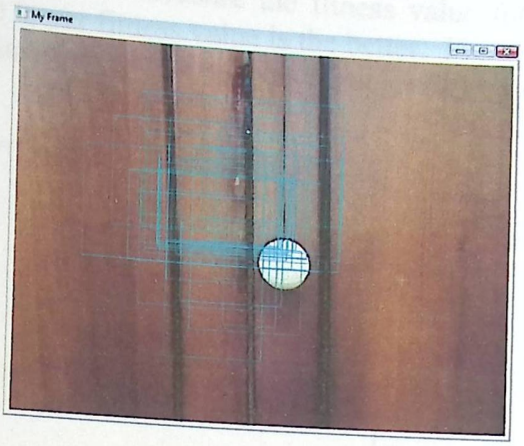
As we noticed, the termination condition in order to reach the goal is reaching predefined number of iterations, which is seven iterations.



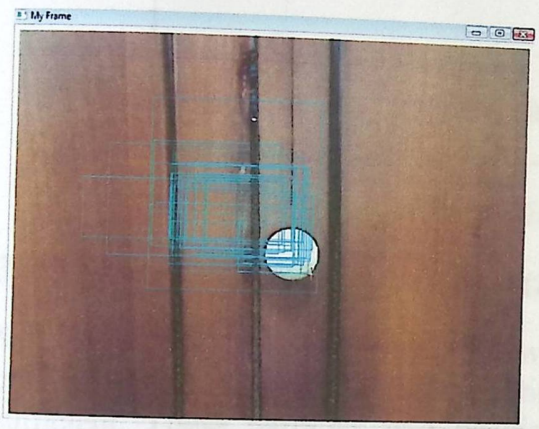
Initialization phase



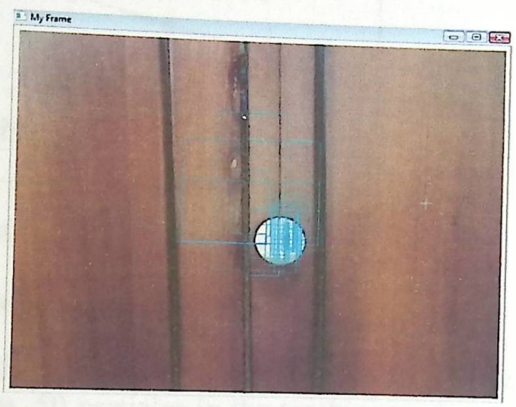
Iteration 2



Iteration 3

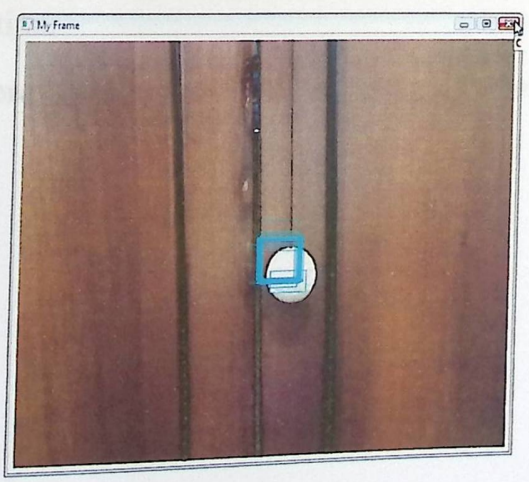


Iteration 4



Iteration 5

Iteration 6



Iteration 7

Figure 4-3 the initialization, updating, convergence steps in PSO

In minimization optimization, we keep the minimum value, as we mentioned in section 3.3, the Bhattacharyya coefficient is used to measure the fitness value for each particle with respect to the goal, and the lower fitness value is the better. Figure 4-4 shows the decreasing of the average fitness value of all the particles in each iteration in one frame.

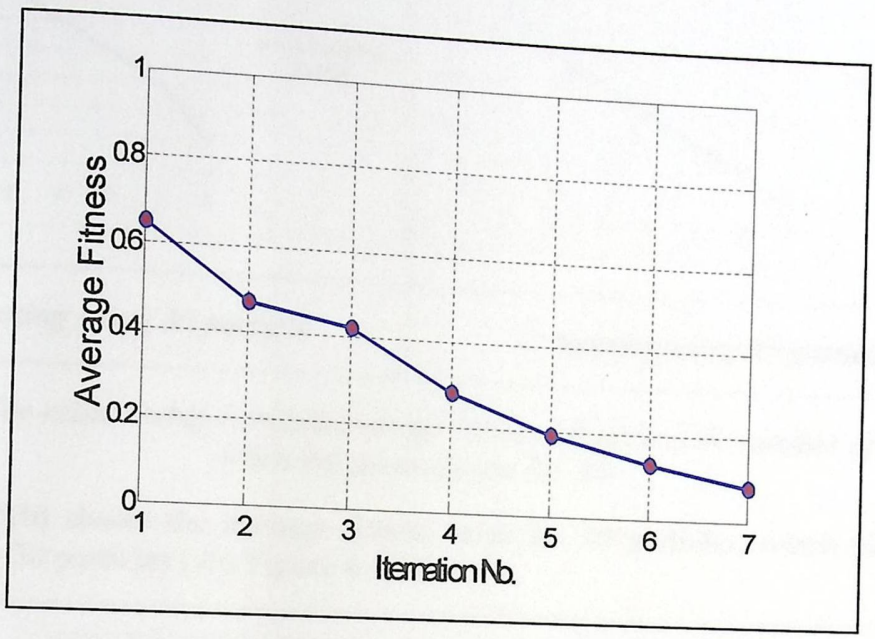
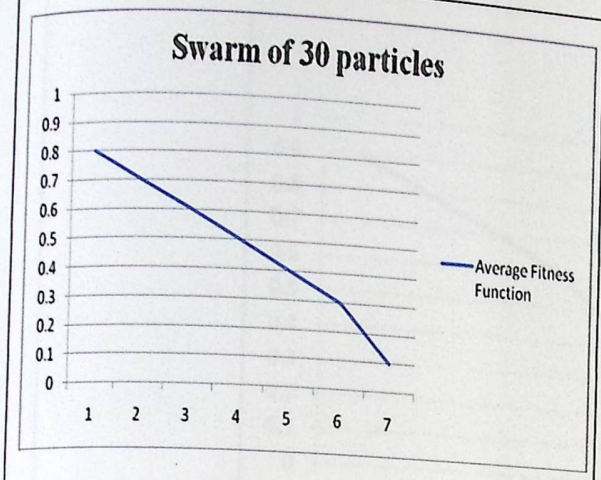


Figure 4-4 the average fitness values in each iteration

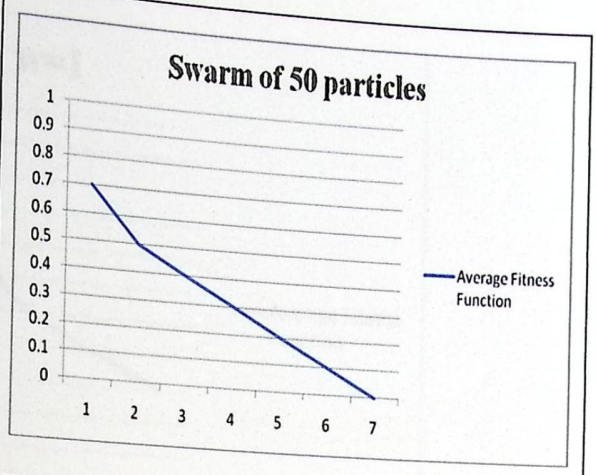
4.2.3 PSO Tracking Based Algorithm Parameters

We tune the PSO parameters until we get the real time system. Beacause of the communication between particles in the swarm, 50 particles in the tracking system is sufficient. These particles will be distributed in frame of 640 by480 pixels. The stopping criterion in each frame in video sequences is reaching the predefined number of iterations, this is sufficient to converge to the goal.The constants of the formula C1, C2 are set to 2. The inertia factor or w is set to less than one then is decreased in each iteration.

The following figures prove the previous values of the parameters.



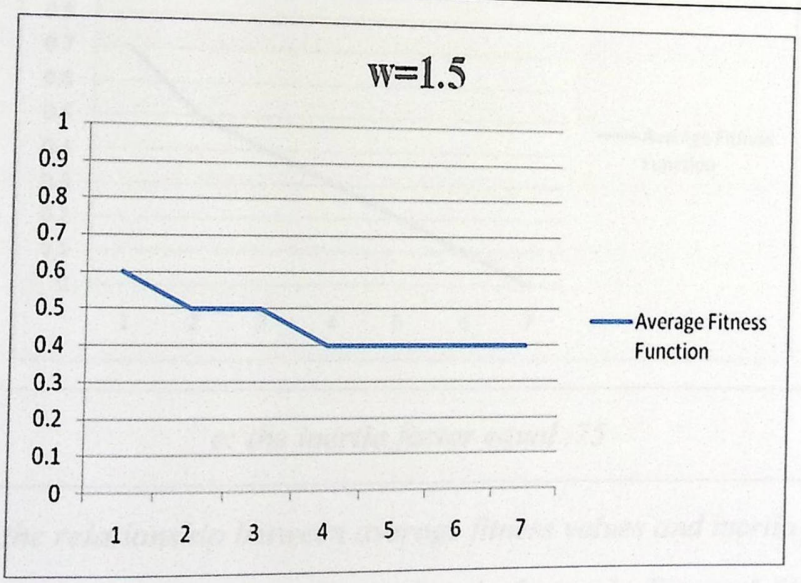
a: tracking using 30 particle



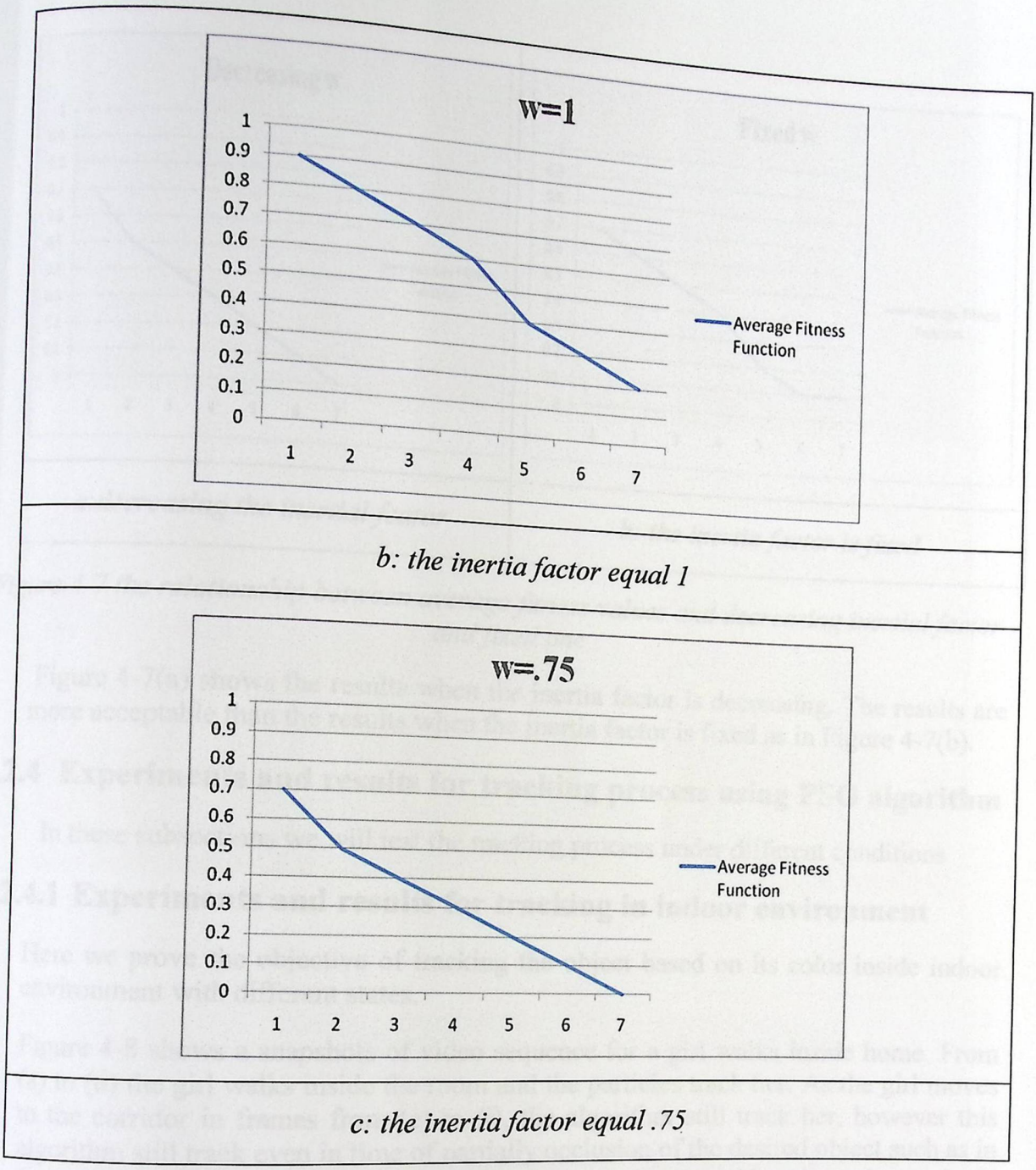
b: tracking using 50 particle

Figure 4-5 The relationship between average fitness values and the number of iteration when the particles are 30, 50.

Figure 4-5(b) shows the average fitness value for 50 particles, which gives better result than 30 particles (see Figure 4-5(a)).



a: the inertia factor equal 1.5

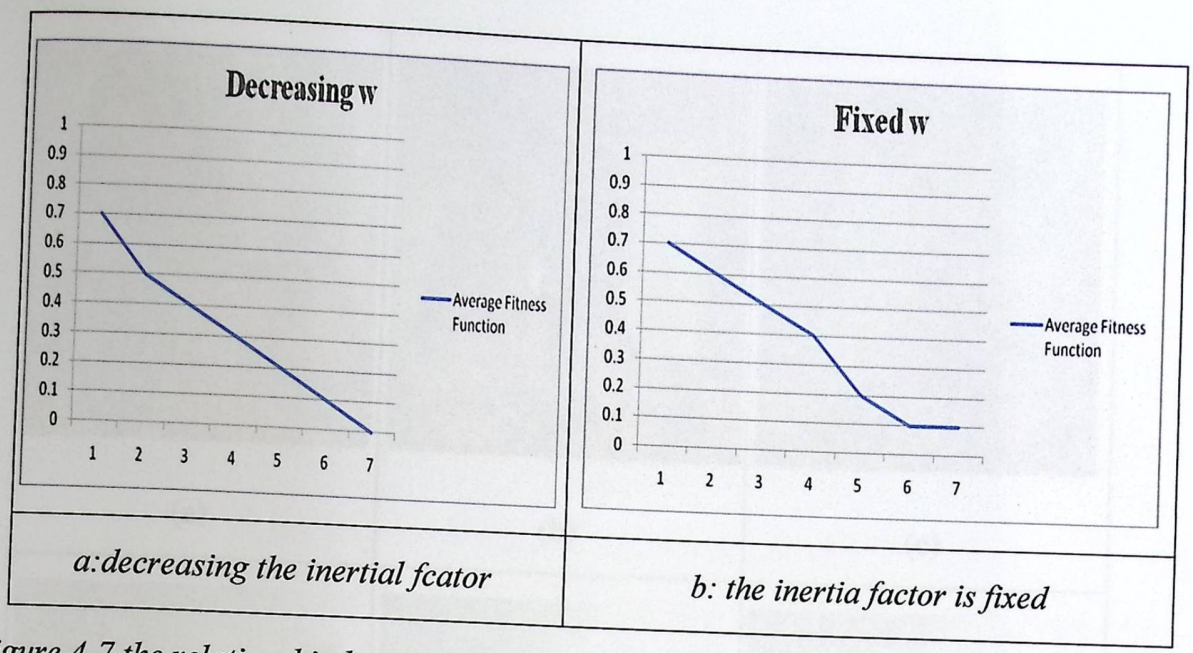


b: the inertia factor equal 1

c: the inertia factor equal .75

Figure 4-6 the relationship between average fitness values and inertia factor

Figure 4-6 shows different values for the inertia factor. In Figure 4-6(a) shows the results when w equal to 1.5. The results are not acceptable because the average fitness value is greater than 0. In Figure 4-6(b) shows the result when the inertia factor equal 1. This experiment gives better results than the previous one, but the average fitness value still greater than 0. Finally the inertia factor is set to .75, which gives the best result; the average fitness value equal zero.



a:decreasing the inertial fcator

b: the inertia factor is fixed

Figure 4-7 the relationship between average fitness values and decreasing inertial factor and fixed one

Figure 4-7(a) shows the results when the inertia factor is decreasing. The results are more acceptable than the results when the inertia factor is fixed as in Figure 4-7(b).

4.2.4 Experiments and results for tracking process using PSO algorithm

In these subsections we will test the tracking process under different conditions

4.2.4.1 Experiments and results for tracking in indoor environment

Here we prove the objective of tracking the object based on its color inside indoor environment with different states.

Figure 4-8 shows a snapshots of video sequence for a girl walks inside home. From (a) to (d) the girl walks inside the room and the particles track her. As the girl moves to the corridor in frames from (e) to (i), the algorithm still track her, however this algorithm still track even in time of partially occlusion of the desired object such as in (h) and (i). In frames (g) she comes back to the room the algorithm detect her and still track her in (k) and (l). In the state of covering the search space the algorithm track her also as in (l).

(a)

(b)

(c)

(d)

(e)

(f)

(g)

(h)

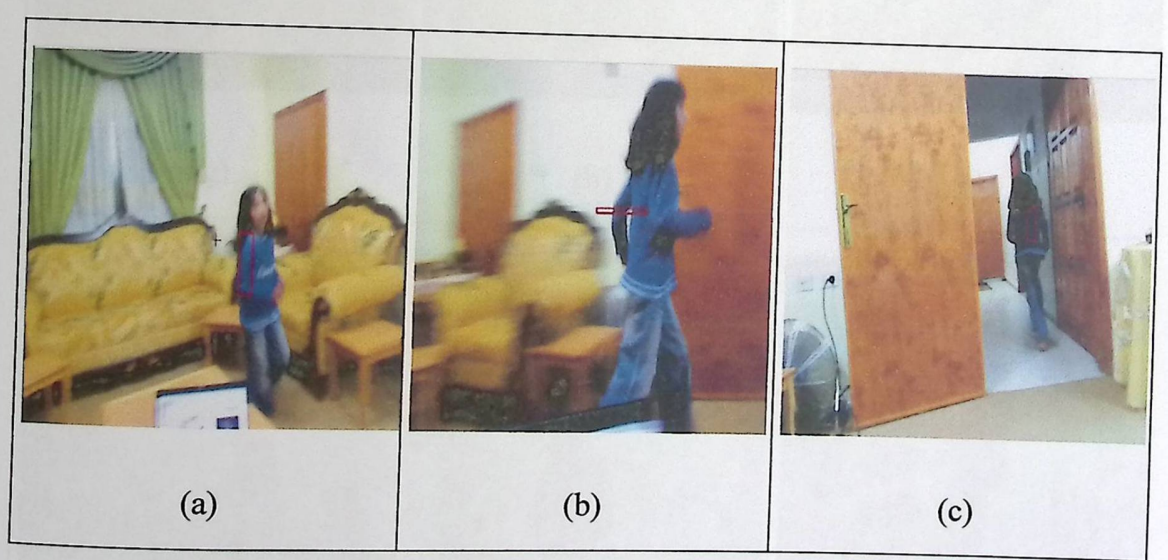
(i)

(j) (k) (l)

tracking inside indoor environment8-4Figure

In the Figure 4-9 another snapshots of video sequence inside indoor environment for a girl runs. Tracking a running object is not easy as a walking one; because the algorithm must track even if the object moves too quickly. Our algorithm is succeeded to track a running object. From (a) to (f) the girl runs and the algorithm is succeeded to track her even when she goes to the corridor far away from the camera like in (d), and also the algorithm detect her when she come back to the room in (e) and (f).

The algorithm is still succeeded when the object is partially occluded as in (g) and (h).



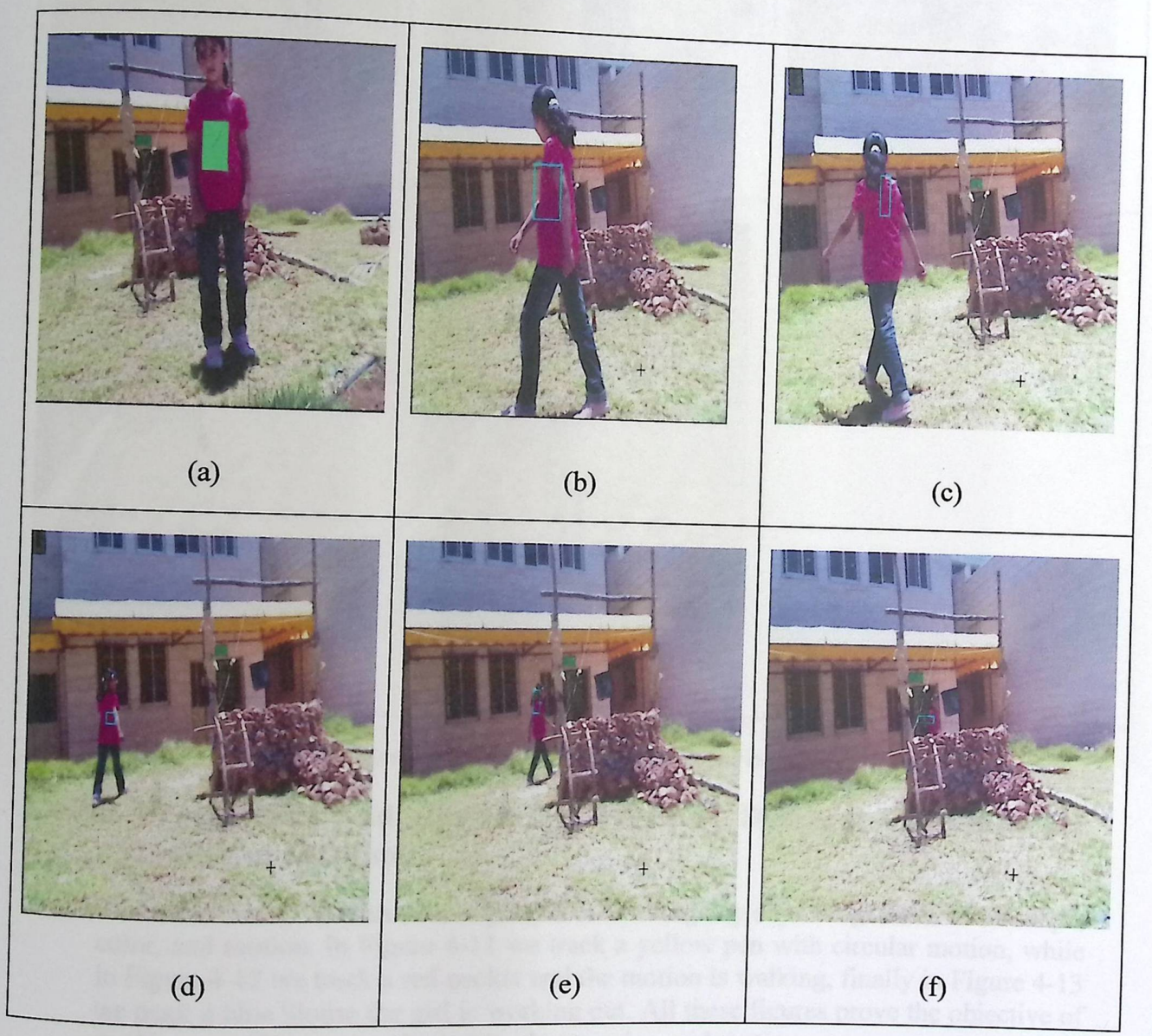
(a) (b) (c)

Figure 4-9 tracking inside indoor environment in running state

4.2.5 Experiments and results for tracking outdoor environment

Tracking outdoor environment is not easy as indoor s; because of the illumination changing at the same place, and because of buildings, trees etc. In the following we prove the objective of tracking object in outdoor environment.

Figure 4-10 is snapshots of video sequence that is captured outdoor. As we notice the success of the algorithm for tracking outside even the partially occlusion of the girl like in (f).



(a) (b) (c)

(d) (e) (f)

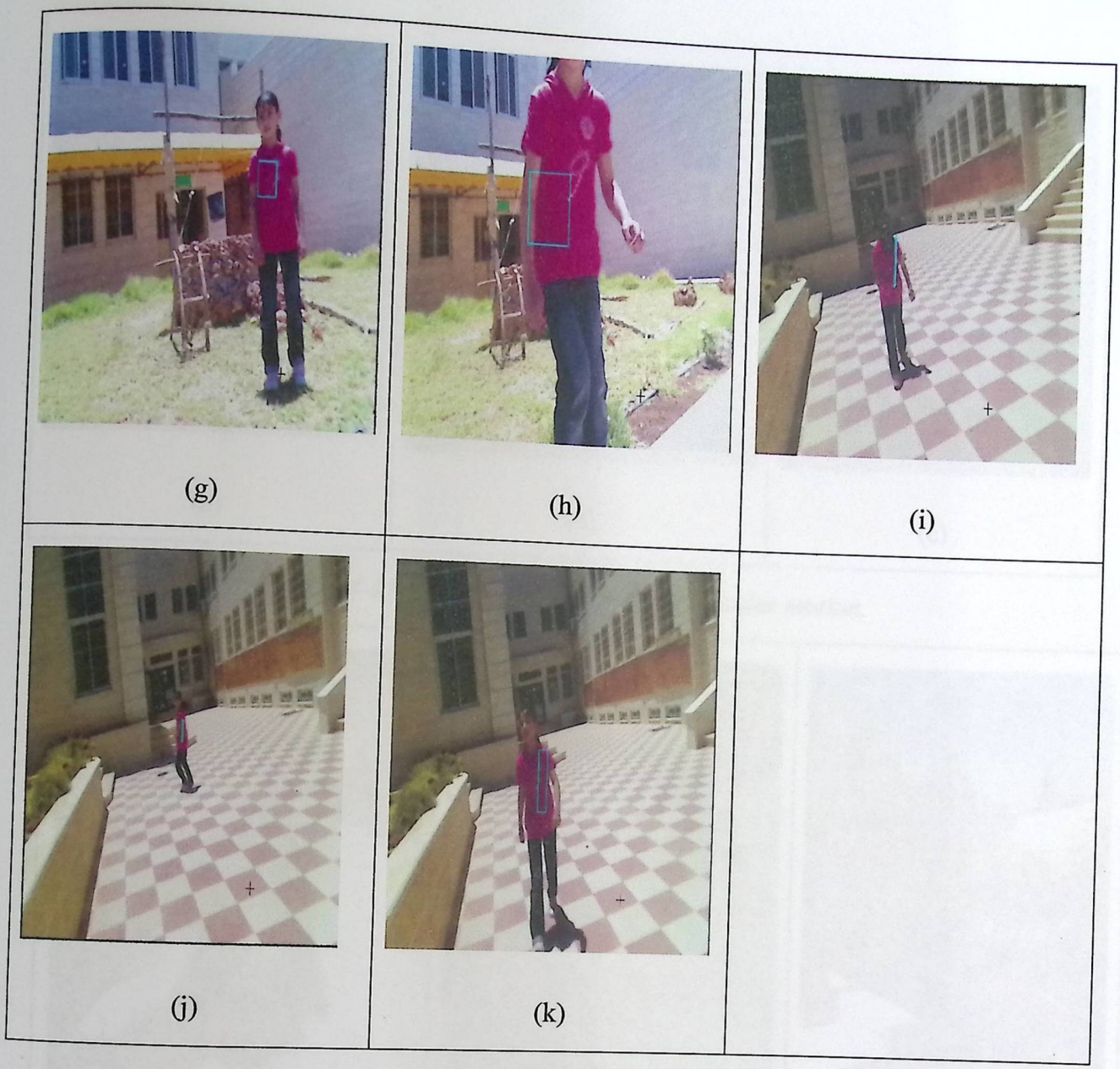


Figure 4-10 Tracking Outdoor Environment

4.2.6 Experiments and Results for Tracking Different Objects either Indoor or Outdoor.

In the following we prove the objective of tracking any object regardless of its shape, color, and motion. In Figure 4-11 we track a yellow pen with circular motion, while in Figure 4-12 we track a red packet and the motion is walking, finally in Figure 4-13 we track a blue blouse the girl is working out. All these figures prove the objective of tracking any object regardless of its shape, color, and motion.

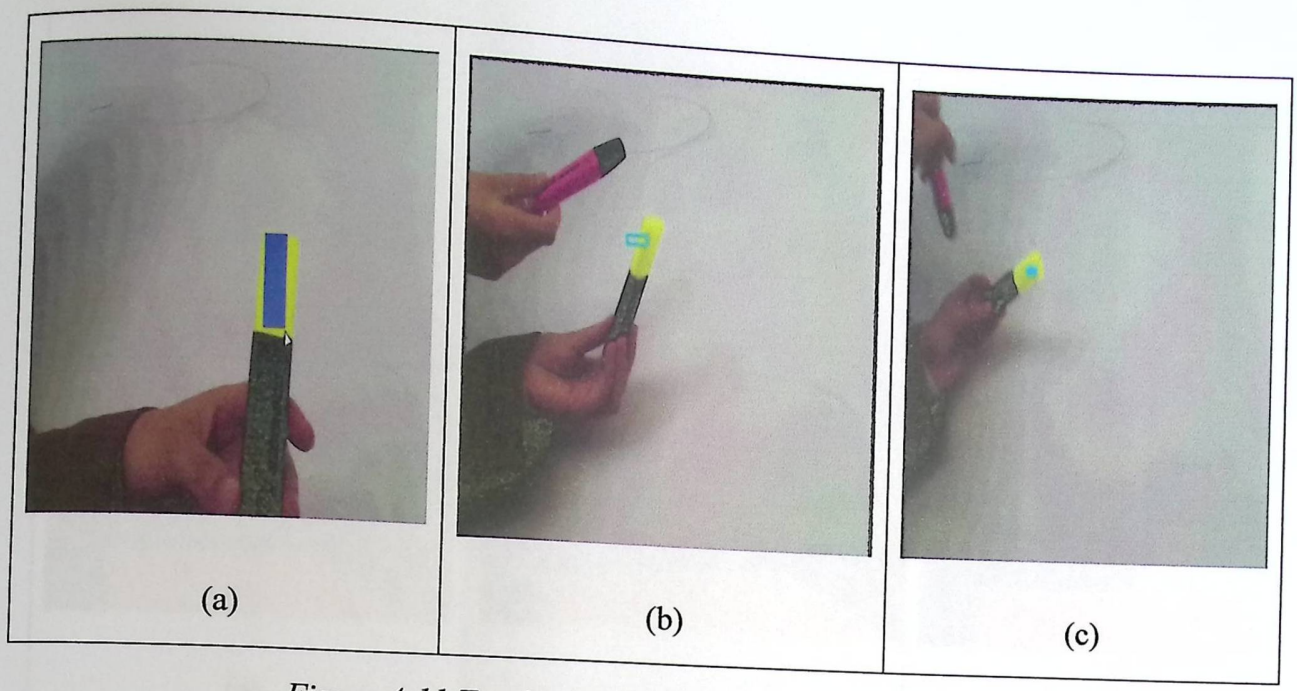


Figure 4-11 Tracking Yellow pen and circular motion.

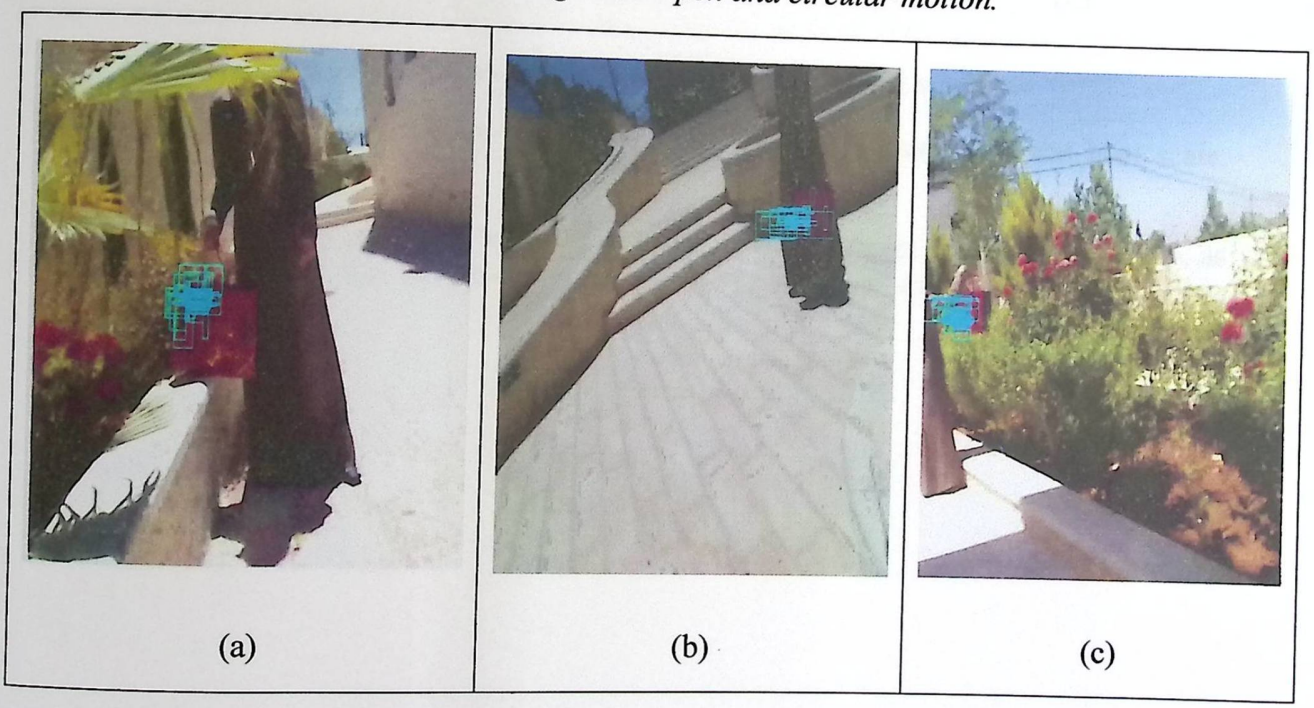


Figure 4-12 Tracking red packet.

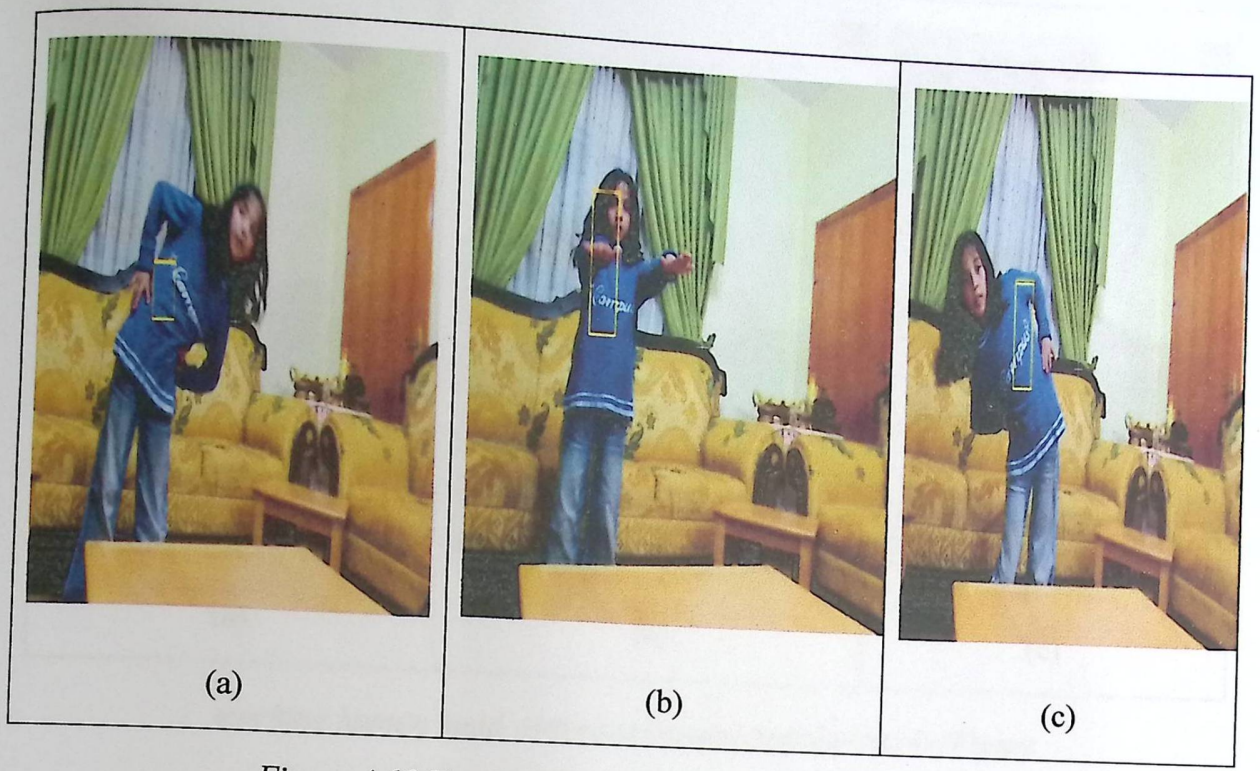
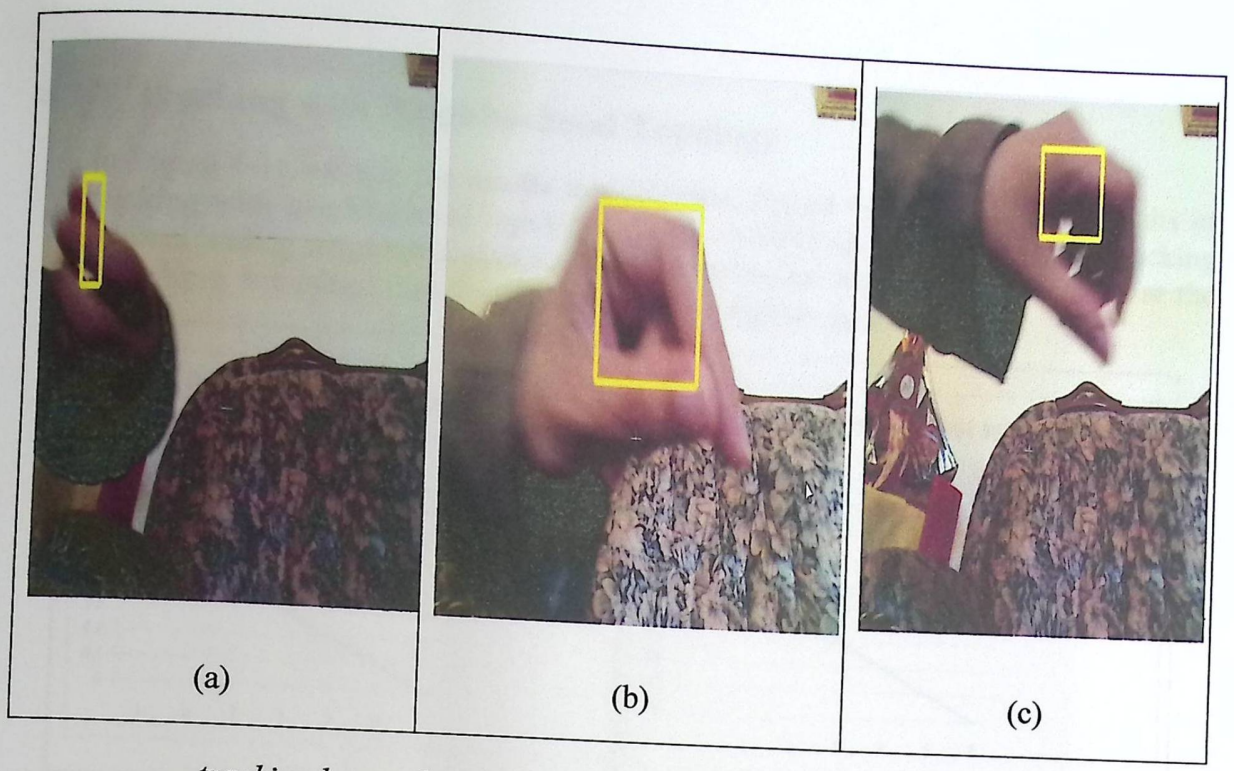


Figure 4-13 Tracking blue blouse and right, left motion

The following figures prove the objective: tracking is invariant to the goal scale, translation, rotation.

In Figure 4-14 we prove the tracking is invariant to the object scaling and rotation, as we can see how the particles still converge to the goal even when is scaled or translated.



tracking human hand with rotation and translation14-4Figure

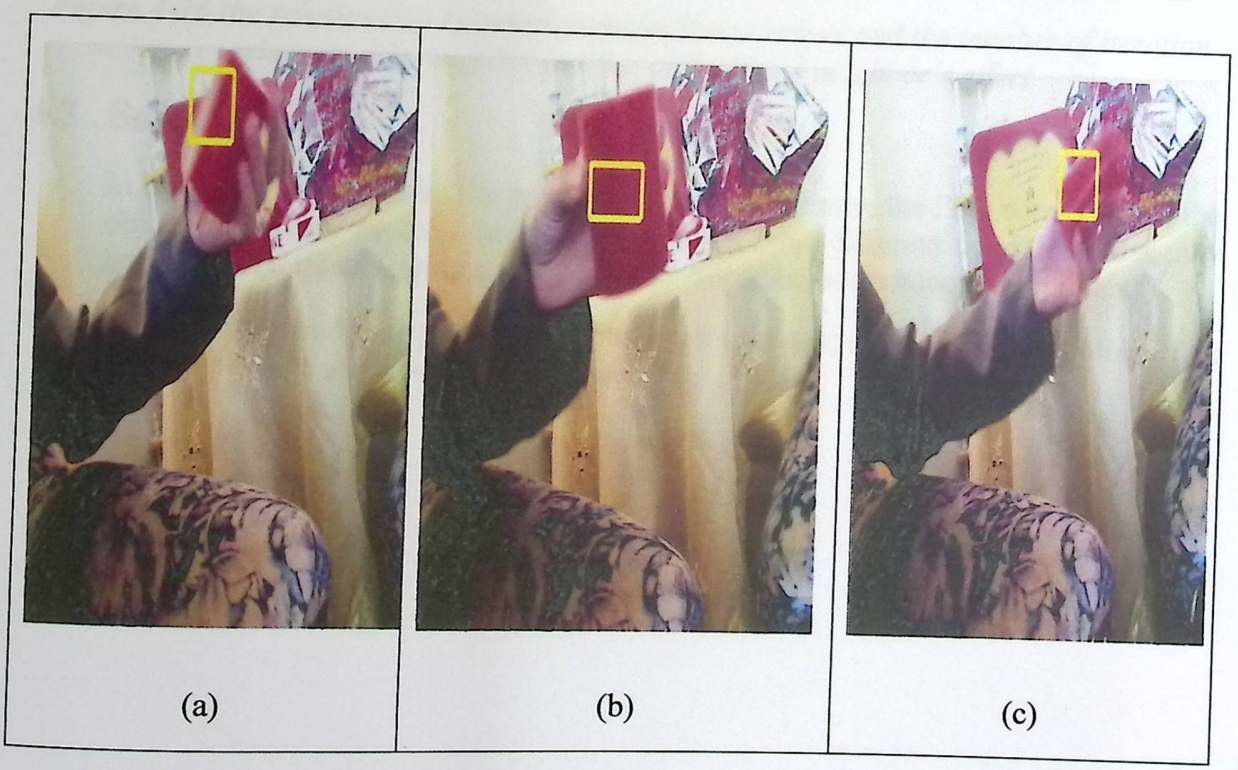
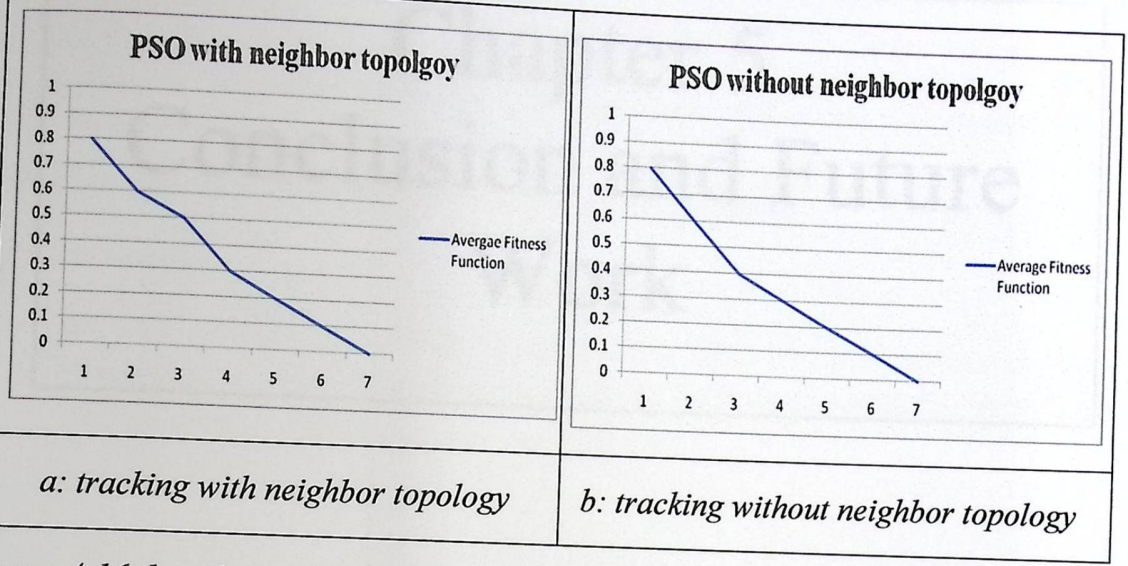


Figure 4-15 Tracking candle with rotation and translation

In Figure 4-15 we prove the tracking is invariant to the object rotation how the particles still converge even of the object rotation.

4.2.7 Tracking with Neighborhood Topology

In Figure 4-16 we test the results in two cases, Figure 4-16(a) shows the results of tracking with neighborhood topology. Figure 4-16(b) shows the results of tracking without adding neighbor's effects. These experiments were done at the same at the same time. We notice that there is no effect two figures seem the same.



a: tracking with neighbor topology | *b: tracking without neighbor topology*

Figure 4-16 the relationship between average fitness values and the number of iteration when we add neighbor's effect and without neighbor's effect

4.3 Summary

In this chapter we proved the preprocessing of the frames, the PSO convergence, the objectives of our project, and also we proved other different conditions that may happen to the tracking process. We tracked inside indoor outdoor environment.

Finally, we tested the system with additional topology; neighborhood topology, this topology didn't add new value to the tracking process.

Chapter 5
Conclusion and Future Work

Conclusion

In this project we built a system for general purpose real time object tracking using particle swarm optimization inside indoor environment and outdoor environment. The system is succeeded to achieve all the objectives. It can track the goal in case of problems, the scaling, translating, rotating, partially and fully occlusion of the desired object.

We tuned the PSO algorithm parameters, we found that inertia factor started with less than one then decreasing it gave efficient results. The constancies are set to two, the number of particles is set to 50 particles, and the swarm of particles converges to the goal within seven iterations.

Using PSO algorithm gave efficient results in tracking systems.

Future work

- We can enhance the system to build any object tracking application such as surveillance system, an application that gives information about the desired object.
- We can track multiple objects at the same time.

References

[1] Wikipedia Encyclopedia, Computer Vision, last modified on 16 June 2009 at 02:37, < http://en.wikipedia.org/wiki/Computer_vision >

[2] J.Kennedy, R.Eberhart, and Y.Shi. (2001). Swarm Intelligence. Academic Press.

[3] Swarm intelligence tutorial, < http://www.swarmintelligence.org> 10/6/2009

[4] J. Kennedy, and R. Eberhart, *Particle swarm optimization*, in Proc. of the IEEE Int. Conf. on Neural Networks, Piscataway, NJ, pp. 1942–1948, 1995.

[5] J.Izquierdo, I.Montalvo, R.Pérez, M. Tavera: Optimization in water systems: a PSO approach, The Society for Computer Simulation, International,2008.

[6] I. Sulistijono, N.Kubota: Human Head Tracking Based on Particle Swarm Optimization and Genetic Algorithm. JACIII 11(6): 681-687 (2007)

[7] R. C. Eberhart and Y. Shi, "Particle Swarm Optimization: Developments, Applications, and Resources," in Proc. Congress on Evolutionary Computation 2001 IEEE service center, Piscataway, NJ., Seoul, Korea., 2001.

[9] Venter, G. and Sobieski, J., "Particle Swarm Optimization"،AIAA 2002-1235, 43rd AIAA/ASME/ASCE/AHS/ASC Structures،Structural Dynamics, and Materials Conference, Denver, CO،.April 2002.

[10] Shi, Y. and Eberhart, R. C. Parameter selection in particle swarm optimization. Evolutionary Programming VII: Proc. EP 98 pp. 591-600. Springer-Verlag, New York, 1998.

[11] Scholar Encyclopedia, Particle Swarm optimization, last modified 19:35, 21 November 2008 http://www.scholarpedia.org/article/Particle_swarm_optimization >,10/6/2009<

[12] R.C. Eberhart and Xiaohui Hu. Human tremor analysis using particle swarm optimization. In Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on, 1999.

[13] Shiqiang Du, Wanshe Li, and Kai Cao. A learning algorithm of artificial neural network based on ga - pso. In Intelligent Control and Automation, 2006. WCICA 2006. The SixthWorld Congress on, pages 3633 – 3637, 2006.

[14] Ching-Yi Chen and Fun Ye. Particle swarm optimization algorithm and its application to clustering analysis. In 2004 IEEE International Conference on Networking, Sensing and Control, pages 789 – 794 Vol.2, 2004.

[15] Chin Aik Koay and D. Srinivasan. Particle swarm optimization-based approach for generator maintenance scheduling. In the 2003 IEEE Swarm Intelligence Symposium, 2003. SIS '03. Proceedings of, pages 167 – 173, 2003.

[16] Tomoya Takahashi, Masaki Yamakita, and Sang-Ho Hyon. An optimization approach for underactuated running robot. In SICE-ICASE, 2006. International Joint Conference, pages 3505 – 3510, 2006.

[17] S. Srinoy and W. Kurutach. Combination artificial ant clustering and k-pso clustering

approach to network security model. In Hybrid Information Technology, 2006. ICHIT'06. Vol 2. International Conference on, pages 128 – 134, 2006.

[18] Riccardo Poli, Analysis of the publications on the applications of particle swarm optimisation, Journal of Artificial Evolution and Applications, v.2008 n.1, p.1-10, January 2008.

[19] A.Rawat T.Toppo , Feature Based Object Tracking Using PTZ Camera.

[20] Microsoft Library,< http://msdn.microsoft.com/en-us/library/cc705040(PROT.10).aspx>, 20/6/2009.

[21] A. Yilmaz, O. Javed and M. Shah, "Object Tracking: A Survey," ACM Journal of Computing Surveys, Vol. 38, No. 4, 2006.

[22] SVI-Wiki (3Dmicroscopy, deconvolution, visualization and analysis,), HSV color space, < http://support.svi.nl/wiki/HsvColor >, 20/6/2009

[23]Wikipedia Ensyclopedi, HSL and HSV, <http://en.wikipedia.org/wiki/HSV_color_space>,20/6/2009

[24] John G. Allen, Richard Y. D. Xu, and Jesse S. Jin, "Object tracking using Camshift algorithm and multiple quantized feature spaces," in

[25] Y. Zheng and Y.Meng, Adaptive Object Tracking using Particle Swarm Optimization, Proceedings of the 2007 IEEE International Symposium on Computational Intelligence in Robotics and Automation Jacksonville, FL, USA, June 20-23, 2007

[26] C. Yang, R. Duraiswami, and L. Davis, "Fast Multiple Object Tracking via a Hierarchical Particle Filter," Proc. IEEE Int'l Conf. Computer Vision (ICCV), 2005.

[27] I. Sulistijono, N.Kubota: Human Head Tracking Based on Particle Swarm Optimization and Genetic Algorithm. JACIII 11(6): 681-687 (2007)

[28] L. Anton-Canalís, E. Sanchez-Nielsen and M. Hernández-Tejera , Swarmtrack: A Particle Swarm Approach to Visual Tracking VISAPP'06, Setúbal, Portugal, Feb. 25-28 2006. Vol.2. pp 221-228. ISBN: 972-8865-40-6.Setúbal, Portugal.

[29] Lei Zhang: Application of particle swarm optimization on batch process scheduling. ACM Southeast Regional Conference (1) 2005.