Object Tracker-A Real Time Object Tracking System with Particle Filter

Ahmet Özdemir

Submitted to the
Institute of Graduate Studies and Research
in partial fulfillment of the requirements for the degree of

Master of Science
in
Computer Engineering

Eastern Mediterranean University
February 2023
Gazimağusa, North Cyprus

Approval of the Institute of Graduate Studies and Research

Prof. Dr. Ali Hakan Ulusoy
Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of Master of Science in Computer Engineering.

Prof. Dr. Zeki Bayram
Chair, Department of Computer Engineering

We certify that we have read this thesis and that in our opinion it is fully adequate in scope and quality as a thesis for the degree of Master of Science in Computer Engineering.

Asst. Prof. Dr. Ahmet Ünveren
Supervisor

Examining Committee

1. Assoc. Prof. Dr. Adnan Acan
2. Assoc. Prof. Dr. Mehtap Köse Ulukök
3. Asst. Prof. Dr. Ahmet Ünveren

ABSTRACT

The filtering problem is to estimate the motion of an object detected by sensors, cameras, voice recorders, etc., based on its specific characteristics such as color, light reflection, and sound waves.

There are many filtering algorithms that have been used by researchers in the past to solve the related problems. In the literature, Lucas Rod'es Guirao (partner Diego Mor'ın) conducted an experiment to estimate the motion of an object and compared the performance of the Kalman filter and the particle filter in their study. They conclude that using the Kalman filter after the particle filter can slightly improve the performance of the particle filter alone, but the combination of Kalman filter and particle filter is still very sensitive to particle deprivation when there are occlusions.

The objective of this thesis is to use the particle filter algorithm to improve the tracking mechanizm in the way that updating particles which tracking the movements of the ball in a faster way.

This thesis presents the modified particle filter optimization for tracking the object efficiently and follow its posterior movements on the exact point to track the object truly. Depending on the proposed particle filter algorithm, the camera is set to track the object behind of the conjectural goal line and the proposed code performs the estimation of particles as a guider of goalkeeper.

Focused small ball considered as an object to estimate its *x* and *y* coordination in a two-dimensional area captured by a fixed camera connected to the computer. A 30-

second live shot is used to track the ball by using the RGB features of the particles to calculate the estimation of the object by a coded Matlab application. The proposed agorithm checks whether the ball has crossed the goal line or not, and then print these all output information on terminal section of Matlab.

Keywords: Particle filter algorithm, Kalman filter, Estimation, Object tracking.

ÖZ

Filtreleme problemi, sensörler, kameralar, ses kayıt vb. cihazları kullanarak hedef objenin kendine ait özgü renk, ışık yansıması ve ses dalgaları gibi özelliklerini kullanarak objenin konum hareketlerini tahmin etme konularını içerir.

Geçmişte araştırmacılar tarafından kendi alanlarına ait problemleri çözmek için kullanılan birçok filtreleme algoritması vardır. Literatürde Lucas Rod´es Guirao (partner Diego Mor´ın) bir araştırma yapmıştır ve bu araştırmada Kalman filter ve particle filter algoritmalarını karşılaştırıp hedef objenin konum hareketlerini tahmin etme yaklaşımlarını incelemişlerdir. Araştırma sonucunda Kalman filter ın particle filter dan sonra kullanılmasının, tek başına particle filter performansını geliştirdiğini göstermiştir ama Kalman filter ve particle filter bileşimi olan Kalmanized particle filter (KPF)'ın ise oklüzyon varken, cismi takip eden taneciklerin yoksunluğu konusunda hala hassas olduğu gözlemlenmiştir.

Bu tezde amaç, particle filter algoritmasını kullanarak takip mekanizmasını, takibi sağlayan taniecikleri hızlı bir yol ile güncelleyerek geliştrimektir.

Bu tez, nesneyi verimli bir şekilde takip eden ve onun bir sonraki hareketlerini doğru bir şekilde tahmin eden bir particle filter optimizasyonu sunar.

Hedef cismin takip edilmesini sağlayacak particle filter optimize etme algoritmasını geliştirerek objenin bir sonraki konumuna daha verimli bir şekilde erişim sağlanması hedeflenmektedir. Bu algoritmaya bağlı olarak, hedef cisim takibi için hayali kale çizgisine sabitlenecek şekilde bir kamera ve taneciklere yön verecek olan kodlanmış bir tahmin etme algoritması kullanıldı.

Bilgisayara bağlı bir web kamera kullanarak, futbol topu olarak varsayılan bir objenin bir sonraki adımını, iki-boyutlu bir alanda *x* ve *y* koordinatları tahmin edilecek. Matlab kodları kullanılarak 30 saniyelik bir canlı kaydın içerisinde obje takibini, objenin renk özelliklerini kullanarak taneciklerin, topun üzerine yerleşip takip etmesi sağlanacak. Daha sonra topun, web kameraya olan mesafesi belirlenmiş, hayali kale çizgisini geçip geçmediği hesaplanarak sonuç bilgileri Matlab terminal kısmında gösterilecektir.

Anahtar Kelimeler: Tanecikli filtre algoritması, Kalman filtresi, Tahmin, Obje takibi

To My Lovely Family

ACKNOWLEDGMENTS

I would like to thanks of gratitude to my supervisor Asst. Prof. Dr. Ahmet Ünveren who helped me in doing a lot of research. I am really thankful to him for the continuous support to my MSc. study, for his patience, motivation and immense knowledge.

I want to thank my family for their endless love and supporting me spiritually throughout my life.

And finally, I would like to thank my dear friend Gizem Baran and my lovely pet Mücito who are always supporting and motivating me in a positive way.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF SYMBOLS

$\delta(.)$	Dirac Delta Function
π	Target Distribution
$\omega_k^{(i)}$	Normalized Particle Weight
$\pi_t(x_{t+1})$	Prediction
$\pi_{t+1}(x_t)$	Updating Current State
$\pi_{t+1}(x_{t+1})$	New Estimation
$\mathfrak{p}(x_k \mid t_{1:k})$	Probability Density Function
$\mathfrak{q}(y_k, t_{1:k})$	Proposal Density Function
$x_t \sim q_t(. \mid x_{t-1}, \theta)$	State Equation
$y_t \sim f_t(. \mid x_t, \phi)$	Observation Equation

Chapter 1

INTRODUCTION

Particle filter problem is commonly used in object tracking area by researchers. The important point of the filtering problem is not just recording the objects' posterior movements by using needed devices as video recorder, light and movement sensors etc. The most important aim is that, following the target object in a safest and fastest way by using particles which are sticking on the objects to find out its direction, velocity and distance between object and input recorder. Many researchers have studied on this subject and tried to find different methods to solve this problem in the past researches [1–6].

Kalman Filter is an implementation of Bayes filter was invented by Rudolph Emil Kalman in 1950. Algorithm of this filter includes two steps which are prediction and update. Prediction step tries to recognize future state by using previous state depending on current state. This state focuses on some needed variables which are position, velocity and acceleration to measure estimation of the object. Update state is the state which using current state to make improvement of present state. This state lays on required predicted state variable and predicted covariance variable calculated by using prediction results to update prediction step which will be current state later on. Difference of Kalman filter against other filters is that Kalman filter is a filter in dynamic system to measure current state and estimate posterior movements of the object by using previous state variables which includes dynamic changes which are position, velocity and acceleration.

Particle Filter which called as Monte Carlo method provides to measure target distribution (π) by using three systems, prediction $(\pi_t(x_{t+1}))$, updating current state $(\pi_{t+1}(x_t))$ and new estimation $(\pi_{t+1}(x_{t+1}))$. These systems are helping to find evolving probability distribution π in the state that increasing dimension. The authors are inspired of two models to improve their research. Bayesian Missing Data-Problem by using Rao-Blackwellization to provide approximation and State-Space Model by using observation equation $(y_t \sim f_t(\cdot \mid x_t, \phi))$ and state equation represented by Markov process $((x_t \sim q_t(\cdot \mid x_{t-1}, \theta)))$. The authors proposed a general use of Rao-Blackwellization to improve performance of approximation, memory usage and speed of computation. In this case, weight variable is added in the problem to improve better approximation of each posterior step with their probability values [1].

Kalmanized particle filter is the filtering method which combines two filtering model to improve estimation of the posterior movements of the target objects researched by [7]. The authors made a comparison between performance of Kalman filter and particle filter in their related research. Nintendo Pinball game has used as an experimental application. One minute video recorded to test it on related filter algorithms. Particle filter and Kalman filter methods applied on this video separately and then they apply the combination of particle filter and Kalman filter to observe the performance of finding expected posterior movement of the object and then the performances has recorded to finalize conlusion. Results are compared in two way which are 'when no particle deprivation occurs' and 'when particle deprivation occurs'. By comparing these two informations the results shows that using Kalman filter after particle filter is approximately improoving the efficiency of the particle filter alone. The authors observe that the Kalmanized particle filter is sensitive to deprivation effects to particles that estimates object. Their results conclude that if particle deprivation does not occurs, mean square error rate is acceptable for particle filter, Kalmanized particle filter (with constant velocity and acceleration separately) but if particle filter deprivation occurs, mean square error rate of applied particle filter, Kalmanized particle filter (with constant velocity and acceleration separately) is too high and cannot acceptable although mean square error rate of Kalman filter (with constant velocity and acceleration separately) and particle filter does not have too much differences between ‘when no particle deprivation occurs’ and ‘when particle deprivation occurs’.

The problem in this thesis, focusing on the state to estimate target distribution which will be called estimation posterior movement of the object by using particle filter algorithm by assigning constant velocity and constant acceleration. Algorithm is improved as Kalmanized particle filter algorithm by using only particle filter algorithm as mentioned in [1]. A small ball defined as an object is focused to estimate its x and y coordination in a two-dimensional area recorded by fixed camera connected to computer. 30 second live record is used to follow the ball by using RGB feature of the particles to calculate estimation of the object by coded matlab application.

This thesis consist of four chapter.

In Chapter 1, required definitions and literature review metioned as Introduction.

In Chapter 2, basic information about particle filter algortihm, modified particle filter algorithms, advantages and disadvantages mentioned as Particle Filter.

In Chapter 3, proposed particle filter algorithm, detailed steps of particle filter model, developing process of the experiment mentioned as Proposed Particle Filter.

In Chapter 4, experiment process, results and future work is mentioned as Conclusion.

Chapter 2

PARTICLE FILTER

Particle filters (sequence Monte Carlo) technique is a collection of Monte Carlo algorithms for tackling filtering issues such as determining future state in dynamical systems under signal processing and Bayesian inference. The main target of the system is calculating posterior movement distributions of the target object by given partial observations.

The history of particle filter is started with the term "particle filters" was found in 1996 by Del Moral and this filter used for fluid mechanics since 1960's [8]. Sequential Monte Carlo method has invented in 1998 by reference [6].

Particle filter leads set of samples which points posterior states of a random process which widely preferred as mathematical models of systems or phenomenal situations in stochastic process. The study of stochastic process consists of mathematical information and specific techniques related to calculus, set theory, probability etc. particle filter model provides a well-known methodology [8] to produce particles from the needed distribution without assumption about state distribution.

Particle filter improves its prediction by updating the approximation in statistical manner. The particles which are distributed randomly are named as set of particles; each sample has a weight assigned to represent probability of that sample. In the resampling step, negligible weighted particles are replaced by new particles which of the particles with higher weights.

For situations involving nonlinear filtering and hidden Markov models, the particle filter model is preferable. Hidden Markov model employs a number of approximation methods, including extended Kalman filters, Markov Chain Monte Carlo approaches, and the linear-Gaussian signal-observation model (Kalman filter).

The objective of particle filter is to predict the future density of observation variables which is related to state-process by some methodological design. The particle filter is invented and improved for hidden Markov model. Using observation computation with regard to state-space, a general particle filter model forecasts the next step distribution of the hidden states:

$$\begin{tikzcd}
Y_0 \arrow[r] \arrow[d] & Y_1 \arrow[r] \arrow[d] & Y_2 \arrow[r] \arrow[d] & Y_3 \arrow[d] & \cdots & \text{signal} \\
Z_0 & Z_1 & Z_2 & Z_3 & \cdots & \text{observation}
\end{tikzcd}$$

given the observation process Z_0 to Z_s at a time step s, the filtering issue focuses on predicting sequentially the values of the hidden states Y_s.

The particle filter model provides conditional probability approximation using experiential measure related with a usual typed particle algorithm. In reverse, the Markov Chain Monte Carlo or importance sampling strategy models the posterior as follows:

$$\mathfrak{p}\left(x_0, x_1, \cdots, x_s \mid y_0, y_1, \cdots, y_s\right).$$

2.1 Particle Filtering

Particle filters utilize many discrete "particles" to indicate the belief distribution across a tracked object's position.

Predicting the moving of the objects in complicated area is a popular research in computer vision. It involves determining how the time based system's state changes by time by having multiple measurements that may contain noise. Calculating the posterior density $\mathfrak{p}(x_k \mid t_{1:k})$ of the present object state x_k conditioned on all observation $t_{1:k} = (t_1, \cdots, t_k)$, up to time k is the goal of Bayesian tracking.

In two phases, the probability density function $\mathfrak{p}(x_k \mid t_{1:k})$ may be produced recursively: prediction and update. If the time-varying state x_k is represented as a first-order Markov process, the probability density function $\mathfrak{p}(x_k \mid t_{1:k})$ is calculated as

$$\mathfrak{p}(x_k \mid t_{1:k}) = \kappa \mathfrak{p}(t_k \mid x_k)\, \mathfrak{p}(x_k \mid t_{1:k-1}) \tag{2.1}$$

$$\mathfrak{p}(x_k \mid t_{1:k-1}) = \int \mathfrak{p}(x_k \mid x_{k-1})\, \mathfrak{p}(x_{k-1} \mid t_{1:k-1})\, dx_{k-1}, \tag{2.2}$$

where κ is an independent normalizing constant of x_k, the likelihood function is denoted by $\mathfrak{p}(t_k \mid x_k)$, the dynamic model is denoted by $\mathfrak{p}(x_k \mid x_{k-1})$ and $\mathfrak{p}(x_k \mid t_{1:k-1})$ is the temporal prior over x_k given past observation, $\mathfrak{p}(x_k \mid x_{k-1})$ and $\mathfrak{p}(t_k \mid x_k)$ it is not necessary for the distribution to be Gaussian in this scenario [9].

Particle filter [1, 10] is a technique for using Monte Carlo simulations to create a recursive Bayesian filter. Rao-Blackwellization is one technique for improving the performance of sequential Monte Carlo (SMC). The Rao-Blackwellized particle filter (RBPF) [3, 11, 12] is a method that seeks to assess parts of the filtering equations analytically while employing Monte Carlo sampling for the remaining equations rather than depending exclusively on sampling.

The fundamental concept is that of particle filtering is to employ a weighted sample set to approach the probability distribution

$$S = \left\{ \left(s^{(n)}, \pi^{(n)} \right) \mid n = 1, \cdots, N \right\}.$$

Each sample s represent one possible state of the object with a corresponding discrete sampling probability represents a different potential state of the object, with a discrete sampling probability π, where $\sum_{n=1}^{N} \pi^{(n)} = 1$ [13].

The sample set's evolution is characterized by advancing each sample in accordance with a system model. Each member of the collection is then weighted in relation to the observations, and N samples are chosen with replacement by selecting a specific sample with probability $\pi^{(n)} = \mathfrak{p}\left(t_l \mid x_l = s_l^{(n)}\right)$ [13]. Using

$$E[S] = \sum_{n=1}^{N} \pi^{(n)} s^{(n)} \tag{2.3}$$

the estimated mean state of an item is computed at each time step [13].

Since it models uncertainty, particle filtering offers a reliable tracking framework. It can keep its options open by assessing many state hypotheses concurrently. It can keep its options open by assessing many state hypotheses at the same time [13].

Particle filter, also known as sequential Monte Carlo techniques are a group of algorithms that are useful in simulations for sampling and estimating state distributions in dynamic systems. They have been used in a variety of areas, including:

1. Robotics: Particle filters have been used to localize robots and estimate their

pose.

2. Computer vision: Particle filters have been used to track objects in video sequences and to estimate the pose of cameras.
3. Navigation: Patricle filters have been used in GPS and inertial navigation systems to estimate the position and orientation of vehicles.
4. Speech processing: Particle filters have been used to track the state of speech production systems and to estimate the spectral parameters of speech signals.
5. Finance: Particle filters have been used to estimate asset price models and to calibrate financial derivatives.
6. Bioinformatics: Particle filters have been used to estimate gene expression levels and to align DNA sequences.
7. Environment monitoring: Particle filters have been used to estimate air quality and to track the movement of pollutants in the atmosphere.
8. Medicine: Particle filters have been used to model the dynamics of physiological systems and to design control strategies for medical devices.

The sets of particles used in the particle filters are appropriately weighted and updated recursively based on the Bayesian rule. Additionally, the standard particle filter algorithm includes a resampling technique based on sequential importance sampling [14]. The specific steps are as seen below:

$$y_k = \mathfrak{f}(y_{k-1}, u_{k-1})$$
$$z_k = \mathfrak{g}(y_k, v_k),$$

where y_k and z_k are the state vector and measurement vector, respectively, at time k. The process noise is denoted by u_k while the measurement noise is denoted by v_k. Similarly, $\mathfrak{f}$ and $\mathfrak{g}$ are two well-known nonlinear functions.

The posterior probability density function $\mathfrak{p}(y_k \mid z_{1:k})$ may be estimated by a set of N particles $\left\{y_k^{(i)}, i = 1, 2, \cdots, N\right\}$ derived from a well-known proposal density function $\mathfrak{q}(y_k, z_{1:k})$ [15]

$$\hat{\mathfrak{p}}(y_k \mid z_{1:k}) = \sum_{i=1}^{N} \omega_k^{(i)} \delta\left(y_k - z_k^{(i)}\right), \tag{2.4}$$

where at time k importance weights (normalized particle weight) are $\omega_k^{(i)}$ the function of Dirac delta $\delta(.)$ is the Dirac delta function. At any moment, the weight can be modified as follows:

$$\omega_k^{(i)} = \omega_{k-1}^{(i)} \frac{\mathfrak{p}\left(z_k \mid y_k^{(i)}\right) \mathfrak{p}\left(y_k^{(i)} \mid y_{k-1}^{(i)}\right)}{\mathfrak{q}\left(y_k^{(i)} \mid y_{k-1}^{(i)}, z_k\right)}. \tag{2.5}$$

At time k, specific particles $\left\{y_k^{(i)}\right\}_{i=1}^{N}$ are chosen from a significance density function during the importance-sampling phase. The particle filter approach is difficult to apply when the optimal significance density function is used. Choosing $\mathfrak{p}(y_k \mid y_{k-1})$ as a suboptimal importance density function:

$$\mathfrak{q}\left(y_k \mid y_{k-1}^{(i)}, z_k\right) = \mathfrak{p}\left(y_k \mid y_{k-1}^{(i)}\right) \tag{2.6}$$

is a realistic and effective option [15].

Substituting (2.6) in (2.5), the weight update equation is written as follows:

$$\omega_k^{(i)} = \omega_{k-1}^{(i)} \mathfrak{p}\left(z_k \mid y_k^{(i)}\right).$$

Finally, the weights of the particles are normalized by $\tilde{\omega}_k^{(i)} = \frac{\omega_k^{(i)}}{\sum_{i=1}^{N} \omega_k^{(i)}}$. If the particles with high weights are over-represented, the number of useful particles is reduced, which in turn results in a reduction in the new particle set's information capacity [16].

$$\hat{N}_{eff} = \frac{1}{\sum_{i=1}^{N} \left(\tilde{\omega}_k^{i}\right)^2},$$

determines the number of effective (meaningful) particles $\hat{N}_{eff}$.

The flowchart of the particle filter algorithm is shown below:

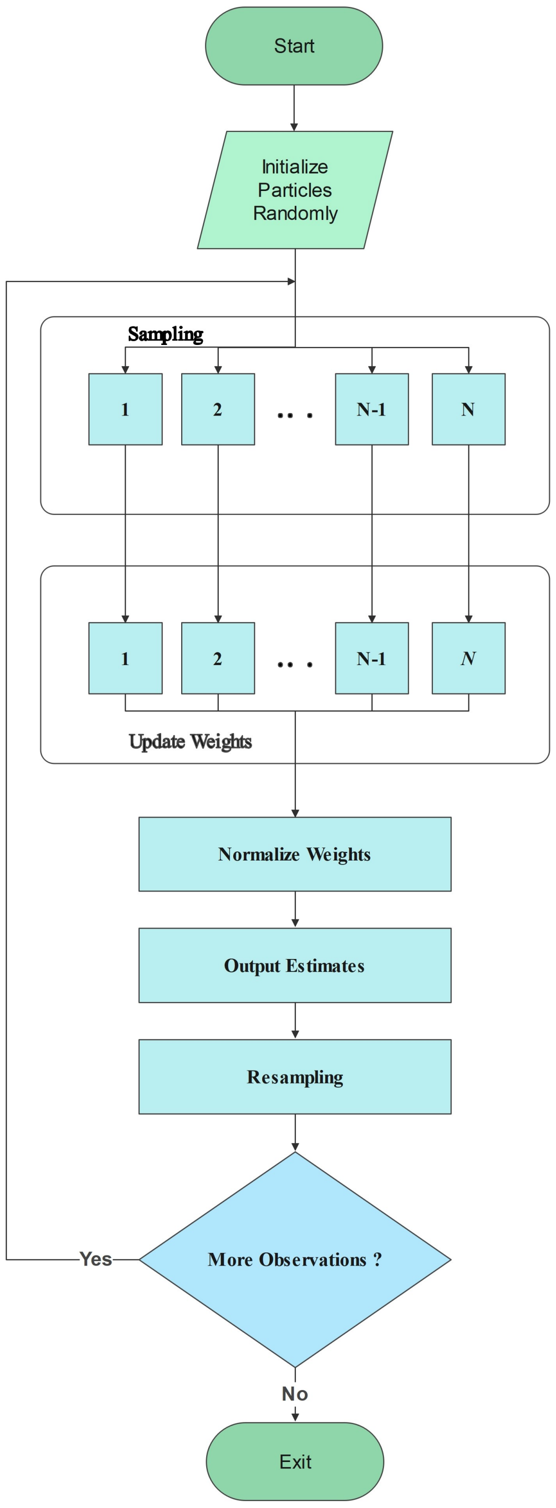


Figure 2.1: Flowchart of the particle filter

The individual steps of the procedure shown in Figure 2.1 are explained by birefly as follows:

Initializing: In this step, a set of particles is initialized, each representing a possible state of the system being estimated. The initial particles can be randomly generated or generated using a prior distribution of the system's state.

Sampling: In this step, a sample of each particle is generated by applying a motion model to the current state of the particle. This model represents how the system's state changes over time.

Update Weights: In this step, the weights of each particle are updated based on the measurement obtained from the system. This measurement is used to evaluate the likelihood of each particle's state representing the actual state of the system.

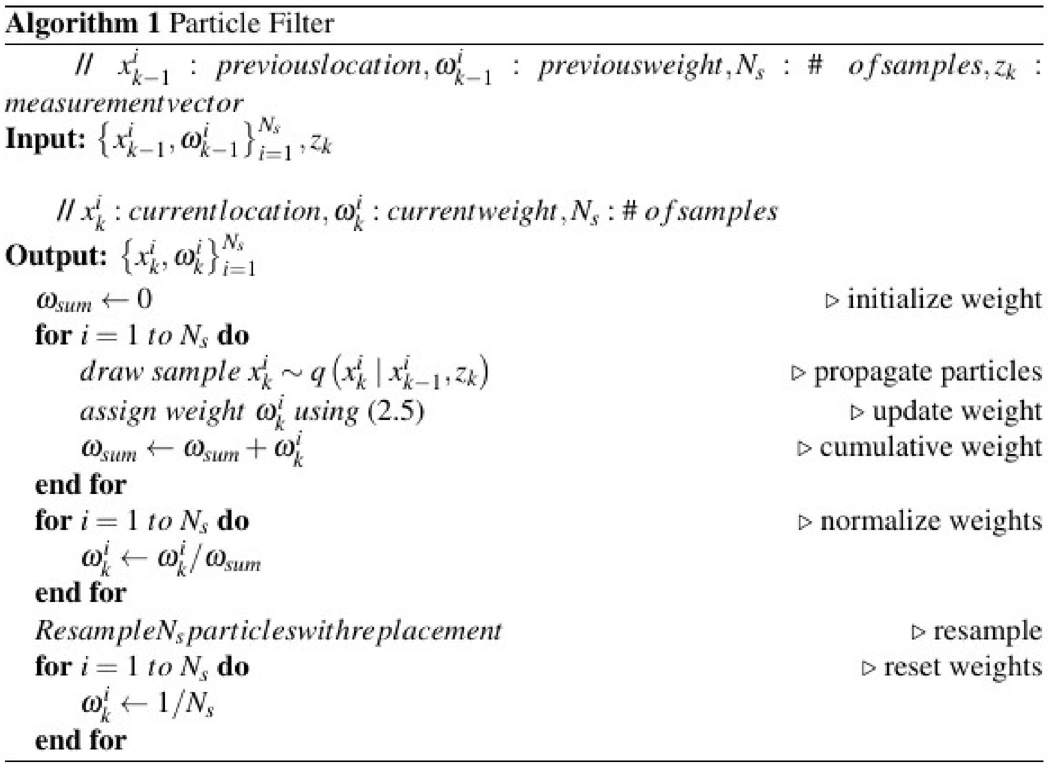
Normalize Weights: In this step, the weights of each particle are normalized to ensure that they sum up to 1. This allows for proper computation of the particle filter's output.

Output Estimation: In this step, the output of the particle filter is estimated by taking the weighted average of all the particles. This provides an estimate of the system's state based on the current measurements and the motion model.

Resampling: In this step, particles are randomly selected from the current set of particles based on their weights. The idea behind resampling is to keep particles that have high weights (i.e., particles that are more likely to represent the actual state of the system) and eliminate particles with low weights (i.e., particles that are less likely to represent the actual state of the system).

Note that these steps are repeated at each time step of the system's operation to provide an updated estimate of the system's state.

The algorithm of the traditional particle filter related to the mathematical expressions is shown below in the Figure 2.2



```
Algorithm 1 Particle Filter
    // $x_{k-1}^i$ : $previouslocation, \omega_{k-1}^i$ : $previousweight, N_s$ : # $ofsamples, z_k$ : $measurementvector$
Input: $\left\{x_{k-1}^i, \omega_{k-1}^i\right\}_{i=1}^{N_s}, z_k$

    // $x_k^i$ : $currentlocation, \omega_k^i$ : $currentweight, N_s$ : # $ofsamples$
Output: $\left\{x_k^i, \omega_k^i\right\}_{i=1}^{N_s}$
  $\omega_{sum} \leftarrow 0$                                   ▷ initialize weight
  for $i = 1$ to $N_s$ do
      draw sample $x_k^i \sim q\left(x_k^i \mid x_{k-1}^i, z_k\right)$      ▷ propagate particles
      assign weight $\omega_k^i$ using (2.5)                    ▷ update weight
      $\omega_{sum} \leftarrow \omega_{sum} + \omega_k^i$                      ▷ cumulative weight
  end for
  for $i = 1$ to $N_s$ do                                ▷ normalize weights
      $\omega_k^i \leftarrow \omega_k^i / \omega_{sum}$
  end for
  $ResampleN_sparticleswithreplacement$                         ▷ resample
  for $i = 1$ to $N_s$ do                                ▷ reset weights
      $\omega_k^i \leftarrow 1/N_s$
  end for
```

Figure 2.2: Algorithm particle filter

Originally, particle filters were developed to use edge-based image features [17, 18]. Color-based particle filters have recently been introduced [12, 19]. The capacity of particle filter to retain a precise approximation of the posterior is critical to its success. To account for deviations in the state space, a large number of particles are required to assure proper sampling, and this number grows exponentially with the dimension of the state. However, the high computational cost associated with particle filters are often unsuitable for real-time applications because to their enormous particle count.

2.2 Advantages and Disadvantages of Particle Filter

Advantages of particle filters:

1. Particle filters are able to estimate state distributions in nonlinear and non-Gaussian systems, where traditional Kalman filters may not work well.
2. Particle filters can handle high-dimensional state spaces, whereas other methods such as the extended Kalman filter become computationally infeasible.
3. Particle filters are easy to implement and can be used with minimal assumptions about the system being modelled.

Disadvantages of particle filters:

1. Particle filters require a significant amount of samples (particles) to accurately estimate state distributions, which can make them computationally expensive.
2. Particle filters can suffer from degeneracy, where the particles collapse into a narrow region and the filter fails to accurately estimate the state distribution.
3. Particle filters may have slow convergence, especially in high-dimensional state spaces.
4. Particle filters may have difficulty with multimodal state distributions, where the state can take on multiple values with similar likelihood.

Chapter 3

PROPOSED PARTICLE FILTER

Particle filter is a sequential Monte Carlo method for estimating the posterior distribution of a system's state given noisy observations. The fundamental concept of a particle filter is to depict the posterior distribution using a set of randomized samples, named particles, and then use these particles to approximate the true posterior distribution. The steps of the particle filter algorithm are as follows:

1. Initialization: The first step is to initialize the particles. This is typically done by sampling from the previous distribution of the state.
2. Resampling: After the initial set of particles is generated, a resampling step is performed to omit particles which have low weight and to replicate particles that have high weight. The aim of this step is to ensure that the particles are represented good at true posterior distribution.
3. Propagation: After resampling, each of the particles are propagated forward in time using the dynamics of the system. The propagation step is often performed using a process model, which describes how the system state improves over time.
4. Weighting: After the particles are propagated, their weights are updated based on the likelihood of the observed information given the current state of the particle set. The goal of this step is to assign superior weights through particles which are more likely to have generated the observed data.
5. Estimation: After the particles have been weighted, the last step is to estimate the state of the system using the particles. This can be done using various techniques,

such as the sample mean or the maximum a posteriori (MAP) estimate.

6. Repeat steps 2-5 for each time step until the desired estimation is reached.

Note that the particle filter algorithm is a recursive algorithm, in which means that it uses the information from the former time step to estimate next state of the system at the current time step.

The literature is reviewed and some good experiments are found in the field of object tracking using the particle filter algorithm, where some different methods are applied. Inspired by the research [20], the particle filter algorithm was developed to calculate the estimation of the target object based on its specific texture color in the RGB feature.

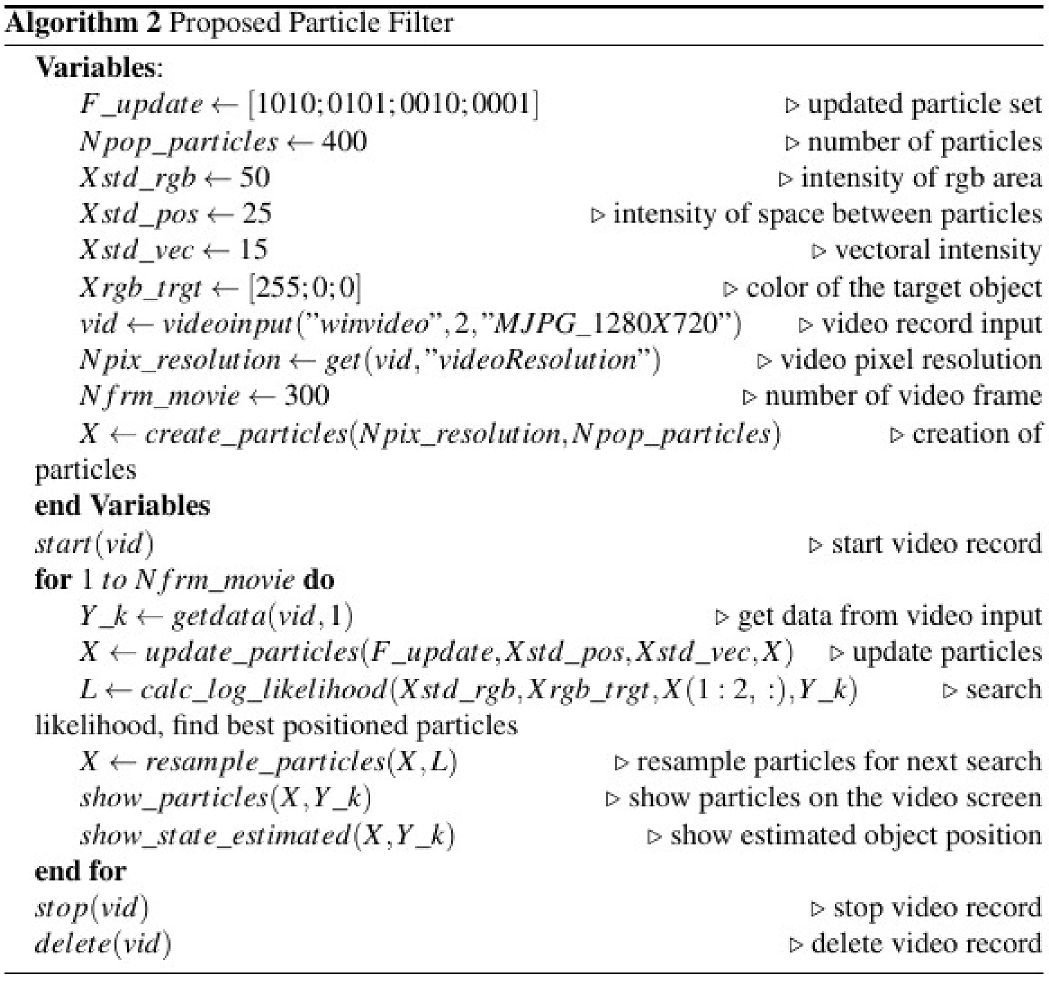
Unlike other methods, the particle filter is a technique used to estimate the probable state of a system based on noisy observations. It utilizes a collection of random samples, called particles, to estimate the true posterior distribution of the system. The particle filtering algorithm is a type of recursive method, i.e., it builds on information from previous steps to estimate the current state of the system.

In the project [20], the authors improve an algorithm called the Kalmanized Particle Filter Algorithm and perform the experiment to estimate the posterior motion of the object. They apply this algorithm to a Nintendo pinball game and try to estimate the target object. The authors applied the particle filter, the Kalman filter, and the combined Kalmanized particle filter respectively to each game. As a conclusion, they found that the combined Kalmanized particle filter is very sensitive to particle deprivation when there are occlusions because the background in the game is noisy and the filter algorithm has problems to focus on the target object and the particle filters may lose the estimation of the target object.

In this thesis an algorithm was developed that includes a particle filter algorithm to track an object that is a red ball. In the simulation, a constant velocity is used for the target object, and the distance measurements refer to the intensity of the number of pixels as a percentage measure used to check how close the ball is to the camera image.

The subject of the thesis is that the webcam image is considered as a goal in a football game and the proposed algorithm tries to estimate the posterior motion of the target object as a certain colored ball in the x and y axes. To check whether the ball, which has a constant velocity, is in the goal, the distance between the camera pixels of the target object is measured. When the ball approaches the camera pixels, the percentage becomes higher and the algorithm can measure whether the ball is in the goal or not.

The proposed method shown in Figure 3.1 gives the detailed pseudocode:



```
Algorithm 2 Proposed Particle Filter
Variables:
    $F\_update \leftarrow [1010;0101;0010;0001]$                ▷ updated particle set
    $Npop\_particles \leftarrow 400$                             ▷ number of particles
    $Xstd\_rgb \leftarrow 50$                                    ▷ intensity of rgb area
    $Xstd\_pos \leftarrow 25$                   ▷ intensity of space between particles
    $Xstd\_vec \leftarrow 15$                                    ▷ vectoral intensity
    $Xrgb\_trgt \leftarrow [255;0;0]$                      ▷ color of the target object
    $vid \leftarrow videoinput("winvideo", 2, "MJPG\_1280X720")$    ▷ video record input
    $Npix\_resolution \leftarrow get(vid, "videoResolution")$    ▷ video pixel resolution
    $Nfrm\_movie \leftarrow 300$                            ▷ number of video frame
    $X \leftarrow create\_particles(Npix\_resolution, Npop\_particles)$    ▷ creation of particles
end Variables
$start(vid)$                                                ▷ start video record
for $1$ $to$ $Nfrm\_movie$ do
    $Y\_k \leftarrow getdata(vid, 1)$                       ▷ get data from video input
    $X \leftarrow update\_particles(F\_update, Xstd\_pos, Xstd\_vec, X)$    ▷ update particles
    $L \leftarrow calc\_log\_likelihood(Xstd\_rgb, Xrgb\_trgt, X(1:2,:), Y\_k)$    ▷ search likelihood, find best positioned particles
    $X \leftarrow resample\_particles(X, L)$          ▷ resample particles for next search
    $show\_particles(X, Y\_k)$                     ▷ show particles on the video screen
    $show\_state\_estimated(X, Y\_k)$                   ▷ show estimated object position
end for
$stop(vid)$                                                  ▷ stop video record
$delete(vid)$                                                ▷ delete video record
```

Figure 3.1: Algorithm proposed particle filter

The functions in the Figure 3.1 are mentioned in detail as follows:

The "create particles" step in particle filtering for object tracking involves generating a set of particles that will be used to represent the state of the target object. The particles are generated using an initialization method and are used as the starting point for the particle filter algorithm. The number of particles generated depends on the desired accuracy of the particle filter and the complexity of the target object's state. In general, a larger number of particles results in a more accurate representation of the target object's state, but it also increases the computational burden of the particle filter. The particles can be generated in several ways, depending on the desired initialization method. One common method is to randomly generate particles within a defined region that is believed to contain the target object. Another method is to use the results of an initial detection or tracking algorithm to generate a set of particles that are centered around the detected object's state. Particles are created randomly by using the particle population and pixel resolution variables in the funciton $CREATE\_PARTICLES$. The function of proposed method shown in Figure 3.2 gives the detailed pseudocode:

```
function CREATE_PARTICLES($Npix\_resolution$ , $Npop\_particles$)
    $X1 \leftarrow randi(Npix\_resolution(2), 1, Npop\_particles)$
    $X2 \leftarrow randi(Npix\_resolution(1), 1, Npop\_particles)$
    $X3 \leftarrow zeros(2, Npop\_particles)$
    $X \leftarrow [X1; X2; X3]$
end function
```

Figure 3.2: Function create particles

The "update particles" step in particle filtering involves propagating each particle's state forward in time, generating a predicted measurement for each particle, calculating the likelihood of each particle representing the actual state of the target object, and updating the weight of each particle based on the likelihood values. These updated particles are then used in the resampling step to ensure that the particle distribution remains representative of the target object's state. Particles are updated by using the position of particles and their vectoral variables related to the set of F_update and coded in function $UPDATE\_PARTICLES$. The function of proposed method shown in Figure 3.3 gives the detailed pseudocode:

```
function UPDATE_PARTICLES($F\_update$ , $Xstd\_pos, Xstd\_vec, X$)
    $N \leftarrow size(X, 2)$
    $X \leftarrow F\_update * X$
    $X(1:2,:) \leftarrow X(1:2,:) + Xstd\_pos * randn(2, N)$
    $X(3:4,:) \leftarrow X(3:4,:) + Xstd\_vec * randn(2, N)$
end function
```

Figure 3.3: Function update particles

The likelihood calculation is a crucial step in particle filtering as it provides a mechanism to assess the probability of each particle representing the target object's state based on the similarity of the predicted measurement and actual measurement. Typically, the likelihood calculation involves comparing the predicted measurement with the actual measurement using a probabilistic model, such as a Gaussian distribution. The resulting likelihood value for each particle is used to compute the posterior probability of the particle representing the target object's state, which can then be used to resample the particles and update their weights. The function *CALC_LOG_LIKELIHOOD* of proposed method shown in Figure 3.4 gives the detailed pseudocode:

```
function CALC_LOG_LIKEKIHOOD($Xstd\_rgb$ , $Xrgb\_trgt$,$X$,$Y$)
    $Npix\_h \leftarrow size(Y,1)$
    $Npix\_w \leftarrow size(Y,2)$
    $N \leftarrow size(X,2)$
    $L \leftarrow zeros(1,N)$
    $Y \leftarrow permute(Y,[3\ 1\ 2])$
    $A \leftarrow -\log(sqrt(2*pi)*Xstd\_rgb)$
    $B \leftarrow -0.5/(Xstd\_rgb^2)$
    $X \leftarrow round(X)$
    for 1 $to$ $N$ do
        $m \leftarrow X(1,k)$
        $n \leftarrow X(2,k)$
        $I \leftarrow (m \geq 1\ \&\ m \leq Npix\_h)$
        $J \leftarrow (n \geq 1\ \&\ m \leq Npix\_w)$

        if $I$ && $J$ then
            $C \leftarrow double(Y(:,m,n))$
            $D \leftarrow C - Xrgb\_trgt$
            $D2 \leftarrow D' * D$
            $L(k) \leftarrow A + B * D2$
        else
            $L(k) \leftarrow -Inf$
        end if
    end for
end function
```

Figure 3.4: Function calculate likelihood

The resampling step in particle filtering is used to ensure that the particle distribution remains representative of the target object's state by eliminating particles with low weights and creating new particles to represent areas of high probability. The resampling step is performed after the likelihood calculation and weight update. The first step in resampling is to normalize the weights of each particle such that they sum to 1. This is done to ensure that the particles with the highest weights have a higher chance of being selected for resampling. Once the weights are normalized, a set of new particles can be generated using a resampling method such as systematic resampling or stratified resampling. In systematic resampling, a random number is generated, and particles are selected with a probability proportional to their weight. In stratified resampling, particles are selected based on dividing the normalized weights into equal intervals and then randomly selecting a particle from each interval. Particles are resampled for the next search to be used as the starting point for the next iteration of the particle filter via *RESAMPLE_PARTICLES* function. The function of proposed method shown in Figure 3.5 gives the detailed pseudocode:

```
function RESAMPLE_PARTICLES($X$ , $L\_log$)
    $L \leftarrow \exp(L\_log - max(L\_log))$
    $Q \leftarrow L/sum(L,2)$
    $R \leftarrow cumsum(Q,2)$
    $N \leftarrow size(X,2)$
    $T \leftarrow rand(1,N)$
    $[\sim,I] \leftarrow histc(T,R)$
    $X \leftarrow X(:,I+1)$
end function
```

Figure 3.5: Function resample particles

The "show particles" function in particle filtering is a visualization tool that displays the particles on a graphical interface and allows the user to observe the evolution of the particle distribution over time and to evaluate the performance of the particle filter.

This function typically displays the particles as points or small markers superimposed on the video or image of the scene. The color or shape of each particle can be used to represent the weight of the particle, with particles having a higher weight being displayed as larger markers or with a different color. The function can be used to evaluate the performance of the particle filter and to diagnose problems with the particle filter, such as a lack of diversity in the particle distribution or a poor representation of the target object's state. Exact X and Y position on the screen is shown on the terminal of the Matlab program with the function $SHOW\_PARTICLES$. The function of proposed method shown in Figure 3.6 gives the detailed pseudocode:

```
function SHOW_PARTICLES($X$ , $Y\_k$)
    $plot(X(2,:),X(1,:),"-")$
end function
```

Figure 3.6: Function show particles

The particles are resampled to ensure that the particles with higher weights are more likely to be represented in the final set of particles. The state estimate is then computed as the weighted average of the particles. This process is repeated at each time step to track the object over time. State estimation is performed by representing the object's state as a set of particles, each with its own position, velocity, and weight. The weights are updated at each time step based on the observations and control inputs, and the particles with higher weights are more likely to represent the true state of the object.

The function $SHOW\_STATE\_ESTIMATED$ computes the weighted average of particles and estimated X and Y position and target color pixel density of the target object is shown on the terminal section. There is a threshold to control the goal line which is set by the value 350. Message "GOAAAAAAAL !" is shown on the terminal section by measuring the distance between webcam and target object in pixel percentage unit by controlling the threshold value. The function of proposed method shown in Figure 3.7 gives the detailed pseudocode:

```
function SHOW_STATE_ESTIMATED($X$ , $Y\_k$)
    $plot(X\_mean(2,:), X\_mean(1,:), "h", "MarkerSize", 16, "MarkerEdgeColor",$
        $"y", "MarkerFaceColor", "y")$
    $estimated\_position \leftarrow ("X" : +X\_mean(2,:) + "Y : " + X\_mean(1,:))$
    $print\{estimated\_position\}$

    $percentRed \leftarrow 100 * (sum(sum(X))/(size(Y\_k,1) * size(Y\_k,2)))$
    $total\_pixel \leftarrow sum(sum(X)$
    $print\{total\_pixel\}$
    $print\{percentRed\}$

    if $percentRed < 350$ then
        $print\{GOAAAAAAAL!\}$
    end if
end function
```

Figure 3.7: Function show state estimated

In this thesis a webcam camera connected to the computer through a USB port is used to capture video for real-time recording. The connection between the camera and the computer is made using Matlab to obtain a real-time recording. The recordings are analyzed in an algorithm coded in Matlab.

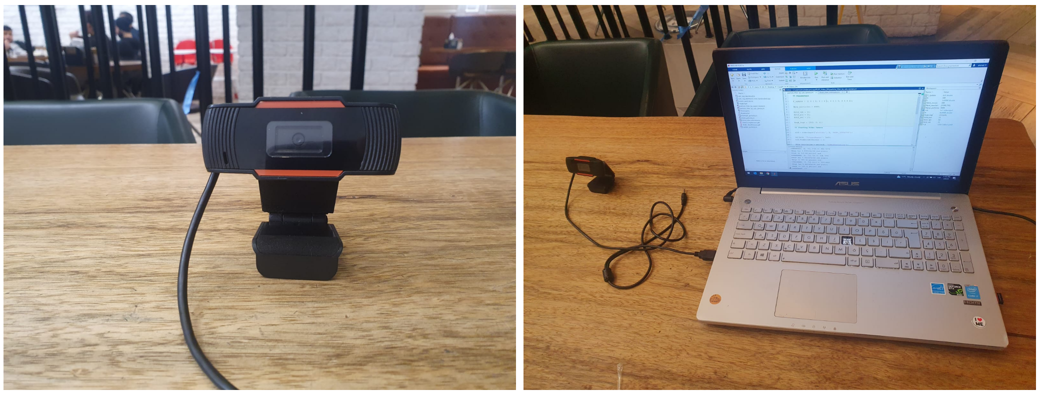


Figure 3.8: Connected devices webcam and computer

A line is considered in front of the camera in the lower area, indicating the position of the goal line. The ball is observed when it is thrown from the side in front of the webcam. The target ball is held by hand with a small thread at the top. On the other side, the ball is held at the starting point. After running the program, recording starts after a short time.

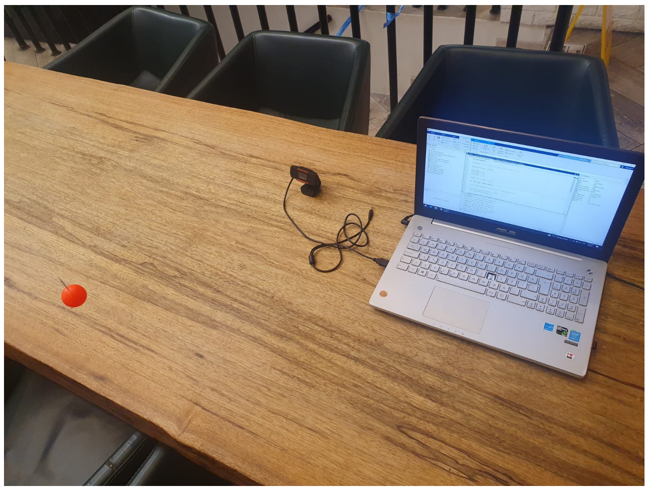
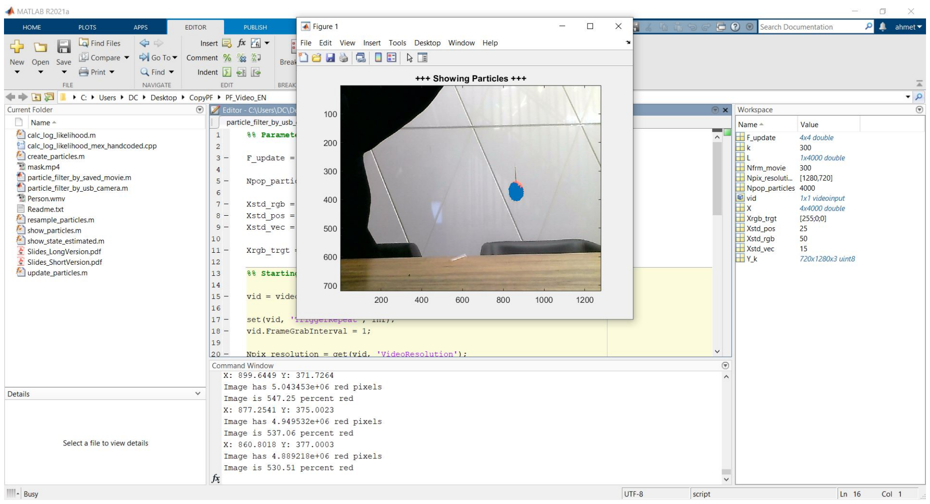


Figure 3.9: Target object and webcam position

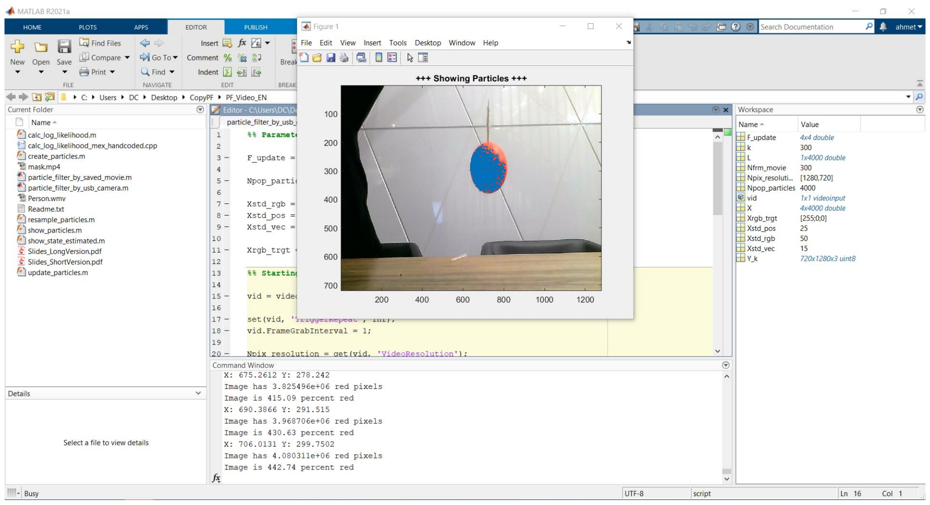
On the first screen, blue colored particles are displayed at random positions in the pixel area of the camera. When the ball is identified at the starting point, the particles move towards the red colored ball and stick to it to follow the target object. On the other side, a yellow star is displayed on the screen. This star indicates the estimation of the posterior movements of the ball.



```
X: 899.6449 Y: 371.7264
Image has 5.043453e+06 red pixels
Image is 547.25 percent red
X: 877.2541 Y: 375.0023
Image has 4.949532e+06 red pixels
Image is 537.06 percent red
X: 860.8018 Y: 377.0003
Image has 4.889218e+06 red pixels
Image is 530.51 percent red
```

Figure 3.10: Identification of target object to the particles

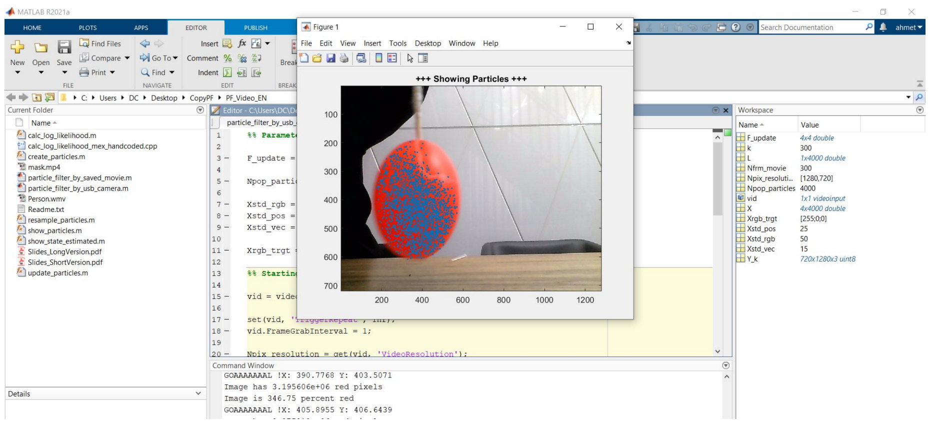
In the beginning the ball is released from the starting point to move it to any point on the camera view. When the ball moves to the goal line, the estimation star moves with the ball to indicate the posterior position of the target object. The final result of the estimation is displayed in the terminal area of Matlab as x and y position. Then the next position of the object is defined at each moment of the object with respect to the pixel change. So the next position of the target object is now estimated.



```
X: 675.2612 Y: 278.242
Image has 3.825496e+06 red pixels
Image is 415.09 percent red
X: 690.3866 Y: 291.515
Image has 3.968706e+06 red pixels
Image is 430.63 percent red
X: 706.0131 Y: 299.7502
Image has 4.080311e+06 red pixels
Image is 442.74 percent red
```

Figure 3.11: Released target object and its posterior positions on the terminal

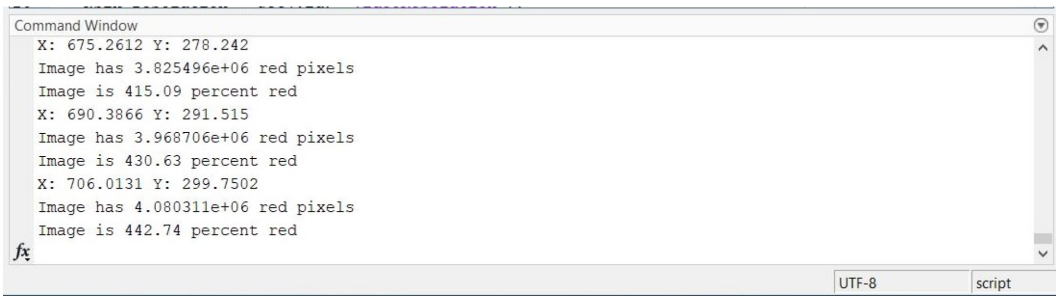
Another feature is that the message "Goal !" is displayed on the terminal when the ball crosses the threshold.



```
GOAAAAAAAL !X: 390.7768 Y: 403.5071
Image has 3.195606e+06 red pixels
Image is 346.75 percent red
GOAAAAAAAL !X: 405.8955 Y: 406.6439
```

Figure 3.12: Target object exceeds the threshold and information message shown on the terminal

The distance is determined based on the pixels of the target object in a percentage unit that depends on how close the ball is to the camera. The message is displayed on the terminal if the number of pixels of the ball on the camera image is large enough in the percentage unit.



```
X: 675.2612 Y: 278.242
Image has 3.825496e+06 red pixels
Image is 415.09 percent red
X: 690.3866 Y: 291.515
Image has 3.968706e+06 red pixels
Image is 430.63 percent red
X: 706.0131 Y: 299.7502
Image has 4.080311e+06 red pixels
Image is 442.74 percent red
```

Figure 3.13: Values of X and Y location of particles, pixel density and rgb color percentage values of the target object shown on the terminal section

Chapter 4

CONCLUSION

This thesis clarifies the development of a real-time object tracking model using a particle filtering algorithm is explained to provide an efficient estimation mechanism in real-time video images. Also, the position of the target object is determined using the red ball in the form of *x* axis and *y* axis. The next position of the target object is calculated using the estimation function shown as a yellow star symbol and output in the terminal area of Matlab. The estimated position can be used in widely used hardware systems for object tracking projects. The distance feature was measured in percentage, relative to the pixel density of the target object using a USB webcam. The proposed particle filter model was developed using Matlab and implemented using a USB webcam and a specific colored small ball. No additional system was required to implement the object tracking model itself.

In the experiment of the proposed particle filtering algorithm, the velocity of the target object was set to a constant velocity. First, the target object is identified in front of the camera and observed that the blue colored particles start searching for the specific colored object in the RGB code and move toward the object. The object is manually released at a constant velocity throught webcam at a constant speed to see the estimated position in the two-dimensional *x* and *y* coordination with respect to the pixels of the USB webcam image. The estimated *x* and *y* axis information is displayed on the terminal in real time. Also, the distance between the located USB webcam and the target object is displayed on the terminal as a percentage of the number of pixels of the target object. A threshold is set to determine whether or not the target object crosses the goal line. If the pixel percentage of the ball exceeds the threshold, the terminal displays the message "GOAL!".

In summary, the future movements of the target object in the current time interval are estimated because the target object is moving. The distance variable is controlled by the percentage of the number of pixels of the target object with respect to the distance to the camera image. In the future work, the algorithm will be developed to convert this simulated system into a hardware system that requires additional devices controlled by the proposed particle filter algorithm as a robot model for the purpose of object tracking and device control.

REFERENCES

[1] Arulampalam M., Maskell S., Gordon N., Clapp T. (2002). A tutorial on particle filters for online nonlinear / non-Gaussian Bayesian tracking. *IEEE Trans. Signal Process* 50 (2), 174-189.

[2] Collins R. T., Seitz S. M., Szeliski R. (2000). Using particles to track camera motion. *Proceedings of the International Conference on Computer Vision* 2, 1274-1281.

[3] Doucet A., De Freitas N., Gordon N. (2001). Sequential Monte Carlo methods in practice, Springer.

[4] Fisher J. W. (2003). Object tracking using particle filters. *Proceedings of the International Conference on Computer Vision*, 849-856.

[5] Gustafsson F., Gunnarsson F., Bergman N., Forssell U., Jansson J., Karlsson R., Nordlund P. J. (2002). Particle filters for positioning, navigation and tracking. *IEEE Transactions on Signal Processing* 50 (2), 425-437.

[6] Jun S. L., Chen R. (2018). Sequential Monte Carlo methods for dynamic systems. *American Statistical Association* 93 (443), 1032-1044.

[7] Casella G., Robert C. P. (1996). Rao-Blackwellisation of sampling schemes. *Biometrika* 83 (1), 81-94.

[8] Del Moral P. (1997). Nonlinear filtering: Interacting particle resolution. *Comptes Rendos de Academie des Sciences Series I-Mathematics* 325 (6), 653-658.

[9] Sarkka S., Vehtari A., Lampinen J. (2007). Rao-Blackwellized particle filter for multiple target tracking. *Information Fusion* 8, 2-15.

[10] Isard M., Blake A. (1998). Condensation-conditional density propagation for visual tracking. *Int. J. Comput. Vision* 29 (1), 5-28.

[11] Akashi H., Kumamoto H. (1977). Random sampling approach to state estimation in switching environments. *Automatica* 13, 429-434.

[12] Reza, A. A., (2017). Particle filter and its variants. *Journal of Engineering and Applied Sciences* 12 (22), 5221-5229.

[13] Nummiaro K., Koller-Meier E., Van Gool L. (2003). An adaptive color-based particle filter. *Image Vision Comput.* 21 (1), 99-110.

[14] Zhu Z. Particle filter algorithm and its application. Beijing: science.

[15] Zhang T., Xu C., Yang M. H. (2017). Multi-task correlation particle filter for robust object tracking. *Proceedings of the International conference on computer vision and pattern recognition*, 4819-4827.

[16] Gao M. L., Li L., Sun X. M., Yin L.J., Li H. T., Luo D. S. (2015). Firefly algorithm based particle filter method for visual tracking. *Optik-International*

Journal for Light and Electron Optics 126 (18), 1705-1711.

[17] Isard M., Blake A. (1998). ICONDENSATION: unifying low-level tracking in a stochastic framework. *Proceedings of European Conference on Computer Vision (ECCV'98)* 1, 893-908.

[18] Julier S. J., Uhlmann J. K. (1995). A new extension of the Kalman filter to nonlinear systems. *Proceedings of the International Conference on Aerospace and Electronic Systems* 3, 1255-1265.

[19] Perez P., Hue C., Vermaak J., Gangnet M. (2002). Color-based probabilistic tracking. *Proceedings of European Conference on Computer Vision (ECCV'02)* I, 661-675.

[20] Guirao L. R., Morin D. (2016). Kalman filter and particle filter for real time object tracking. *KTH Royal Institute*, 1-7.