İSTANBUL TECHNICAL UNIVERSITY ★ INSTITUTE OF SCIENCE AND TECHNOLOGY

3D MODEL BASED STOCHASTIC TRACKING OF LICENSE PLATES IN VIDEO SEQUENCES

Ph.D. Thesis by
İlhan Kubilay YALÇIN, M.Sc.

Department : **Computer Engineering**

Programme: **Computer Engineering**

JUNE 2007

İSTANBUL TECHNICAL UNIVERSITY ★ INSTITUTE OF SCIENCE AND TECHNOLOGY

3D MODEL BASED STOCHASTIC TRACKING OF LICENSE PLATES IN VIDEO SEQUENCES

Ph.D. Thesis by
İlhan Kubilay YALÇIN, M.Sc.
504002301

Date of submission : 30 January 2007
Date of defence examination: 1 June 2007

JUNE 2007

<u>**İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ**</u>

VİDEO DİZİLERİNDEKİ ARAÇ PLAKALARININ ÜÇ BOYUTLU MODEL YARDIMIYLA STOKASTİK YÖNTEMLERLE İZLENMESİ

DOKTORA TEZİ

Y. Müh. İlhan Kubilay YALÇIN

504002301

Tezin Enstitüye Verildiği Tarih : 30 Ocak 2007

Tezin Savunulduğu Tarih : 1 Haziran 2007

HAZİRAN 2007

PREFACE

I am very grateful to my dissertation supervisor Prof. Dr. Muhittin Gökmen for his encouragement, guidance and ideas over the past seven years. I also wish to thank the research assistant Abdulkerim Çapar for his sincere companionship.

Most of all, I would like to thank my mother Münevver and my father Ferit for their graceful support during all my education. I would like to express my deepest thanks especially to my wife Aysel for her continuous motivation and encouragement without which I could not finish my dissertation.

JUNE 2007 İlhan Kubilay YALÇIN

CONTENTS

LIST OF ABBREVIATIONS

3D	: Three-Dimensional
2D	: Two-Dimensional
AIM	: Accumulative Intensity Morphing
ASIR	: Auxiliary Sampling Importance Re-sampling
AVI	: Automated Vehicle Identification
DE	: Differential Evolution
DEMC	: Differential Evolution Markov Chain
DPC	: Direct Parameter Calibration
EA	: Evolutionary Algorithm
EKF	: Extended Kalman Filter
FEMZ	: Farksal Evrim Markov Zinciri
FOV	: Field of View
HMM	: Hidden Markov Model
GA	: Genetic Algorithm
GRV	: Gaussian Random Variables
IR	: Infrared Camera
LF	: Likelihood Function
LMMSE	: Least Minimum Mean Square Error
LP	: License Plate
LS	: Least Squares
LSE	: Least Squares Estimator
MAP	: Maximum A Posteriori
MCMC	: Markov chain Monte Carlo
MLE	: Maximum Likelihood Estimator
MMSE	: Minimum Mean Square Error
MRF	: Markov Random Field
OCR	: Optical Character Recognizer
PDF	: Probability Distribution Function
ROI	: Region Of Interest
SGA	: Simple Genetic Algorithm
SIR	: Sampling Importance Re-sampling
SIS	: Sequential Importance Sampling
SOM	: Self Organizing Map
SW	: Sliding Windows
VLPR	: Visual License Plate Recognition

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS

s_k	: System state vector at time step *k*
s_k^n	: A sample (particle) at time step *k*
s_G^n	: A sample (particle) in generation *G*
b_k	: Measurement vector at time step *k*
B	: All the measurements $b_{1:k}$
$A(t)$	: System transition matrix at time *t*
$B(t)$	: System input gain at time *t*
$C(t)$	: System measurement matrix at time *t*
$D(t)$	: System process noise gain at time *t*
$\Lambda(s)$	: Likelihood function
$u(t)$	: System input at time *t*
Q	: Process noise covariance matrix
R	: Measurement noise covariance matrix
f_k	: State transition function at time step *k*
h_k	: Measurement function at time step *k*
w_k	: System noise at time step *k*
v_k	: Measurement noise at time step *k*
$\Gamma(k)$	: Noise gain at time step *k*
x, y, z	: Cartesian coordinates
α, β, γ	: Euler angles
$N(0, \sigma)$	: Gaussian noise zero mean sigma standard deviation
c_k	: License plate speed at time step *k*
P_k	: Covariance at time step *k*
K_k	: Kalman gain at time step *k*
π_k^n	: Likelihood value for a sample at time step *k*
u_{G+1}^n	: Crossed over vector for generation *G*
v_{G+1}^n	: Perturbated vector for generation *G*

3D MODEL BASED STOCHASTIC TRACKING OF LICENSE PLATES IN VIDEO SEQUENCES

SUMMARY

The aim of this work is 3D tracking of license plates from video in order to determine the state (spatial position and 3D orientation) of the license plate from sequential frames of the video. This can be accomplished in a brute force by testing every possible orientation and translation and then selecting the one that best fits the current frame. If all six degrees of spatial freedom of the object are to be determined, the state space of the object is six dimensional. Setting the number of possible values of each degree of freedom to 100, the task of tracking by brute force then requires 100^6 comparisons of a state with the image data. Even with such a limited resolution and a six dimensional feature space, it is clear that, it is computationally impossible to perform tracking in real time by brute force. That is where stochastic tracking is meaningful. A stochastic process is one whose behavior is non-deterministic in that the next state of the environment is partially but not fully determined by the previous state of the environment. Instead of comparing every possible configuration of the object with each video frame, the idea behind stochastic tracking is to make a set of guesses of the state, compare these guesses with the current frame, and use the result of this comparison as the basis for a new set of guesses when the next frame comes. The new guesses are made by selecting the best guesses from the last frame and applying a model of the movement of the object from one frame to the next. The set of guesses (called particles or samples) will frame by frame converge around the correct state of the object.

In recent years, there has been a great interest in applying Particle Filtering to computer vision problems. The specialized Particle Filtering method for computer vision problems is introduced as Condensation or Sequential Importance Sampling. Condensation algorithm utilizes factored sampling and given dynamic models to propagate an entire probability distribution for object position and shape over time. It can perform successfully robust tracking of object motion. On the other hand, its convergence greatly depends on the trade off between the number of particles/hypotheses and the fitness of the dynamic model. For example, in cases where the dynamics are complex or poorly modeled, thousands of samples are usually required for real applications. In order to improve the performance of the Condensation algorithm, we propose DEMC Particle Filter, which is an integration of the Differential Evolution and Particle Filtering. We utilize DEMC Particle Filter for tracking the license plates in 3D from monocular camera view. We compared the performance of the Auxiliary Particle Filter, Condensation Algorithm, Genetic Condensation Algorithm and DEMC Particle Filter. With the same computation complexity, DEMC Particle Filter outperforms other three algorithms regarding error rate and robustness.

VİDEO DİZİLERİNDEKİ ARAÇ PLAKALARININ ÜÇ BOYUTLU MODEL YARDIMIYLA STOKASTİK YÖNTEMLERLE İZLENMESİ

ÖZET

Bu çalışmanın amacı, araç plakalarının üç boyutlu uzay içerisindeki konum ve yönelimlerinin bulunması için video görüntüsünden izlenmesidir. İzleme işlemi her bir olası konum ve yönelimi deneyerek ve o andaki video karesine en iyi uyanı seçerek gerçekleştirilebilir. Eğer nesnenin altı dereceli uzaysal serbestliği belirlenmek isteniyorsa, durum uzayı altı boyutlu alınabilir. Her serbestlik derecesi için olası değerleri 100 kabul edersek, bu nesneyi olası her durumu deneyerek izleyebilmek için görüntü verisi üzerinde 100^6 karşılaştırma yapmamız gerekmektedir. Bu sınırlı çözünürlük ve altı dereceli serbestlik uzayında dahi, bu şekilde gerçek zamanlı izleme yapmanın mümkün olmadığı açıktır. Bu aşamada stokastik izleme kavramı anlamlı olmaktadır. Stokastik süreç bir önceki system durumunun bir sonraki durumu tam olarak belirlemediği determinist olmayan bir süreçtir. Stokastik izlemenin ardında yatan düşünce, her olası nesne durumunu denemek yerine, durum hakkında tahminlerde bulunmak ve bu tahminleri o anki video karesi ile karşılaştırarak sonuçları bir sonraki video karesi için tahmin yapmakta kullanmaktır. Bir sonraki video karesi için yeni tahminler son video karesi için yapılan karşılaştırmalarda en iyi sonucu veren tahminler kullanılarak ve bu tahminler öngörülen dinamik model ile zamanda ilerletilerek yapılır. Bu tahmin kümesi (parçacık veya örnekler) zaman içinde nesnenin gerçek durumuna yakınsayacaktır.

Son yıllarda, bilgisayar ile görüntü işleme problemlerinde Parçacık Filtreleri'nin kullanımına yönelik bir ilgi görülmektedir. Bilgisayarla görüntü işleme problemlerinde kullanılan özel Parçacık Filtresi'ne Yoğunlaştırma algoritması veya Ardışıl Önem Örnekleme denmektedir. Yoğunlaştırma algoritması Çarpan Örneklemesi ve bütün olasılık dağılımını zaman içinde ilerletmek için dinamik bir model kullanmaktadır. Bu yöntem hareketli nesneler için gürbüz bir izleme olanağı sunmaktadır. Öte yandan, bu algoritmanın yakınsaması büyük oranda parçacık sayısı ve dinamik modelin doğruluğu arasındaki ilişkiye bağlıdır. Örnek olarak, dinamik modelin karmaşık olduğu veya kötü modellenmiş bir sistemde, gerçek bir uygulama için binlerce parçacık gerekmektedir. Yoğunlaştırma algoritmasını iyileştirmek amacıyla FEMZ Parçacık Filtresi'ni öneriyoruz, bu algortima Farksal Evrim ve Parçacık Filtresinin bir birleşimidir. FEMZ Parçacık Filtresi'ni üç boyutlu uzayda tek kamerayla araç plakası konum ve yöneliminin izlenmesi için kullandık. Yardımcı Parçacık Filtresi, Yoğunlaştırma Algoritması, Genetik Yoğunlaştırma Algoritması ve FEMZ Parçacık Filtresi'nin izleme başarımlarını karşılaştırdık. Aynı hesaplama yükü altında, FEMZ Parçacık Filtresi diğer üç algoritmaya göre çok daha iyi başarım göstermektedir.

1. INTRODUCTION

Automated Vehicle Identification (AVI) is still an important research issue and drawing attention in machine vision community. Its potential commercial applications are automatic barrier systems, automatic payment of parking fee or highway toll fee, automatic detection of a stolen vehicle, automatic calculation of traffic density and so on.

License plate enables us to identify a vehicle and its owner, because it has peculiar information. License plate recognition is the most effective method for identification of the vehicle. A suitable and promising solution to vehicle identification is visual recognition of the license plate from camera view. This approach is applicable because it does not require vehicles to carry additional equipment such as special RF transmitters.

However, visual license plate detection and recognition is a very difficult task. It is quite a challenging problem because vehicles are running in an outdoor environment, where lighting conditions can change rapidly, weather conditions can cause poor image quality, license plates can be dirty or in poor condition and occlusions can occur frequently. Therefore, Visual License Plate Recognition (VLPR) systems may fail because of uncontrollable external conditions. Beside the challenging nature of the problem, the high-dimensional nature of the VLPR problem may impose a significant computational load on the target processing platform.

VLPR systems can be separated into three major parts. These are license plate detection (segmentation), license plate tracking and license plate recognition. In these three parts, license plate detection and tracking can be considered the most important step in the VLPR systems. License plate detection involves a search over the given image for candidate regions, which has the extracted license plate pattern characteristics. Many techniques are proposed for extracting plate region, where vertical edge density caused by license text, corner extraction and plate color features are generally evaluated for locating the license plate. These techniques also utilize a search method for finding the appropriate license plate regions. These methods include Genetic algorithms [5,6], Means Shift algorithm [4] and various different heuristic search techniques, which we will give examples in the next sections. Searching the license plate regions and locating in each sequential video frame is an expensive processing. Instead, once an initial location for the license plate is given, it is possible to track the license plate in consecutive video frames. In our work, we focused on mostly the tracking step of the VLPR systems. In order to improve the performance of the well-known Condensation algorithm, we propose DEMC Particle Filter, which is an integration of the Differential Evolution and Particle Filtering. We utilize DEMC Particle Filter for tracking the license plates in 3D from monocular camera view.

This thesis is organized as follows: In the following subsections, we will give a brief summary on the related research in literature and the statement of objectives for this thesis. Section 2 describes the basics of Statistical Bayesian Estimation. The section includes the Bayesian approach, introduces several estimators and state space form. Section 3 provides information on Markov Chain Monte Carlo Methods. Particle filtering is also introduced in this section. Section 4 describes the concept of Stochastic Optimization. Genetic algorithms and evolutionary algorithms are all included in this section. Section 4.4 is about DEMC sampling. These sections contain basic theoretical background for our application. Section 5 introduces our application of 3D License Plate Tracking and its problem statement. Section 6 includes the basics of 3D Model Based Tracking. Section 7 introduces our novel approach DEMC Particle Filter. Section 8 is about utilizing the DEMC Particle Filter for our application. Section 9 presents the experimental results. Finally, section 10 is the conclusion remarks.

1.1 Related Work

Even though, we divided a typical VLPR system into three parts, namely detection, tracking and recognition, all the processing starts with detecting a license plate in the video frame. There are many creative approaches for detecting and locating the license plates in a scene, nevertheless most of these techniques utilize the presence of a text in the license plate frame.

The text locating-based technique depends on the fact that there is frequent change of the intensity caused by the characters in the plate. The edge of the characters is extracted and connected together using morphological operation to represent the plate region [1-7]. However, in the complex scene the normal morphological techniques such as closing or dilating produce noisy output and result in too many candidates for plate regions. The edge or plate boundary detection suffers dramatically because of uncertainty of the edge, caused by various types of the plate as well as identifying plate area from incomplete or broken edges. The process ends up with producing too many candidate regions, which is difficult for segmentation. In order to overcome the difficulties of morphing, Accumulative Intensity Morphing (AIM) algorithm is introduced in [1]. The basic concept of the algorithm is to create a connected region from a group of pixels that conform to pre-defined conditions.

Another text locating-based technique is proposed in [2] to extract and recognize license plates of motorcycles and vehicles on highways. In the first stage, a block-difference method is used to detect moving objects. In the second stage, an edge screening method based on the projection of edge magnitudes is used to find two peaks in the projection histograms to bound license plates. In the third stage, character images are segmented and recognized.

An alternative text locating-based technique in [3] proposes a method of extracting license plate based on wavelet transform. The proposed system consists of three main stages, wavelet transform, roughly extracting candidate regions based on high frequency features and locating the exact license plate.

Mean shift algorithm is a nonparametric statistical method for seeking the main modes of a point sample distribution. The mean shift estimate of the gradient of a density function and the associated iterative procedure of mode seeking is applied to image segmentation in [4]. The candidate regions are obtained by applying mean shift segmentation. Three features are defined and extracted in order to detect the license plate area from segmented candidate areas, namely, rectangularity, aspect ratio and edge density.

Genetic search algorithm is used for the license plate locating problem in [5]. In order to overcome the degradation of license plates in vehicle images under various illumination conditions, a windowed local area is adaptively binarized and the feature vector is obtained from a bank of filters. After that, the real integer coded genetic algorithm is proposed to search the whole image for the most promising area.

A color-based approach is normally useful and fast. However, the plate color must be different from background color and color is not stable when the lighting conditions change especially when using IR cameras at night. Color-based approaches basically fail due to change in lighting conditions. For example, color of an object varies significantly from outdoor to indoor environment.

A distributed genetic algorithm based segmentation approach to plate region extraction is presented in [6]. In the segmentation stage, a label image is generated with a distributed genetic algorithm. The chromosome is composed of two parts. One part a chromosome is the label assigned to the pixel on which it is lying. The other part is coding for the feature vector. For each chromosome, the algorithm computes a distance between the actual feature vector measured on the image at its location and the vector for which it is coding. Next step is to replace each chromosome by a neighbor (possibly itself), according to the fitness of the neighboring chromosomes. For each chromosome, the algorithm looks for a neighboring coding, which has the same label. Current chromosome is recombined by this neighbor. Finally from the label image, the plate region is extracted by applying some heuristic rules.

The work in [7] presents a method that uses a license plate as an inherent car feature. The work assumes that, the license plate color is normally distributed and builds a statistical representation of the license plate color space. Their technique allows classification using a known number of classes including a rejection class. Statistical decision considers the area, position, dimension and orientation of the extracted candidate regions.

The plate detection algorithm can process the whole image. Such a treatment would be expensive in term of processing time. For this reason, a tracking technique should be applied allowing us to make a prediction of the probable plate's position in the next image, thus greatly reducing the image area (called the ROI Region Of Interest) in which the detection process must be carried out. The tracking technique is utilized for estimating the plate center's position in the next frame.

The major development in filter theory was introduced by Kalman who noticed that for linear Gaussian problems can be solved recursively. His solution was the famous

Kalman Filter [23]. After the Kalman Filter, a number of different ideas for nonlinear and non-Gaussian filtering problems have been proposed. Some of these approximate filters include the popular Extended Kalman Filter, the Gaussian Sum Filter and other filters [24] which approximate the posterior by a mixture density. Grid based filters are also an important approach for posterior approximation. It is seen that all the filters suffer from serious drawbacks. It is as a result of these problems that there has been much interest in Particle filtering.

Most visual tracking systems either rely on contours or feature points because they are both naturally present in objects and easy to extract. Both these features have advantages and inconveniences.

Feature point matching of a 3D object model can be performed using selected control points. These control points are used for matching instead of the edge segments. The matching strategy can be simple and effective by just searching in four directions with desired pixel value [8,9].

Feature points are very well adapted to textured objects and robust to geometrical distortion and to light changes. Unfortunately, they become rare and unstable on poorly textured objects, and they are not invariant to scale changes. By contrast, contour points are informative for scenes with sharp edges and strong contrast changes, but less so in cluttered and textured scenes. In practice, there is no such sharp distinction between textured objects and objects with sharp edges. Therefore, the two information sources are, complementary, and it is useful combining them for camera tracking.

While feature points can be reliably characterized by the neighboring texture, contours information is much more ambiguous, and it is necessary to consider several possibilities when matching models against image contours.

In the model-based approach [8], the motion state is recovered from the 3D configuration with the maximal similarity. This problem has been formulated as an optimization problem and can also be treated in a probabilistic framework as the state estimation of a dynamic system. Since closed-form solutions of a highly non-linear dynamic system are intractable, Extended Kalman Filter, Unscented Kalman Filter, sequential Monte Carlo methods such as Particle Filtering was introduced to solve this problem.

The work in [8] presents an effective way to combine the information provided by edges and other feature points for the purpose of robust real-time 3D tracking. This lets the tracker handle both textured and untextured objects. They propose a method for handling multiple hypotheses for potential edge-locations. In order to efficiently consider multiple hypotheses, they introduce a new robust estimator built on the Tukey estimator.

A system to track vehicles on images taken from a mobile platform is described in [9]. The problem is addressed by modeling a 3D dynamic system, where both the acquisition platform and the tracked vehicles are represented in a state vector. From measurements obtained in every frame, this state vector is re-estimated using an Unscented Kalman Filter, instead of the Extended Kalman Filter. They assume that vehicles progress on a flat surface.

The work in [10] proposes a model based tracking method, called Appearance Guided Particle Filtering. A probability propagation model is derived from a Bayesian formulation for this framework, and a sequential Monte Carlo method is introduced for its realization. They apply the proposed method to articulated hand tracking. They study the state estimation of a dynamic system under the assumption that there are some known attractors, in addition to the initial state, in the state space. An attractor is referred as a state space vector whose observation is known. For a visual tracking problem, attractors are some reference images of the objects with known motion states, and serve as prior knowledge to guide the tracking in a high-dimensional space.

A real-time vehicle management system using a vehicle tracking and a car plate number identification technique is proposed in [11]. The system uses two cameras: one for tracking vehicles and another for capturing license plate. They track the vehicles by applying the Condensation algorithm over the vehicle's movement image captured from the first camera. To render the Condensation algorithm more effective, they build a discrete vehicle shape model by training vehicle patterns with a SOM (Self Organizing Map).

If there exists a strong perspective in traffic video, it is necessary to recover the effects of the license plate projection. Different viewpoints in 3D space produces distorted license plate images in a camera. For this reason, a method is developed in [25] to segment and to recognize characters of license plate objects undergoing various perspective views. This method is for segmenting license plate characters on a moving vehicle in actual outdoor environment and is based upon object contours.

There are also dedicated publications to moving license plate recognition. For example in [26], Yuntao Cui and Qian Huang present a new approach to extract characters on a license plate of a moving vehicle, given a sequence of perspective-distortion-corrected license plate images. Different from many existing single-frame approaches, their method simultaneously utilizes spatial and temporal information. They model the extraction of characters as a Markov Random Field (MRF), where the randomness is utilized for describing the uncertainty in pixel label assignment.

Another important problem for locating license plate in video sequences, is the low resolution of the video. In [27], a new method is proposed to extract license plate from the surveillance video, which is shot at lower resolution (320x240) as well as degraded by video compression. Morphological operations of bottom-hat and morphology gradient are utilized to detect the LP candidates, and effective schemes are applied to select the correct one.

We tried to summarize the previously published work in the literature about license plate detection and tracking. Considering all the previous effort in this area of research, we try to identify the objectives of this thesis in the next section.

1.2 Statement of Objectives

VLPR systems play an important role in traffic/vehicle surveillance/monitoring systems. There are many computer vision-based systems that recognize license plates most of which, mainly focus on the development of reliable optical character recognizer (OCR) without the emphasis on the prior problems of localizing the license plates from moving vehicles.

Primary objective of the thesis is locating license plate in 3D for consecutive video frames utilizing 3D static shape, 3D dynamic motion models and stochastic tracking algorithms. By means of projecting the estimated license plate's 3D position and orientation on to the 2D image and extracting the 2D image region, we will be able to provide a series of license plate extractions, thus a multi input system for OCR functionality.

Locating license plate in 3D will provide us predicting the license plate location in consecutive frames more precisely, because the drift of the license plate in images is the result of a 3D motion in time. Utilizing stochastic tracking algorithms will yield better estimation of a multi-modal probability distribution, thus adapts the system to multi-modal nature of the image segmentation problem.

Furthermore, obtaining the 3D position and orientation of the license plate will provide us the ability to calculate the vehicles speed, parallel to the road surface. This information is very crucial for estimating the position of the license plate in next video frame. Besides, the speed of the vehicle is an important information for traffic surveillance and monitoring functionality.

In addition to all the other considerations about tracking license plates, in the crowded urban regions, occlusion is a serious problem while tracking the license plates. Occlusions by pedestrians and other vehicles is a frequent event for license plates. We also aim to develop our approach robust to the temporary occlusions while tracking.

2. STATISTICAL BAYESIAN ESTIMATION

Statistical estimation differs from classical estimation, which deals with a deterministic but unknown constant [28,29,30,31]. There are two models, one can use in the estimation of a parameter. A nonrandom parameter can be estimated by non-Bayesian or Fisher approach. If the parameter is random variable with an associated PDF (Probability Distribution Function), we may use Bayesian approach. Here, the parameter to be estimated, is a random variable whose particular realization is estimated in statistical estimation. Bayes' theorem resulted in many recursive Bayesian estimation approaches. The motivation for applying Bayesian approaches is its twofold nature. Initially, we need to have available some prior knowledge about our estimation for feeding it into our estimator. For doing this, we need to assume a random variable with a given PDF. The Bayesian approach, when applicable, can therefore improve the estimation accuracy. In classical estimation, initial assumption is generally a unique value of the parameter. By the help of assigning a PDF to the parameter, we can devise strategies to find the estimator. The designed estimator is than said to be optimal with respect to the assumed prior PDF.

In order to study the Bayesian estimators, the concept of Bayesian cost function should be explained. Minimization of this function results in different estimators. We will introduce MMSE (Minimum Mean Square Error) and MAP (Maximum A Posteriori) etc. estimators as examples. These two estimators are the most common ones used in practice.

Before going into details of the estimators, we need to explain Bayesian approach.

2.1 Bayesian Approach

The term "parameter" is used to designate a quantity (scalar or vector valued) that is assumed to be time invariant. If it changes with time, it can be called as a "time-varying parameter", but its time variation must be "slow" compared to the state variables of a system. The estimation of the state vector of a stochastic linear dynamic system is considered later starting from Section 2.3.

In the Bayesian approach, we start with the prior PDF of the parameter from which we will derive the posterior PDF using the Bayes' formula:

$$p(s \mid B) = \frac{p(B \mid s)p(s)}{p(B)} = \frac{1}{c} p(B \mid s)p(s), \quad where\ B = b_{1:k} \tag{2.1}$$

Here c is a normalization constant, s is the system state and B is the measurements. The posterior PDF can be used to estimate the state s.

In contrast to the above, in the non-Bayesian approach there is no prior PDF associated with the parameter and thus one cannot define a posterior PDF for it. In this case, one has the PDF of the measurements conditioned on the parameter, called the likelihood function (LF) of the parameter,

$$\Lambda_B(s) \triangleq p(B \mid s)\,. \tag{2.2}$$

A measure of how "likely" a parameter value is given by the obtained observations. The likelihood function serves as a measure of the evidence from the data.

2.2 Maximum Likelihood and Maximum A Posteriori Estimators

A well-known method of estimation of nonrandom parameter is the ML estimator that maximizes the function:

$$\hat{s}^{ML}(B) = \arg\max_s \Lambda(s) = \arg\max_s p(B \mid s) \tag{2.3}$$

Here s is an unknown constant being a function of the observations.

The Maximum Likelihood Estimator MLE is the solution of the function:

$$\frac{d\Lambda_B(s)}{ds} = \frac{dp(B \mid s)}{ds} = 0 \tag{2.4}$$

The corresponding estimate for a random parameter is the Maximum A Posteriori MAP estimator, which requires the minimization of the risk function:

$$\hat{s}^{MAP}(B) = \arg\max_s p(s \mid B) = \arg\max_s [p(B \mid s)p(s)]\,. \tag{2.5}$$

When using the Bayes' formula, the last equality above does not require a normalization constant for the maximization. The MAP estimate depends on the observations and finally it is a random variable.

The non-Bayesian MLE is nothing but the Bayesian MAP estimate with complete prior ignorance, which is presented by the prior PDF. This is the point where Bayesian and non-Bayesian estimators differ.

Until now, we focused on the estimation of a single parameter, given some noisy measurements. In the following sections, we will focus on the estimation of the state vector of a stochastic dynamic system starting from the linear case.

2.3 State Space Form

For understanding the problem of tracking, consider the evolution of the state sequence $\{s_k, k \in N\}$ of a dynamic system, in most general non-linear situation, state space model for discrete-time stochastic systems is of the form,

$$s_k = f_k(s_{k-1}, u_k, v_{k-1}) \tag{2.6}$$

where $f_k : \Re^{n_s} * \Re^{n_v}$ into $\Re^{n_s}$ is a possibly nonlinear function of the state s_{k-1}, $\{v_{k-1}, k \in N\}$ is the process noise sequence, n_s, n_v are dimensions of the state and process noise vectors, respectively. N is the set of natural numbers and u_k is the known input. The objective of tracking is to recursively estimate s_k from measurements

$$b_k = h_k(s_k, w_k) \tag{2.7}$$

where $h_k : \Re^{n_b} * \Re^{n_w}$ into $\Re^{n_b}$ is a possibly nonlinear function, $\{w_k, k \in N\}$ is the measurement noise sequence, and n_b, n_w are dimensions of the measurement and measurement noise vectors, respectively. In particular, we seek filtered estimates of s_k based on the set of all available measurements $b_{1:k} = \{b_i, i = 1, ..., k\}$ up to time k.

Using Bayesian approach, the tracking problem is to calculate recursively the expectations on the state s_k given the measurements $b_{1:k}$. Thus, it is required to construct the PDF $p(s_k \mid b_{1:k})$. We assume that the initial PDF $p(s_0)$ is known as the prior. From Bayesian perspective, the PDF can be obtained recursively in two stages: prediction and update.

The needed PDF $p(s_{k-1} | b_{1:k-1})$ is known at time $k-1$. The prediction step involves using the system model $s_k = f_k(s_{k-1}, u_k, v_{k-1})$ to obtain the prior PDF of the state for time step k via the Chapman-Kolgorov equation:

$$p(s_k | b_{1:k-1}) = \int p(s_k | s_{k-1}) p(s_{k-1} | b_{1:k-1}) ds_{k-1} \qquad \textbf{(2.8)}$$

At time step k, a measurement b_k becomes available, and this may be used to update the prior (update stage) via Bayes' rule:

$$p(s_k | b_{1:k}) = \frac{p(b_k | s_k).p(s_k | b_{1:k-1})}{p(b_k | b_{1:k-1})} \qquad \textbf{(2.9)}$$

These recurrence relations form the basis for the optimal Bayesian solution.

2.4 Least Squares and Minimum Mean Square Error Estimation

Given a batch of measurements from a static system, a common estimation procedure for nonrandom parameters is the Least Squares (LS) method. Given the (scalar and nonlinear) measurements

$$b_k = h(s_k) + w_k \,. \qquad \textbf{(2.10)}$$

The Least Squares Estimator (LSE) is defined as:

$$\hat{s}^{LSE}(B) = \arg\min_s \{\sum_j^k (b_j - h(s_j, w_j))^2\} \qquad \textbf{(2.11)}$$

For random parameters, the Bayesian case of the above is the Minimum Mean Square Error (MMSE) estimator, given as:

$$\hat{s}^{MMSE}(B) = \arg\min_{\hat{s}} E[(\hat{s} - s)^2 | B] \qquad \textbf{(2.12)}$$

The solution is the conditional mean of s

$$\hat{s}^{MMSE}(B) = E[s | B] \triangleq \int_{-\infty}^{+\infty} s.p(s | B)\, ds \,. \qquad \textbf{(2.13)}$$

Batch Linear Squares Estimator and its iterated version can be used to estimate static systems, which produces a batch of output resulted from a parameter or a set of parameters.

2.5 Estimation of Linear Dynamic Systems, Kalman Filter

The state-space representation of continuous-time linear stochastic systems can be written as:

$$\dot{s}(t) = A(t)s(t) + B(t)u(t) + D(t)\tilde{v}(t) \tag{2.14}$$

Here, $s(t)$ is the system state, $A(t)$ is the system matrix, $B(t)$ is the input gain, $D(t)$ is the noise gain, $u(t)$ is the input and $\tilde{v}(t)$ is the process noise.

The output of the system is, in general

$$b(t) = C(t)s(t) + \tilde{w}(t) \tag{2.15}$$

where $b(t)$ is the output, $C(t)$ is the measurement matrix and $\tilde{w}(t)$ is the measurement noise.

In the stochastic case, the noises are usually assumed to be zero mean, white and mutually independent. If the noise is not zero mean, its mean (if known) can be taken as a known input [32].

The state equation has the following solution:

$$s(t) = F(t,t_0)s(t_0) + \int_{t_0}^{t} F(t,\tau)[B(\tau)u(\tau) + D(\tau)\tilde{v}(\tau)]d\tau \tag{2.16}$$

For a time-invariant system, assuming $t_0 = 0$, we have

$$F(t) \triangleq F(t,0) = e^{At}. \tag{2.17}$$

Some of the computational methods for the evaluation of the matrix e^{At} are infinite series, Laplace transform and polynomial interpolation.

In the state space representation of discrete-time systems, it is assumed that the input is piecewise constant, so that

$$u(t) = u(t_k) \quad where \quad t_k \le t \le t_{k+1}. \tag{2.18}$$

Then the state equation can be written as,

$$s(t_{k+1}) = F(t_{t+1},t_k)s(t_k) + G(t_{k+1},t_k)u(k) + v(t_k). \tag{2.19}$$

Sampling a time invariant continuous time system at arbitrary times, state transition matrix can be formulated as,

$$F(k) \triangleq F(t_{k+1} - t_k) = e^{(t_{k+1}-t_k)A} \tag{2.20}$$

and the input gain is,

$$G(k) \triangleq G(t_{k+1}, t_k) = \int_{t_k}^{t_{k+1}} e^{(t_{k+1}-\tau)A} B d\tau \,. \tag{2.21}$$

Finally the discrete time process noise is,

$$v(k) \triangleq v(t_k) = \int_{t_k}^{t_{k+1}} e^{(t_{k+1}-\tau)A} D\tilde{v}(\tau) d\tau \,. \tag{2.22}$$

We need to give further two equations for noise covariance matrices Q and R:

$$E[v(k)v(j)'] = Q(k)\delta_{kj} \tag{2.23}$$

$$E[w(k)w(j)'] = R(k)\delta_{kj} \tag{2.24}$$

Here Q is the process noise covariance matrix and R is the measurement noise covariance matrix.

Finally, the (dynamic) model for discrete-time linear stochastic systems can be written with the simplified index-only time notation as,

$$s(k+1) = F(k)s(k) + G(k)u(k) + v(k) \,. \tag{2.25}$$

The discrete-time measurement equation is, with a similar notation,

$$b(k) = H(k)s(k) + w(k) \,. \tag{2.26}$$

If the discrete-time system is time-invariant, that is,

$$F(k) = F, \quad G(k) = G \tag{2.27}$$

and the solution becomes very simple as:

$$s(k) = F^k s(0) + \sum_{i=0}^{k-1} F^{k-i-1}[Gu(i) + v(i)] \tag{2.28}$$

Sometimes we can include a noise gain to the state transition equation in the form of,

$$s(k+1) = F(k)s(k) + G(k)u(k) + \Gamma(k)v(k) \,. \tag{2.29}$$

Before going into the details of the Kalman filter we will note two properties of the system state. First, the propagation of the system state mean can be given as,

$$\bar{s}(k+1) = F(k)\bar{s}(k) + G(k)u(k) + \Gamma(k)\bar{v}(k) \,. \tag{2.30}$$

Defining the covariance of the state as $P_{ss}(k)$, covariance propagation equation is,

$$P_{ss}(k+1) = F(k)P_{ss}(k)F(k)' + \Gamma(k)Q(k)\Gamma(k)' . \qquad \textbf{(2.31)}$$

The discrete-time Kalman filter computes recursively the MMSE estimate of the state of a dynamic system. With some assumptions such as, the matrices F, G, H, Q, and R are assumed known and possibly time varying. The system can be time varying and the noises non-stationary. The initial state is modeled as a random variable, Gaussian distributed with known mean and covariance. The two noise sequences and the initial state are assumed to be zero mean white Gaussian noise, which are mutually independent. This all make a linear Gaussian model and a Gauss Markov process.

The Kalman filter is named after Rudolph E. Kalman, who in 1960 published his famous paper describing a recursive solution to the discrete-data linear filtering problem [23].

The Kalman filter is essentially a set of mathematical equations that implement a predictor-corrector type estimator that is optimal in the sense that it minimizes the estimated error covariance then some presumed conditions are met. Since the time of its introduction, the Kalman filter has been the subject of extensive research and application, particularly in the area of autonomous or assisted navigation. This is likely due in large part to advances in digital computing that made the use of the filter practical, but also to the relative simplicity and robust nature of the filter itself. Rarely do the conditions necessary for optimality actually exist, and yet the filter apparently works well for many applications in spite of this situation.

The Kalman filter estimates a process by using a form of feedback control thus the filter estimates the process state at some time and then obtains feedback in the form of noisy observations. The equations for Kalman filter fall into two groups namely time update equations and observation update equations. The time update equations are responsible for projecting forward in time the current state and error covariance estimates to obtain the a priori estimates for the next time step. The observation update equations are responsible for the feedback.

The time update equations can also be thought as predictor equations, while the observations update equations can be thought as corrector equations. The equations for the time and observation updates are presented below.

Time Update

$$s(k \mid k-1) = F(k).s(k-1 \mid k-1) + G(k).u(k) \qquad \textbf{(2.32)}$$

$$P_{ss}(k \mid k-1) = F(k).P_{ss}(k-1 \mid k-1).F(k)^T + Q(k) \qquad \textbf{(2.33)}$$

Observation Update

$$K(k) = P_{ss}(k \mid k-1).H(k)^T.(H(k).P_{ss}(k \mid k-1).H(k)^T + R(k))^{-1} \qquad \textbf{(2.34)}$$

$$s(k \mid k) = s(k \mid k-1) + K(k).(b(k) - H(k).s(k \mid k-1)) \qquad \textbf{(2.35)}$$

$$P_{ss}(k \mid k) = (-K(k).H(k)).P_{ss}(k \mid k-1) \qquad \textbf{(2.36)}$$

The optimal filter gain $K(k)$ is proportional to the state prediction variance and inversely proportional to the innovation $(b(k) - H(k).s(k \mid k-1))$ variance. Thus, the gain is large if the state prediction is inaccurate (has a large variance) and the measurement is accurate (has a relatively small variance)

The gain is small if the state prediction is accurate (has a small variance) and the measurement is “inaccurate” (has a relatively large variance). A large gain indicates a “rapid” response to the measurement in updating state, while a small gain yields a slower response to the measurement. In frequency domain, it can be shown that these properties correspond to a higher/lower bandwidth of the filter. A filter whose optimal gain is higher yields less “noise reduction” as one would expect from a filter with a higher bandwidth.

Under the Gaussian assumption for the initial state (or initial state error) and all the noises entering into the system, the Kalman filter is the optimal MMSE state estimator. If these random variables are not Gaussian and one has only their first two moments, then the Kalman filter algorithm is the best linear state estimator, that is, the Least Minimum Mean Square Error LMMSE state estimator.

2.6 Estimation of Nonlinear Systems, Extended Kalman Filter

The Kalman filter addresses the general problem of trying to estimate the state of a discrete time controlled process, which is governed by a linear stochastic difference equation. If the process to be estimated and/or the observation relationship to the process is non-linear, then Kalman filter is not sufficient for estimation. A Kalman filter that linearizes about the current mean and covariance is referred to as an

Extended Kalman Filter [32]. Consider the basic state-space estimation framework as in,

$$s(k) = f_k(s(k-1), u(k), w(k-1)) \quad \textbf{(2.37)}$$

$$b(k) = h_k(s(k), v(k)). \quad \textbf{(2.38)}$$

To estimate a process with non-linear difference and observation relationships, we write new governing equations that linearize the estimate.

$$s(k) = \hat{s}(k) + A(k)(s(k-1) - \hat{s}(k-1)) + W(k).w(k-1) \quad \textbf{(2.39)}$$

$$b(k) = \hat{b}(k) + H(k)(s(k) - \hat{s}(k)) + V(k).v(k). \quad \textbf{(2.40)}$$

Here A is the Jacobian matrix of partial derivatives of f_k with respect to s, W is the Jacobian matrix of partial derivatives of f_k with respect to w, H is the Jacobian matrix of partial derivatives of h_k with respect to s and V is the Jacobian matrix of partial derivatives of h_k with respect to v.

Time Update

$$s(k \mid k-1) = f_k(s(k-1 \mid k-1), u(k), 0) \quad \textbf{(2.41)}$$

$$P(k \mid k-1) = A(k).P(k-1 \mid k-1).A(k)^T + W(k)Q(k-1)W(k)^T \quad \textbf{(2.42)}$$

Observation Update

$$K(k) = P(k \mid k-1)H(k)^T.(H(k)P(k \mid k-1)H(k)^T + V(k)R(k)V(k)^T)^{-1} \quad \textbf{(2.43)}$$

$$s(k \mid k) = s(k \mid k-1) + K(k)(b(k) - h_k(s(k \mid k-1), 0)) \quad \textbf{(2.44)}$$

$$P(k \mid k) = (-K(k).H(k))P(k \mid k-1) \quad \textbf{(2.45)}$$

Given the noisy observation $b(k)$, a recursive estimation for $s(k)$ can be expressed in the form,

$$\hat{s}(k) = (\textit{prediction of } s(k)) + K(k).(b(k) - (\textit{prediction of } b(k)). \quad \textbf{(2.46)}$$

This recursion provides the optimal minimum mean-squared error (MMSE) estimate for $s(k)$ assuming the prior estimate $\hat{s}(k-1)$ and current observation $b(k)$ are Gaussian Random Variables (GRV). We need not assume linearity of the model. The optimal terms in this recursion are given by,

$$\begin{aligned}
&\hat{s}(k)^{-} = E[F(\hat{s}(k-1), w(k-1))] \\
&K(k) = P_{s_k b_k} P^{-1}_{\bar{b}_k \bar{b}_k} \\
&\hat{b}(k) = E[H(k)(\hat{s}(k)^{-}, v(k))]
\end{aligned} \quad \textbf{(2.47)}$$

where the optimal prediction of $s(k)$ is written as $\hat{s}(k)^{-}$, and corresponds to the expectation of a nonlinear function of the random variables $\hat{s}(k-1)$ and $w(k-1)$. The optimal gain term $K(k)$ is expressed as a function of posterior covariance matrices (with $\bar{b}(k) = b(k) - \hat{b}(k)^{-}$). Note these terms also require taking expectations of a nonlinear function of the prior state estimates.

The Kalman filter calculates these quantities exactly in the linear case, and can be viewed as an efficient method for analytically propagating a GRV through linear system dynamics. For nonlinear models, however, the EKF approximates the optimal terms as,

$$\begin{aligned}
&\hat{s}(k)^{-} \approx f_k(\hat{s}(k-1), \bar{w}) \\
&K(k) \approx \hat{P}_{s(k)b(k)} \hat{P}^{-1}_{\bar{b}(k)\bar{b}(k)} \\
&\hat{b}(k) \approx h_k(\hat{s}_k^{-}, \bar{v})
\end{aligned} \quad \textbf{(2.48)}$$

where predictions are approximated simply as the function of the prior mean value for estimates. The covariance is determined by linearizing the dynamic equations and then determining the posterior covariance matrices analytically for the linear system. In other words, in the EKF the state distribution is approximated by a GRV which is then propagated analytically through the "first-order" linearization of the nonlinear system. As such, the EKF can be viewed as providing "first-order" approximations to the optimal terms. These approximations, however, can introduce large errors in the true posterior mean and covariance of the transformed (Gaussian) random variable, which may lead to sub-optimal performance and sometimes divergence of the filter.

2.7 Grid Based and Approximate Grid Based Methods

Grid-based methods provide the optimal recursion of the filtered density $p(s_k)$ if the state space is discrete and consists of a finite number of states $s_{k-1}^{i}, i = 1,...,N$.

If the state space is continuous but can be divided into N cells, then grid-based method can be utilized to approximate the posterior PDF at time $k-1$, which is given by:

$$p(s_{k-1} \mid b_{1:k-1}) \approx \sum_{i=1}^{N} w_{k-1|k-1}^{i} \delta(s_{k-1} - s_{k-1}^{i}) \qquad \textbf{(2.49)}$$

Further more, the prediction and update equations can be formulated as

$$p(s_k \mid b_{1:k-1}) \approx \sum_{i=1}^{N} w_{k|k-1}^{i} \delta(s_{k-1} - s_{k-1}^{i}) \qquad \textbf{(2.50)}$$

$$p(s_k \mid b_{1:k}) \approx \sum_{i=1}^{N} w_{k|k}^{i} \delta(s_{k-1} - s_{k-1}^{i}) . \qquad \textbf{(2.51)}$$

The grid points represent regions of continuous state space, for this reason calculating the weights requires integration over these regions. A further approximation in evaluation of the weights can be made. We assume that these weights are computed at the center of the cell corresponding to s_k^i. So the weights can be computed by:

$$w_{k|k-1}^{i} \triangleq \sum_{j=1}^{N} w_{k-1|k-1}^{j} p(s_k^i \mid s_{k-1}^{j}) \qquad \textbf{(2.52)}$$

$$w_{k|k}^{i} \approx \frac{w_{k|k-1}^{i} p(b_k \mid s_k^j)}{\sum_{j=1}^{N} w_{k|k-1}^{j} p(b_k \mid s_k^j)} \qquad \textbf{(2.53)}$$

The grid should be dense to get a good approximation to the continuous state space. As the dimensionality of the state space increases, the computational cost of the approach increases significantly. Another disadvantage of grid-based methods is that the state space must be predefined and, thus, cannot be partitioned unevenly to give greater resolution in high probability density regions, unless prior knowledge is used.

Hidden Markov Model (HMM) filters are an application of such approximate grid-based methods in a fixed-interval smoothing context and have been used extensively in speech processing.

3. MARKOV CHAIN MONTE CARLO METHODS

Markov Chain Monte Carlo (MCMC) is a powerful means for generating random samples that can be used in computing statistical estimates and marginal and conditional probabilities [24,39]. Although MCMC has general applicability, one area where it has had a revolutionary impact is Bayesian analysis. MCMC has greatly expanded the range of problems for which Bayesian methods can be applied.

Sequential Monte Carlo methods found limited use in the past, except for the last decade, primarily due to their very high computational complexity and the lack of adequate computing resources of the time. The fast advances of computers in the last several years and the outstanding potential of Particle Filters have made them recently a very active area of research.

We will introduce importance sampling, rejection sampling, Metropolis method and the Gibbs sampling as a basis for the Monte Carlo methods.

Mainly the problems addressed by the Monte Carlo methods are generating samples $\{s^n\}_{n=1}^{N}$ given a probability distribution function $p(s)$ and estimating expectations of functions under this distribution is given as:

$$E[f(s)] = \int f(s)p(s)ds \tag{3.1}$$

The probability distribution function $p(s)$ is called as the target density in Monte Carlo methods. For example, this can be the posterior probability of a model's parameters given some measurements.
If the first problem is solved, estimating expectations of functions under this distribution can be solved such as,

$$E[f(s)] = \frac{1}{N}\sum_{n} f(s^n). \tag{3.2}$$

An important property of the Monte Carlo methods is its independence of the dimensionality of the space sampled. However, high dimensionality can cause other difficulties.

We will assume that the density from which we want to draw samples $p(s)$ can be calculated, at least within a constant ratio given by,

$$p(s) = \frac{p^*(s)}{Z} \qquad \textbf{(3.3)}$$

because we do not have the normalizing factor Z ,

$$Z = \int p(s)ds\,. \qquad \textbf{(3.4)}$$

Importance sampling is not a method for generating samples from $p(s)$, but estimating the expectations of a function $f(s)$. Let's assume that the density $p(s)$ is too complicated but we have a simpler density $q(s)$.

In importance sampling, we will generate *N* samples from proposal density $q(s)$. If these samples were sampled from $p(s)$, we could estimate any given function. However, when we generate samples from $q(s)$, there will be a problem of over and under representation of the areas, where $q(s)$ and $p(s)$ strongly differ. Therefore, we introduce weights,

$$w_n = \frac{p^*(s^n)}{q^*(s^n)} \qquad \textbf{(3.5)}$$

which we use to adjust the importance of each sample in our estimator:

$$E[f(s)] = \frac{\sum_n w_n f(s^n)}{\sum_n w_n} \qquad \textbf{(3.6)}$$

It can be proven that the estimate converges to the true value as *N* increases [39].

Rejection sampling is another Monte Carlo sampling technique. Again, we assume $p(s)$ to be complex density, which can be evaluated with a constant ratio. We have a simpler proposal density $q(s)$. We have a further assumption that,

$$c.q^*(s) > p^*(s) \quad for \quad \forall s \qquad \textbf{(3.7)}$$

where c is a positive constant. We generate two random numbers. The first s' is generated from $q(s)$ and the other is generated from a uniform distribution in the interval $[0, c.q(s')]$. If $u > p^*(s')$, then s' is rejected. Otherwise s' is included into the set of samples generated from $p(s)$. Rejection sampling works well if $q(s)$ is a good approximation of $p(s)$.

In large and complex problems, it is difficult to provide a $q(s)$ that is a good approximation of $p(s)$. The Metropolis algorithm [39] is useful in these cases. Metropolis algorithm utilizes a proposal density $q(s)$, which depends on the current state $s(t)$. In the simplest case, proposal density can be a Gaussian centered on the current state $q(s';s(t))$. Like the previous two Monte Carlo methods, we assume that $p(s)$ can be calculated at least within a constant ratio. In order to decide whether to accept or reject the new generated sample from $q(s';s(t))$, we have the following quantity:

$$a = \frac{p^*(s').q(s(t);s')}{p^*(s(t)).q(s';s(t))} \tag{3.8}$$

If $a \geq 1$ then the new generated sample is accepted, otherwise the new sample is accepted with probability a.

The Metropolis method is an example of a Markov Chain Monte Carlo MCMC method [39]. Compared to rejection sampling where generated candidate samples are independent from desired distribution, MCMC methods involve a Markov process in which a sequence of samples is generated, where each sample depends on the previous. Since successive samples are correlated with each other, the Markov chain should be run for a considerable time in order to generate effectively independent samples from the target distribution.

Gibbs sampling [24] also known as "heat bath" method, is a method for sampling from distributions over at least two dimensions. Gibbs sampling is a Metropolis method in which the proposal distribution is defined in terms of conditional distribution of the joint distribution $p(s)$. The target distribution is complex but we assume that its conditional distributions are easy to draw samples from.

In general, for a sample dimension of *L,* a single iteration involves sampling from one dimension at a time.

$$\begin{aligned} s_1^{(t+1)} &\sim p(s_1 \mid s_2^t, s_3^t, \ldots, s_L^t) \\ s_2^{(t+1)} &\sim p(s_2 \mid s_1^t, s_3^t, \ldots, s_L^t) \\ s_3^{(t+1)} &\sim p(s_3 \mid s_1^t, s_2^t, \ldots, s_L^t) \quad \ldots \end{aligned} \tag{3.9}$$

Gibbs sampling is like a Metropolis method in which all the proposals are accepted.

3.1 Particle Filtering

Particle filtering [48,49] is a sequential MCMC methodology where the basic idea is the recursive computation of relevant probability distributions using the concepts of importance sampling and approximation of probability distributions with discrete random measures.

The particle filter represents the underlying PDF that describes the state of an object by a set of random samples from the space on which the PDF is defined. Every sample is commonly referred to as a particle. Associated with each particle is a weight. The particle locations and weights are used to achieve "Monte-Carlo" approximations to integrals involving the unknown PDF that is being determined.

Bayesian recursive propagation of the posterior density is only a conceptual solution in that in general, it cannot be determined analytically. Kalman filter is an optimal solution to the problem under linearity and Gaussian white noise assumptions. There are also sub-optimal solutions like Extended Kalman filter, approximate grid based methods and Particle filters which approximate the optimal Bayesian solution.

Particle filter can be summarized as follows,

- Initially, all particles have equivalent weights attached to them.
- At the prediction step, the state of every particle is updated according to the motion model. An accurate dynamical model is essential for robust tracking and for achieving real-time performance.
- During the measurement step, new information that became available about the system is used to adjust the particle weights for every particle. The weight is set to be the likelihood of this particle state describing the true current state of the object, which can be computed via Bayesian inference to be proportional to the probability of the observed measurements given the particle state.
- The sample points are then redistributed to obtain uniform weighting for the next algorithm iteration by re-sampling them from the computed posterior probability distribution.

- At any time, such characteristics (position, speed etc.) can be directly computed, if desired, by using the particle set and weights as an approximation to the true PDF.

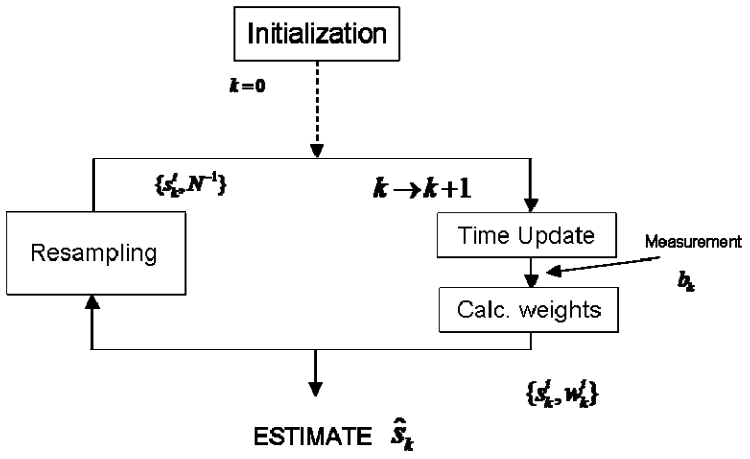


Figure 3.1: Particle Filtering Flow Chart

The flow of the algorithm is also shown in Figure 3.1, where $\{s_k^i, w_k^i\}$ are the samples and the associated weights at time step k and N is the number of samples. The algorithm works well in many cases where the Kalman filter (or the extended Kalman filter) would fail due to poor approximation of the process PDF by the Gaussian (for example, when the PDF is multimodal). An advantage of the formulation is that it can be easily applied even when the state update model and the measurement model are nonlinear since they are only evaluated in the forward direction, and need not be inverted. This allows the use of error functions that make sense for the problem, including nonlinear ones.

The first particle filtering method that we will explain is the well-known Sequential Importance Sampling (SIS) algorithm. SIS forms the basis for most sequential Monte Carlo filters.

Before detailing the algorithm, let $\{s_k^i, w_k^i\}_{i=1}^N$ is a characterization of the posterior $p(s_k \mid b_{1:k})$ where s_k^i are a set of N samples with associated weights w_k^i. The weights are normalized that $\sum_i w_k^i = 1$. The posterior density at time step k can be approximated as:

$$p(s_k \mid b_{1:k}) \approx \sum_{i=1}^{N} w_k^i \delta(s_k - s_k^i) \qquad \textbf{(3.10)}$$

The weights are assigned according to *importance sampling*. Suppose $p(s)$ and $q(s)$ are the target and the proposal densities. $p(s) \propto \pi(s)$ can be evaluated by a constant. Then a weighted approximation of the target density can be given as,

$$p(s_k) \approx \sum_{i=1}^{N} w^i \delta(s - s^i) \qquad \textbf{(3.11)}$$

where

$$w^i \propto \frac{\pi(s^i)}{q(s^i)} \qquad \textbf{(3.12)}$$

are the normalized weights assigned to each particle. We will further define the proposal and the target density as:

$$w_k^i \propto \frac{p(s_k^i \mid b_{1:k})}{q(s_k^i \mid b_{1:k})} \qquad \textbf{(3.13)}$$

If we can factorize the proposal density such that,

$$q(s_k \mid b_{1:k}) = q(s_k \mid s_{k-1}, b_{1:k}) q(s_{k-1} \mid b_{1:k}), \qquad \textbf{(3.14)}$$

we can derive the weight update equation as:

$$w_k^i \propto \frac{p(b_k \mid s_k^i) p(s_k^i \mid s_{k-1}^i) p(s_{k-1}^i \mid b_{1:k-1})}{q(s_k^i \mid s_{k-1}^i, b_{1:k}) q(s_{k-1}^i, b_{1:k-1})} = w_{k-1}^i \frac{p(b_k \mid s_k^i) p(s_k^i \mid s_{k-1}^i)}{q(s_k^i \mid s_{k-1}^i, b_{1:k})} \qquad \textbf{(3.15)}$$

By using the fact that,

$$p(s_k \mid b_{1:k}) \propto p(b_k \mid s_k) p(s_k \mid s_{k-1}) p(s_{k-1} \mid b_{1:k}). \qquad \textbf{(3.16)}$$

Finally, the posterior filtered density $p(s_k \mid b_{1:k})$ can be approximated as,

$$p(s_k \mid b_{1:k}) \approx \sum_{i=1}^{N} w_k^i \delta(s_k - s_k^i). \qquad \textbf{(3.17)}$$

SIS algorithm consists of recursive propagation of the weights and particles as each measurement is received sequentially.

A common problem with the SIS particle filter is the degeneracy phenomenon, where after a few iterations; all but one particle will have negligible weight. A measure for the degeneracy is proposed as the effective sample size N_{eff}, which is defined as,

$$N_{eff} = \frac{N}{1+Var(w_k^{*i})} \quad \textbf{(3.18)}$$

where the w_k^{*i} are the true weights,

$$w_k^{*i} = \frac{p(s_k^i \mid b_{1:k})}{q(s_k^i \mid s_{k-1}^i, b_{1:k})}. \quad \textbf{(3.19)}$$

We can not calculate true weights exactly, but an estimate for N_{eff} is given as,

$$N_{eff} = \frac{1}{\sum_{i=1}^{N}(w_k^i)^2} \quad \textbf{(3.20)}$$

where w_k^i are the normalized weights obtained from the weight update step. A small N_{eff} indicates severe degeneracy. There are two main approaches to overcome the degeneracy problem. The first is the optimum choice of the importance density. The optimum importance function minimizes the variance of the true weights w_k^{*i} and it is given by:

$$\begin{aligned} & q(s_k \mid s_{k-1}^i, b_k)_{optimum} = p(s_k \mid s_{k-1}^i, b_k) \\ & = \frac{p(b_k \mid s_k \mid s_{k-1}^i)p(s_k \mid s_{k-1}^i)}{p(b_k \mid s_{k-1}^i)} \end{aligned} \quad \textbf{(3.21)}$$

Utilizing the above equation in weight update stage, we conclude,

$$w_k^i \propto w_{k-1}^i p(b_k \mid s_{k-1}^i). \quad \textbf{(3.22)}$$

The optimal importance density suffers from two important problems. It requires the evaluation of the probability $p(s_k \mid s_{k-1}^i, b_k)$ and it requires an integration over the new state.

It is a common choice to accept the importance density to be the prior,

$$q(s_k \mid s_{k-1}^i, b_k) = p(s_k \mid s_{k-1}^i) \quad \textbf{(3.23)}$$

which yields,

$$w_k^i \propto w_{k-1}^i p(b_k \mid s_k^i). \quad \textbf{(3.24)}$$

This is the choice for importance density for Condensation Algorithm, which will be introduced later in the thesis.

The second method for overcoming degeneracy is “re-sampling”. The basic idea behind re-sampling is to eliminate particles that have small weights and to concentrate on particles with large weights. There are several re-sampling techniques such as stratified sampling, residual sampling, systematic sampling and so on. Although re-sampling step reduces the effects of the degeneracy problem, it introduces new problems. The particles with high weights are selected many times. This yields loss of diversity, which is known as sample impoverishment.

Implementation of the systematic re-sampling algorithm is detailed below as an example,

Re-sampling Algorithm

```
• Initialize the cumulative distribution function CDF values starting from $c_1 = 0$
• For $i = 2 : N$
    o Calculate CDF as $c_i = c_{i-1} + w_k^i$
• Draw a starting point as $u_1 = U[0, N^{-1}]$
• For $j = 1 : N$
    o Calculate next $u_j = u_1 + N^{-1}(j-1)$
    o Until $u_j > c_i$ , increment i.
    o Update sample $s_k^{j*} = s_k^i$
    o Update weight $w_k^j = N^{-1}$
    o Update parent $i^j = i$
```

A generic particle filter algorithm steps are formulated below,

Generic Particle Filter

```
• For $i = 1:N$
    o Draw $s_k^i \sim q(s_k \mid s_{k-1}^i, b_k)$
    o Calculate weights $w_k^i$ for each particle
• Sum the weights $sum = \sum_{i=1}^{N} w_k^i$
• Normalize weights $w_k^i = w_k^i / sum$
• Calculate $N_{eff}$
• If $N_{eff} < N_T$
    o Resample
```

Here we will detail only two specific implementations of the generic particle filter that are Sampling Importance Re-sampling (SIR) and the Auxiliary Sampling Importance Re-sampling. SIR filter is known as Condensation Algorithm (Conditional Density Propagation) in computer vision society.

SIR filter includes the following steps,

SIR Filter

```
• For $i = 1:N$
    o Draw $s_k^i \sim p(s_k \mid s_{k-1}^i)$
    o Calculate weights $w_k^i = p(b_k \mid s_k^i)$ for each particle
• Sum the weights $sum = \sum_{i=1}^{N} w_k^i$
• Normalize weights $w_k^i = w_k^i / sum$
    • Resample
```

ASIR (Auxiliary SIR or Auxiliary Particle Filter) filter [52] is also a variant of the SIR filter. This filter can be derived from SIR by assuming the importance density as $q(s_k, i \mid b_{1:k})$. By applying Bayes' rule we obtain,

$$q(s_k, i \mid b_{1:k}) \propto p(b_k \mid s_k)\, p(s_k \mid s_{k-1}^i) w_{k-1}^i \quad \textbf{(3.25)}$$

which can be written by an approximation,

$$q(s_k, i \mid b_{1:k}) \propto p(b_k \mid s_k)\, p(s_k \mid s_{k-1}^i) w_{k-1}^i \quad \textbf{(3.26)}$$

where $\mu_k^i = E[s_k \mid s_{k-1}^i]$ which is a characterization of s_k. Applying this importance density to the weight update step, we obtain,

$$w_k^j \propto w_{k-1}^{i^j} \frac{p(b_k \mid s_k^j)\, p(s_k^j \mid s_{k-1}^{i^j})}{q(s_k^j, i^j \mid b_{1:k})} = \frac{p(b_k \mid s_k^j)}{p(b_k \mid \mu_k^{i^j})}. \quad \textbf{(3.27)}$$

Finally, the steps of the ASIR filter can be summarized as follows,

ASIR Filter

```
• For $i = 1:N$
    o Calculate $\mu_k^i = E[s_k \mid s_{k-1}^i]$
    o Calculate weights $w_k^i = q(i \mid b_{1:k}) \propto p(b_k \mid \mu_k^i) w_{k-1}^i$ for each particle
• Sum the weights $sum = \sum_{i=1}^{N} w_k^i$
• Normalize weights $w_k^i = w_k^i / sum$
• Resample
• For $j = 1:N$
    o Draw $s_k^j \sim p(s_k \mid s_{k-1}^{i^j})$
    o Calculate weights $w_k^j = \frac{p(b_k \mid s_k^j)}{p(b_k \mid \mu_k^{i^j})}$ for each particle
• Sum the weights $sum = \sum_{i=1}^{N} w_k^i$
• Normalize weights $w_k^j = w_k^j / sum$
```

4. STOCHASTIC OPTIMIZATION METHODS

Genetic and Evolutionary algorithms [33-38] are two well-known stochastic optimization techniques. Before introducing these global optimization techniques, we need to mention the concepts of stochastic optimization. There are basically two main properties of the stochastic optimization techniques. These are,

- There is random noise in the measurements.
- There is a random choice made in the search direction as the algorithm iterates toward a solution.

In some optimization problems, the model cannot be completely defined because it depends on quantities that are unknown at the time of formulation. Rather than just use a best guess for the uncertain quantities, we may obtain more useful solutions by utilizing additional knowledge about these quantities into the model. Stochastic optimization algorithms use this level of the uncertainty to produce solutions that optimize the expected performance of the model. Many algorithms for stochastic optimization do, however, proceed by formulating one or more deterministic sub problems. Stochastic and robust optimizations have seen a great deal of recent research activity.

Many deterministic algorithms (Gradient methods : Steepest Descent, Simplex Methods : Nelder-Mead, Newton Methods : Levenberg-Marquart, Quasi-Newton Methods : DFP, BFGS) [36] for nonlinear optimization problems seek only a local solution, a point at which the objective function is smaller than at all other feasible nearby points. They do not always find the global solution, which is the point with lowest function value among all feasible points. Global solutions are needed for many problems, they are difficult to recognize and even more difficult to locate.

We will begin with the most basic algorithm, namely “blind search”. The simplest random search method is one where we repeatedly sample over the problem space such that the current sampling for state s does not take into account the previous samples. This "blind search" approach does not adapt the current sampling strategy to information that has been gathered in the search up to the present time. The approach can be implemented in batch (nonrecursive) form simply by laying down a number of points and taking the value yielding the lowest target value as our estimate of the optimum. The approach can also be implemented in recursive form as we illustrate below,

Blind Search Algorithm

- Initialization : Choose initial value of θ (the parameter to be optimized), randomly or deterministically as θ_0. Calculate the cost function $L(\theta_0)$
- Generate a new value θ_{k+1}^{new}, according to a chosen probability distribution. If $L(\theta_{k+1}^{new}) < L(\theta_k)$ set $\theta_{k+1} = \theta_{k+1}^{new}$ else $\theta_{k+1} = \theta_k$
- Stop the iteration if maximum number of evaluations is reached or a user defined criteria is satisfied.

Blind search is the simplest random search in that the sampling generating the new θ value at each iteration is over the entire domain of interest. The sampling does not take into account of where the previous estimates of θ have been.

The random sampling can be a function of the position of the current best estimate for θ. In this way, the search is more localized in the neighborhood of that estimate, allowing for a better exploitation of information that has previously been obtained about the shape of the loss function. Such algorithms are sometimes referred to as localized algorithms to emphasize their dependence on the local environment near the current estimate for θ. Below is the most naïve of the localized stochastic optimization method. More sophisticated approaches can also be implemented.

Localized Random Search Algorithm

- Initialization : Choose initial value of θ the parameter to be optimized, randomly or deterministically as θ_0. Calculate the cost function $L(\theta_0)$
- Draw an independent random vector d_k and add it to the current value $\theta_k^{new} = \theta_{k-1} + d_k$.
- If $L(\theta_{k+1}^{new}) < L(\theta_k)$ set $\theta_{k+1} = \theta_{k+1}^{new}$ else $\theta_{k+1} = \theta_k$
- Stop the iteration if maximum number of evaluations is reached or a user defined criteria is satisfied.

More sophisticated stochastic algorithms include Annealing Type algorithms, Evolutionary algorithms and Genetic algorithms.

4.1 Genetic Algorithms

Evolutionary and Genetic algorithms (GAs) [40] draw inspiration from the natural search and selection processes leading to the survival of the fittest individuals. Genetic algorithm search methods are rooted in the mechanisms of evolution and natural genetics. The interest in heuristic search algorithms with underpinnings in natural and physical processes began as early as the 1970s. Simulated annealing, Genetic algorithms, and evolutionary strategies are similar in their use of a probabilistic search mechanism directed toward decreasing cost or increasing payoff. These three methods have a high probability of locating the global solution optimally in a multimodal search landscape. (A multi-modal cost function has several locally optimal solutions as well). However, each method has a significantly different mode of operation. The principal difference between Genetic algorithms and evolutionary strategies is that genetic algorithms rely on crossover, a mechanism of probabilistic and useful exchange of information among solutions, to locate better solutions, while evolutionary strategies use mutation as the primary search mechanism.

In the literature, Holland's Genetic algorithm is commonly called the Simple Genetic Algorithm (SGA). Essential to the SGA's working is a population of binary strings. Each string of 0s and 1s is the encoded version of a solution to the optimization problem. Using genetic operators (crossover and mutation) the algorithm creates the subsequent generation from the strings of the current population.

A simple Genetic Algorithm is presented below,

Simple Genetic Algorithm

- Initialize population
- Evaluate population
- While termination criteria is not reached
    - Select solutions for next generation
    - Perform crossover and mutation

GA has been successfully applied to solving optimization problems, such as wire routing, scheduling, adaptive control, game playing, cognitive modeling, transportation problems, traveling salesman problems, optimal control problems, database query optimization etc.

4.2 Condensation and Genetic Condensation Algorithms

In recent years, there has been a great deal of interest in applying Particle Filtering, also known as Condensation or Sequential Importance Sampling [13], to computer vision problems. Applications on parameterized or non-parameterized contour tracking and human tracking have demonstrated its usefulness. In the context of contour-based object recognition, multiple hypotheses for such correspondences are always considered. Condensation is a more efficient way to maintain multiple hypotheses over time while tracking, where particles represent the probability distribution of the target position. The method is relatively robust to noise, and recovers from tracking misses in intermediate frames. In addition, they are relatively simple to implement, and allow one to conveniently combine multiple feature types in the same tracker.

The Condensation algorithm is a specific realization of the SIR Particle Filter. The $s_k^i \sim p(s_k \mid s_{k-1}^i)$ importance density is approximated by $s_k^n = f(s_{k-1}^n) + N(0,\sigma)$ and re-sampling is applied in each measurement step. By combining a tractable dynamic model with visual observations, it can accomplish highly robust tracking of object motion. Below, you may find a sketch of the algorithm.

Condensation Algorithm

- Initialization: For *n = 1,2,...,N* generate samples from the prior in order to obtain $\{s_0^n, \pi_0^n\}$, where s_0^n are the initial samples, π_0^n are the initial likelihoods of the samples and *N* is the number of samples.
- π_{k-1}^n are the associated likelihoods of the samples s_{k-1}^n at time step *k-1* . From sample set $\{s_{k-1}^n, \pi_{k-1}^n\}$, compose the new sample set $\{s_k^n, \pi_k^n\}$. Iterate for *k=1, 2, ...* as follows:
    - Propagate samples using state transition equation $s_k^n = f(s_{k-1}^n) + N(0,\sigma)$.
    - Update the likelihoods (weights) by $\pi_k^n = p(b_k^n \mid s_k^n)$, where $p(b_k^n \mid s_k^n)$ is the likelihood value of the measurement b_k^n given sample s_k^n. Normalize π_k^n 's so that $\sum_n \pi_k^n = 1$.
    - Resample the samples s_k^n with probability π_k^n in order to select *N* samples $\{s_k^n, \pi_k^n\}, n = 1,2,...,N$.

As an example to the improvement efforts on Condensation algorithm, Genetic Condensation algorithm [6] will be presented. Basically, Condensation algorithm solves an optimization problem along the time axis. In this sense, Evolutionary and Genetic algorithms for searching an optimal solution are very similar to Condensation algorithm. They are based on principle of evolution (propagation in Condensation) and the survival of the fittest (re-sampling in Condensation). As a difference, the EAs and GAs benefit from a wide range of structures using a set of operators that are often defined specially for a given problem.

In order to evaluate our DEMC Particle Filtering approach, we will compare it to the standard Condensation and Genetic Condensation algorithms. We slightly changed the structure of the algorithm. The work presented in [6], applies the GA optimization step after the re-sampling. This approach seriously diminishes the optimization opportunity of the GA step. The main reason is the reduced variance of the samples after the re-sampling step. We prefer to apply the GA optimization step just before the re-sampling. We experienced that this approach increases the performance in tracking considerably. The sketch of the Genetic-Condensation algorithm can be given as follows,

Genetic Condensation Algorithm

- Initialization: For *n = 1,2,...,N* generate samples from the prior in order to obtain $\{s_0^n, \pi_0^n\}$
- Compose the new sample set $\{s_k^n, \pi_k^n\}$ by iterating for *k=1, 2, ...* as follows:
  - Propagate samples using state transition equation $s_k^n = f(s_{k-1}^n) + N(0,\sigma)$.
  - Update the likelihoods (weights) by $\pi_k^n = p(b_k^n \mid s_k^n)$, where $p(b_k^n \mid s_k^n)$ is the likelihood value of the measurement y_k^n given sample s_k^n.
  - Apply crossover on the sample set, iterate through the sample set as follows,
    - Select two samples randomly s_k^A, s_k^B from the sample set, apply stochastic crossover,

$$s_k^{newA} = \gamma s_k^A + (1-\gamma)s_k^B + \eta$$

$$s_k^{newB} = \gamma s_k^B + (1-\gamma)s_k^A + \eta$$

    - where η is a normal distributed random variable and γ is an uniform distributed random variable $\in [0..1]$.

- Replace the parents by their children according to Metropolis algorithm, if $p(b_k \mid s_k^{new}) > max(p(b_k \mid s_k^A), p(b_k \mid s_k^B))$ or with a probability of,

$$\frac{p(b_k \mid s_k^{new})}{max(p(b_k \mid s_k^A), p(b_k \mid s_k^B))}$$

- Apply mutation on the sample set, iterate through the sample set as follows,
  - Select a sample s_k^A from the sample set, mute it according to, $s_k^{new} = \gamma s_k^A + \eta$ where η defined as $N(0, \Sigma)$.
  - Accept the new sample if $p(b_k \mid s_k^{new}) > p(b_k \mid s_k^A)$ or with a probability of,

$$\frac{p(b_k \mid s_k^{new})}{p(b_k \mid s_k^A)}$$

- Compute the efficient sample size,

$$N_{eff} = \frac{1}{\sum_n \pi_k^{n2}}$$

- If $N_{eff} \geq Threshold$, do not resample
  - else resample from $\{s_k^n, \pi_k^n\}$ and set the likelihood values equal to $1/N$.

DEMC Particle Filter will be defined in the next section. Both DEMC Particle Filter and Genetic Condensation algorithms are the same in the sense of improving the re-sampling step of the Condensation algorithm.

4.3 Differential Evolution

The Differential Evolution algorithm [41,42] can be classified as a floating-point encoded evolutionary optimization algorithm. Traditional GAs use a fixed type of perturbation, such as adding random numbers to individual parameters. The problem with this approach is that it fails to account for the fact that what might be a small perturbation for one parameter might be gigantic for another. DE avoids this problem by using the population itself as the source of appropriately scaled perturbations.

Differential Evolution (DE) has proven to be a promising candidate for minimizing real valued, multi-modal objective functions. Besides its good convergence properties, DE is very simple to understand and to implement. DE is also particularly easy to work with, having only a few control variables, which remain fixed throughout the entire minimization procedure. DE is a parallel direct search method, which utilizes N D-dimensional parameter vectors $x_i \in G, i = 0,1,2,...,N-1$ as a population for each generation G, for each iteration of the minimization. N does not change during the minimization process. The initial population is chosen randomly and should try to cover the entire parameter space uniformly. As a rule, a uniform probability distribution for all random decisions will be assumed unless otherwise stated. Basically, DE generates new parameter vectors by adding the weighted difference between two population vectors to a third vector. If the resulting vector yields a lower objective function value than a predetermined population member, the newly generated vector replaces the vector, with which it was compared, in the next generation; otherwise, the old vector is retained. Below, you can find a sketch of the DE algorithm.

For each sample s_G^n, $(n = 1,2,...,N)$ and G is the current generation and N is the number of samples, a perturbated vector v_{G+1}^n is generated according to,

$$v_{G+1}^n = s_G^{r1} + F.(s_G^{r2} - s_G^{r3}) \qquad \textbf{(4.1)}$$

with $r1, r2, r3 \in [1, N]$ and $F > 0$. The randomly chosen integers $r1, r2, r3$ are mutually different and also chosen to be different from the running index *n*. *F* is a real and constant factor $\in [0,2]$, which controls the amplification of the differential variation $(s_G^{r2} - s_G^{r3})$. In order to increase the potential diversity of the perturbated parameter vector, crossover is introduced. The vector,

$$u_{G+1}^{n} = (u_{G+1}^{0,n}, u_{G+1}^{1,n}, ..., u_{G+1}^{D-1,n}) \qquad \textbf{(4.2)}$$

with

$$u_{G+1}^{j,n} = \begin{cases} v_{G+1}^{j,n} \; for\; j = \langle k \rangle_D, \langle k+1 \rangle_D, ..., \langle k+L-1 \rangle_D \\ s_G^{j,n} \qquad\qquad for\, all\, other \qquad j \in [0, D-1] \end{cases} \qquad \textbf{(4.3)}$$

is formed. The acute brackets $\langle \; \rangle_D$ denote the modulo function with modulus D, which is the vector dimension. The starting index k is a randomly chosen integer from interval $[0, D-1]$. The integer L, which denotes the number of parameters that are going to be exchanged, is drawn from interval $[0, D-1]$ with probability $Pr(L = v) = (CR)^v$. $CR \in [0,1]$ is the crossover probability and constitutes a control parameter for the optimization. The random decisions for both *k* and *L* are made individually for each trial vector v. In order to decide whether the new vector u_{G+1}^{n} shall become a population member of generation *G+1*, it will be compared to s_G^n. Defining the $p(u)$ as the likelihood function, if vector u_{G+1}^{n} yields a greater likelihood value $p(u_{G+1}^{n})$ than $p(s_G^n)$, s_{G+1}^{n} is set to u_{G+1}^{n}, otherwise the old value s_G^n is retained.

4.4 DEMC Sampling

Differential Evolution Markov Chain (DEMC) [47] combines the basic ideas of DE and MCMC. DEMC is a population MCMC algorithm, where multiple chains run in parallel. DEMC solves an important problem in MCMC, namely that of choosing an appropriate scale and orientation for the jumping distribution. In DEMC, the jumps are simply a fixed multiple of the differences of two random parameter vectors that are currently in the population. The selection process of DEMC works via the usual Metropolis ratio, which defines the probability with which a proposal is accepted.

In DEMC *N* chains run in parallel where jump for one of the chains is determined by the all other *N-1* chains. Taking the difference of vectors of two randomly chosen chains can be the simplest method of balancing the exploration of the space. The difference vector should be multiplied with a factor γ and added to the vector of the current chain. The difference vector contains the required information on scale and orientation. Each proposal is shown to define a Metropolis step, in which each jump is as likely as the reverse jump, given the current state of the remaining chains. The *N*-chain is therefore a single random walk Markov chain on an *N* × *d*-dimensional space. The method can be coded in a few lines, requiring only a function to draw uniform random numbers and a function to calculate the fitness of each proposal vector. Below is the pseudo code of the DEMC,

DEMC Sampling Algorithm

```
• Iterate for G=1,…,NG
    o Assign Temperature according to a cooling schedule
    o Cycle through members of the population $\{s_G^n\} \quad n = 1,...,N$
        • R1 is a uniform number between 1 and N
        • R2 is a uniform number between 1 and N, different than R1
        • $s_{new}^n = s_G^n + c.(s_G^{R1} - s_G^{R2}) + uniform[-b,b]$
        • $r = \frac{fitness(s_{new}^n)}{fitness(s_G^n)}$
        • if $(\log(r) > Temperature * \log(uniform[0,1])$ swap $s_{new}^n, s_G^n$
        • $s_{new}^n$ is a new sample drawn from the target density.
```

Where *NG* is the number of generations and *N* is the number of samples in the generation.

For utilizing DE for drawing samples from a target density, the proposal and acceptance scheme must be such that there is detailed balance with respect to $\pi(s)$. DE proposal scheme is not appropriate for such a balance. DE1 is more appropriate where s^{R0} is replaced by s^n. To ensure all the parameter space can be reached, scheme DE1 is modified such that,

$$s^{new} = s^n + \gamma(s^{R1} - s^{R2}) + e \qquad \textbf{(4.4)}$$

where e is drawn from a symmetric distribution with a small variance compared to that of the target, but with unbounded support, e.g. with $e \sim N(0,b)$ where b is small. The key of DEMC is to introduce a probabilistic acceptance rule in DE: proposal and acceptance with probability $\min(1,\mathrm{r})$ where $\mathrm{r} = \pi(\mathrm{s}^{\mathrm{p}})/\pi(\mathrm{s}^{\mathrm{n}})$.

5. 3D LICENSE PLATE TRACKING USING DEMC PARTICLE FILTER

Numerous algorithms for tracking objects in video sequences have been proposed; they can be classified as feature-based when the goal is to track image feature between consecutive frames without the use of any model or they can be model-based when a 2D or 3D model of the target is known. We will focus on the 3D model-based approach for license plate tracking in VLPR systems.

In the next section, we give brief information on a typical VLPR system architecture.

5.1 VLPR System Architecture

As mentioned in previous sections, a typical VLPR system consist of three major parts, namely license plate detection (segmentation), tracking and recognition parts. The information flow between these parts is shown in Figure 5.1. Basically, the diagram shows the collaboration between the segmentation and the tracking blocks. The tracking block provides the ROI (Region Of Interest) to the segmentation block. The segmentation block works on the given ROI and returns the segmentation results to the tracking block in order to estimate the next LP (License Plate) location in the next frame. Finally, tracking block transfers the filtered LP location for recognition purposes to the recognition block. Figure 5.1 also shows the acquisition system composed of a camera and a frame grabber. In the next sections, we will explain the utilized acquisition system and the model.

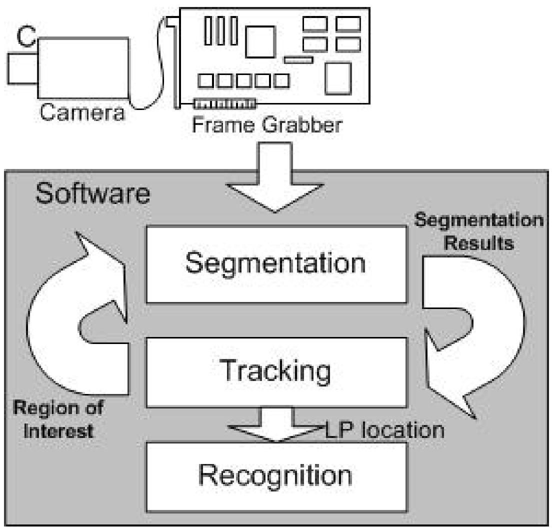


Figure 5.1: VLPR System

5.2 Acquisition System

Our outdoor data acquisition setup is simply shown in Figure 5.2. The acquisition system consists of a monochrome camera, mounted on a tripod, facing the road ahead with a slight inclination.

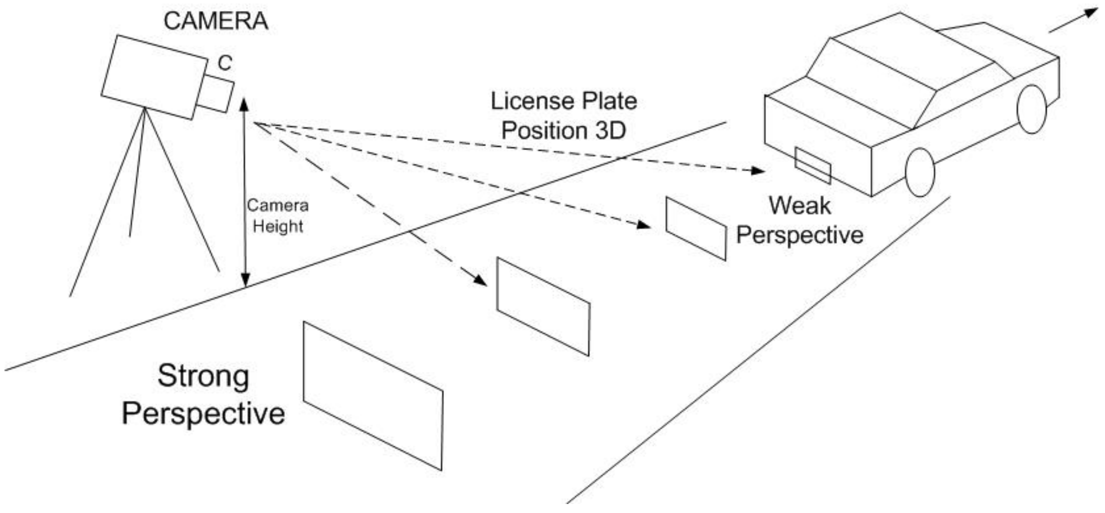


Figure 5.2 : VLPR System Outdoor Setup

The camera has a fixed pre-mounted optics with focal length between 28-280 mm. Images are digitized at a resolution of 704x288 pixels

The focal length of a lens is defined as the distance in mm from the optical center of the lens to the focal point, which is located on the sensor or film as shown in Figure 5.3.

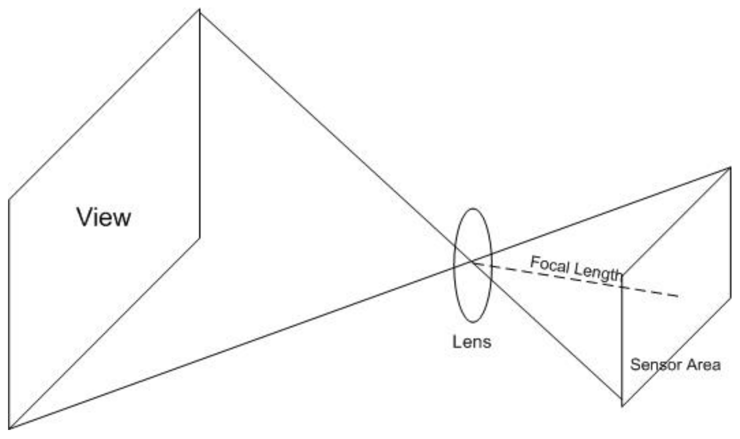


Figure 5.3 : Optical System and Focal Length

The camera lens projects part of the scene onto the film or sensor. The field of view (FOV) is determined by the angle of view from the lens out to the scene and can be measured horizontally or vertically. Larger sensors or films have wider FOVs and can capture more of the scene.

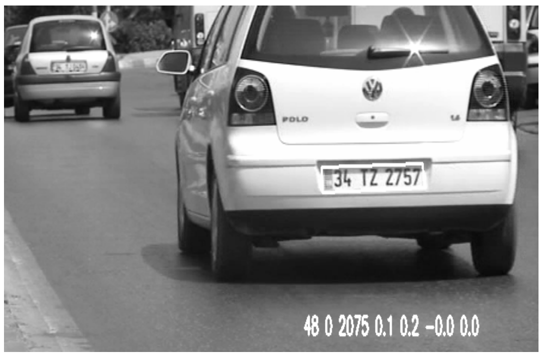
A change in focal length allows you to come closer to the subject or to move away from it and has therefore an indirect effect on perspective. The optical zoom is defined as,

$$\text{Optical zoom} = \frac{\text{maximum focal length}}{\text{minimum focal length}} \quad . \qquad \textbf{(5.1)}$$

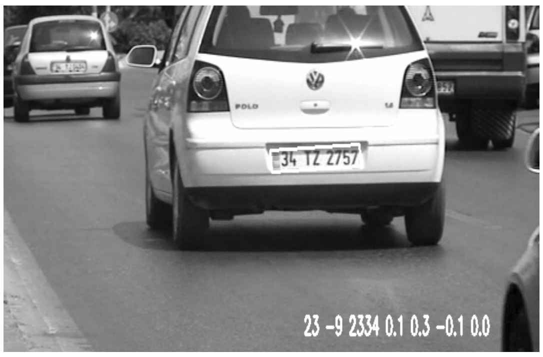
For instance, the optical zoom of a 28-280mm zoom lens is 280mm/28mm or 10X. This means that the size of a subject projected on the film or sensor surface will be ten times larger at maximum focal length (280mm) than at maximum wide angle (28mm). Optical zoom should not be confused with digital zoom.

Given the information above we may have a look at an example scene of the described setup in Figure 5.4.

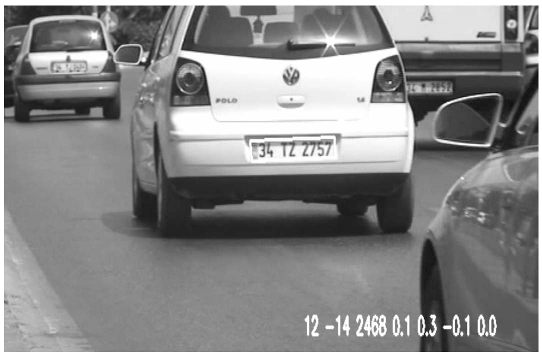
(a)



(b)



(c)



(d)

Figure 5.4 : Example Sequence (a) Frame 16 (b) Frame 24 (c) Frame 32 (d) Frame 36

5.3 Camera Model

We utilize a pinhole camera model [21,43,44] because of its simplicity. This model connects a point of the 3D space to its perspective projection on the camera plane. This transformation is linear in the projective space and is described by the intrinsic and extrinsic parameters. The intrinsic camera parameters, such as the focal length (f), the pixels size (k_u, k_v) and the image coordinates of the projection center $(u_0, v_0)^T$ describe an affine transformation representing a scaling, a rotation and a translation between the camera and the image references. The extrinsic parameters describe the rigid transformation from the world reference to the camera reference. This transformation is entirely defined by a 3x3 rotation matrix and a 3x1 translation vector. Using homogeneous coordinates in the projective space we get:

$$\begin{pmatrix} x \\ y \\ s \end{pmatrix} = \begin{pmatrix} \alpha_u & 0 & u_0 & 0 \\ 0 & \alpha_v & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} . \begin{pmatrix} R & T \\ 0^T & 1 \end{pmatrix} . \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \qquad \textbf{(5.2)}$$

Where $(x\ y\ s)^T$ are the homogeneous coordinates of the projection of point P on the image, $\alpha_u = \frac{f}{k_u}$ and $\alpha_v = \frac{f}{k_v}$ (with k_u and k_v the pixels width and height, respectively), R is the rotation matrix, $(X\ Y\ Z\ 1)^T$ is the homogeneous coordinates of point P in the scene reference, T is the translation vector and combined matrix is the perspective projection matrix (that expresses the transformation between the camera coordinate system and the image coordinate system). The above model describes a projective transformation from 3D space to 2D space (camera retina's plane). A 3x4 projection matrix describes this transformation. Determining the model parameters are called camera calibration. This can be done by observing a 3D known object (classical Calibration), or by using 2D pattern like a checkerboard with a predefined size. However, locating a point in the 3D space implies this one to be viewed from at least two different locations. This requirement can be met either by observing the scene point with two different cameras or by moving a single camera along a known trajectory.

A rotation matrix R can always be written as the product of three matrices representing rotations around the *X, Y*, and *Z* axes. There are several conventions on the order in which these rotations are carried out. For example, taking α, β, γ to be rotation angles around the X, Y, and Z axis respectively yields,

$$R_x = \begin{pmatrix} 1 & 0 & 0 \\ 0 & cos\alpha & -sin\alpha \\ 0 & sin\alpha & cos\alpha \end{pmatrix}, \quad R_y = \begin{pmatrix} cos\beta & 0 & sin\beta \\ 0 & 1 & 0 \\ -sin\beta & 0 & cos\beta \end{pmatrix}$$
$$R_z = \begin{pmatrix} cos\gamma & -sin\gamma & 0 \\ sin\gamma & cos\gamma & 0 \\ 0 & 0 & 1 \end{pmatrix}, \quad R = R_x . R_y . R_z \qquad \textbf{(5.3)}$$

$$T = \begin{pmatrix} \Delta x \\ \Delta y \\ \Delta z \end{pmatrix} \qquad \textbf{(5.4)}$$

Even though Euler angles α, β, γ can do a creditable job for a large range of camera orientations, they have a well-known drawback: When two of the three rotations axes align, one rotation has no effect. This problem is known as gimbal lock, and this singularity can result in ill-conditioned optimization problems.

By this projection the real world coordinates $(X\ Y\ Z)$ are transformed into image coordinates $(x\ y\ s)$ and the pixel coordinates on the image p_x, p_y can be calculated by $p_x = x/s$ and $p_y = y/s$.

In most 3D tracking methods, the internal parameters are assumed to be fixed and known, which means that the camera cannot zoom, because it is difficult to distinguish a change in focal length from a translation along the camera Z-axis. These parameters can also be estimated during an offline camera calibration stage, from the images themselves. Classical calibration methods make use of a calibration pattern of known size inside the field of view.

The perspective projection model is not always sufficient to represent all the aspects of the image formation since it does not take into account the possible distortion from the camera lens, which may be non negligible, especially for wide angle cameras. Fortunately, the lens distortion can be modeled as a 2D deformation of the image.

In this section, we introduced the camera model, since, we need to further explain the utilization of the model in 3D model based tracking concept. In Section 6, an introduction on 3D model based tracking and the theoretical background of the approach is given. In next section, the details of the camera calibration methods are detailed.

5.4 Camera Calibration

The idea behind the calibration is discovering the projection equations that links 3D points to their projections on 2D image plane [21,43,44]. Camera calibration methods rely on one or more images of calibration pattern. In those calibration pattern images, 3D objects of known geometry exist, possibly known position and orientation. Maybe, the most well known calibration method is the Direct Parameter Calibration [21] (DPC).

Consider a 3D point P defined by its coordinates $[X^w, Y^w, Z^w]^T$ in the world reference frame. Let $[X^c, Y^c, Z^c]^T$ be its coordinates in the camera reference frame, where the origin of the camera frame is the center of the projection and the Z is the optical axis. We do not know the extrinsic parameters, 3x3 rotation matrix R and the translation matrix T. The projection is given as,

$$\begin{bmatrix} X^c \\ Y^c \\ Z^c \end{bmatrix} = R. \begin{bmatrix} X^w \\ Y^w \\ Z^w \end{bmatrix} + T \tag{5.5}$$

in addition, its components can be written as,

$$\begin{aligned} X^c &= r_{11}.X^w + r_{12}.Y^w + r_{13}.Z^w + T_x \\ Y^c &= r_{21}.X^w + r_{12}.Y^w + r_{23}.Z^w + T_y \\ Z^c &= r_{31}.X^w + r_{32}.Y^w + r_{33}.Z^w + T_z. \end{aligned} \tag{5.6}$$

Neglecting the radial distortion, we can re-write the equations,

$$\begin{aligned} x_{im} &= -\frac{f}{k_u}.\frac{X^c}{Z^c} + u_0 \\ y_{im} &= -\frac{f}{k_v}.\frac{Y^c}{Z^c} + v_0 \end{aligned} \tag{5.7}$$

with the focal length (f), the pixels size (k_u, k_v) and the image coordinates $(u_0, v_0)^T$. Plugging (3.5) and (3.6) we get,

$$\begin{aligned} x - u_0 &= \frac{r_{11}.X^w + r_{12}.Y^w + r_{13}.Z^w + T_x}{r_{31}.X^w + r_{32}.Y^w + r_{33}.Z^w + T_z} \\ y - v_0 &= \frac{r_{21}.X^w + r_{12}.Y^w + r_{23}.Z^w + T_y}{r_{31}.X^w + r_{32}.Y^w + r_{33}.Z^w + T_z}. \end{aligned} \tag{5.8}$$

We assume that the image center coordinates are known. The aspect ratio α is defined as f_x / f_y. The eight unknowns can be packed in a vector $v = (v_1, v_2, ..., v_8)$ resulting [21],

$$A.v = 0 \tag{5.9}$$

where,

$$\begin{array}{ll} v_1 = r_{21} & v_5 = \alpha r_{11} \\ v_2 = r_{22} & v_6 = \alpha r_{12} \\ v_3 = r_{23} & v_6 = \alpha r_{13} \\ v_4 = T_y & v_8 = T_x \end{array}, \tag{5.10}$$

$$A = \begin{bmatrix} x_1 X_1^w & x_1 Y_1^w & x_1 Z_1^w & x_1 & -y_1 X_1^w & -y_1 Y_1^w & -y_1 Z_1^w & -y_1 \\ x_2 X_2^w & x_2 Y_2^w & x_2 Z_2^w & x_2 & -y_2 X_2^w & -y_2 Y_2^w & -y_2 Z_2^w & -y_2 \\ . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . \\ x_N X_N^w & x_N Y_N^w & x_N Z_N^w & x_N & -y_N X_N^w & -y_N Y_N^w & -y_N Z_N^w & -y_N \end{bmatrix} \tag{5.11}$$

with N corresponding pairs of points, this leads us to a homogeneous system of N linear equations. Those pairs should not be coplanar, and A has rank greater than 7. The solution is trivial and can be calculated by Single Value Decomposition. We have still undetermined parameters (T_z, f_x). We will solve the following N linear equations for determining the missing two parameters,

$$Q.\begin{pmatrix} T_z \\ f_x \end{pmatrix} = b \tag{5.12}$$

where,

$$Q = \begin{bmatrix} x_1 & (r_{11}.X_1{}^w + r_{12}.Y_1{}^w + r_{13}.Z_1{}^w + T_x) \\ x_2 & (r_{11}.X_2{}^w + r_{12}.Y_2{}^w + r_{13}.Z_2{}^w + T_x) \\ . & . \\ . & . \\ . & . \\ x_N & (r_{11}.X_N{}^w + r_{12}.Y_N{}^w + r_{13}.Z_N{}^w + T_x) \end{bmatrix} \tag{5.13}$$

$$b = \begin{pmatrix} -x_1.(r_{31}.X_1^w + r_{32}.Y_1^w + r_{33}.Z_1^w) \\ . \\ . \\ . \\ -x_N.(r_{31}.X_N^w + r_{32}.Y_N^w + r_{33}.Z_N^w) \end{pmatrix}. \tag{5.14}$$

The least squares solution to (5.12) exists and the solution for (T_z, f_x) can be found. as:

$$\begin{pmatrix} \hat{T}_z \\ \hat{f}_x \end{pmatrix} = (A^T A)^{-1} A^T b \qquad \textbf{(5.15)}$$

Until now, we examined three-dimensional reference object-based calibration. Camera calibration is performed by observing a calibration object whose geometry in 3D space is known with very good precision.

In our application of license plate tracking, we applied the flexible technique in [50], which only requires the camera to observe a planar pattern shown at least two different orientations. The pattern can be printed on a laser printer and attached to a reasonable planar surface. Either the camera or the planar pattern can be moved by hand. The motion need not to be known. The proposed approach, which uses 2D metric information, lies between the photogrammetric calibration, which uses explicit 3D model, and self-calibration, which uses motion rigidity or equivalently implicit 3D information.

Considering the pinhole camera model, we can briefly rewrite the projection equation as,

$$s\tilde{m} = A[R\ t]\tilde{M} \qquad \textbf{(5.16)}$$

where s is a scaling factor, $\tilde{m} = [u, v, 1]^T$, $\tilde{M} = [X, Y, Z, 1]^T$. The matrix A (camera intrinsic matrix) is defined as,

$$A = \begin{bmatrix} \alpha & \gamma & u_0 \\ 0 & \beta & v_0 \\ 0 & 0 & 0 \end{bmatrix}. \qquad \textbf{(5.17)}$$

$[R\,t]$ is called as the camera extrinsic matrix. which relates the world coordinate system to the camera coordinate system.

Given a model plane and we assume it is on *Z=0*, we can define a homography *H* as,

$$s\tilde{m} = H\tilde{M} \quad , where \quad H = A[r_1\ r_2\ t]. \qquad \textbf{(5.18)}$$

We will define a matrix *B* such as,

$$B = A^{-T} A^{-1} = \begin{bmatrix} B_{11} & B_{12} & B_{13} \\ B_{21} & B_{22} & B_{23} \\ B_{31} & B_{32} & B_{33} \end{bmatrix}. \qquad \textbf{(5.19)}$$

Note that B is symmetric, defined by a 6D vector,

$$b = [B_{11} B_{12} B_{22} B_{13} B_{23} B_{33}]^T \qquad \textbf{(5.20)}$$

Let the *i*-th column vector of H be $h_i = [h_{i1}, h_{i2}, h_{i3}]^T$ .Then, we have

$$h_i^T B h_j = v_{ij}^T b \, . \qquad \textbf{(5.21)}$$

Here v_{ij} is given as,

$$v_{ij} = [h_{i1}h_{j1}, h_{i1}h_{j2} + h_{i2}h_{j1}, h_{i2}h_{j2}, h_{i3}h_{j1} + h_{i1}h_{j3}, h_{i3}h_{j2} + h_{i2}h_{j3}, h_{i3}h_{j3}]^T \, . \qquad \textbf{(5.22)}$$

Considering the homography constraints from a given homography, we can rewrite the equations as,

$$\begin{bmatrix} v_{12}^T \\ (v_{11} - v_{22})^T \end{bmatrix} . b = 0 \, . \qquad \textbf{(5.23)}$$

If *n* images are given with model plane observed, we can combine *n* such equations and we get,

$$V.b = 0 \, . \qquad \textbf{(5.24)}$$

Here *V* is a *2n x 6* matrix. In case of $n \geq 3$ we can find a unique solution for *b* up to a scale factor.

Once *b* is determined, we can compute all camera intrinsic parameters from matrix *B*, which is defined as $B = \lambda A^{-T} A$.

and the intrinsic parameters are,

$$\begin{aligned} v_0 &= (B_{12}B_{13} - B_{11}B_{23})/(B_{11}B_{22} - B_{12}^2) \\ \lambda &= B_{33} - [B_{13}^2 + v_0(B_{12}B_{13} - B_{11}B_{23})]/B_{11} \\ \alpha &= \sqrt{\lambda / B_{11}} \\ \beta &= \sqrt{\lambda B_{11} / (B_{11}B_{22} - B_{12}^2)} \\ \gamma &= -B_{12}\alpha^2\beta / \lambda \\ u_0 &= \gamma v_0 / \alpha - B_{13}\alpha^2 / \lambda . \end{aligned} \qquad \textbf{(5.25)}$$

The extrinsic parameters can be computed, if *A* is known,

$$r_1 = \lambda A^{-1} h_1,$$
$$r_2 = \lambda A^{-1} h_2,$$
$$r_3 = r_1 x r_2, \qquad \textbf{(5.26)}$$
$$t = \lambda A^{-1} h_3,$$
$$\lambda = 1 / \left\| A^{-1} h_1 \right\| = 1 / \left\| A^{-1} h_2 \right\|.$$

Because of the noise in the measurements, rotation matrix does not in general fulfill the properties of a rotation matrix. Assume that the image points are corrupted by independent and identical noise. With *n* images and *m* points in each image, the maximum likelihood estimate can be obtained by minimizing the following function,

$$\sum_{i=1}^{n} \sum_{j=1}^{m} \left\| m_{ij} - \hat{m}(A, R_i, t_i, M_j) \right\|^2 \qquad \textbf{(5.27)}$$

where $\hat{m}(A, R_i, t_i, M_j)$ is the projection of the point M_j in the image *i*. Minimization the given function is a nonlinear minimization problem, which can be solved by Levenberg-Marquart algorithm [50].

6. 3D MODEL BASED TRACKING

Location of the camera may cause different perspective views, which plays an important role in the recognition process at the final step of the VLPR systems. Most VLPR system nevertheless utilize 2D plane, but with a limited range of visual angle and focus distance. Because considerable perspective in 3D space results distorted license plate images.

3D displacement of the objects or object parts in the scene, results a 2D change in the image, which is typically modeled by 2D tracking applications. This yields a need for an adaptive model in order to handle appearance changes due to perspective effects or deformations. This appearance model finally includes a centroid for images position in 2D, a scale factor and last but not the least; the model should utilize an affine transformation for relocating object feature points.

More complex models exist like splines, deformable templates, 2D deformable meshes or 2D articulated models. Basically, these models are far from recovering the actual object state in space despite their sophisticated nature.

On the other hand, 3D tracking targets recovering all six degrees of freedom continuously, given the camera location and orientation relative to the scene. Totally 3D displacement and the orientation of the object in the scene is discovered.

The 3D information can be represented in the form of a CAD model of a scene object, a set of planar parts, or even a rough 3D model such as an ellipsoid. Such models can be created using either automated techniques or commercially available products.

In the particular context of our application, the license plate is a rectangle defined by its four corner points, which also defines its width and its height (50 cm to 10 cm).

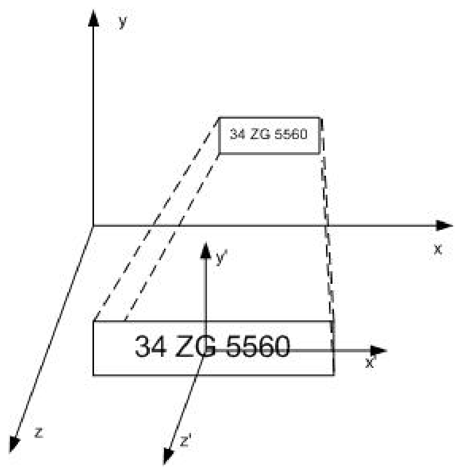


Figure 6.1 : License Plate Projection

The Figure 6.1 describes the projection process of the license plate on the camera's image plane. In the case of a license plate tracking application, it is reasonable to assume that the vehicle moves parallel to the road plane. The camera is statically mounted. In this case, the 3D motion of the license plate is restricted and the prediction and estimation can be optimized according to this 3D restricted 6-DOF behavior.

Model-based approaches try to extract the drift information by finding object pose that correctly projects some fixed features of a given 3D model onto the 2D image. These features can be edges, line segments, or points.

3D tracking is more complex and has the difficulty of matching high dimension 3D models to the image features. The matching of 3D model to image features is an optimization problem. The optimization space is large and convergence is not guarantied, if the initial estimation is not close to the solution. In order to improve the estimation process, Bayesian approach needs to be used. In the next section, the utilized system state in Bayesian framework is explained.

6.1 System State

The state of a license plate in 3D can be represented with six variables such as the Cartesian coordinates X, Y, Z of the license plate's center in 3D and the Euler angles α (the rotation around x axis), β (the rotation around y axis) and finally γ (the rotation around z axis) rotation angles of the license plate's plane. The state vector can be represented by $s_k = [X\ Y\ Z\ \alpha\ \beta\ \gamma]^T$ where k is the time step. The state vector variables are shown in Figure 6.2. The origin of the coordinate system is the center of the 2D image plane.

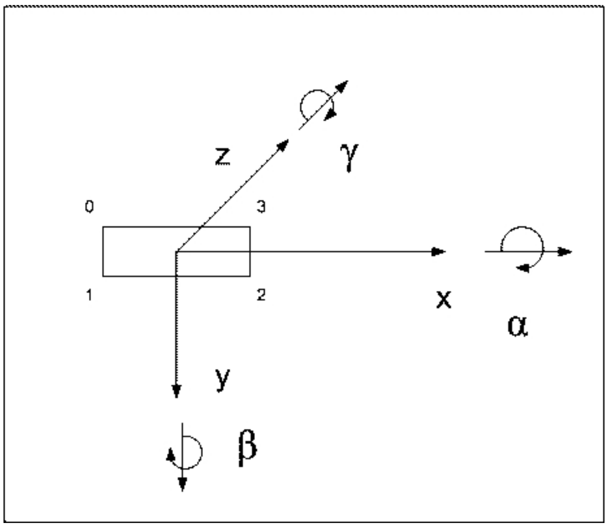


Figure 6.2 : Coordinate System

$\{s_k, k \in N\}$ is given as the state sequence, where s_k is the state at time step k. We may consider the system being in one of the states $s_k^n, n \in N$ with the probability $p(s_k^n)$ at time step k. This notation will be used in the proceeding sections. In the context of our application, we assume that the road conforms to a planar surface, the movement of a vehicle is parallel to this plane. Euler angles changes are accordingly limited considering the real world scenarios. That simply dictates that the variance of γ is considerably less than the other Euler angles. γ can only change if there exists an ego-motion of the camera, which is not the case in our setup.

6.2 System Model

The state evolution along time can be modeled by a discrete-time nonlinear dynamic system. The license plate is assumed to move towards its plane's normal vector direction. The system dynamic equation is non-linear such as $s_k = f_k(s_{k-1}, w_{k-1})$ and can be approximated with a stochastic differential equation as,

$$s_{k+1} = s_k + \begin{bmatrix} c_k.sin\beta_k \\ -c_k.sin\alpha_k.cos\beta_k \\ c_k.cos\alpha_k.cos\gamma_k \\ 0 \\ 0 \\ 0 \end{bmatrix} + N(0,\sigma), \quad with \quad s_k = \begin{bmatrix} X_k \\ Y_k \\ Z_k \\ \alpha_k \\ \beta_k \\ \gamma_k \end{bmatrix} \qquad \textbf{(6.1)}$$

where α, β, γ are the Euler angles and c_k is the velocity of the license plate in its normal direction. w_{k-1} is represented by a zero mean, σ standard deviation Gaussian $N(0,\sigma)$. The system dynamic equation can also be stated as $s_k = f(s_{k-1}, c_{k-1}, w_{k-1})$ considering the c_k as a state of the system. Estimating the correct value of c_k is a crucial point of the tracking. In our very first trials, we also included the velocity of the license plate into the state vector. This made the state vector seven dimensional. The velocity of the plate is not directly measured from the video sequence. For this reason, estimation of the velocity is not an easy task. Increasing the dimension of the state vector and trying to estimate a hidden state, was not yielding good tracking results. In our tracking application, the c_k velocity magnitudes are approximated by an augmented Kalman filter, which processes the proceeding location estimate differences as velocity measurements and outputs a filtered c_k velocity magnitude. The time update equations are,

$$\begin{aligned} \hat{c}_k^- &= \hat{c}_{k-1} \\ P_k^- &= P_{k-1} + Q. \end{aligned} \qquad \textbf{(6.2)}$$

In addition, the measurement update equations are,

$$\begin{aligned} K_k &= P_k^-(P_k + R)^{-1} \\ \hat{c}_k &= \hat{c}_k^- + K_k(d_k - \hat{c}_k^-) \\ P_k &= (1 - K_k)P_k^- \end{aligned} \qquad \textbf{(6.3)}$$

where d_k is calculated by taking the Euclidian distance of two successive coordinates $(x\ y\ z)$ of license plate in video sequence.

6.3 Observation Model

The observation model $b_k = h_k(s_k, v_k)$ relates the system state s_k with the information extracted from images b_k. Given the image b_k, solving the equation $s_k = h_k^{-1}(b_k)$, would yield the desired solution if we had a deterministic description of the measurement process. Instead, given the image b_k, we may define a likelihood function $p(b_k | s_k)$ assuming the system in state s_k. Therefore, figuring out the probability $p(b_k | s_k)$ is important for understanding update step.

Vertical edge detection is a popular method for exposing the desired features in license plate segmentation. It can be observed that the edges are more uniformly distributed with in the plate area than in other part of the image. Although the pattern of the vertical edges in the plate is unique compared to the other parts, directly searching for this pattern is not an easy task.

Most of the work on license plate localization, somehow utilizes the edge information. For example, an interesting approach, Sliding Cocentric Windows (SW) segmentation technique is introduced in [12]. The method is developed in order to describe the local irregularity in the image using image statistics such as standard deviation and/or mean value.

Despite the edge extraction based methods, we apply simple template matching for likelihood calculation. The information extraction starts by defining the polygon determined by the license plate corners projected on to the camera plane as shown in Figure 6.1. A warping operation is defined from these points to a rectangle whose dimensions are the same as the template. The template matching is applied after adjusting the extracted image region and the template to zero mean and unit variance.

The details of this operation will be explained in Section 8 .

7. DEMC PARTICLE FILTER

Tracking problem is an optimization problem and can also be treated in a probabilistic framework for the state estimation of a dynamic system. Sequential Monte Carlo methods such as Particle Filtering, were introduced to solve non-linear dynamic systems. Isard and Blake in [13] introduced the concept of particle filtering to visual tracking, named Condensation Algorithm. Particle Filtering was introduced to the vision community in the form of the Condensation algorithm. Improvements for the Condensation algorithm were considered in [14], [15], and [16]. The algorithm was applied to tracking people in video and face tracking. The reason these algorithms have attracted much interest is that they offer a framework for dynamic state estimation, where the underlying probability density functions need not be Gaussian, and state and measurement equations can be nonlinear, which are commonly encountered in computer vision.

Condensation is a very popular technique for visual tracking problem. However, its convergence greatly depends on the balance between the number of particles, hypotheses and the fitness of the dynamic model. The important problem of the Condensation algorithm is to choose proper samples to approach the actual samples position. Many improvements are offered to tackle the limitations of Condensation algorithm [14-16].

Condensation algorithm has several limitations. In cases where the dynamics are complex or poorly modeled, thousands of samples are usually required for real applications. However, the performance of Condensation depends on both the number of particles and the accuracy of the dynamic model. Given a specific error margin, the number of the particles required is generally determined by the dimension and structure of the state space. A typical 6-DOF tracking problem may require thousands of particles.

Another limitation of Condensation is that in the sampling step the set of samples are propagated without taking account of new measurements. As a result, a large number of samples may be needed to accurately represent the distribution. This is particularly the case for peaked likelihood or case where the new measurements appear in the tail of the prior. After the new weights are evaluated, there are only few samples may be counted in and a large number of them will be neglected.

Genetic Condensation Algorithm is presented in [17], which utilizes the mutation and crossover operators of the genetic algorithm to find more appropriate samples by calculating weights. They improve robustness, accuracy and flexibility in Condensation for visual tracking.

The work in [18] presents a hybrid sampling solution that combines the sampling in the image feature space and in the state space via RANSAC and Particle filtering, respectively. Number of particles can be reduced to dozens for a full 3D tracking problem, which contains considerable noise of different types. The algorithm has been applied to the problem of 3D face pose tracking with changing moderate or intense expressions.

Inspired by the optimization mechanism in DE and Condensation, we propose a new method, namely, DEMC PARTICLE FILTER to tackle the limitations of Condensation algorithm. Several re-sampling methods for Condensation are proposed in literature. DEMC is an alternative for re-sampling step in Condensation algorithm. DEMC is more effective than standard re-sampling techniques and avoids sample degeneracy and impoverishment without violating multimodal nature of Condensation algorithm.

DEMC sampling is inspired from DE techniques. DE is characterized by keeping the N candidates for optimal solution at each iteration. As a matter of fact, there exists a close relationship between Condensation algorithm and EA's. The structure involved in Condensation is quite similar to that in EA's. The correspondence between Condensation and EA's terminology can be summarized as follows.

First, we can say, a time step k is similar to a generation in DE. s_k^n, which is identified as samples in the Condensation, is considered as the strings in DE. Re-sampling procedure in Condensation corresponds to the selection of DE. Accordingly, in the DE, the weights can be called the fitness of the string, which undergoes selection. The maximum likelihood principle is interpreted as a rule to choose the model that maximizes the overall fitness under a circumstance of given data and may be interpreted from the Bayesian point of view. In particular, a fluctuation caused by a perturbation can be exactly regarded as the system noise, which is caused by a non-Gaussian probability density function.

Two methods were introduced in literature to overcome the degeneracy problem in literature. The first one is the utilization of optimum importance density as explained in section 3.1. The optimum importance density is given as:

$$q(s_k \mid s_{k-1}^i, b_k)_{optimum} = p(s_k \mid s_{k-1}^i, b_k) \tag{7.1}$$

which can be simplified as,

$$p(s_k \mid s_{k-1}^i, b_k) = \frac{p(b_k \mid s_k)\, p(s_k \mid s_{k-1}^i)}{p(b_k \mid s_{k-1}^i)} \tag{7.2}$$

For a given s_{k-1}^i (which is the sample inherited from previous time step), by the means of DEMC sampling, we can draw samples from the optimum importance density.

If s_{k-1}^i is given and we know how many times it has been re-sampled in the previous time step, we can perform DEMC sampling in order to generate the needed particles. We will use $p(s_k \mid s_{k-1}^i)$ as the proposal density for generating initial values to the DEMC chains. The initial samples are generated by $s_k^n = f(s_{k-1}^n) + N(0,\sigma)$ as in Condensation algorithm. The fitness function is reduced to $p(b_k \mid s_k)\, p(s_k \mid s_{k-1}^i)$ because the denominator is the same for all generated proposals. The proposals are generated and rejected or accepted according to the DEMC algorithm.

After the DEMC sampling step the generated samples have higher likelihood values which significantly reduces the sample degeneracy. The mechanism of DEMC sampling also increases the variance of the generated samples, which diminishes the sample impoverishment.

We have slightly modified the proposal generation of the DEMC algorithm. We also utilize the best sample in the chain in order to generate new proposals by using an analogy to the DE/rand-to-best/1/exp version of DE algorithm. We aim to reduce the iteration number for DEMC sampling by shifting the proposals to the best sample. From the optimization point of view, this increases the speed of convergence to the optimum value.

Below you can find a sketch of the DEMC Particle Filter.

DEMC Particle Filter

- Initialization: Generate samples $\{s_0^n, \pi_0^n\}$ from initial distribution for $n = 1,2,...,N$, where s' are the samples, $\pi's$ are the weights assigned to each sample and *N* is the number of samples.
- Iterate for *k=1,2,...* and at time step *k*, propagate the probability density as follows:
    - NSS$_n$ is the number of samples that needs to be sampled from sample $\{s_{k-1}^n\}$ assigned by the previous re-sampling step. Draw samples $\{s_k^{m,n}\}$ given $\{s_{k-1}^n\}$ according to $p(s_k \mid s_{k-1})$ where $m = 1,\ldots, NSS_n$. $\{s_k^{m,n}\}$ are the samples that are propagated from $\{s_{k-1}^n\}$ according to $s_k^{m,n} = f(s_{k-1}^n) + v_m$ where $v_m \sim N(0,\sigma)$.
    - Iterate for *i=1,...,I.* where *I* is the number of DE iterations. For each *m,n,*
        - Calculate the new weight, $\pi_k^{m,n} = p(b_k \mid s_k^{m,n})p(s_k^{m,n} \mid s_{k-1}^n)$, where $p(b_k \mid s_k^{m,n})$ is the likelihood value of the measurement b_k given sample $s_k^{m,n}$.
        - Generate a perturbated sample $s_k^{p,n}$, which is generated according to, $s_k^{p,n} = s_k^{m,n} + a.(s_k^{Rbest,n} - s_k^{m,n}) + c.(s_k^{R1,n} - s_k^{R2,n}) + uniform(-b,b)$ as described in DE algorithm. $s_k^{n\,Rbest}$ is the sample with greatest likelihood in set $\{s_k^{m,n}\}$.
        - Assign a weighted temperature where,

$$T = TempCoef.p(s_k^{p,n} \mid s_{k-1}^n).$$

        - If $\log(\pi_k^{m,n}) > T.\log(u)$ where $u \sim uniform(0,1)$, $s_k^{p,n}$ is a new

DEMC Particle Filter continues

- For each sample *n*, find the best s_{k-1}^{n} with maximum π_k^n by,

$$\pi_k^n = p\ (b_k \mid s_{k-1}^n) = \sum_{m=1}^{M} p\ (b_k \mid s_k^{m,n}).p\ (s_k^{m,n} \mid s_{k-1}^n)$$

- Calculate the estimation of s_k, by $\hat{s}_k = \sum_{m=1}^{M} s_k^{m,n=best} / NSS_{n=best}$
- Combine all $\{s_k^{mn}\}$ into $\{s_k^n\}$ and store the new weights as, π_k^n and normalize so that $\sum_n \pi_k^n = 1$.
- Resample *N/3* samples from $\{s_k^n\}$ according to π_k^n and assign three times the calculated re-sampling as NSS_n for each sample.

The temperature value in DEMC sampling step is adjusted by scaling with $p(s_k^{p,n} \mid s_{k-1}^n)$, which reduces the probability of generating samples that are less probable according to the time propagation.

DEMC Particle Filter is summarized above; the flow of the particle filter is also presented in the Figure 7.1

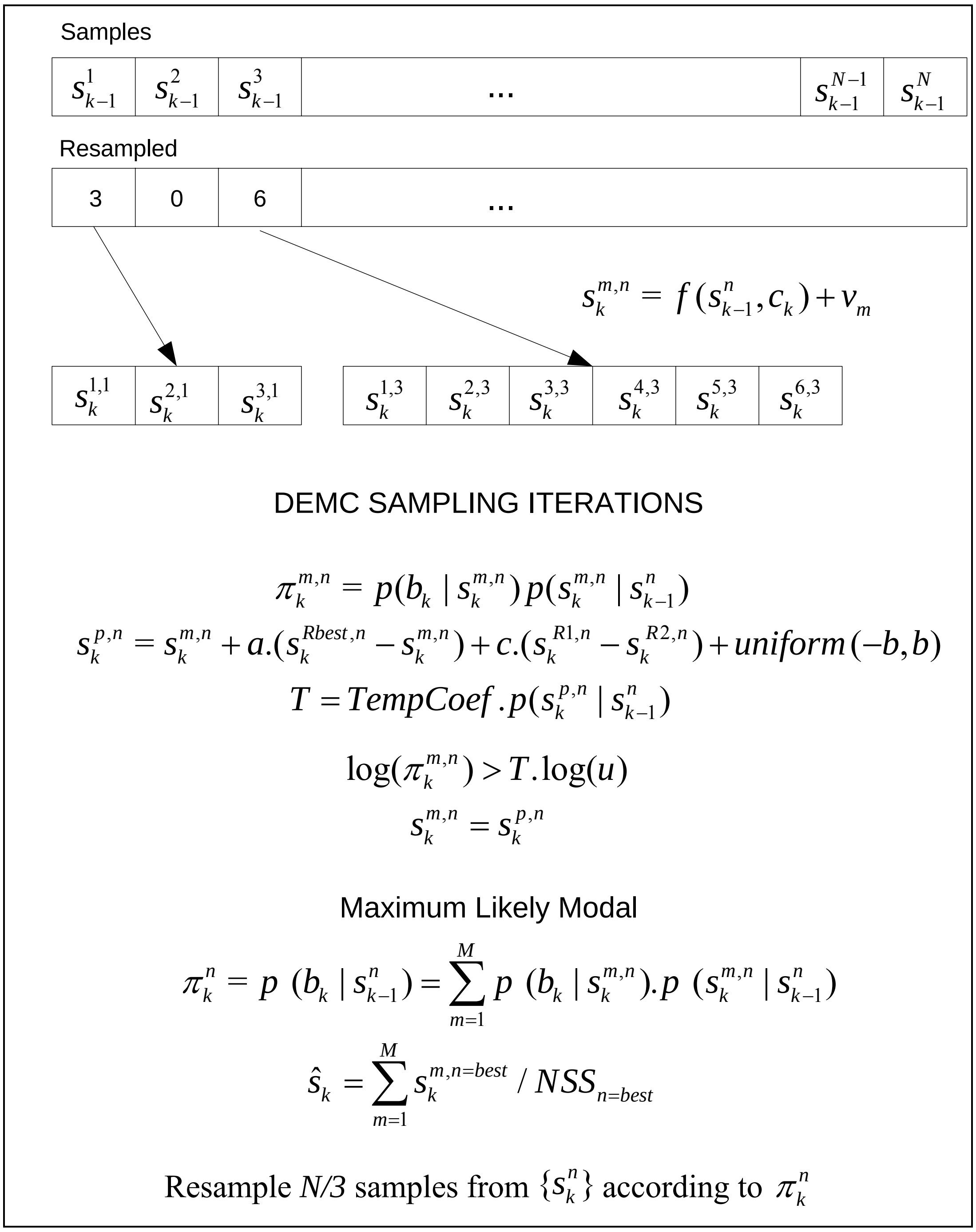

Samples
s_{k-1}^{1} s_{k-1}^{2} s_{k-1}^{3} ... s_{k-1}^{N-1} s_{k-1}^{N}
Resampled
3 0 6 ...
$s_k^{m,n} = f(s_{k-1}^{n}, c_k) + v_m$
$s_k^{1,1}$ $s_k^{2,1}$ $s_k^{3,1}$
$s_k^{1,3}$ $s_k^{2,3}$ $s_k^{3,3}$ $s_k^{4,3}$ $s_k^{5,3}$ $s_k^{6,3}$
DEMC SAMPLING ITERATIONS
$\pi_k^{m,n} = p(b_k \mid s_k^{m,n})\, p(s_k^{m,n} \mid s_{k-1}^{n})$
$s_k^{p,n} = s_k^{m,n} + a.(s_k^{Rbest,n} - s_k^{m,n}) + c.(s_k^{R1,n} - s_k^{R2,n}) + uniform(-b,b)$
$T = TempCoef\,.p(s_k^{p,n} \mid s_{k-1}^{n})$
$\log(\pi_k^{m,n}) > T.\log(u)$
$s_k^{m,n} = s_k^{p,n}$
Maximum Likely Modal
$\pi_k^{n} = p\ (b_k \mid s_{k-1}^{n}) = \sum_{m=1}^{M} p\ (b_k \mid s_k^{m,n}).p\ (s_k^{m,n} \mid s_{k-1}^{n})$
$\hat{s}_k = \sum_{m=1}^{M} s_k^{m,n=best} / NSS_{n=best}$
Resample $N/3$ samples from $\{s_k^n\}$ according to π_k^n

Figure 7.1 : DEMC Particle Filter Flow Diagram

7.1 Occlusion Detection in DEMC Particle Filter

Occlusion detection can also be augmented to the DEMC Particle Filter. Let $\hat{s}_k$ be the filtered state at time step k. Occlusion detection can be formulated as,

$$L(b_k) = \frac{p(b_k \mid H_1)}{p(b_k \mid H_0)}$$
$$L(b_k) > \tau \qquad \textbf{(7.3)}$$

$L(b_k)$ is the Neuman-Pearson criteria for detection. If $L(b_k)$ is greater than a given threshold τ, we assume there is a detection. If not, there is supposed to be no license plate visible in the video frame. Neuman-Pearson criteria can be calculated in the DEMC Particle Filter as follows,

$$L(b_k) \approx \frac{\sum_{n=1}^{N} p_{H1}(b_k \mid s_k^n)}{\sum_{n=1}^{N} p_{H0}(b_k \mid s_k^n)} \approx \frac{p_{H1}(b_k \mid \hat{s}_k)}{p_{H0}(b_k \mid \hat{s}_k)} \qquad \textbf{(7.4)}$$

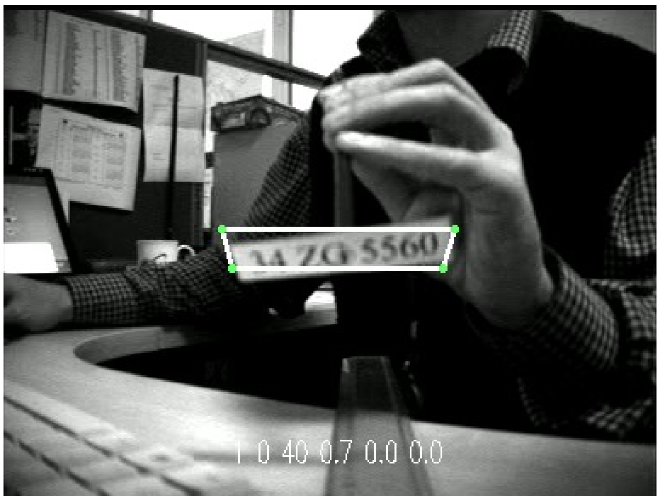
$L(b_k)$ is calculated before each re-sampling step. If $L(b_k) > \tau$ is satisfied, re-sampling is performed, else re-sampling is omitted.

8. UTILIZING DEMC PARTICLE FILTER

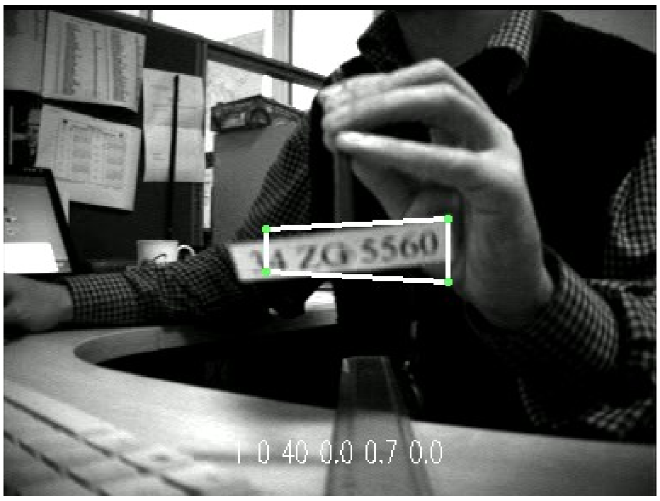
We supplied the theoretical background for 3D projection, state space models and stochastic estimation concept in previous sections. We also detailed the stochastic tracking concept and stochastic tracking algorithms. As mentioned already, our main objective is 3D tracking of license plates from monocular camera view. In this section, we explain the application of DEMC Particle Filtering to a real world scenario, namely 3D license plate tracking.

We will try to elaborate the main ideas, which have been explained theoretically in previous sections. First question arising is as following. “What do the samples s_k mean in our application of license plate tracking?”

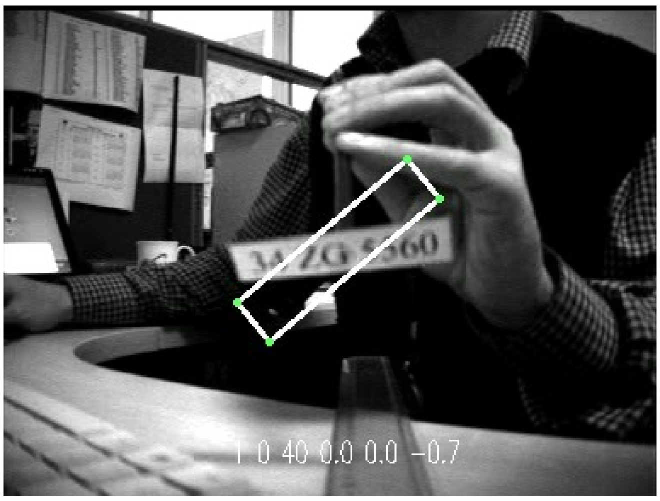
For better understanding we may look at some sample state vectors and their representation for real world in Figure 8.1.



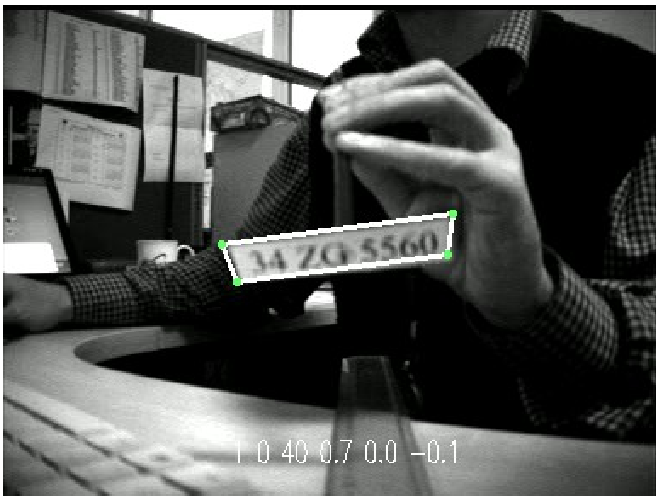
(a)


34 ZG 5560
1 0 40 0.0 0.7 0.0

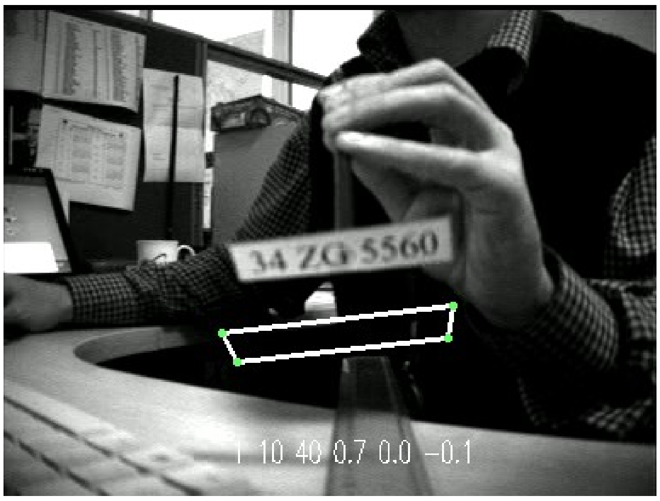
(b)


34 ZG 5560
1 0 40 0.0 0.0 -0.7

(c)


34 ZG 5560
1 0 40 0.7 0.0 -0.1

(d)



(e)

Figure 8.1 : Samples and their Representations

(a) $X = 1\ Y = 0\ Z = 40\ \alpha = 0.7\ \beta = 0\ \gamma = 0$

(b) $X = 1\ Y = 0\ Z = 40\ \alpha = 0\ \beta = 0.7\ \gamma = 0$

(c) $X = 1\ Y = 0\ Z = 40\ \alpha = 0\ \beta = 0\ \gamma = -0.7$

(d) $X = 1\ Y = 0\ Z = 40\ \alpha = 0.7\ \ \beta = 0\ \gamma = -0.1$

(e) $X = 1\ Y = 10\ Z = 40\ \alpha = 0.7\ \beta = 0\ \gamma = -0.1$

Thus, it can easily be seen that, each sample corresponds to a translation and rotation of the license plate. Each sample is a hypothesis for the position and orientation of the license plate. On the other hand, we do not have only one suggestion for the state of the license plate, we have a set of samples s_k^n each with different state variables and each make a different suggestion for the license plate position and orientation. Now we can answer to the next question. “What do we do with all those samples?”

First thing to mention is that, samples evolve in time. For example,

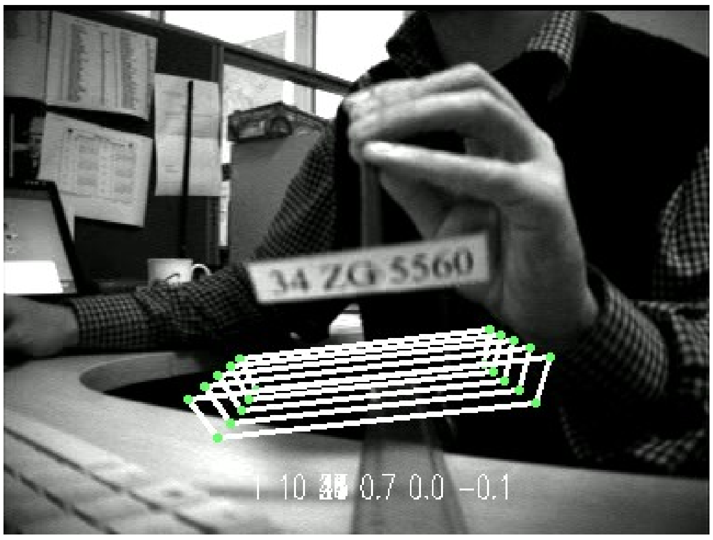


Figure 8.2 : License Plate Dynamics

in Figure 8.2, a sample is moving forward thus, approaching the camera. This is a consequence of the propagation of the samples according to the dynamic model given by $s_k^n = f(s_{k-1}^n, c_k) + N(0,\sigma)$.

Remembering the Kalman time update equations, we may notice the similarity of the propagation in Condensation algorithm and time update step in Kalman filter. After the time update step, we have a measurement update step in Kalman equation set. This step corresponds to the re-sampling step of the Condensation algorithm. For the DEMC Particle Filter, re-sampling step also includes DE optimization iterations. In order to understand the re-sampling step, we need to be familiar with the probability distribution concept and its propagation in time. Analogical to sampling in time, we may think that the samples s_k^n are samples from the probability distribution $p(s_k)$, therefore s_k^n's are representing a probability distribution. On the other hand, $p(s_k)$ is measurement dependent and given as $p(s_k \mid b_k)$. From Bayesian point of view we use $p(b_k \mid s_k)$ in our algorithms. We may talk about a deterministic drift of the samples by dynamic model and then a correction according the given measurement on the probability distribution. By means of this recursive updates, the algorithm is able to track a probability distribution and so an object. The mean of this probability distribution is the prediction and corrected (filtered) output of the filter. As an add-on or enhancement to the Condensation algorithm, DEMC Particle Filter applies additional DE optimization iterations at the re-sampling step. This is where the DEMC Particle Filter has advantage on Condensation algorithm and reduces the number of required samples and avoids sample degeneracy. Additional DE optimization iterations propagate the samples to more probable areas of the distribution so better represents the probability distribution. We can observe such a propagation of samples to more probable position in Figure 8.3. From frame 1 to 4 the bold square is the current mean of the distribution and the thin squares are the 10 samples projected on to the image. The samples propagate to the given template pattern, which is the triangular logo (see on the top left corner of the images) on the pencil box.

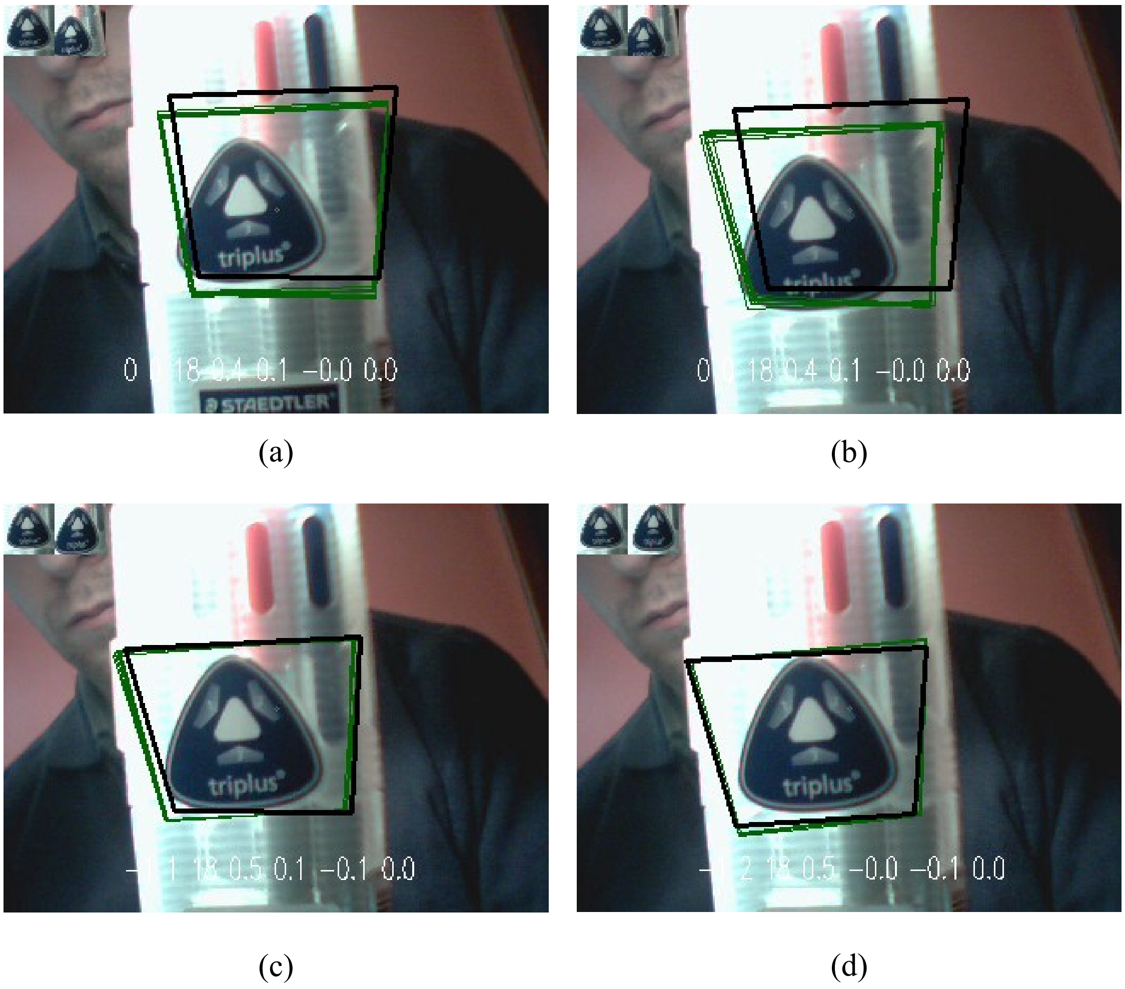


Figure 8.3 : Propagation of the Samples in Time (a) Frame 1 (b) Frame 2 (c) Frame 3 (d) Frame 4

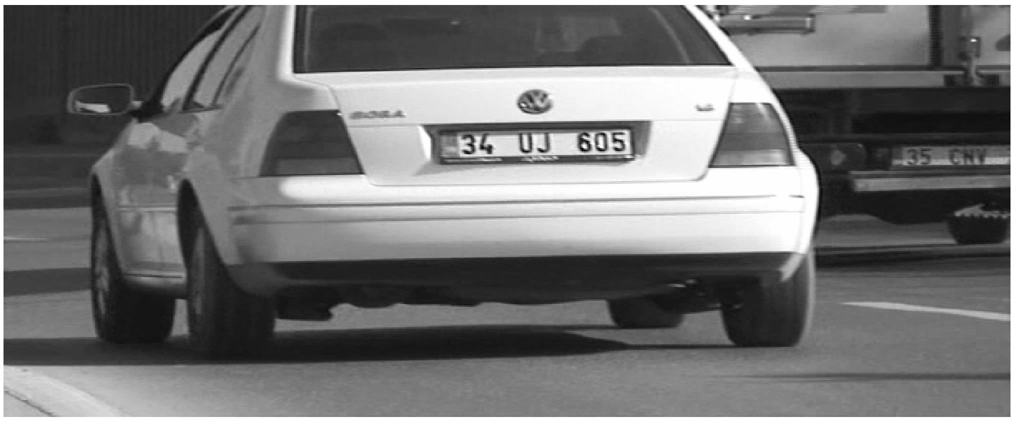


Figure 8.4 : Video Frame Sample

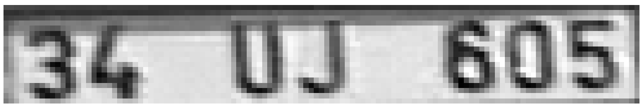


Figure 8.5 : Template

8.1 Likelihood Calculation

$\pi_k^n = p(b_k^n \mid s_k^n)$ likelihood values are obtained by simple template matching. The matching is performed against the template that is extracted from the initial frame. A sample video frame is shown in Figure 8.4 . The template extracted from this frame can be as shown in Figure 8.5. There is no update on this template during the tracking in our application. The template is normalized in order to obtain an image with zero mean and unit variance. The template matching is then performed as follows,

$$likelihood(x,y) = \sum_{n=0}^{T_x} \sum_{m=0}^{T_y} [\hat{I}_{x,y}(n,m) - T(n,m)]^2 \qquad \textbf{(8.1)}$$

where T_x, T_y are the template dimensions, $T(n,m)$ is the zero mean, unit variance template and $\hat{I}_{x,y}(n,m)$ is the extracted image region around *(x,y)* which is processed to have zero mean and unit variance. Because of the 3D nature of the system state, this image region also needs to be warped according to the sample's Euler angles.

Likelihood values calculated for a window, which is shifted around the true license plate coordinates (in a 100x50 pixels neighborhood), is shown in Figure 8.6. Please notice that in this likelihood calculation, no warping operation is performed before template matching.

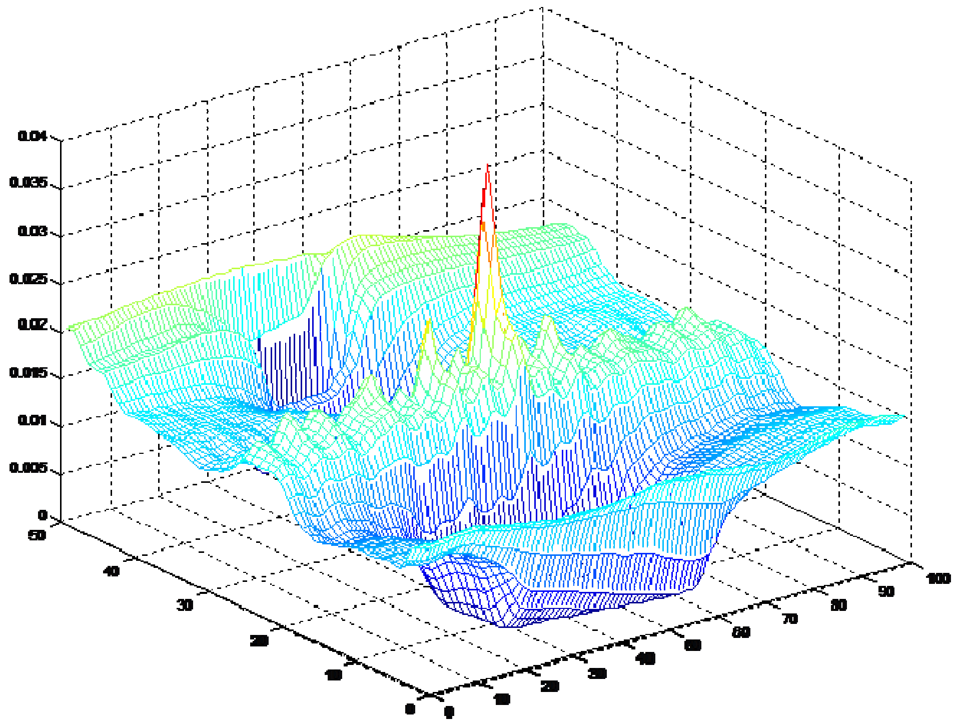


Figure 8.6 : Likelihood Function Plot

There is an important step that has to be performed before template matching; that is the warping of extracted image before matching to the given template. The warping is performed according to the sample values, namely the 3D coordinates X, Y, Z and the α, β, γ Euler angles. 3D license plate corners are first projected to the image plane by camera model and each 2D corner coordinate is fed to the warping function. The perspective transform is calculated from given 2D corner coordinates. The result of warping operation is shown in Figure 8.7. In this example the template is warped, instead in the application we need to warp the extracted image region.

We will try to evaluate the performance of four algorithms, namely Auxiliary Particle Filter, Condensation Algorithm, Genetic Condensation Algorithm and DEMC Particle Filter in the next section for the application of license plate tracking.

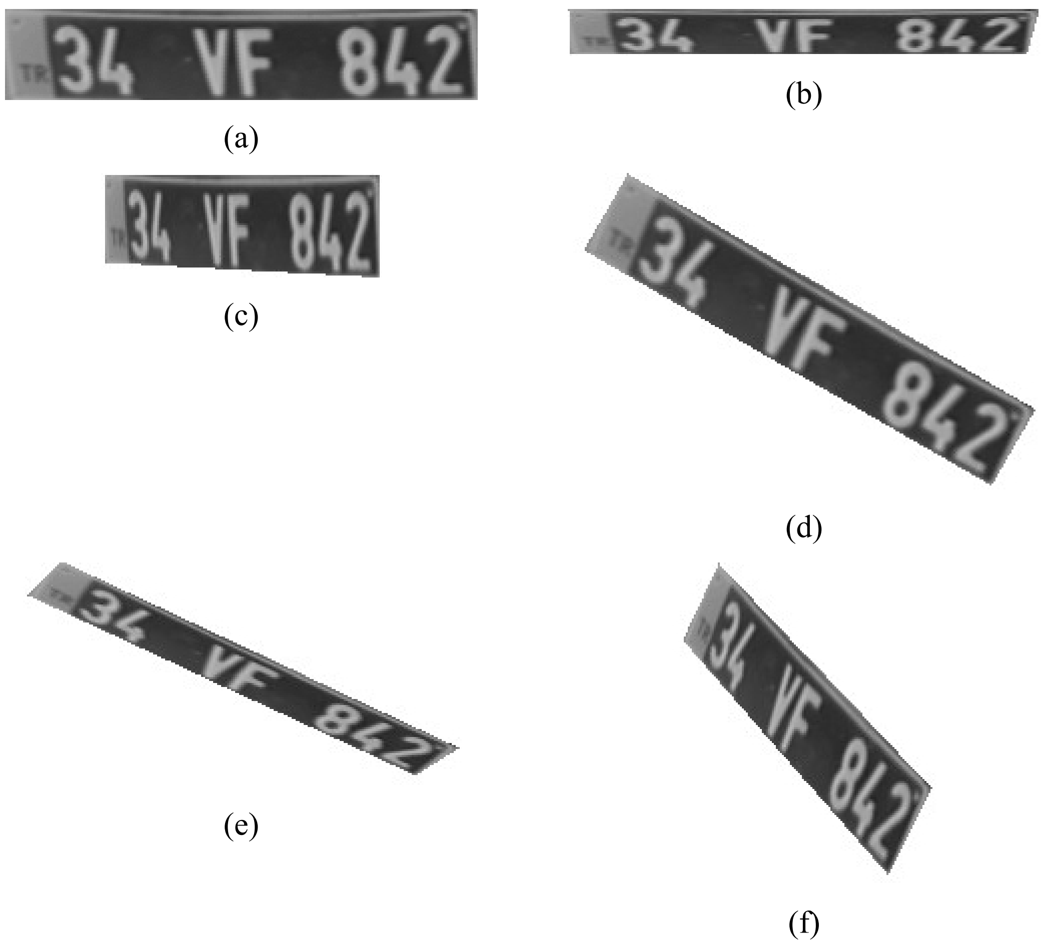


Figure 8.7 : 3D Projection of the License Plate in Various Euler Angles (a) $\alpha = 0, \beta = 0, \gamma = 0$ (b) $\alpha = 60^{\circ}, \beta = 0, \gamma = 0$ (c) $\alpha = 0, \beta = 60^{\circ}, \gamma = 0$ (d) $\alpha = 0, \beta = 0, \gamma = 30^{\circ}$ (e) $\alpha = 60^{\circ}, \beta = 0, \gamma = 30^{\circ}$ (f) $\alpha = 0, \beta = 60^{\circ}, \gamma = 30^{\circ}$

9. EXPERIMENTAL RESULTS

Before evaluating DEMC Particle Filter, it is necessary to explain the requirements of a robust tracking algorithm.

First, a tracking algorithm must decrease the error rate of the noisy measurements. Thus, the error rate of the output of the algorithm must be less than the input, namely noisy measurements. This decrease in error rate is due to the employed system model and its accuracy in prediction of the next state of the system. Another requirement of the tracking algorithm is its robustness, in other words, its strength in estimating the right ROI in the application and insisting on the right path of solution avoiding jumping to other possible system solutions. We utilize a tracking algorithm in order to restrict our attention into a defined ROI, thus we want to eliminate the rest of the image from processing. Thus, the tracking algorithm's processing complexity should be less then processing all the input. Otherwise, the problem can be solved by brute force. Last but not the least, tracking algorithm should tolerate to a degree of missing measurements. Thus, for example in the presence of occlusion, the algorithm should possibly sustain its stability for a required period of time.

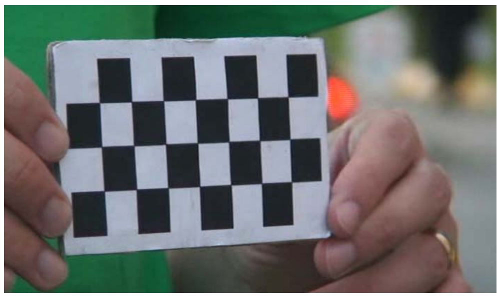
In order to evaluate the DEMC Particle Filter, we have implemented Auxiliary Particle Filter, Condensation Algorithm and Genetic Condensation Algorithm. We evaluated these algorithms by tracking license plates on 12 test video sequences.

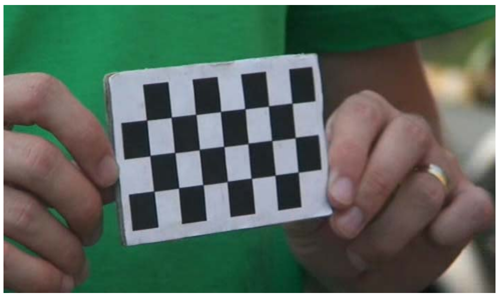
The tests are performed with calibrated camera and manually determined license plate 2D corner coordinates as a ground truth. This ground truth is also utilized for track initialization. Therefore, the initial frame for each vehicle has no error, but the proceeding frames yield corner displacement errors according the tracking performance of the filters.

9.1 Calibration for Each Video Sequence

In order to utilize our framework of tracking on the test sequences, our algorithm requires camera parameters. We do not apply a coordinate transform between the camera and the outdoor environment because we did not need to define a world coordinate system other than camera coordinate system. Thus, we only require the focal length (f), the pixels size (k_u, k_v) and the image coordinates of the projection center $(u_0, v_0)^T$. The projection center is assumed to be the sensor center, which dictates that the projection center is the half of the image dimensions. We utilized the calibration method in [50]. This method requires a couple of calibration images, which include the calibration pattern clearly seen. The calibration pattern is a 8x4 chessboard in our application. The dimensions of the calibration pattern is fed to the algorithm and at least 10 calibration images are used for each test video. Example calibration images can be seen in Figure 9.1.

(a) Example Calibration Image 1


(b) Example Calibration Image 2


(c) Example Calibration Image 3

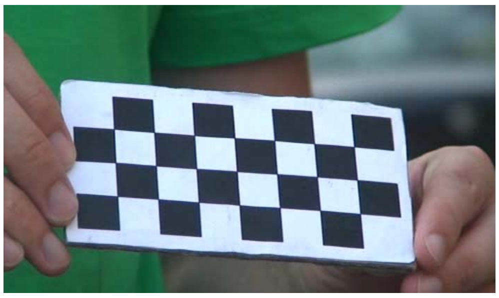
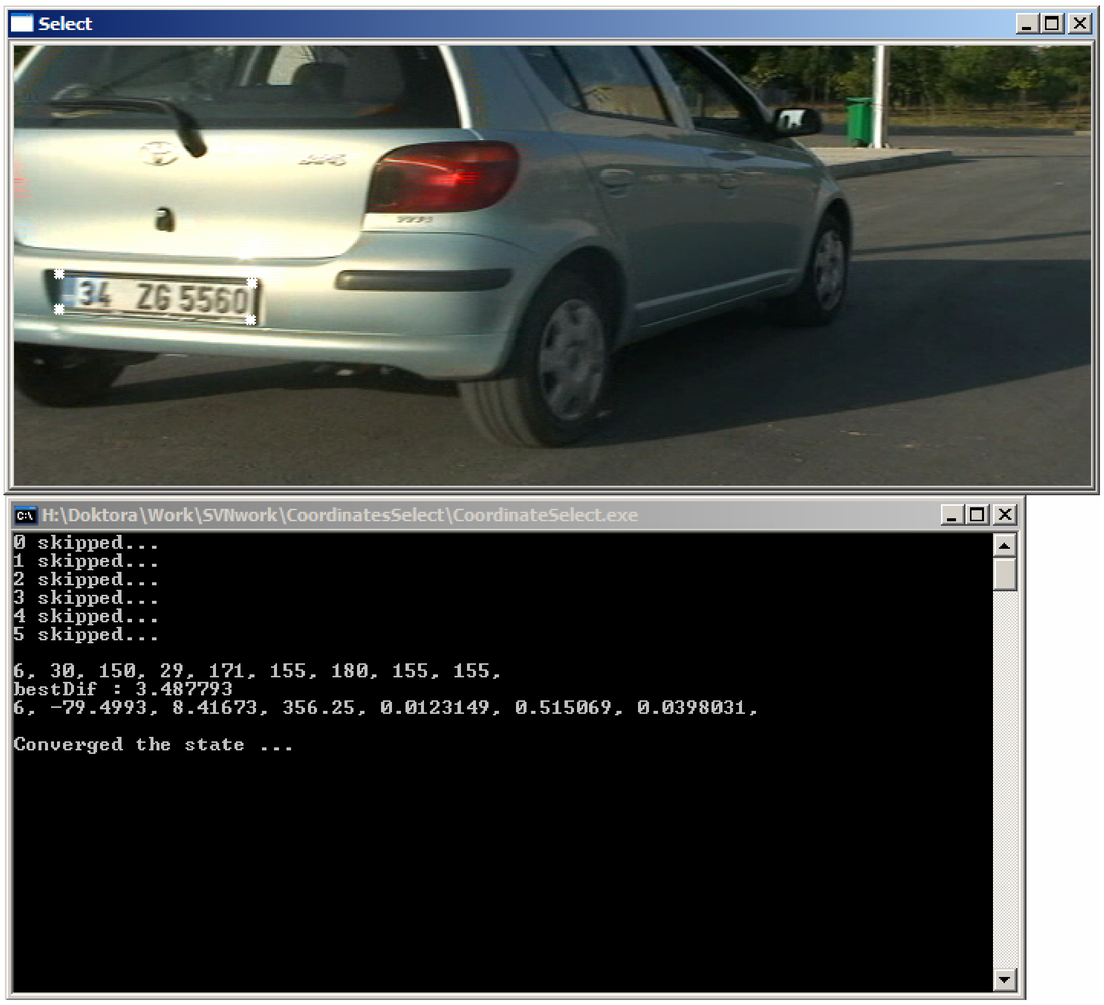

(d) Example Calibration Image 4

Figure 9.1 : Example Calibration Images (a) Image 1 (b) Image 2 (c) Image 3 (d) Image 4

The calibration algorithm outputs the parameters $\alpha_u = \frac{f}{k_u}$ and $\alpha_v = \frac{f}{k_v}$ which we will use in the test data extraction and tracking itself.

9.2 Composing the Test Data for Video Sequences

Before testing each particle on the test sequences, the ground truth for each frame must be extracted. This operation is performed manually using a GUI program, which helped to select the license plate corners.


Select
H:\Doktora\Work\SVNwork\CoordinatesSelect\CoordinateSelect.exe

```
0 skipped...
1 skipped...
2 skipped...
3 skipped...
4 skipped...
5 skipped...

6, 30, 150, 29, 171, 155, 180, 155, 155,
bestDif : 3.487793
6, -79.4993, 8.41673, 356.25, 0.0123149, 0.515069, 0.0398031,

Converged the state ...
```

Figure 9.2 : License Plate Coordinates Selection Program

This utility has also a process of controlling the selected license plate corners by optimizing it according to the 3D model of the license plate. Thus, the program includes an optimization step, which converges to the 3D state of the license plate. The optimized cost function that is given as:

$$\arg\min_{(XYZ\alpha\beta\gamma)} = \sum_{i=1}^{4} (x_i^{projected} - x_i^{selected})^2 + (y_i^{projected} - y_i^{selected})^2 \qquad \textbf{(9.1)}$$

The optimization is performed on the state $s \triangleq (XYZ\alpha\beta\gamma)^T$, the difference of the four projected and selected corners is minimized.

If the convergence is supplied in a limited number of iterations and with a required error limit, the four corners are accepted as the license plate corner coordinates. Also after the optimization, we get the approximated 3D state of the license plate, which is used for initialization of the filters at the very beginning of tracking.

9.3 Performance Comparison

DEMC Particle Filter, Auxiliary Particle Filter, Condensation Algorithm and Genetic Condensation Algorithm are evaluated on 12 test videos, which include 108 vehicles. Each vehicle is tracked maximum of 25 frames after the initialization frame. Totally 2205 frames of tracking performed for evaluation. The filters are utilized with the same number of particles and with the same number of likelihood calculation. Considering the number of particles *N*, Condensation algorithm has complexity of *O(N)*, Auxiliary Particle Filter has *O(2N),* DEMC Particle Filter has *O(3N)* (in case of 2 DEMC iterations), Genetic Condensation has *O(4N)* complexity. If we want to equalize the number of likelihood computations and if Condensation algorithm has 360 particles, Auxiliary Particle Filter will have 180 particles, DEMC Particle Filter will have 120 particles and Genetic Condensation will have 90 particles, which is the case in the test setup.

The general parameters for the DEMC Particle Filter is given as,

Table 9.1: Parameters for DEMC Particle Filter

Parameter	**Value**
a	0.8
c	0.2
Temperature Coefficient	1e-3
b	[0.05,0.05,0.05,0.005,0.005,0.005]
Number of iterations I	2

and the general parameters for the Genetic Condensation algorithm is given as,

Table 9.2: Parameters for Genetic Condensation Algorithm

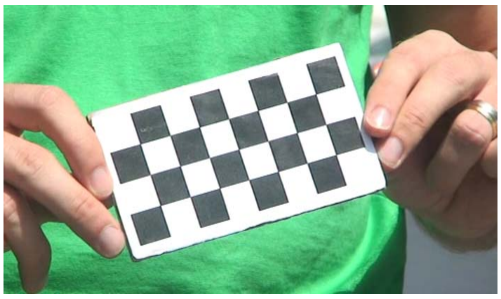
Parameter	**Value**
b	[4,4,15,0.01,0.01,0.005]

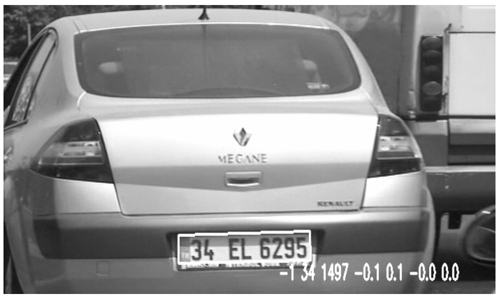
Initial speed for all the filters at the initialization step is given as 20 cm per frame, which makes 18 km/h (at 25 fps). The speeds of vehicles at the test sequences are generally between 10-50 km/h. These speeds are very common in urban area. If the tracking is performed in highways, the configuration for initial speed, process noise variance etc. should be tuned accordingly.

The videos are captured by a Sony Handy cam, which can capture 704x576 at the frame rate of 25 fps. This video is interleaved; therefore, we needed to discard even rows of the image, which resulted in a 704x288 resolution. The camera parameters are calibrated according to this resolution.

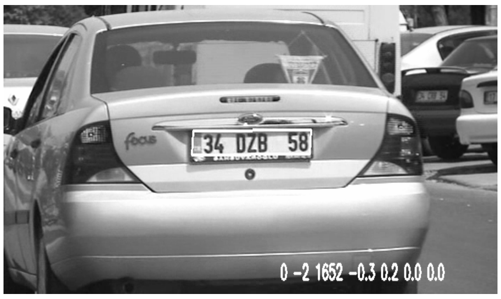
The tracking is performed on grayscale images. Color information is not utilized in tracking. The main reason for not utilizing color information is to make the approach also usable with IR cameras, which at the end will allow us to work at night.

Sample frames from 12 test videos are presented below. The tracking results are listed after each test video sequence. The corner displacement error is given as pixels per corner.

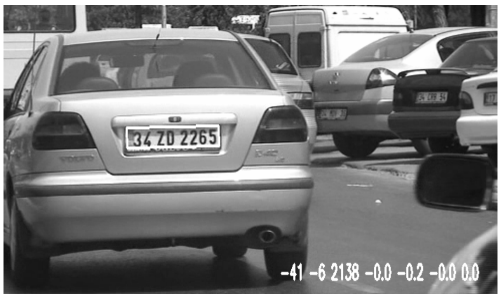

(a) Calibration Image



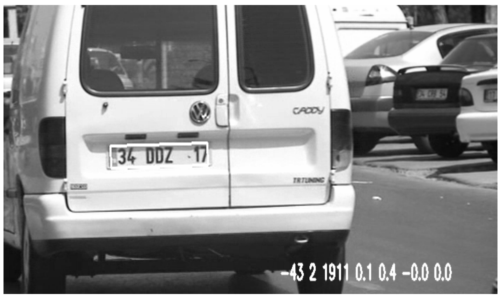
(b) Video 1 Frame 12



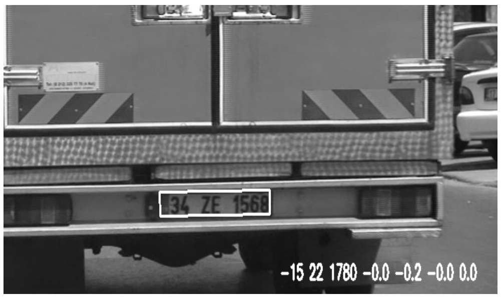
(c) Video 1 Frame 44



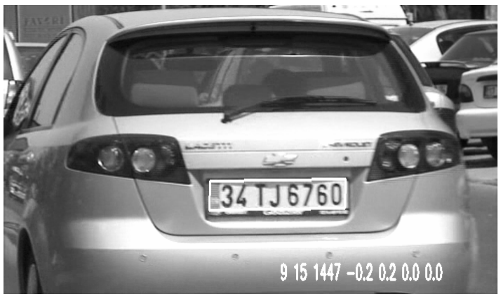
(d) Video 1 Frame 116



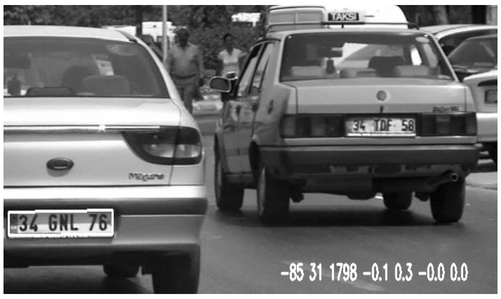
(e) Video 1 Frame 156



(f) Video 1 Frame 252



(g) Video 1 Frame 376



(h) Video 1 Frame 476

Figure 9.3 : Test Video Sequence 1 (a) Calibration Image (b) Frame 12 (c) Frame 44 (d) Frame 116 (e) Frame 156 (f) Frame 252 (g) Frame 376 (h) Frame 476

Video Sequence : 1

Number of Vehicles : 10

Number of Tracked Frames : 210

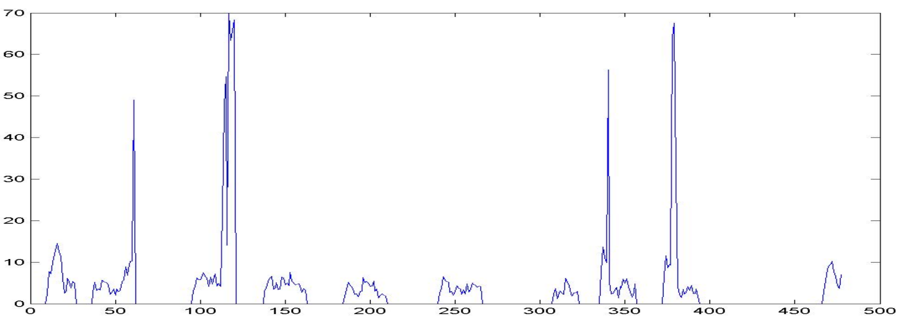
Process Noise Standard Deviation σ in equation 6.1 : [10,5,30,0.02,0.02,0.01]

Table 9.3: Test Video 1 Tracking Results with Same Computation Load

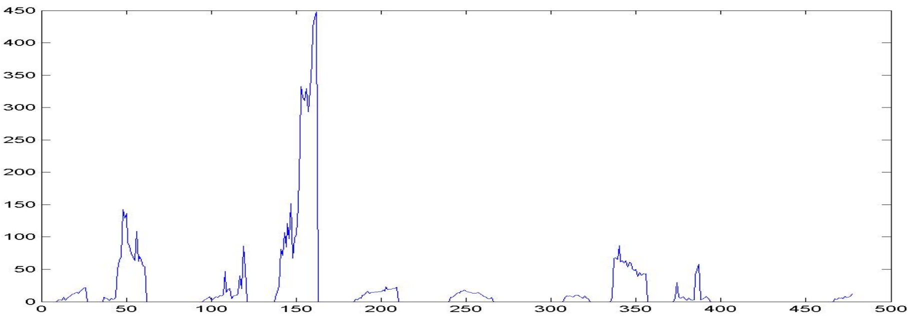
Algorithm	**Number of Particles**	**Number of Likelihood Calculations**	**Total Corner Displacement Error**	**Average Displacement Error per Corner**
Condensation	360	360	6498	7.73
Auxiliary Particle Filter	180	360	37056	44.11
DEMC Particle Filter	120	360	3399	4.04
Genetic Condensation	90	360	4302	5.12

Table 9.4: Test Video 1 Tracking Results with Same Number of Particles

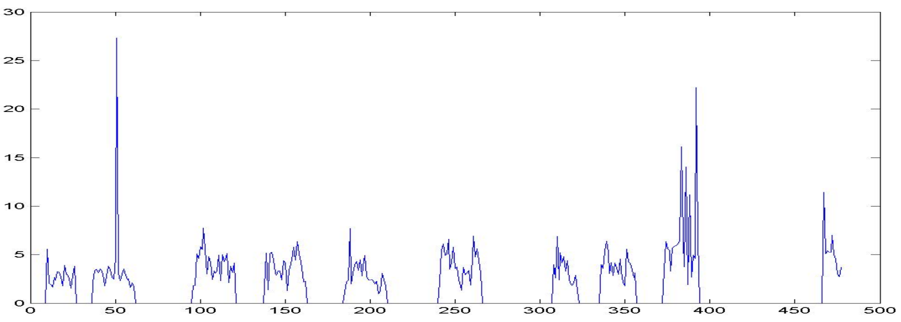
Algorithm	**Number of Particles**	**Number of Likelihood Calculations**	**Total Corner Displacement Error**	**Average Displacement Error per Corner**
Condensation	360	360	6498	7.73
Auxiliary Particle Filter	360	720	12140	14.45
DEMC Particle Filter	360	1080	2438	2.90
Genetic Condensation	360	1440	2062	2.45



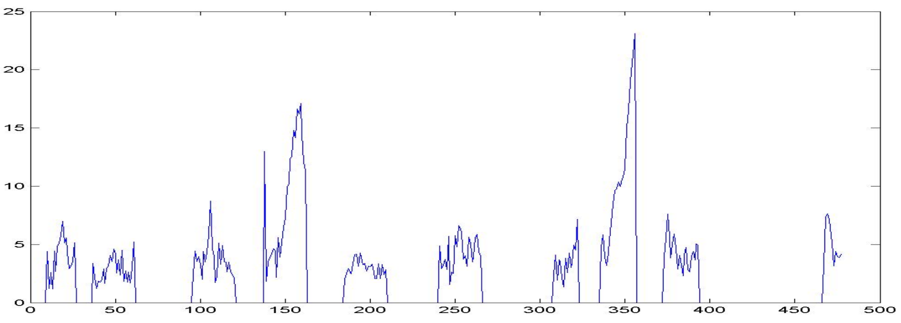
(a)



(b)

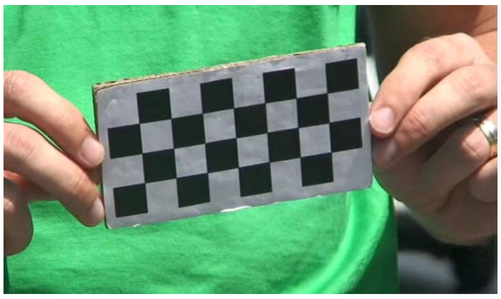


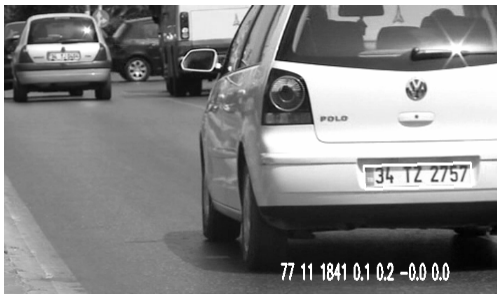
(c)



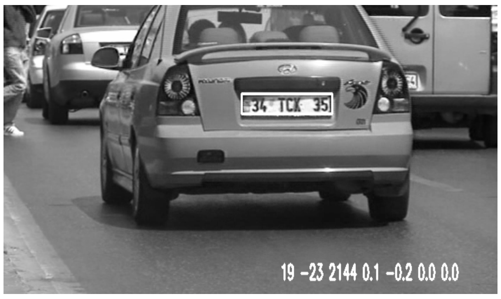
(d)

Figure 9.4 : Test Video Sequence 1 Corner Displacement Errors with Same Computation Load (a) Condensation (b) Auxiliary Condensation (c) DEMC Particle Filter (d) Genetic Condensation

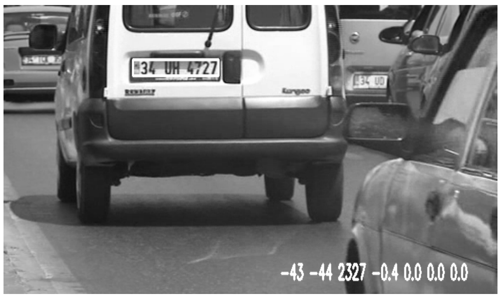

(a) Calibration Image



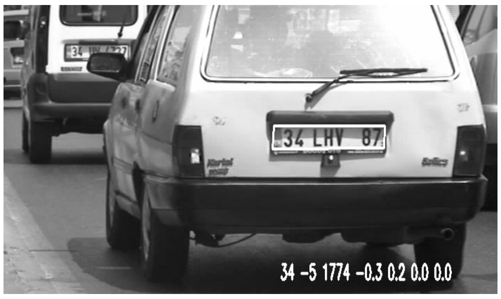
(b) Video 2 Frame 16



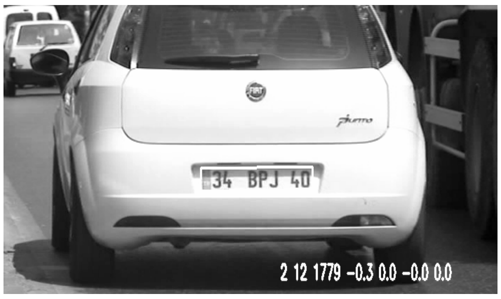
(c) Video 2 Frame 172



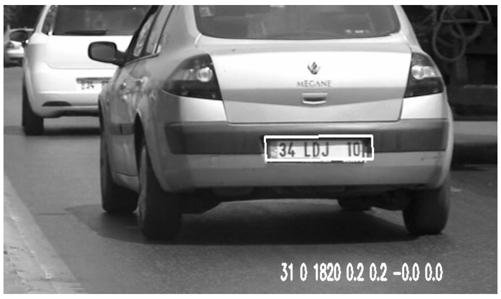
(d) Video 2 Frame 376



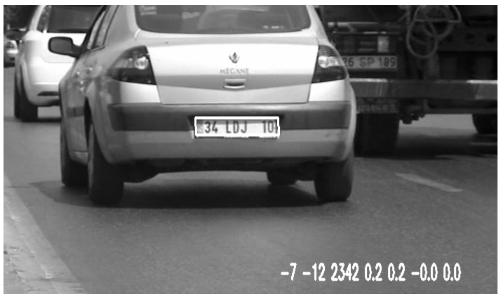
(e) Video 2 Frame 408



(f) Video 2 Frame 612



(g) Video 2 Frame 680



(h) Video 2 Frame 692

Figure 9.5 : Test Video Sequence 2 (a) Calibration Image (b) Frame 16 (c) Frame 172 (d) Frame 376 (e) Frame 408 (f) Frame 612 (g) Frame 680 (h) Frame 692

Video Sequence : 2

Number of Vehicles : 7

Number of Tracked Frames : 175

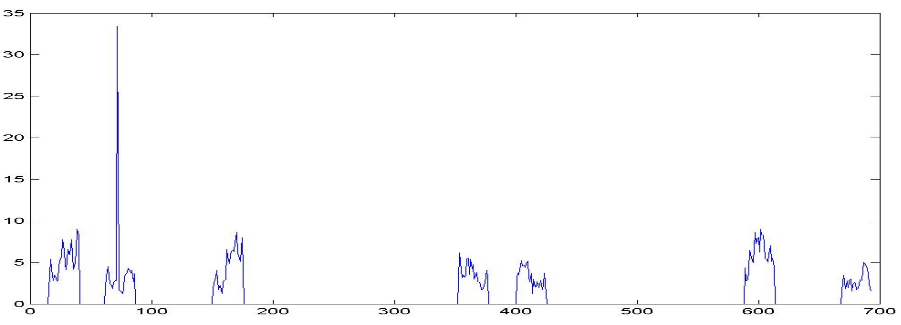
Process Noise Standard Deviation σ in equation 6.1: [10,5,30,0.02,0.02,0.01]

Table 9.5: Test Video 2 Tracking Results with Same Computation Load

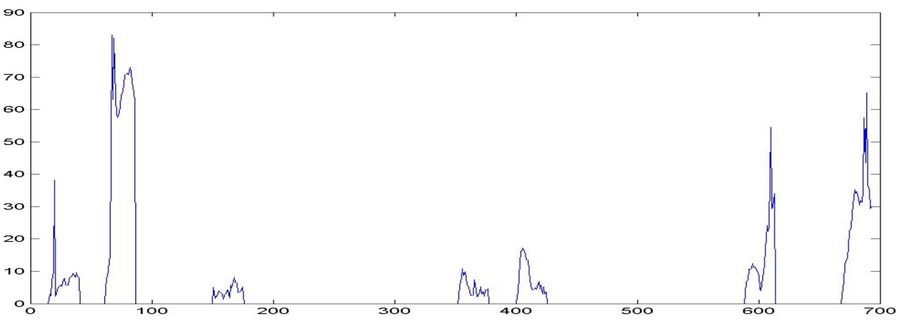
Algorithm	**Number of Particles**	**Number of Likelihood Calculations**	**Total Corner Displacement Error**	**Average Displacement Error per Corner**
Condensation	360	360	2973	4.24
Auxiliary Particle Filter	180	360	12578	17.96
DEMC Particle Filter	120	360	2676	3.82
Genetic Condensation	90	360	5591	7.98

Table 9.6: Test Video 2 Tracking Results with Same Number of Particles

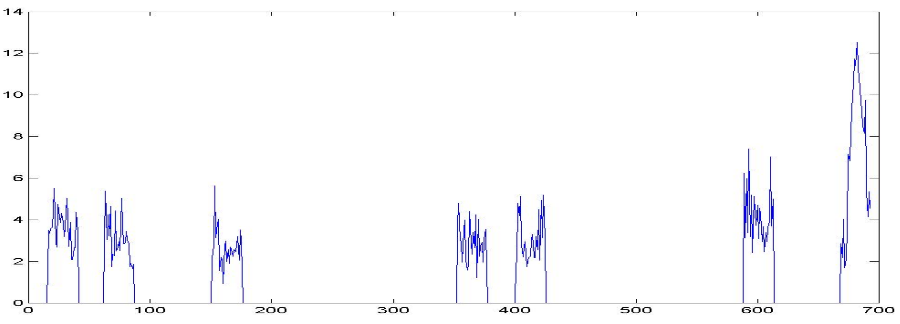
Algorithm	**Number of Particles**	**Number of Likelihood Calculations**	**Total Corner Displacement Error**	**Average Displacement Error per Corner**
Condensation	360	360	2973	4.24
Auxiliary Particle Filter	360	720	26747	38.21
DEMC Particle Filter	360	1080	1854	2.64
Genetic Condensation	360	1440	1530	2.18



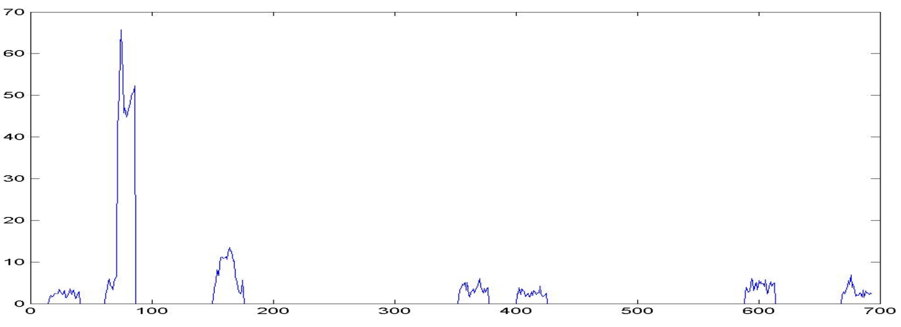
(a)



(b)



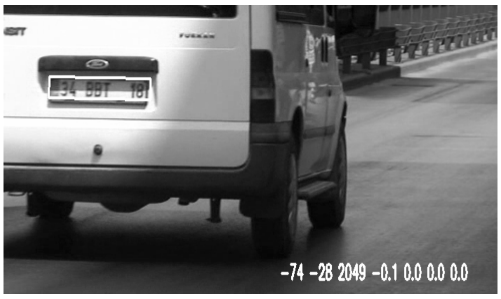
(c)



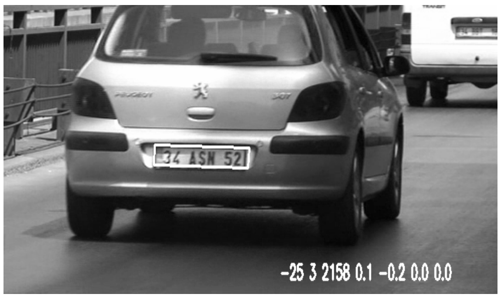
(d)

Figure 9.6 : Test Video Sequence 2 Corner Displacement Errors with Same Computation Load (a) Condensation (b) Auxiliary Condensation (c) DEMC Particle Filter (d) Genetic Condensation

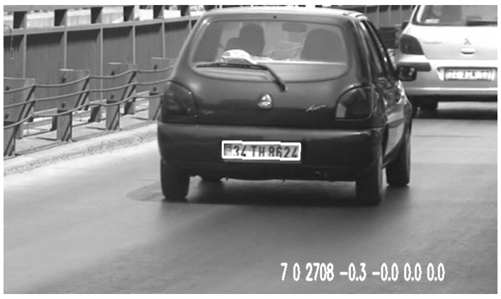
(a) Calibration Image



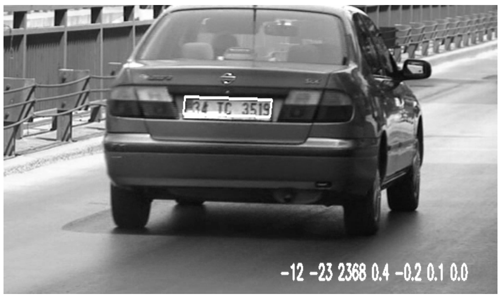
(b) Video 3 Frame 12



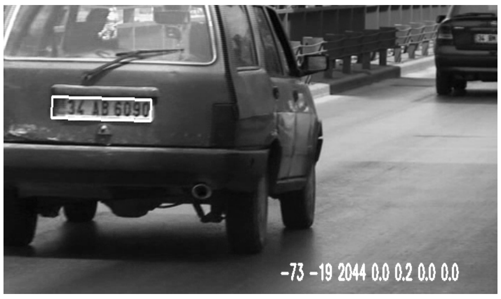
(c) Video 3 Frame 76



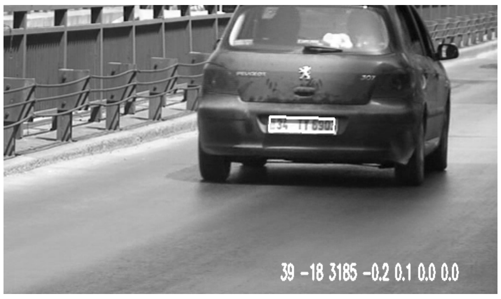
(d) Video 3 Frame 132



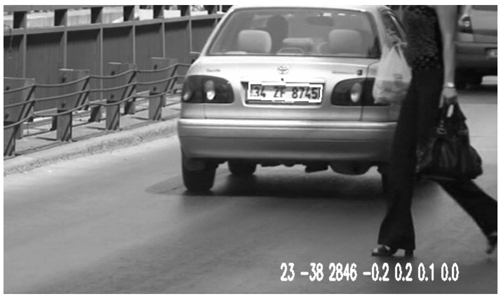
(e) Video 3 Frame 232



(f) Video 3 Frame 344



(g) Video 3 Frame 448



(h) Video 3 Frame 576

Figure 9.7 : Test Video Sequence 3 (a) Calibration Image (b) Frame 12 (c) Frame 76 (d) Frame 132 (e) Frame 232 (f) Frame 344 (g) Frame 448 (h) Frame 576

Video Sequence : 3

Number of Vehicles : 9

Number of Tracked Frames : 225

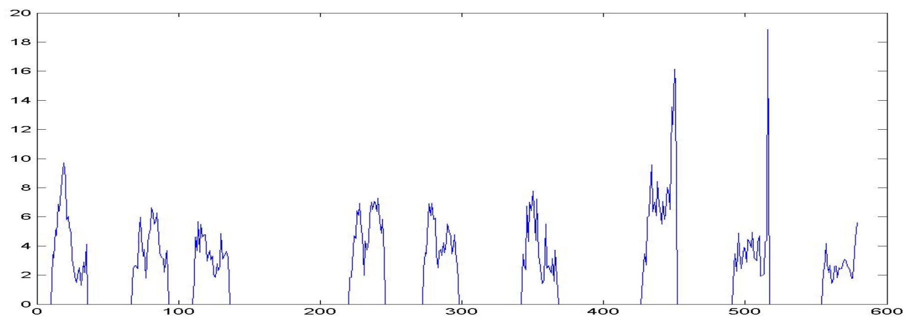
Process Noise Standard Deviation σ in equation 6.1: [10,5,30,0.02,0.02,0.01]

Table 9.7: Test Video 3 Tracking Results with Same Computation Load

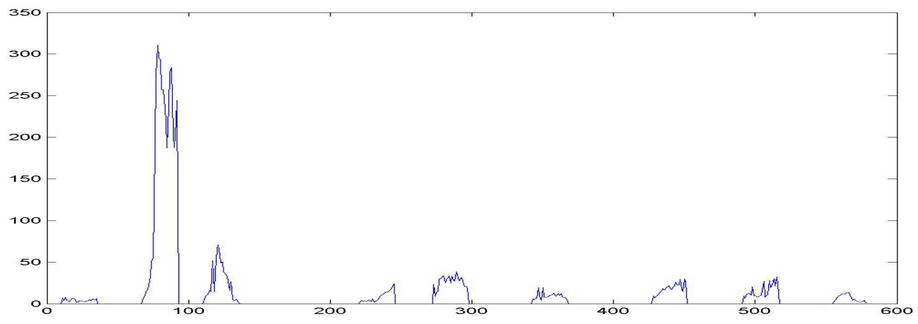
Algorithm	**Number of Particles**	**Number of Likelihood Calculations**	**Total Corner Displacement Error**	**Average Displacement Error per Corner**
Condensation	360	360	3955	4.39
Auxiliary Particle Filter	180	360	28781	31.97
DEMC Particle Filter	120	360	3548	3.94
Genetic Condensation	90	360	4080	4.53

Table 9.8: Test Video 3 Tracking Results with Same Number of Particles

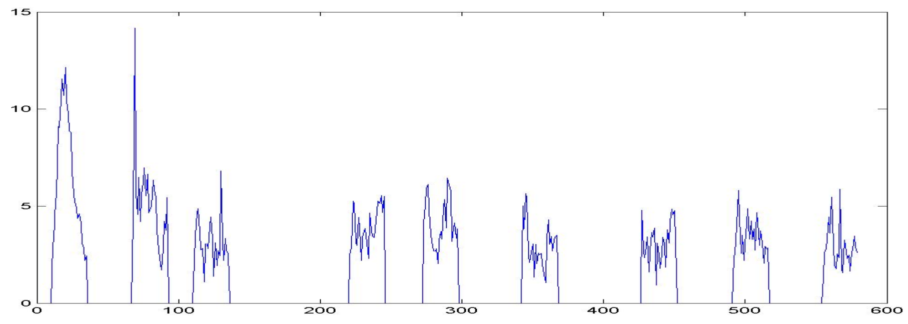
Algorithm	**Number of Particles**	**Number of Likelihood Calculations**	**Total Corner Displacement Error**	**Average Displacement Error per Corner**
Condensation	360	360	3955	4.39
Auxiliary Particle Filter	360	720	5885	6.53
DEMC Particle Filter	360	1080	2925	3.25
Genetic Condensation	360	1440	2630	2.92



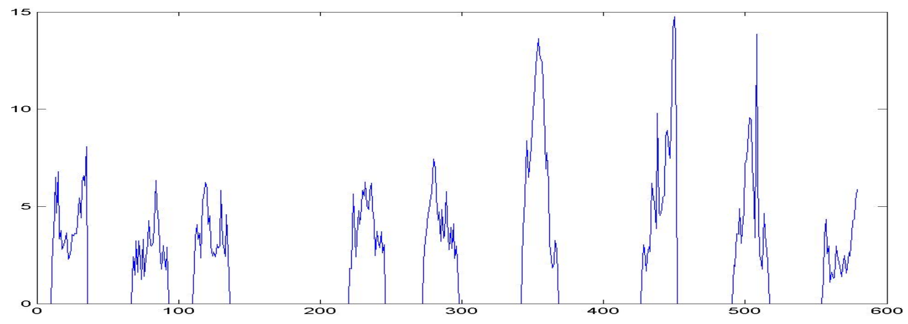
(a)



(b)

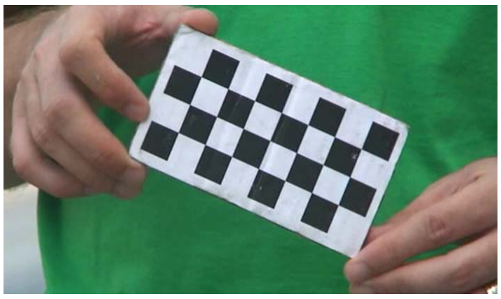


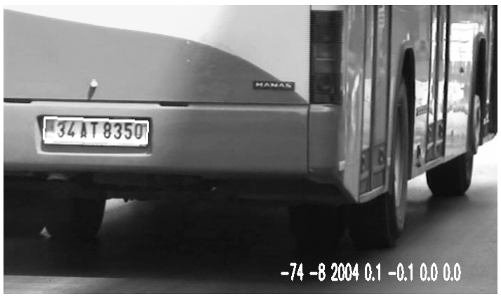
(c)



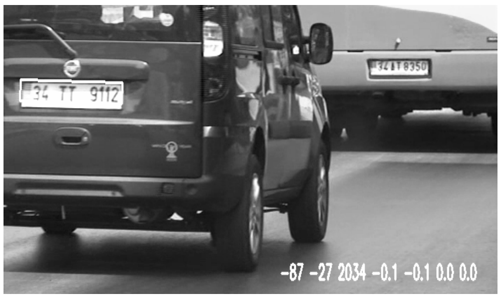
(d)

Figure 9.8 : Test Video Sequence 3 Corner Displacement Errors with Same Computation Load (a) Condensation (b) Auxiliary Condensation (c) DEMC Particle Filter (d) Genetic Condensation

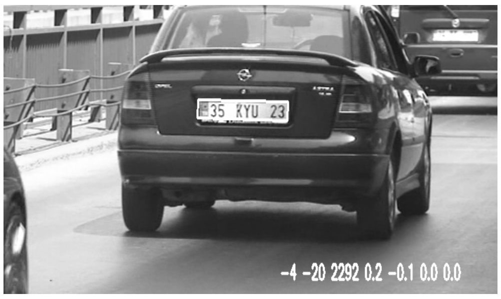

(a) Calibration Image



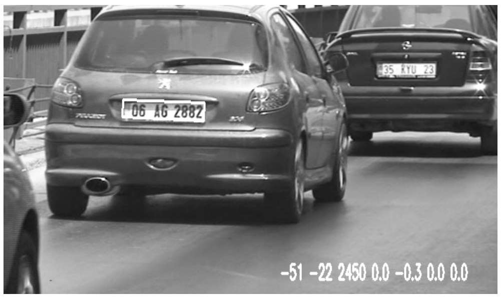
(b) Video 4 Frame 16



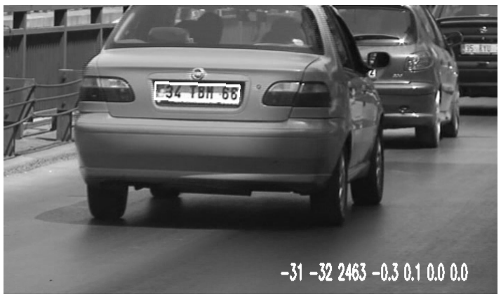
(c) Video 4 Frame 72



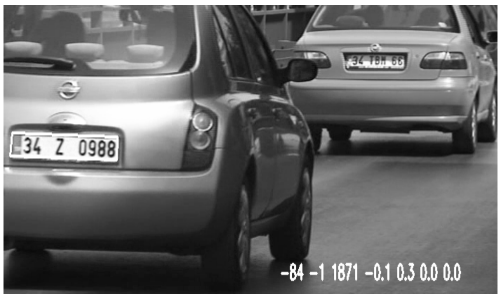
(d) Video 4 Frame 160



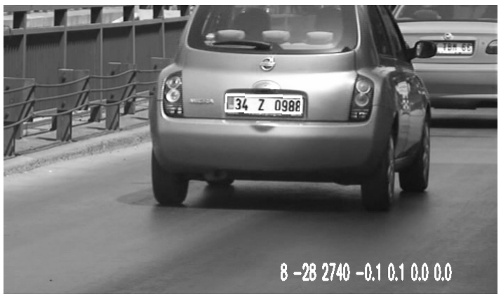
(e) Video 4 Frame 204



(f) Video 4 Frame 248



(g) Video 4 Frame 284



(h) Video 4 Frame 308

Figure 9.9 : Test Video Sequence 4 (a) Calibration Image (b) Frame 16 (c) Frame 72 (d) Frame 132 (e) Frame 160 (f) Frame 204 (g) Frame 284 (h) Frame 308

Video Sequence : 4

Number of Vehicles : 6

Number of Tracked Frames : 150

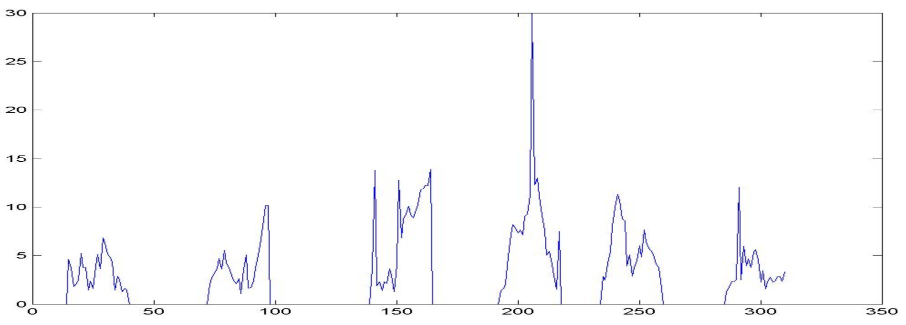
Process Noise Standard Deviation σ in equation 6.1: [10,5,30,0.02,0.02,0.01]

Table 9.9: Test Video 4 Tracking Results with Same Computation Load

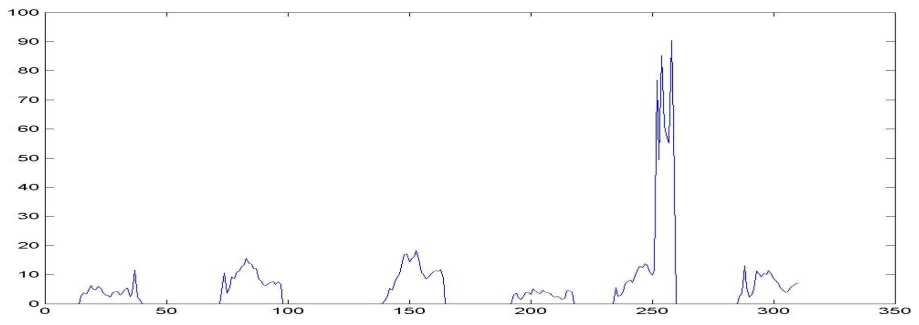
Algorithm	Number of Particles	Number of Likelihood Calculations	Total Corner Displacement Error	Average Displacement Error per Corner
Condensation	360	360	3204	5.34
Auxiliary Particle Filter	180	360	6207	10.34
DEMC Particle Filter	120	360	2180	3.63
Genetic Condensation	90	360	3043	5.07

Table 9.10: Test Video 4 Tracking Results with Same Number of Particles

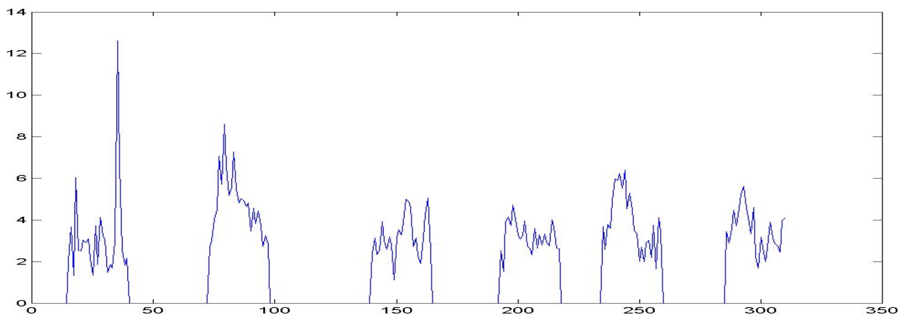
Algorithm	Number of Particles	Number of Likelihood Calculations	Total Corner Displacement Error	Average Displacement Error per Corner
Condensation	360	360	3204	5.34
Auxiliary Particle Filter	360	720	3027	5.04
DEMC Particle Filter	360	1080	1639	2.73
Genetic Condensation	360	1440	1540	2.56



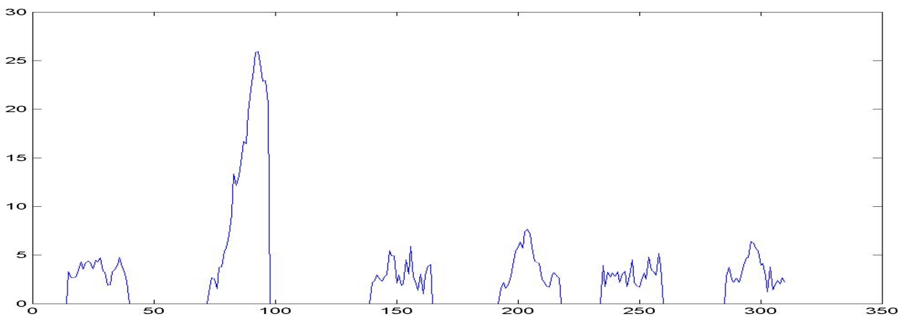
(a)



(b)

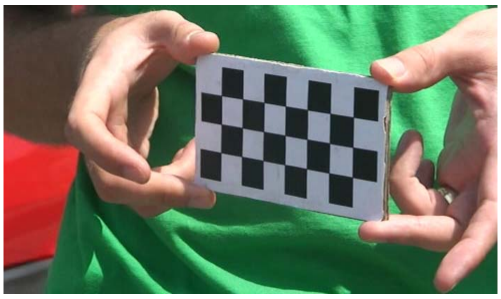


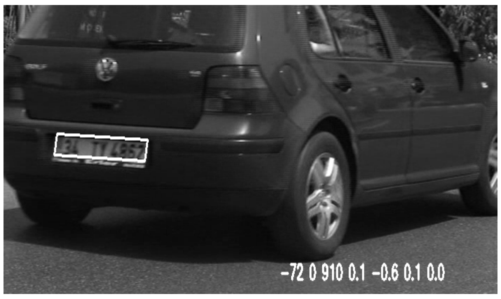
(c)



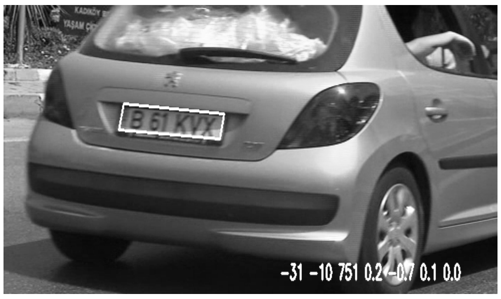
(d)

Figure 9.10 : Test Video Sequence 4 Corner Displacement Errors with Same Computation Load (a) Condensation (b) Auxiliary Condensation (c) DEMC Particle Filter (d) Genetic Condensation

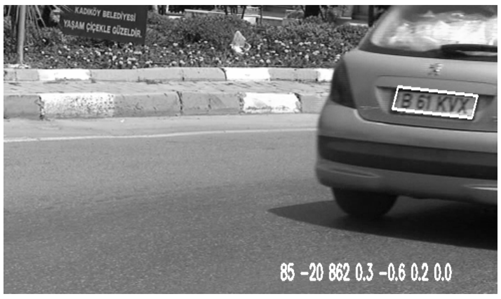

(a) Calibration Image



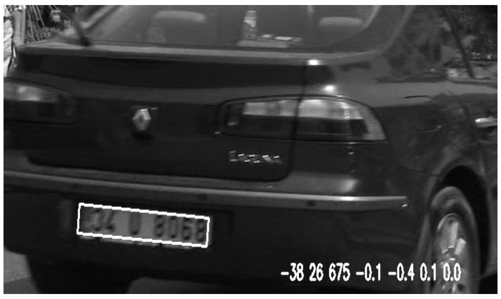
(b) Video 5 Frame 24



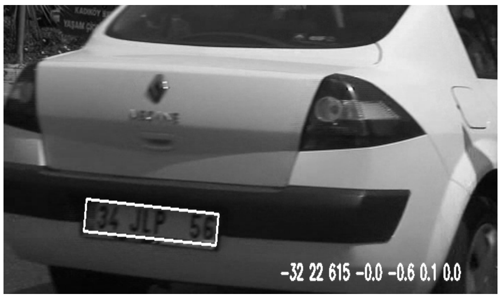
(c) Video 5 Frame 172



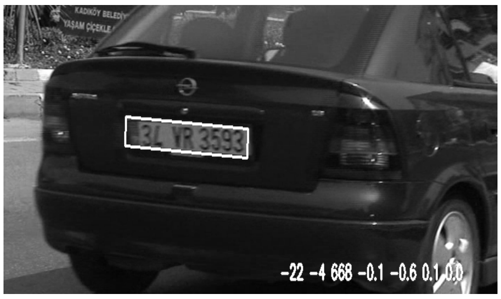
(d) Video 5 Frame 184



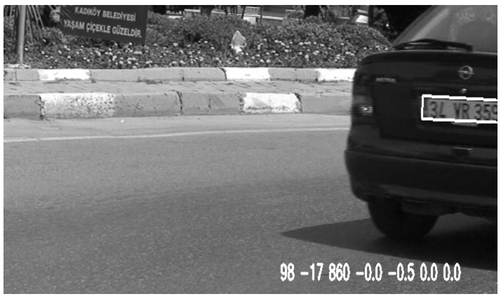
(e) Video 5 Frame 308



(f) Video 5 Frame 348



(g) Video 5 Frame 436



(h) Video 5 Frame 448

Figure 9.11 : Test Video Sequence 5 (a) Calibration Image (b) Frame 24 (c) Frame 172 (d) Frame 184 (e) Frame 308 (f) Frame 348 (g) Frame 436 (h) Frame 448

Video Sequence : 5

Number of Vehicles : 10

Number of Tracked Frames : 129

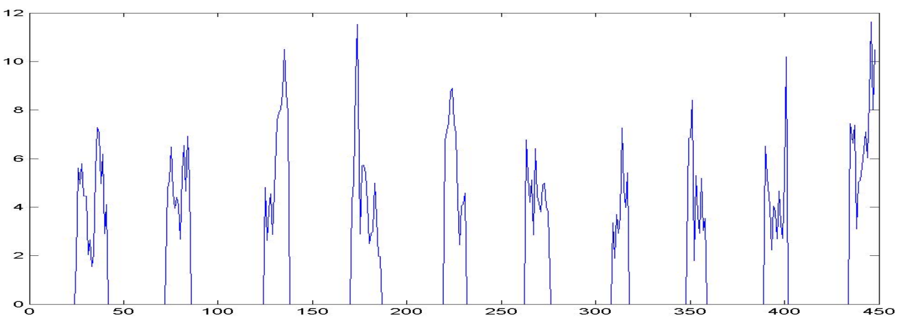
Process Noise Standard Deviation σ in equation 6.1: [10,5,30,0.02,0.02,0.01]

Table 9.11: Test Video 5 Tracking Results with Same Computation Load

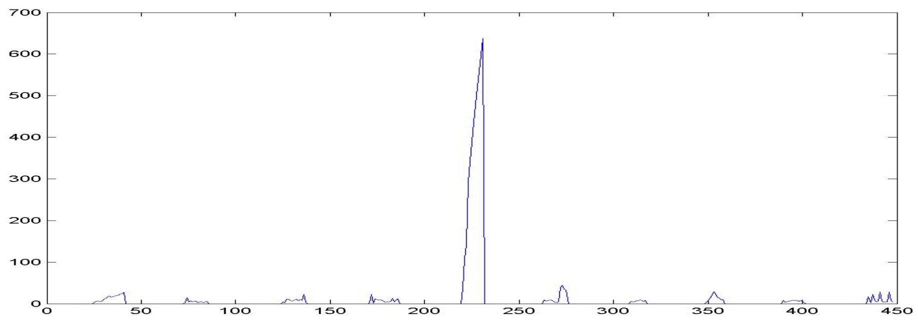
Algorithm	**Number of Particles**	**Number of Likelihood Calculations**	**Total Corner Displacement Error**	**Average Displacement Error per Corner**
Condensation	360	360	2668	5.17
Auxiliary Particle Filter	180	360	22942	44.46
DEMC Particle Filter	120	360	2503	4.85
Genetic Condensation	90	360	3209	6.21

Table 9.12: Test Video 5 Tracking Results with Same Number of Particles

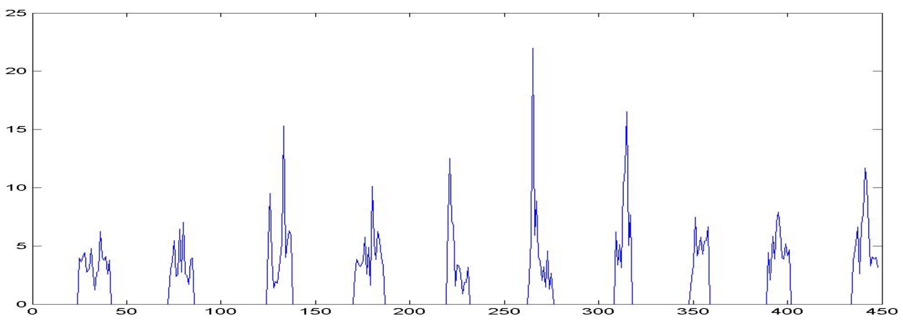
Algorithm	**Number of Particles**	**Number of Likelihood Calculations**	**Total Corner Displacement Error**	**Average Displacement Error per Corner**
Condensation	360	360	2668	5.17
Auxiliary Particle Filter	360	720	2438	4.72
DEMC Particle Filter	360	1080	1844	3.57
Genetic Condensation	360	1440	1649	3.19



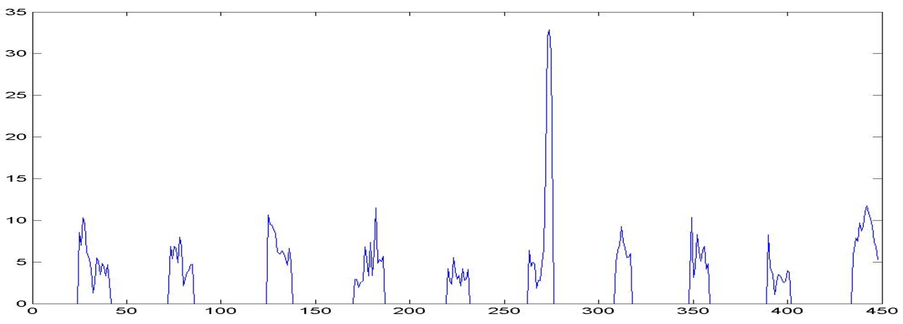
(a)



(b)

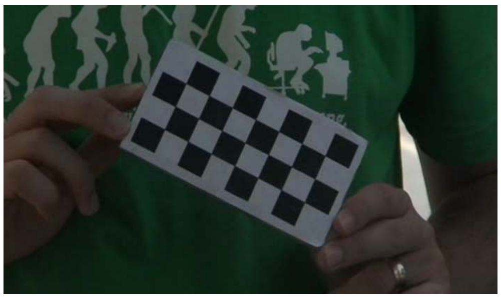


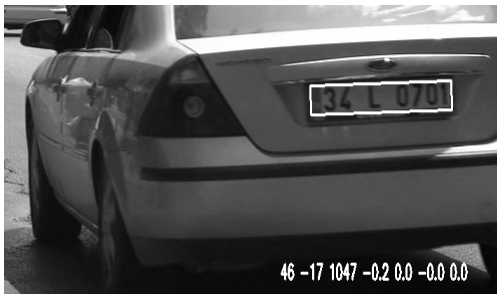
(c)



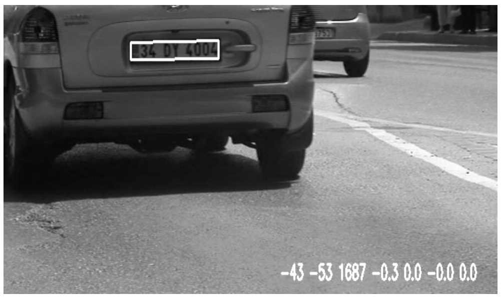
(d)

Figure 9.12 : Test Video Sequence 5 Corner Displacement Errors with Same Computation Load (a) Condensation (b) Auxiliary Condensation (c) DEMC Particle Filter (d) Genetic Condensation

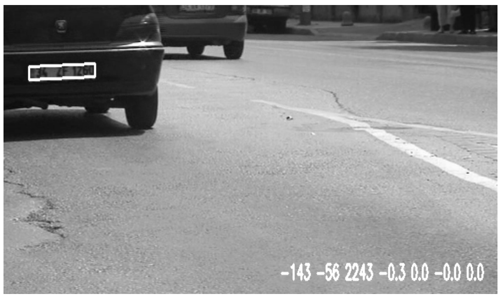

(a) Calibration Image



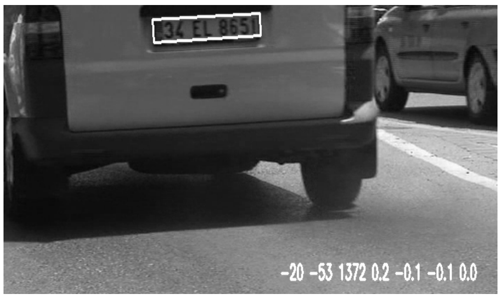
(b) Video 6 Frame 20



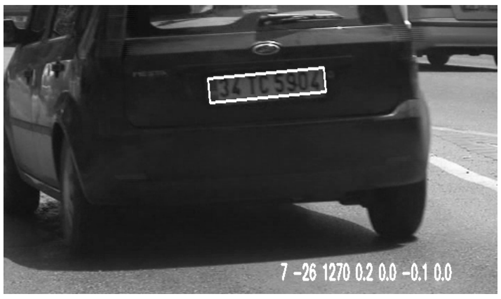
(c) Video 6 Frame 80



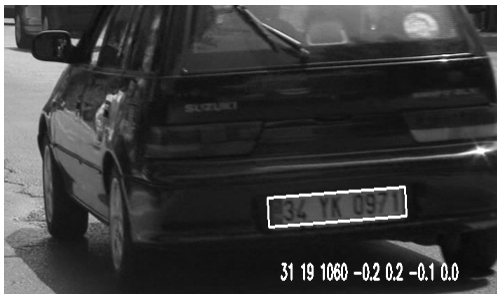
(d) Video 6 Frame 160



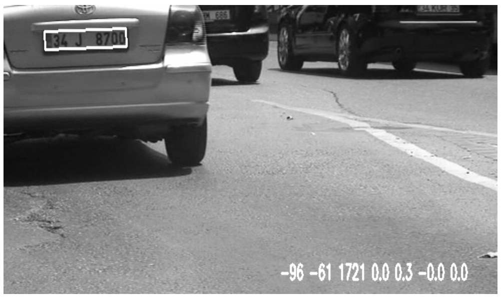
(e) Video 6 Frame 216



(f) Video 6 Frame 272



(g) Video 6 Frame 376



(h) Video 6 Frame 928

Figure 9.13 : Test Video Sequence 6 (a) Calibration Image (b) Frame 20 (c) Frame 80 (d) Frame 160 (e) Frame 216 (f) Frame 272 (g) Frame 376 (h) Frame 928

Video Sequence : 6

Number of Vehicles : 15

Number of Tracked Frames : 376

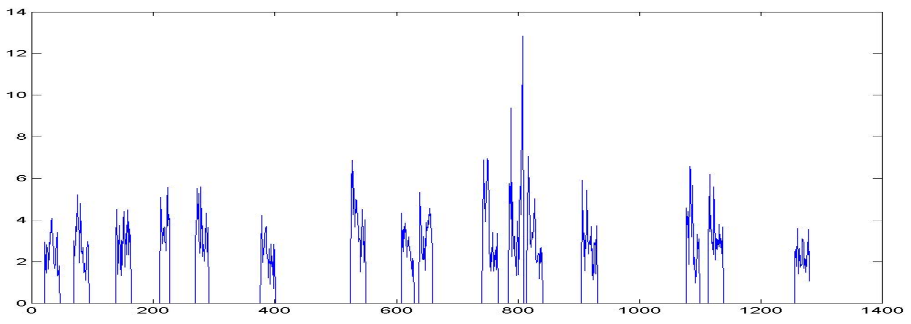
Process Noise Standard Deviation σ in equation 6.1: [10,5,30,0.02,0.02,0.01]

Table 9.13: Test Video 6 Tracking Results with Same Computation Load

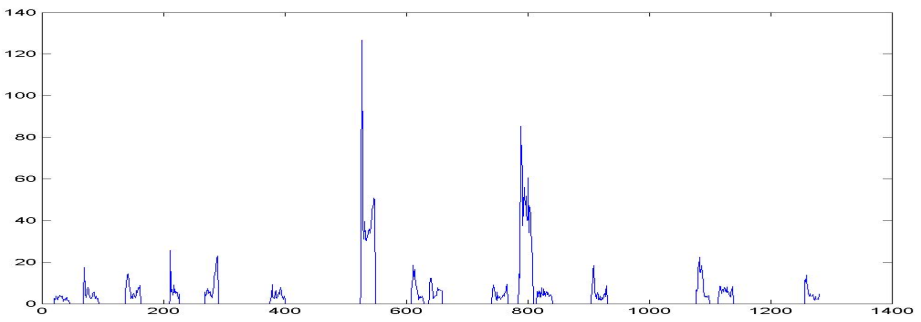
Algorithm	Number of Particles	Number of Likelihood Calculations	Total Corner Displacement Error	Average Displacement Error per Corner
Condensation	360	360	4829	3.21
Auxiliary Particle Filter	180	360	15923	10.58
DEMC Particle Filter	120	360	5636	3.74
Genetic Condensation	90	360	5088	3.38

Table 9.14: Test Video 6 Tracking Results with Same Number of Particles

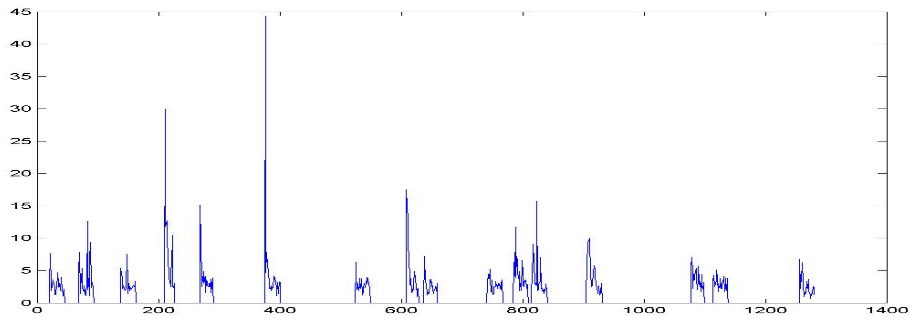
Algorithm	Number of Particles	Number of Likelihood Calculations	Total Corner Displacement Error	Average Displacement Error per Corner
Condensation	360	360	4829	3.21
Auxiliary Particle Filter	360	720	8666	5.76
DEMC Particle Filter	360	1080	4001	2.66
Genetic Condensation	360	1440	3540	2.35



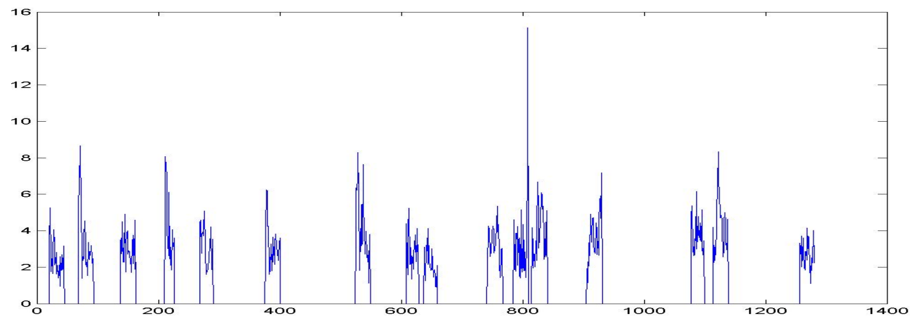
(a)



(b)



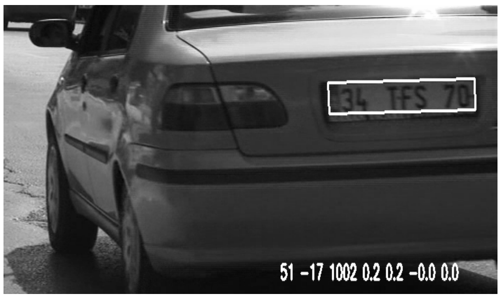
(c)



(d)

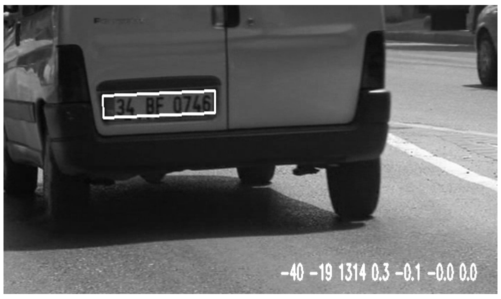
Figure 9.14 : Test Video Sequence 6 Corner Displacement Errors with Same Computation Load (a) Condensation (b) Auxiliary Condensation (c) DEMC Particle Filter (d) Genetic Condensation

(a) Calibration Image



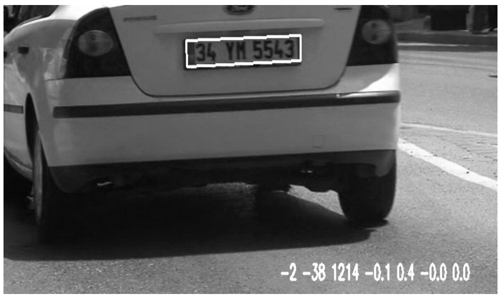
```
51 -17 1002 0.2 0.2 -0.0 0.0
```

(b) Video 7 Frame 8



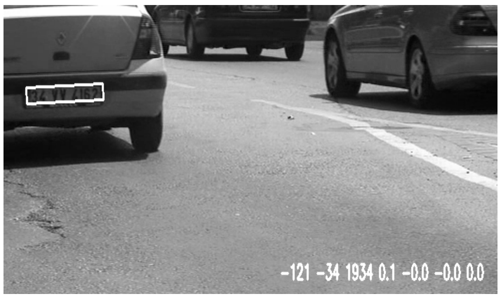
```
-40 -19 1314 0.3 -0.1 -0.0 0.0
```

(c) Video 7 Frame 52



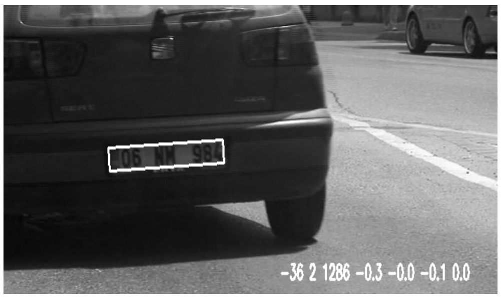
```
-2 -38 1214 -0.1 0.4 -0.0 0.0
```

(d) Video 7 Frame 92



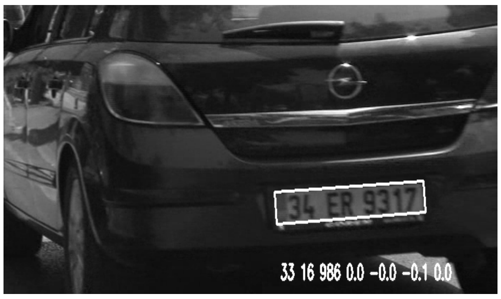
```
-121 -34 1934 0.1 -0.0 -0.0 0.0
```

(e) Video 7 Frame 164



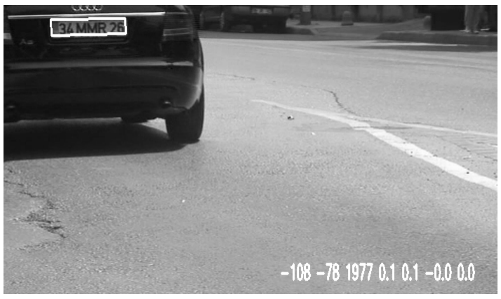
```
-36 2 1286 -0.3 -0.0 -0.1 0.0
```

(f) Video 7 Frame 496



```
33 16 986 0.0 -0.0 -0.1 0.0
```

(g) Video 7 Frame 528



```
-108 -78 1977 0.1 0.1 -0.0 0.0
```

(h) Video 7 Frame 640

Figure 9.15 : Test Video Sequence 7 (a) Calibration Image (b) Frame 8 (c) Frame 52 (d) Frame 92 (e) Frame 164 (f) Frame 496 (g) Frame 528 (h) Frame 640

Video Sequence : 7

Number of Vehicles : 11

Number of Tracked Frames : 241

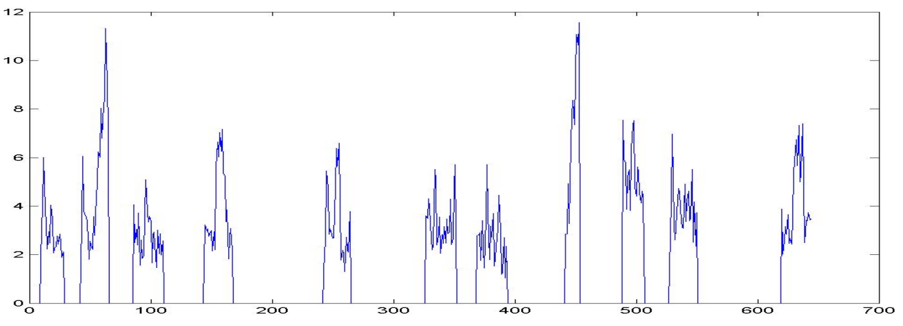
Process Noise Standard Deviation σ in equation 6.1: [10,5,30,0.02,0.02,0.01]

Table 9.15: Test Video 7 Tracking Results with Same Computation Load

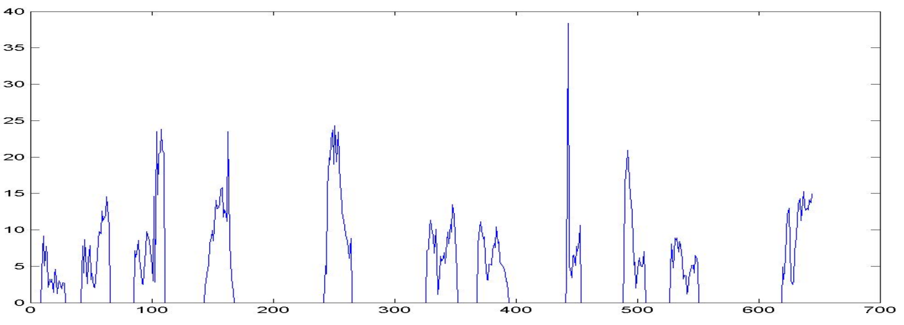
Algorithm	**Number of Particles**	**Number of Likelihood Calculations**	**Total Corner Displacement Error**	**Average Displacement Error per Corner**
Condensation	360	360	3758	3.89
Auxiliary Particle Filter	180	360	8412	8.72
DEMC Particle Filter	120	360	4319	4.48
Genetic Condensation	90	360	4165	4.32

Table 9.16: Test Video 7 Tracking Results with Same Number of Particles

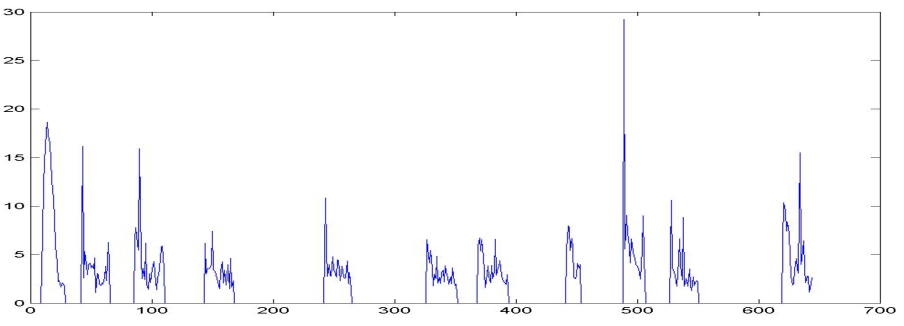
Algorithm	**Number of Particles**	**Number of Likelihood Calculations**	**Total Corner Displacement Error**	**Average Displacement Error per Corner**
Condensation	360	360	3758	3.89
Auxiliary Particle Filter	360	720	3566	3.69
DEMC Particle Filter	360	1080	3030	3.14
Genetic Condensation	360	1440	2481	2.57



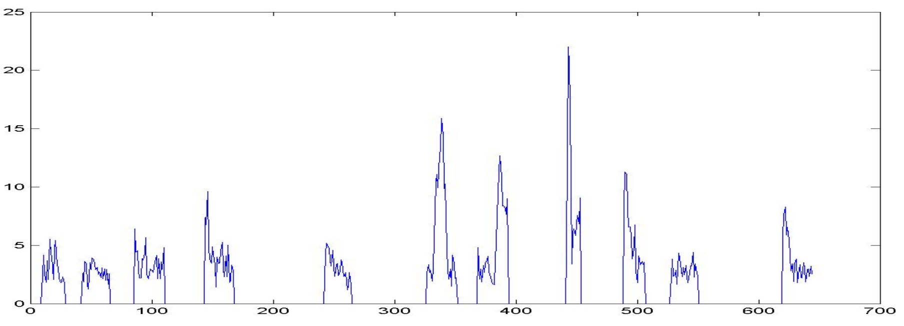
(a)



(b)

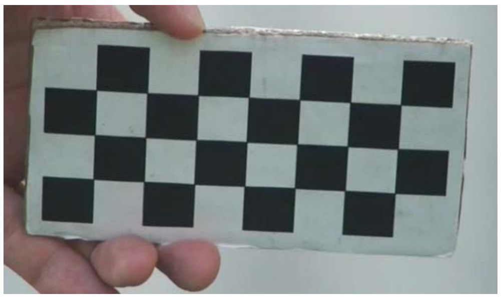


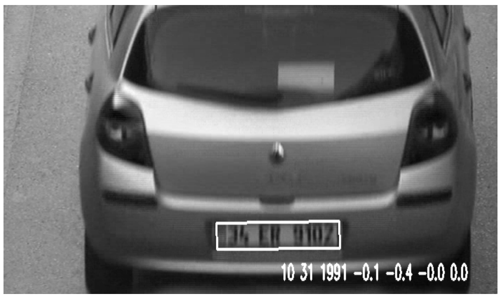
(c)



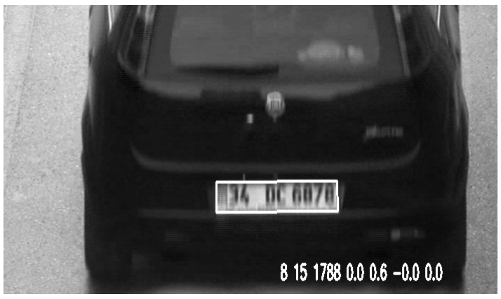
(d)

Figure 9.16 : Test Video Sequence 7 Corner Displacement Errors with Same Computation Load (a) Condensation (b) Auxiliary Condensation (c) DEMC Particle Filter (d) Genetic Condensation

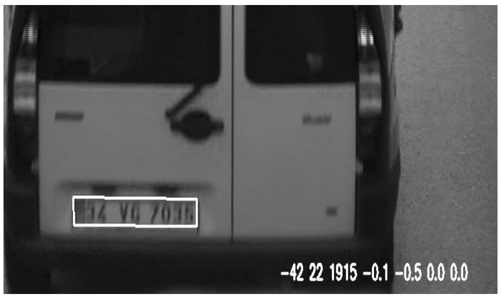

(a) Calibration Image



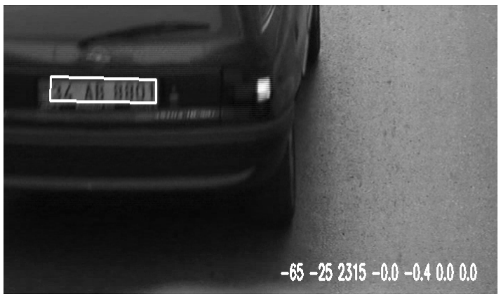
(b) Video 8 Frame 8



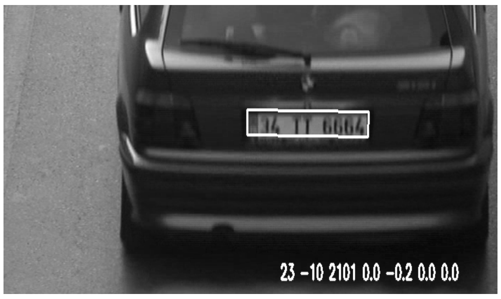
(c) Video 8 Frame 64



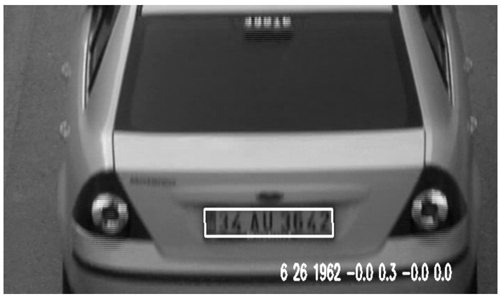
(d) Video 8 Frame 104



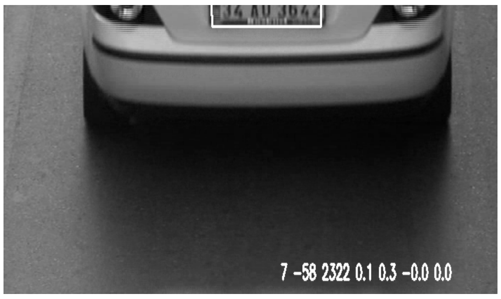
(e) Video 8 Frame 160



(f) Video 8 Frame 196



(g) Video 8 Frame 236



(h) Video 8 Frame 244

Figure 9.17 : Test Video Sequence 8 (a) Calibration Image (b) Frame 8 (c) Frame 64 (d) Frame 104 (e) Frame 160 (f) Frame 196 (g) Frame 236 (h) Frame 244

Video Sequence : 8

Number of Vehicles : 6

Number of Tracked Frames : 45

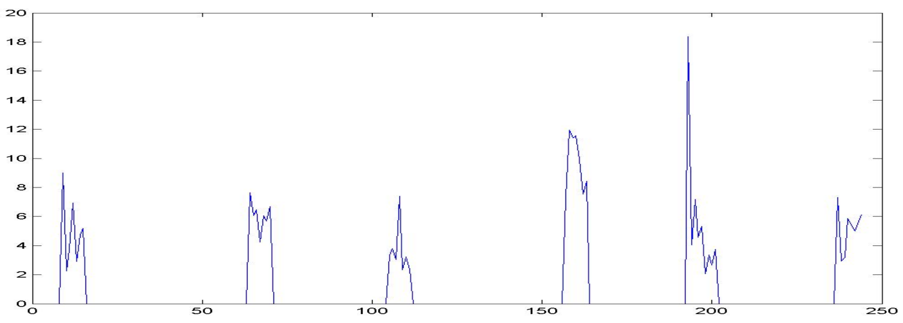
Process Noise Standard Deviation σ in equation 6.1: [5,10,30,0.02,0.02,0.01]

Table 9.17: Test Video 8 Tracking Results with Same Computation Load

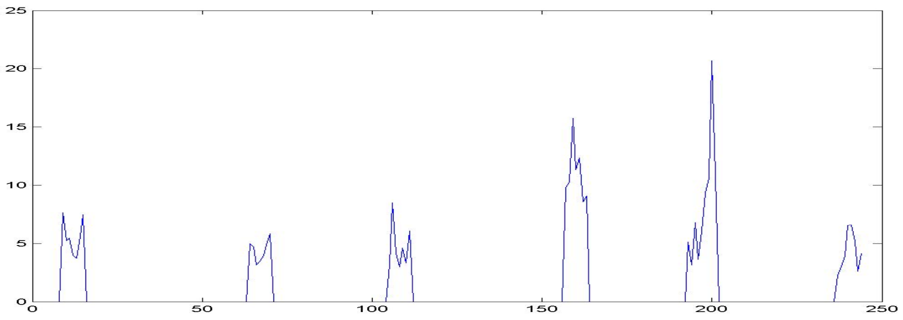
Algorithm	**Number of Particles**	**Number of Likelihood Calculations**	**Total Corner Displacement Error**	**Average Displacement Error per Corner**
Condensation	360	360	1055	5.86
Auxiliary Particle Filter	180	360	1162	6.45
DEMC Particle Filter	120	360	885	4.91
Genetic Condensation	90	360	789	4.38

Table 9.18: Test Video 8 Tracking Results with Same Number of Particles

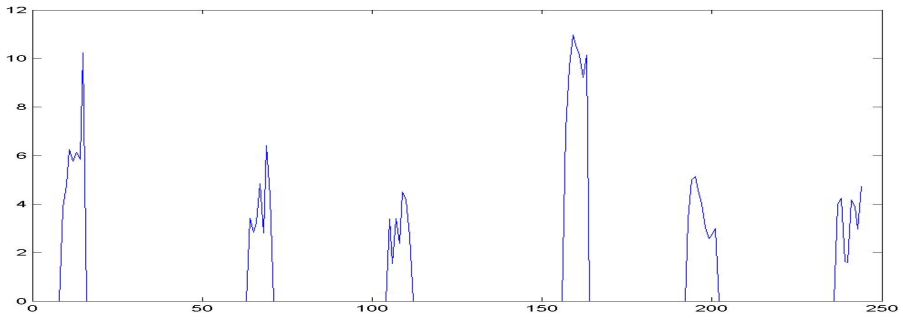
Algorithm	**Number of Particles**	**Number of Likelihood Calculations**	**Total Corner Displacement Error**	**Average Displacement Error per Corner**
Condensation	360	360	1055	5.86
Auxiliary Particle Filter	360	720	843	4.68
DEMC Particle Filter	360	1080	826	4.58
Genetic Condensation	360	1440	693	3.85



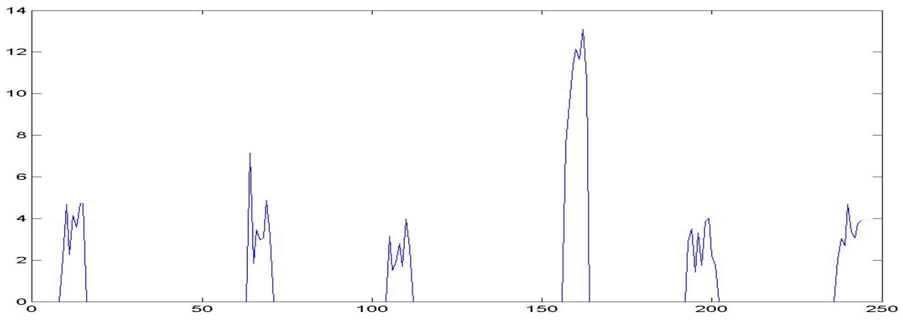
(a)



(b)

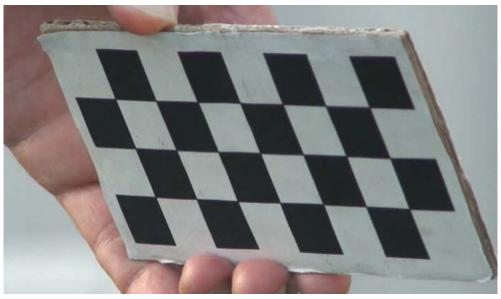


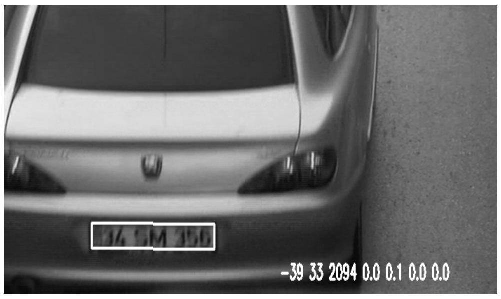
(c)



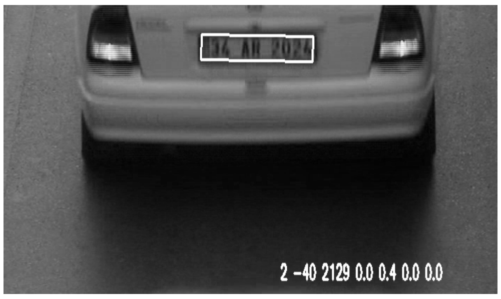
(d)

Figure 9.18 : Test Video Sequence 8 Corner Displacement Errors with Same Computation Load (a) Condensation (b) Auxiliary Condensation (c) DEMC Particle Filter (d) Genetic Condensation

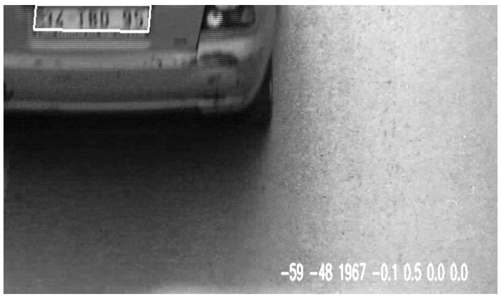

(a) Calibration Image



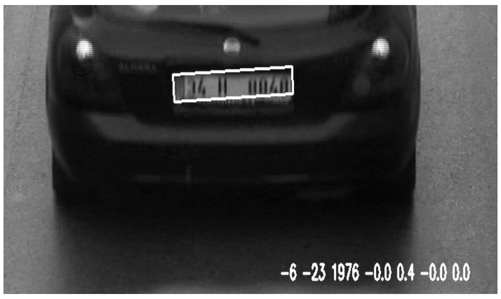
(b) Video 9 Frame 16



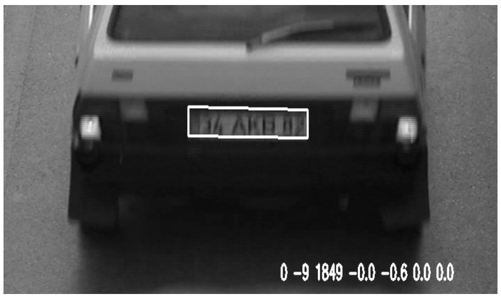
(c) Video 9 Frame 48



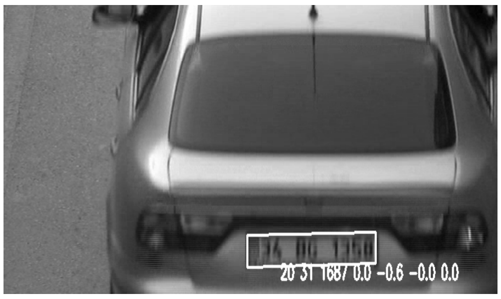
(d) Video 9 Frame 88



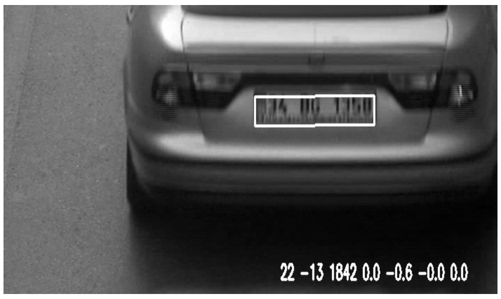
(e) Video 9 Frame 160



(f) Video 9 Frame 256



(g) Video 9 Frame 432



(h) Video 9 Frame 436

Figure 9.19 : Test Video Sequence 9 (a) Calibration Image (b) Frame 16 (c) Frame 48 (d) Frame 88 (e) Frame 160 (f) Frame 256 (g) Frame 432 (h) Frame 436

Video Sequence : 9

Number of Vehicles : 7

Number of Tracked Frames : 41

Process Noise Standard Deviation σ in equation 6.1: [5,10,30,0.02,0.02,0.01]

Table 9.19: Test Video 9 Tracking Results with Same Computation Load

Algorithm	Number of Particles	Number of Likelihood Calculations	Total Corner Displacement Error	Average Displacement Error per Corner
Condensation	360	360	1178	7.18
Auxiliary Particle Filter	180	360	1285	7.83
DEMC Particle Filter	120	360	709	4.32
Genetic Condensation	90	360	612	3.73

Table 9.20: Test Video 9 Tracking Results with Same Number of Particles

Algorithm	Number of Particles	Number of Likelihood Calculations	Total Corner Displacement Error	Average Displacement Error per Corner
Condensation	360	360	1178	7.18
Auxiliary Particle Filter	360	720	907	5.53
DEMC Particle Filter	360	1080	760	4.63
Genetic Condensation	360	1440	539	3.28

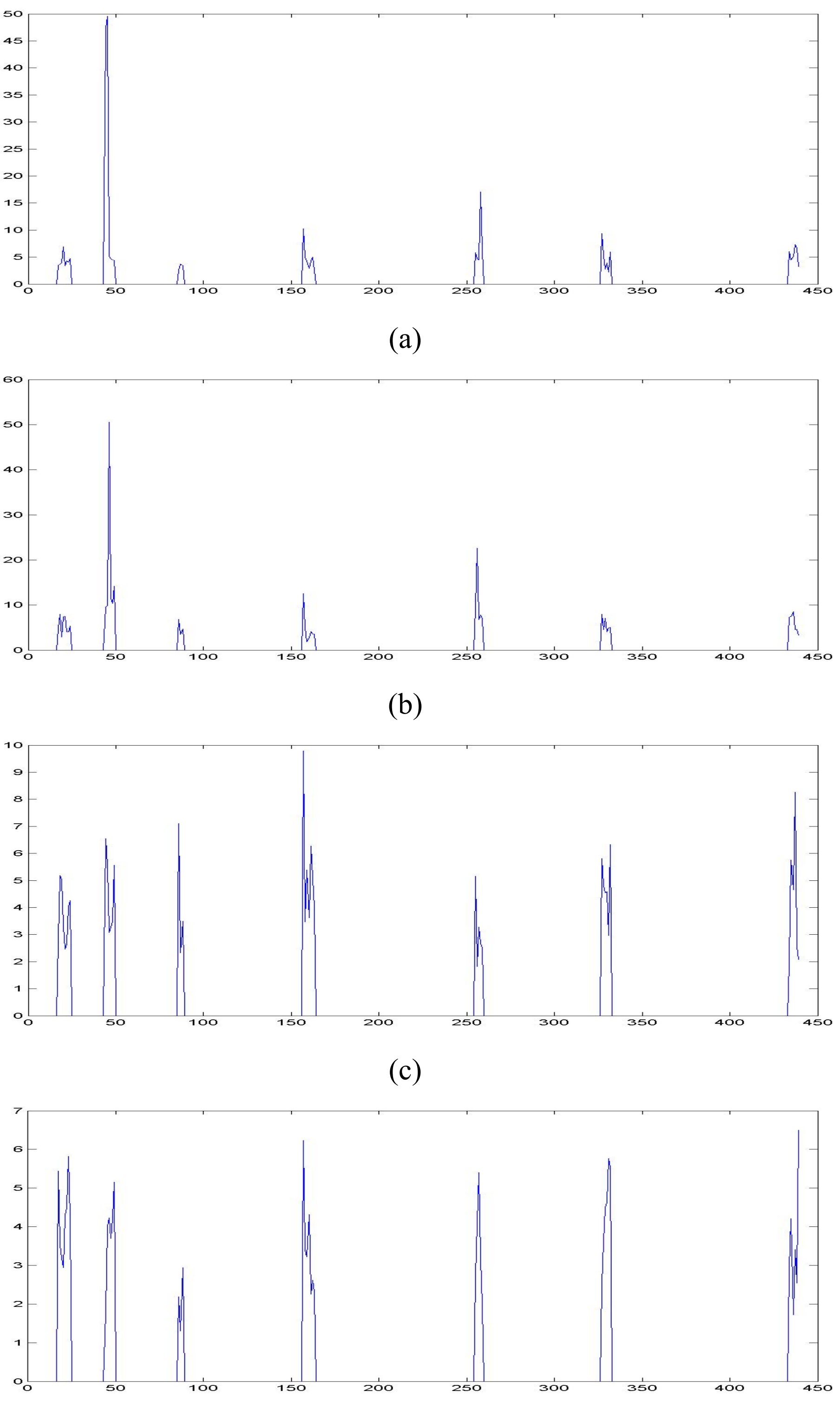
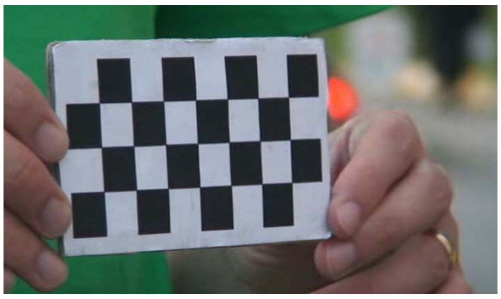
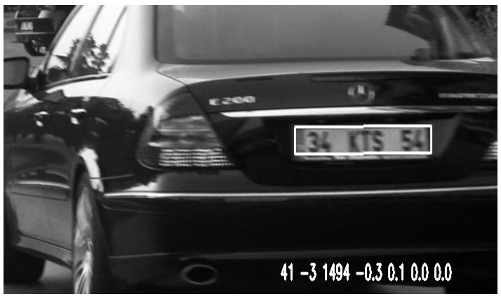
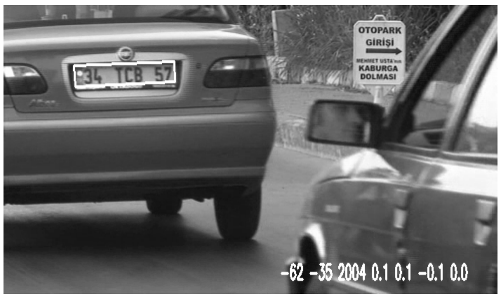


Figure 9.20 : Test Video Sequence 9 Corner Displacement Errors with Same Computation Load (a) Condensation (b) Auxiliary Condensation (c) DEMC Particle Filter (d) Genetic Condensation

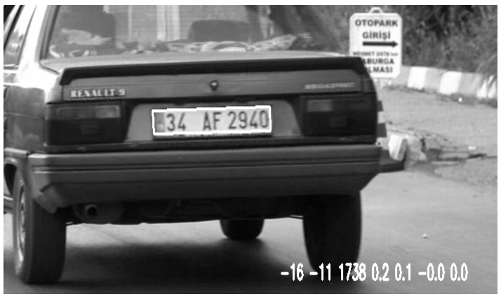

(a) Calibration Image



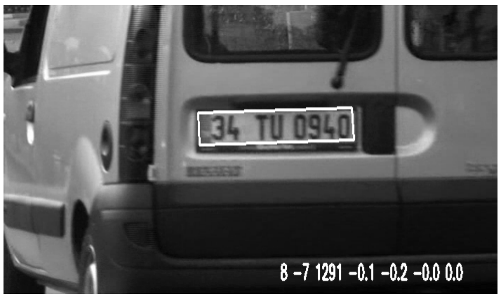
(b) Video 10 Frame 20



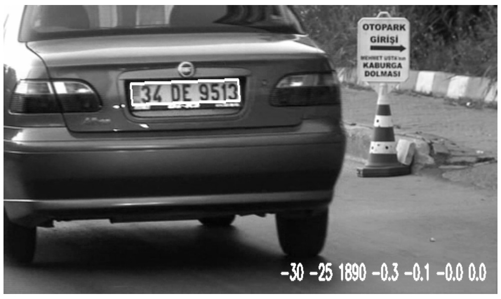
(c) Video 10 Frame 88



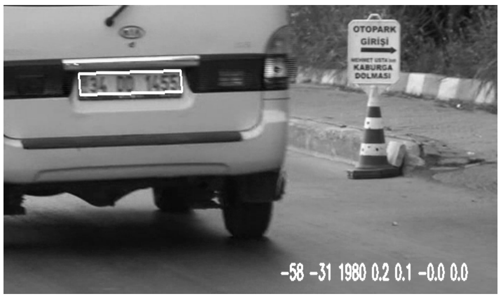
(d) Video 10 Frame 120



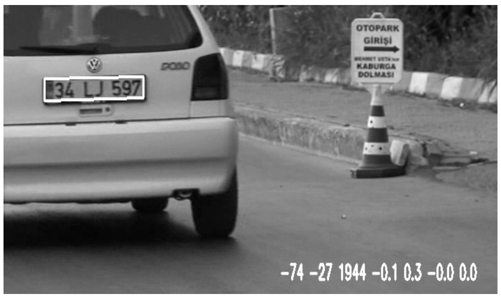
(e) Video 10 Frame 176



(f) Video 10 Frame 244



(g) Video 10 Frame 276



(h) Video 10 Frame 520

Figure 9.21 : Test Video Sequence 10 (a) Calibration Image (b) Frame 20 (c) Frame 88 (d) Frame 120 (e) Frame 176 (f) Frame 244 (g) Frame 276 (h) Frame 520

Video Sequence : 10

Number of Vehicles : 9

Number of Tracked Frames : 205

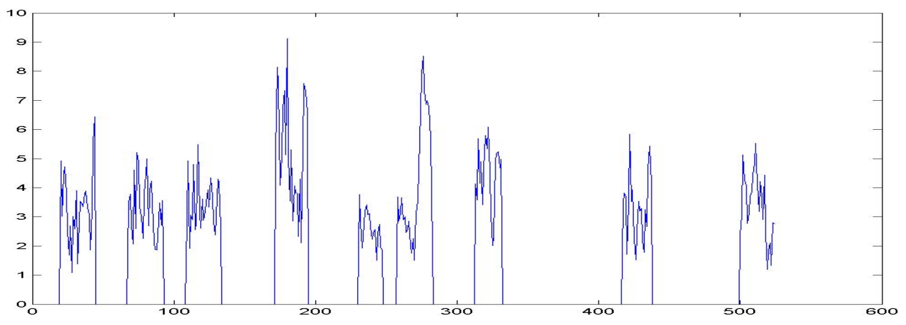
Process Noise Standard Deviation σ in equation 6.1: [10,5,30,0.02,0.02,0.01]

Table 9.21: Test Video 10 Tracking Results with Same Computation Load

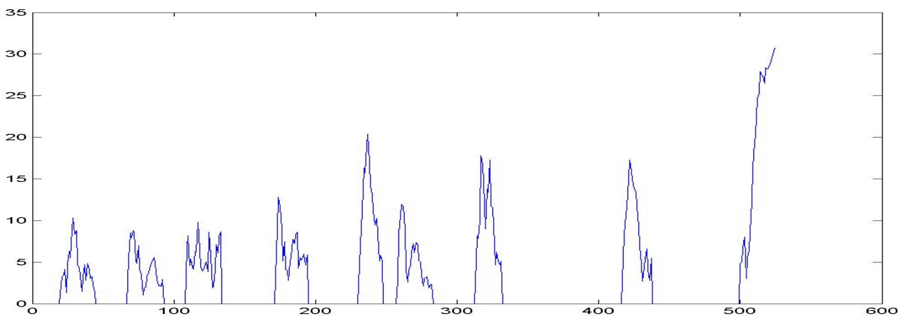
Algorithm	**Number of Particles**	**Number of Likelihood Calculations**	**Total Corner Displacement Error**	**Average Displacement Error per Corner**
Condensation	360	360	3060	3.73
Auxiliary Particle Filter	180	360	6808	8.30
DEMC Particle Filter	120	360	2868	3.49
Genetic Condensation	90	360	3111	3.79

Table 9.22: Test Video 10 Tracking Results with Same Number of Particles

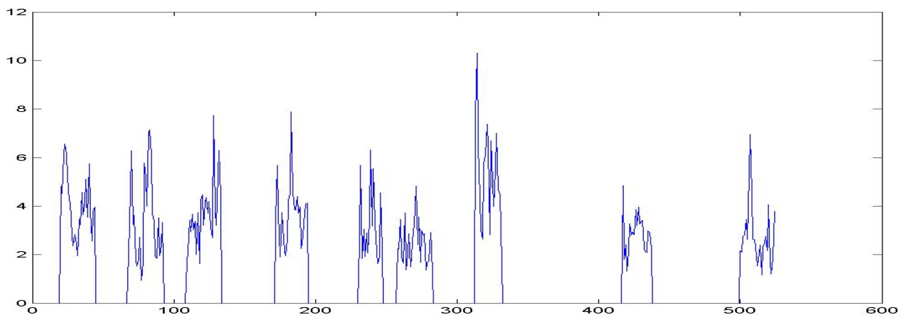
Algorithm	**Number of Particles**	**Number of Likelihood Calculations**	**Total Corner Displacement Error**	**Average Displacement Error per Corner**
Condensation	360	360	3060	3.73
Auxiliary Particle Filter	360	720	3111	3.79
DEMC Particle Filter	360	1080	2377	2.89
Genetic Condensation	360	1440	1959	2.38



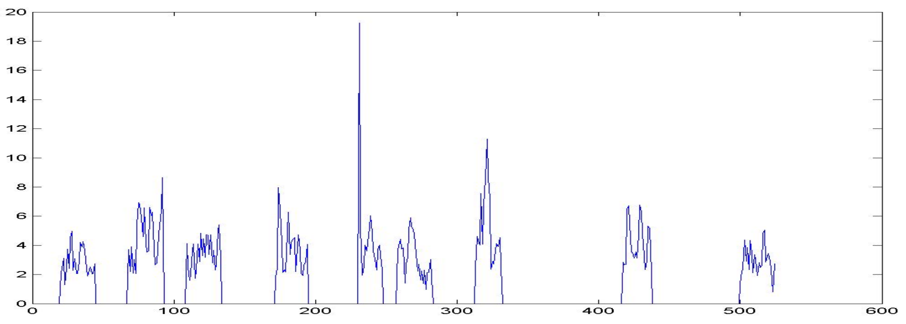
(a)



(b)

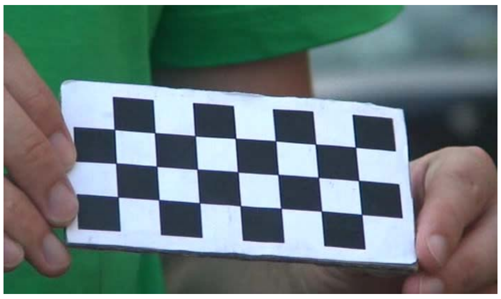


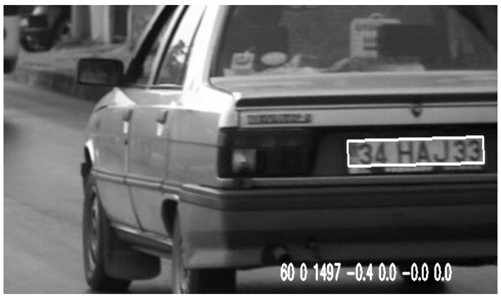
(c)



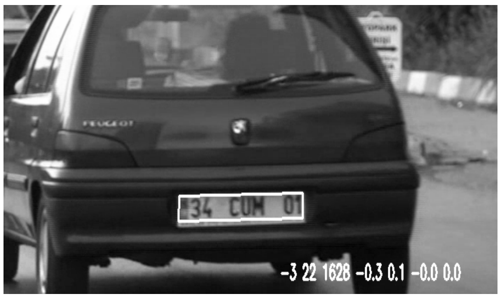
(d)

Figure 9.22 : Test Video Sequence 10 Corner Displacement Errors with Same Computation Load (a) Condensation (b) Auxiliary Condensation (c) DEMC Particle Filter (d) Genetic Condensation

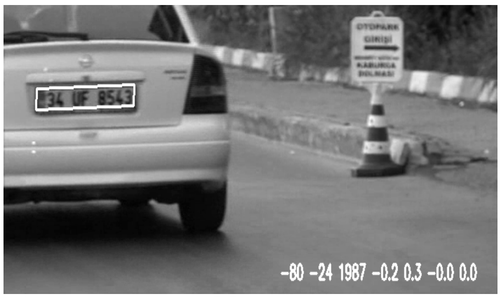

(a) Calibration Image



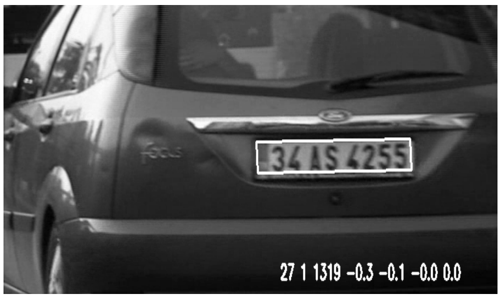
(b) Video 11 Frame 16



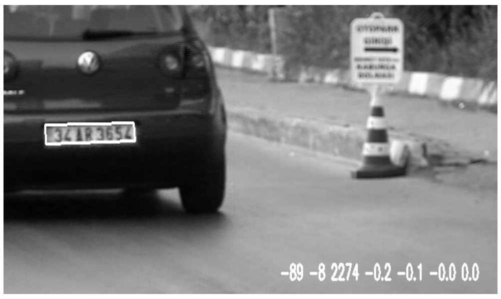
(c) Video 11 Frame 68



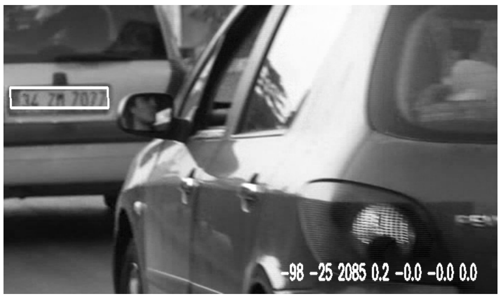
(d) Video 11 Frame 136



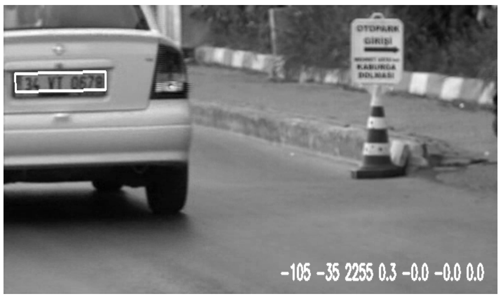
(e) Video 11 Frame 232



(f) Video 11 Frame 340



(g) Video 11 Frame 480



(h) Video 11 Frame 688

Figure 9.23 : Test Video Sequence 11 (a) Calibration Image (b) Frame 16 (c) Frame 68 (d) Frame 136 (e) Frame 232 (f) Frame 340 (g) Frame 480 (h) Frame 688

Video Sequence : 11

Number of Vehicles : 10

Number of Tracked Frames : 238

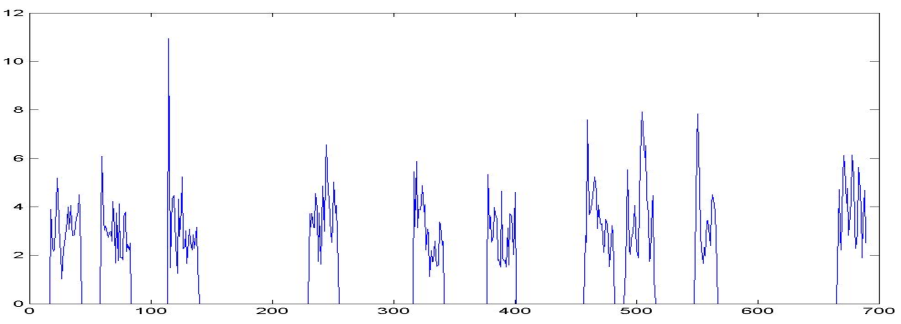
Process Noise Standard Deviation σ in equation 6.1: [10,5,30,0.02,0.02,0.01]

Table 9.23: Test Video 11 Tracking Results with Same Computation Load

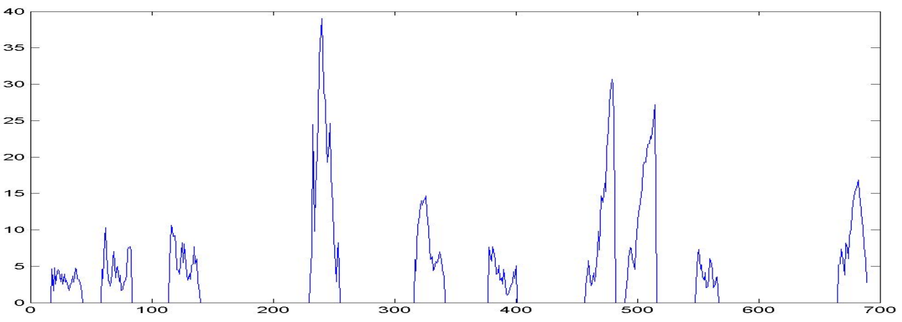
Algorithm	Number of Particles	Number of Likelihood Calculations	Total Corner Displacement Error	Average Displacement Error per Corner
Condensation	360	360	3230	3.39
Auxiliary Particle Filter	180	360	8375	8.79
DEMC Particle Filter	120	360	3100	3.25
Genetic Condensation	90	360	3192	3.35

Table 9.24: Test Video 11 Tracking Results with Same Number of Particles

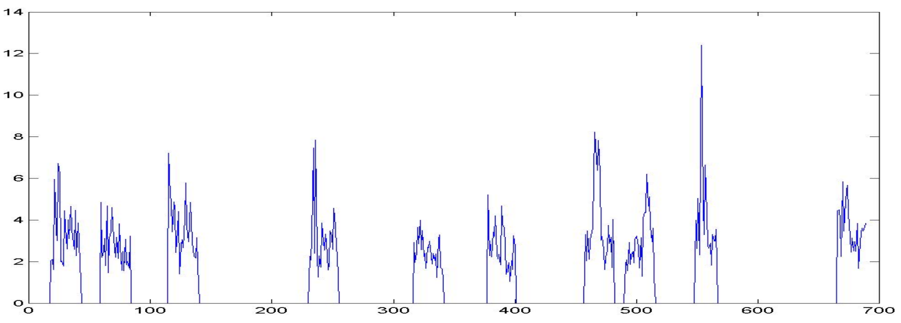
Algorithm	Number of Particles	Number of Likelihood Calculations	Total Corner Displacement Error	Average Displacement Error per Corner
Condensation	360	360	3230	3.39
Auxiliary Particle Filter	360	720	3653	3.83
DEMC Particle Filter	360	1080	2551	2.67
Genetic Condensation	360	1440	2215	2.32



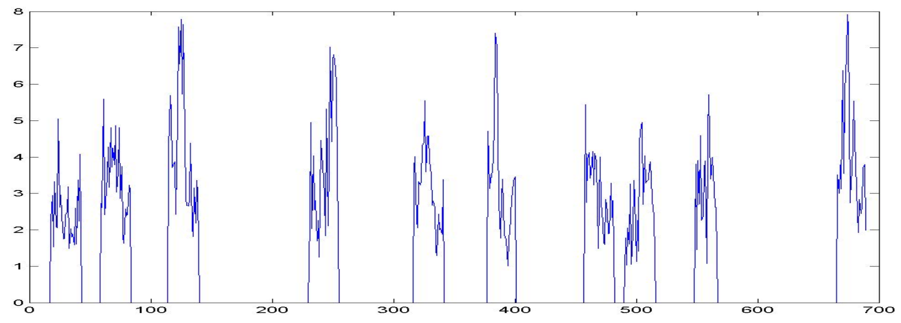
(a)



(b)

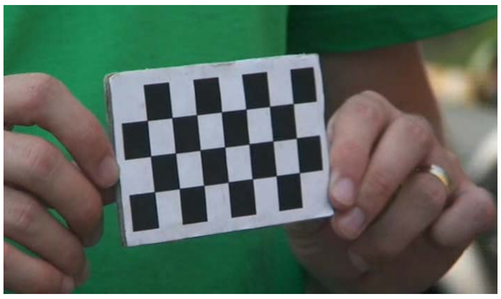


(c)

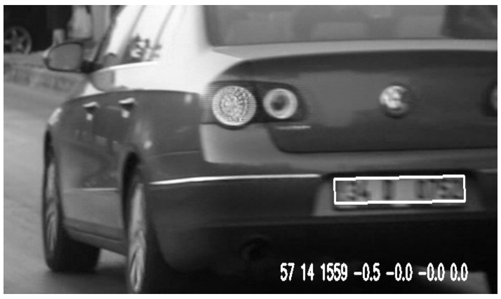


(d)

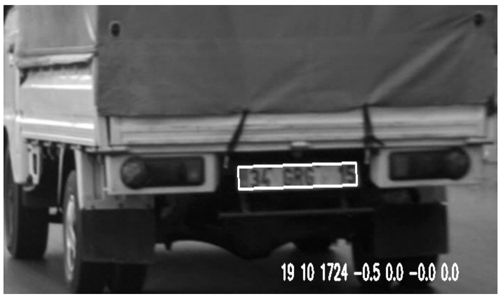
Figure 9.24 : Test Video Sequence 11 Corner Displacement Errors with Same Computation Load (a) Condensation (b) Auxiliary Condensation (c) DEMC Particle Filter (d) Genetic Condensation



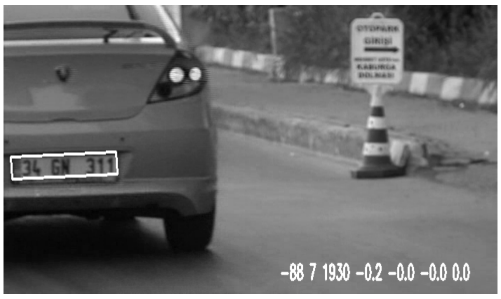
(a) Calibration Image



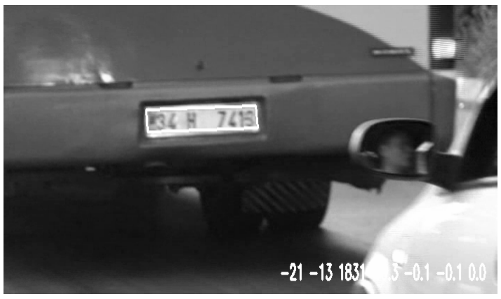
(b) Video 12 Frame 8



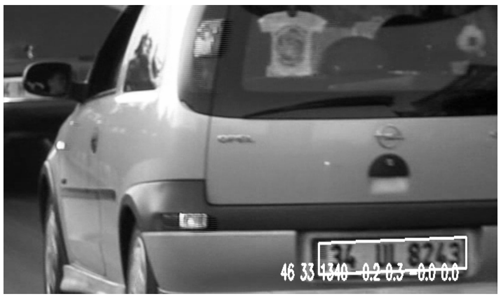
(c) Video 12 Frame 100



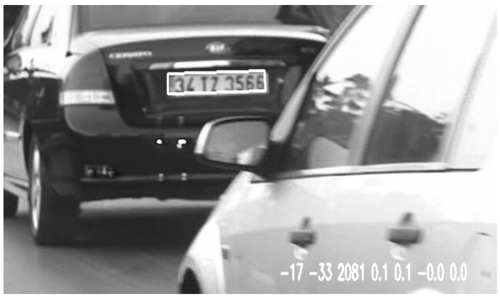
(d) Video 12 Frame 152



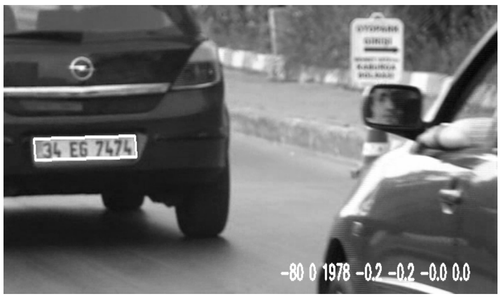
(e) Video 12 Frame 316



(f) Video 12 Frame 336



(g) Video 12 Frame 404



(h) Video 12 Frame 492

Figure 9.25 : Test Video Sequence 12 (a) Calibration Image (b) Frame 8 (c) Frame 100 (d) Frame 152 (e) Frame 316 (f) Frame 336 (g) Frame 404 (h) Frame 492

Video Sequence : 12

Number of Vehicles : 8

Number of Tracked Frames : 170

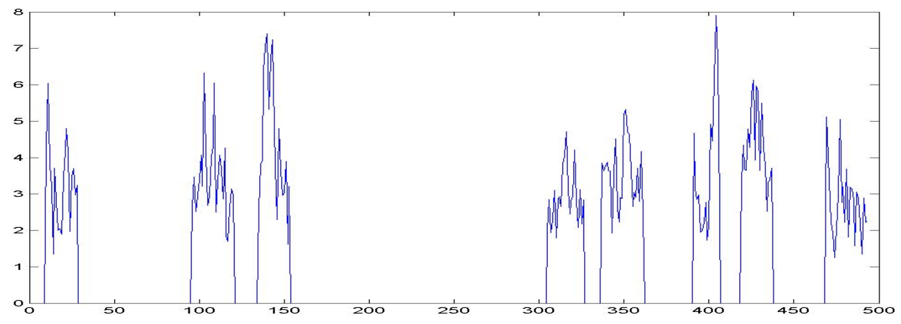
Process Noise Standard Deviation σ in equation 6.1: [10,5,30,0.02,0.02,0.01]

Table 9.25: Test Video 12 Tracking Results with Same Computation Load

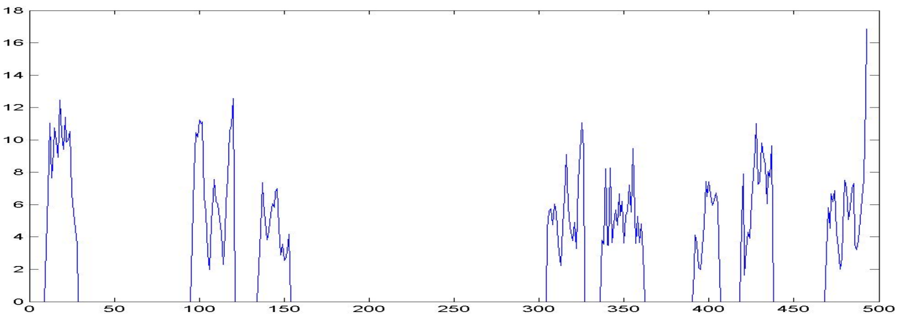
Algorithm	**Number of Particles**	**Number of Likelihood Calculations**	**Total Corner Displacement Error**	**Average Displacement Error per Corner**
Condensation	360	360	2387	3.51
Auxiliary Particle Filter	180	360	4188	6.15
DEMC Particle Filter	120	360	2841	4.17
Genetic Condensation	90	360	2529	3.71

Table 9.26: Test Video 12 Tracking Results with Same Number of Particles

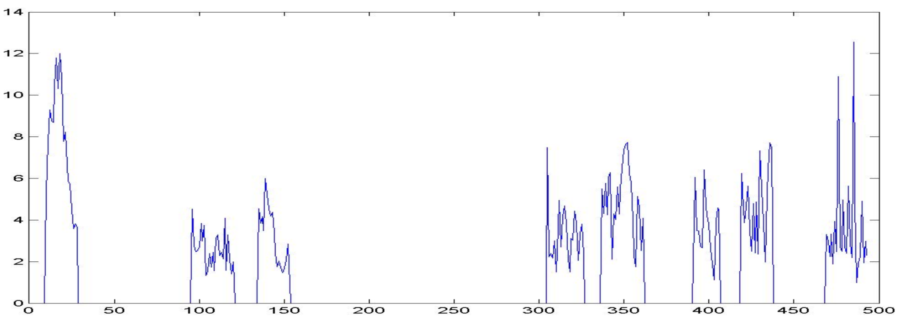
Algorithm	**Number of Particles**	**Number of Likelihood Calculations**	**Total Corner Displacement Error**	**Average Displacement Error per Corner**
Condensation	360	360	2387	3.51
Auxiliary Particle Filter	360	720	2697	3.96
DEMC Particle Filter	360	1080	2115	3.11
Genetic Condensation	360	1440	1771	2.60



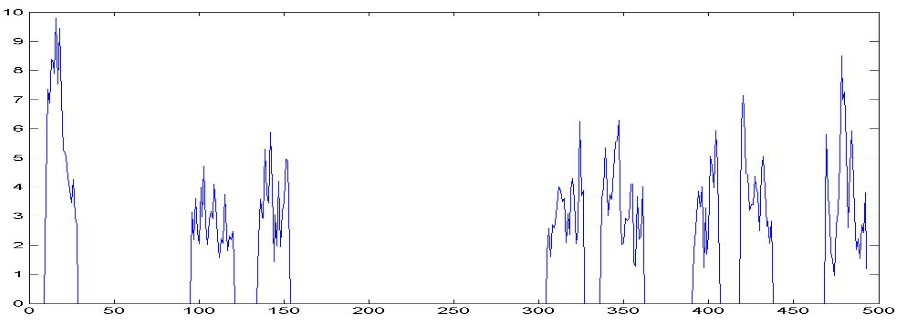
(a)



(b)



(c)



(d)

Figure 9.26 : Test Video Sequence 12 Corner Displacement Errors with Same Computation Load (a) Condensation (b) Auxiliary Condensation (c) DEMC Particle Filter (d) Genetic Condensation

The tracking result for 12 individual test videos is given. The overall score of performance can be gathered in tables as below,

Table 9.27: Overall Tracking Result with Same Computation Load

Algorithm	**Number of Particles**	**Number of Likelihood Calculations**	**Average Displacement Error per Corner (pixels)**
Condensation	360	360	***4.80***
Auxiliary Particle Filter	180	360	***17.13***
DEMC Particle Filter	120	360	***4.05***
Genetic Condensation	90	360	***4.64***

Table 9.28: Overall Tracking Results with Same Number of Particles

Algorithm	**Number of Particles**	**Number of Likelihood Calculations**	**Average Displacement Error per Corner (pixels)**
Condensation	360	360	***4.80***
Auxiliary Particle Filter	360	720	***8.35***
DEMC Particle Filter	360	1080	***3.23***
Genetic Condensation	360	1440	***2.72***

The results show that the with same computation load, DEMC Particle filter performs best among all other filters with 4.05 pixels average corner error. It is very important to observe that the DEMC Particle Filter performs better with the same number of likelihood calculations. On the other hand, Genetic Condensation algorithm also performs better than standard Condensation algorithm with the same computation load. The Auxiliary Particle Filter is reported to be impractical when the likelihood function has low variance [49]. Template matching is such a peaked likelihood function that has steep increase close to the optimum but considerably low values at the neighborhood of the optimum. Auxiliary Particle Filter is the worst filter with 17.13 pixels corner error. Auxiliary Particle Filter requires more particles for being applicable to the license plate tracking problem.

The tests are repeated with the same number of particles. With the same number of particles, DEMC Particle Filter and Generic Condensation algorithm performs significantly better than the standard Condensation algorithm. The best filter in this situation is the Genetic Condensation algorithm with 2.72 pixels error per corner. However, it must always be kept in mind that the total likelihood calculation per time step of the Genetic Condensation is 360 more than the DEMC Particle Filter, 720 more than Auxiliary Particle Filter and 1080 more than Condensation algorithm.

9.4 DEMC Particle Filter Performance with Different Number of Particles

It is also necessary to study the behavior of the DEMC Particle Filter with different number of particles. The performance of the filters gets better with the increasing number of particles, until a satisfying number of particles is reaches. After that point, we cannot get a performance increase, even if we continue to increase the particle number. For each application, this number of particles should be searched for efficiency.

We performed the tracking of the test video 1 with DEMC Particle Filter with different number of particles. During these trials, the parameters of the DEMC Particle Filter are kept constant and the only varying input to the filtering was the number of particles.

We started from 60 and increased the number of particles as 120, 180, 240, 300, 360, 420, as shown in Table 9.29. We obtained a steady performance increase until 360 particles. After that number, increasing the number of particles did not result in performance increase.

We applied 120 as the number of particles in the performance evaluation tests because this is the minimum number of particles that we can sustain a robust tracking performance.

Video Sequence : 1

Number of Vehicles : 10

Number of Tracked Frames : 210

Process Noise Standard Deviation : [10,5,30,0.02,0.02,0.01]

Table 9.29: DEMC P. F. Tracking Results with Different Number of Particles

Algorithm	Number of Particles	Number of Likelihood Calculations	Total Corner Displacement Error	Average Displacement Error per Corner
DEMC Particle Filter	60	180	20509	24.41
DEMC Particle Filter	120	360	3399	4.04
DEMC Particle Filter	180	540	3352	3.99
DEMC Particle Filter	240	720	2874	3.42
DEMC Particle Filter	300	900	2828	3.36
DEMC Particle Filter	360	1080	2438	2.90
DEMC Particle Filter	420	1260	2599	3.09

9.5 DEMC Particle Filter Performance with Different Parameters

The performance of the DEMC particle filter with different parameters is shown in Table 9.30. It is important to calibrate the filter to the application with different parameter set trials. We tested the performance of the algorithm again with the test video sequence 1. Changing the parameter a does not give any increase in the performance. 0.8 is an optimum value for the filter, which is also recommended as 0.85 by the Differential Evolution algorithm itself.

Nevertheless, different values of the parameter *c* can give better results dependent to the application. We reached a good performance with *c* assigned to 0.2. Decreasing the parameter *c* increases the probability of completely missing the target while tracking. However, decreasing the parameter *c* increases the precision of the localization. In other words, the localization performance increases with decreasing *c* and variance of the particles increases with increasing *c*.

As mentioned in DEMC sampling algorithm parameter *b* should be very small. Therefore, it is not very suitable to make calibration of parameter *b*. The only remaining parameter that we can change, is the temperature. The temperature should be adjusted according to the characteristics of the likelihood function. If the likelihood function is very distinctive and can separate the feature space strictly, it is quite helpful to decrease the temperature. However, if the likelihood function has a big variance on the feature set, we may think of increasing it and by this means, we can increase the variance of the particles.

Video Sequence : 1

Number of Vehicles : 10

Number of Tracked Frames : 210

Process Noise Standard Deviation : [10,5,30,0.02,0.02,0.01]

Table 9.30: DEMC P. F. Tracking Results with Different Parameters

Algorithm	Number of Particles	a	c	b	Temp	Total Corner Displacement Error
DEMC Particle Filter	360	0.8	0.2	5,5,5,0.5,0.5,0.5 *1e-2	1e-3	2438
DEMC Particle Filter	360	0.8	0.2	5,5,5,0.5,0.5,0.5 *1e-2	1	2608
DEMC Particle Filter	360	0.5	0.2	5,5,5,0.5,0.5,0.5 *1e-2	1e-3	7326
DEMC Particle Filter	360	0.8	0.5	5,5,5,0.5,0.5,0.5 *1e-2	1e-3	4486
DEMC Particle Filter	360	0.9	0.1	5,5,5,0.5,0.5,0.5 *1e-2	1e-3	2341

9.6 Speed Measurement by License Plate Tracking

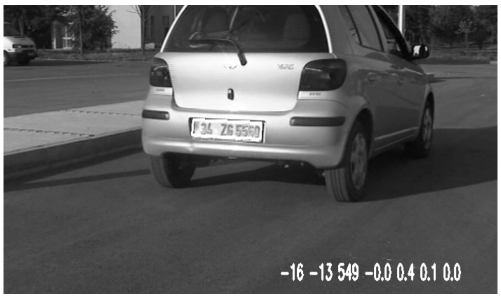
The natural result of tracking in 3D is the ability of measuring the 3D velocity of the license plate. The most useful information in motion is the speed of the license plate, which is also the speed of the vehicle. We can make the speed measurement of the vehicle in our approach as explained in section 6.2. In order to test this ability, we have recorded the videos of the vehicles with known speed values. The test video for speed measurement validation includes four pass of the same car with different speed values. The speeds are in time order 15 km/h, 28 km/h, 38 km/h and 21 km/h.

The videos are tracked by DEMC Particle Filter with 360 particles. We have utilized a high number of particles because here we do not make a performance test considering the particle number; we are testing the ability of speed measurement.

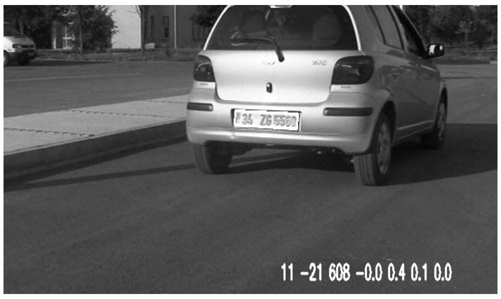
The video sequence and the measured speed is shown in Figure 9.27, Figure 9.28, Figure 9.29 and Figure 9.30

(a) Vehicle Speed 15 km/h Frame 8

(b) Vehicle Speed 15 km/h Frame 12

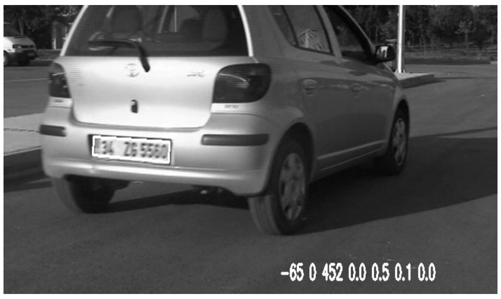


(c) Vehicle Speed 15 km/h Frame 16



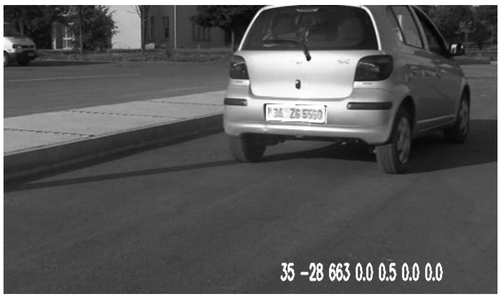
(d) Vehicle Speed 15 km/h Frame 20

Figure 9.27 : Test Vehicle Speed 15 km/h (a) Frame 8 (b) Frame 12 (c) Frame 16 (d) Frame 20 **(Measured Speed 15.77 km/h)**

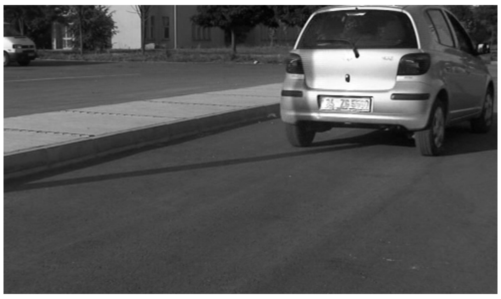


(a) Vehicle Speed 28 km/h Frame 296

(b) Vehicle Speed 28 km/h Frame 300

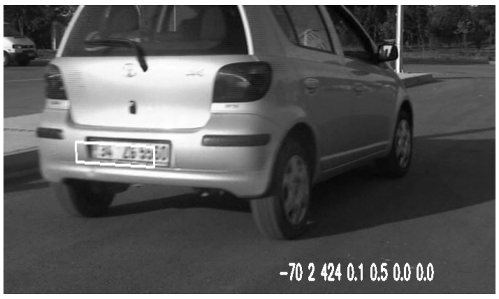


(c) Vehicle Speed 28 km/h Frame 304

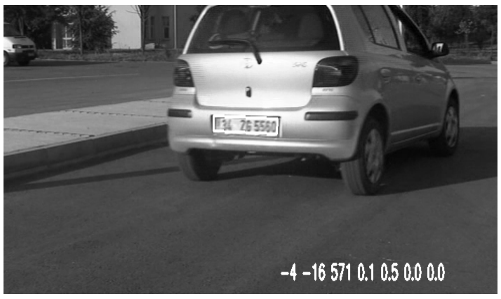


(d) Vehicle Speed 28 km/h Frame 308

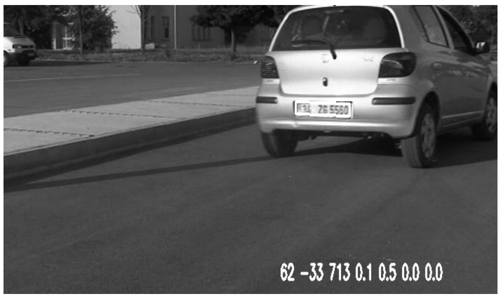
Figure 9.28 : Test Vehicle Speed 28 km/h (a) Frame 296 (b) Frame 300 (c) Frame 304 (d) Frame 308 **(Measured Speed 25.93 km/h)**



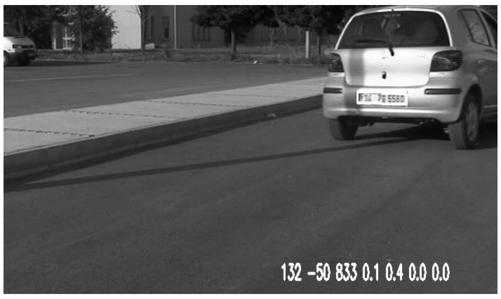
(a) Vehicle Speed 38 km/h Frame 504



(b) Vehicle Speed 38 km/h Frame 508

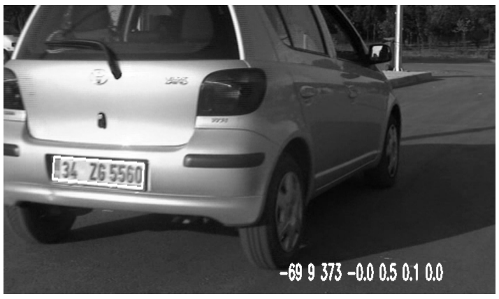


(c) Vehicle Speed 38 km/h Frame 512

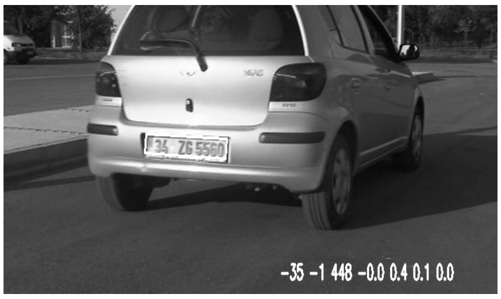


(d) Vehicle Speed 38 km/h Frame 516

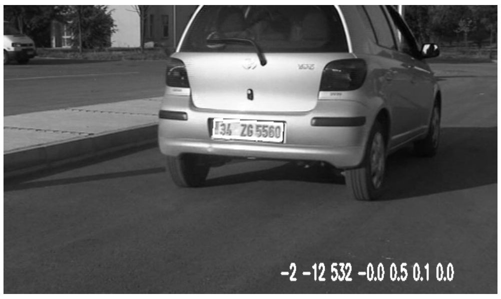
Figure 9.29 : Test Vehicle Speed 38 km/h (a) Frame 504 (b) Frame 508 (c) Frame 512 (d) Frame 516 **(Measured Speed 33.81 km/h)**



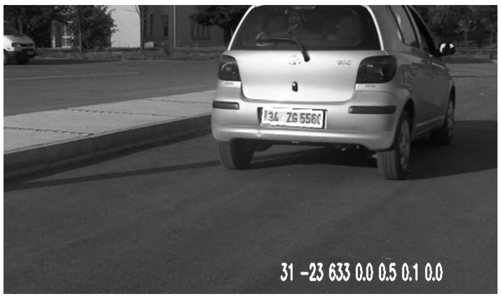
(a) Vehicle Speed 21 km/h Frame 688



(b) Vehicle Speed 21 km/h Frame 692



(c) Vehicle Speed 21 km/h Frame 696



(d) Vehicle Speed 21 km/h Frame 700

Figure 9.30 : Test Vehicle Speed 21 km/h (a) Frame 688 (b) Frame 692 (c) Frame 696 (d) Frame 700 **(Measured Speed 19.87 km/h)**

The velocity of the vehicle is measured by considering the whole tracking sequence, thus the 3D shift is divided to the time duration between the first and the last frames.

9.7 Occlusion Detection

It is a very frequent situation that occlusion of the license plate occurs by pedestrians and the other vehicles in urban area. An important advantage of utilizing a tracking filter is gaining robustness to the occlusions.

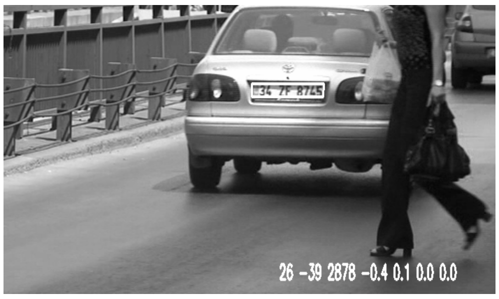
However, in order to be robust to the occlusions, the filter should have a mechanism of detecting the occlusions. Because the filter should be preserving its last kinematics and continue updating the particles discarding the wrong or absent measurements.

We have recorded a test sequence in order to present this feature of the DEMC Particle Filter. In this test sequence, a pedestrian occludes the license plate completely for 5 frames.

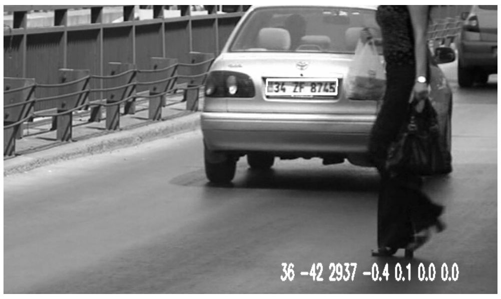
The standard Condensation algorithm has no occlusion detection feature. We have augmented this feature to the DEMC Particle Filter as explained in section 7.1. The Figure 9.31 shows the result of Condensation Algorithm tracking without occlusion detection. On the other hand, Figure 9.32 shows the result of tracking by DEMC Particle Filtering.

The Condensation algorithm continues to resample even if there is no visible license plate in the frame. This results in the shift of the estimated state to the road regions with comparably similar brightness levels to the license plate.

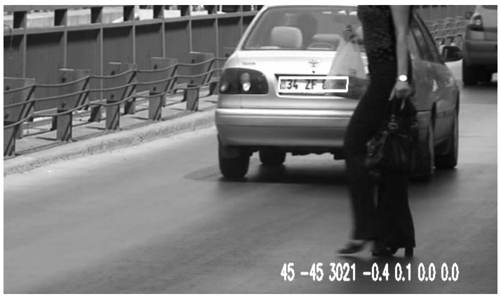
Nevertheless, the DEMC Particle Filter detects the occlusion and stops re-sampling of the particles, which result in the scattering of the particles with the last kinematics. After the occlusion is over, the probable particles are re-sampled quickly, which again refreshes the estimated state.



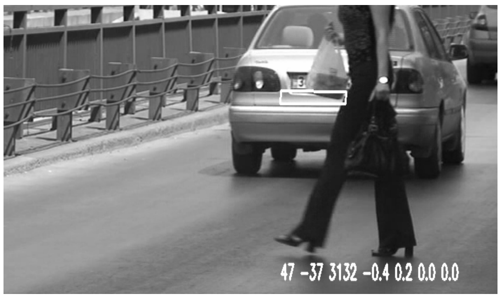
(a) Video Occlusion Frame 2



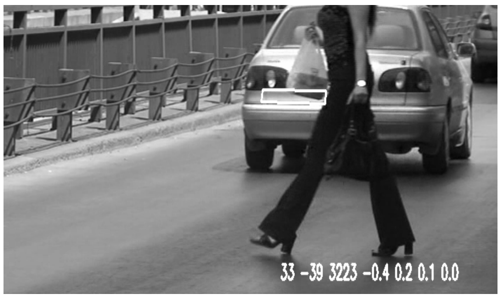
(b) Video Occlusion Frame 4



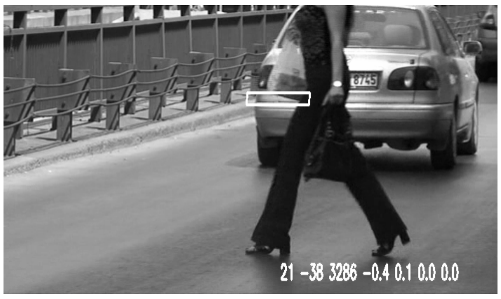
(c) Video Occlusion Frame 6



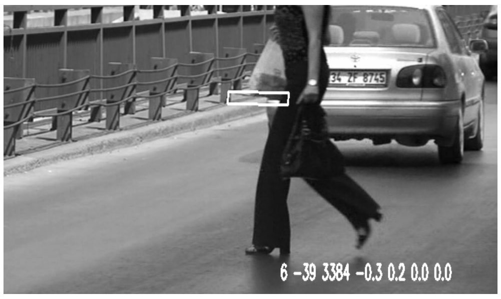
(d) Video Occlusion Frame 8



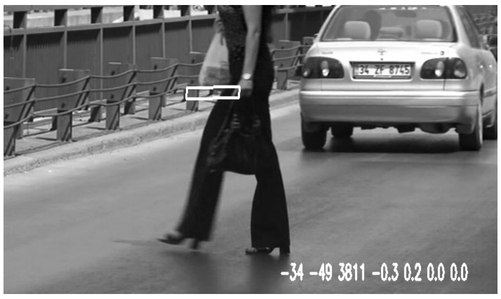
(e) Video Occlusion Frame 10



(f) Video Occlusion Frame 12

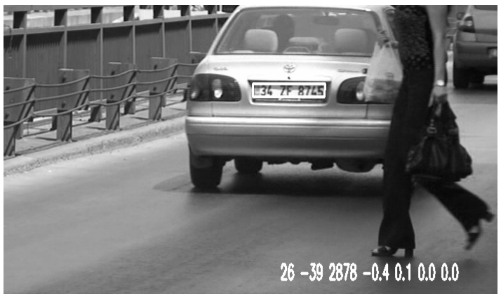


(g) Video Occlusion Frame 14

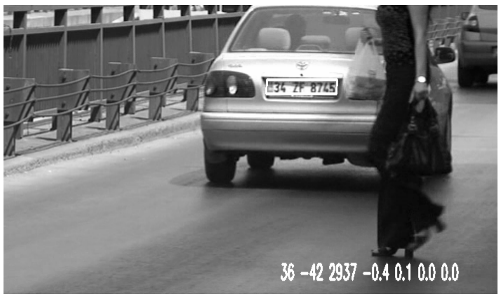


(h) Video Occlusion Frame 20

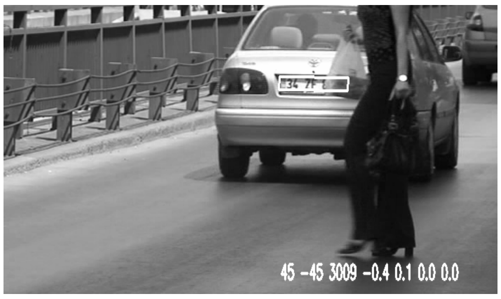
Figure 9.31 : Test Video Occlusion (without Occlusion Detection) (a) Frame 2 (b) Frame 4 (c) Frame 6 (d) Frame 8 (e) Frame 10 (f) Frame 12 (g) Frame 14 (h) Frame 20



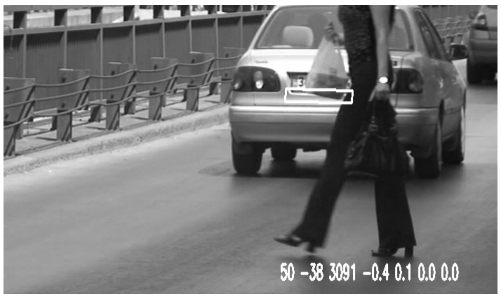
(a) Video Occlusion Frame 2



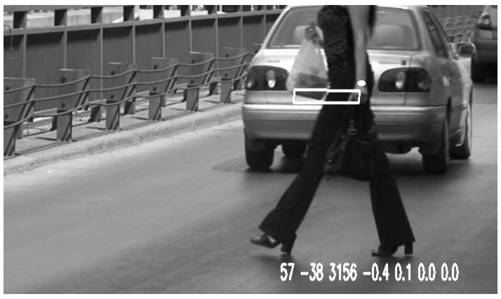
(b) Video Occlusion Frame 4



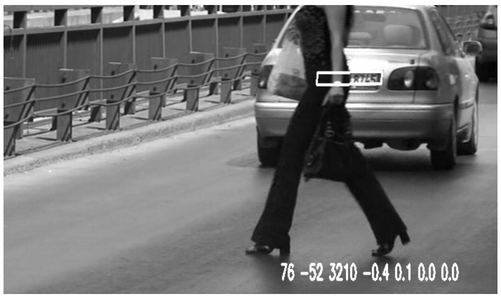
(c) Video Occlusion Frame 6



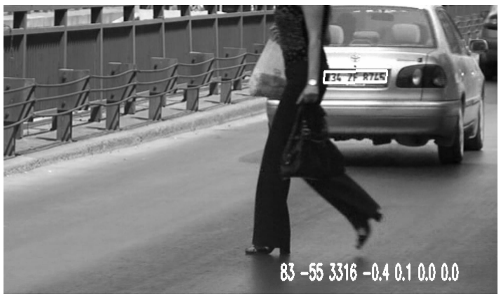
(d) Video Occlusion Frame 8



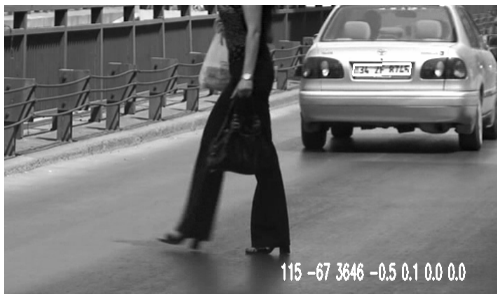
(e) Video Occlusion Frame 10



(f) Video Occlusion Frame 12



(g) Video Occlusion Frame 14



(h) Video Occlusion Frame 20

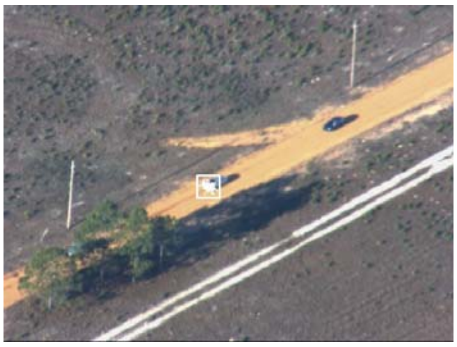
Figure 9.32 : Test Video Occlusion (with Occlusion Detection) (a) Frame 2 (b) Frame 4 (c) Frame 6 (d) Frame 8 (e) Frame 10 (f) Frame 12 (g) Frame 14 (h) Frame 20

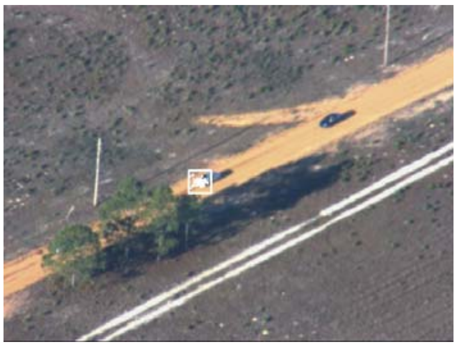
In Figure 9.33 and Figure 9.34, a test sequence is shown, which is recorded from a helicopter. This aerial video sequence is taken from the VIVID website [53]. The tracked vehicle disappears behind the trees and appears again after the occlusion. Again, Condensation and DEMC Particle Filter tracking results are presented in Figure 9.33 and Figure 9.34. DEMC Particle Filter manages to continue tracking after the occlusion, but Condensation algorithm gets stuck at the first location that the occlusion starts.

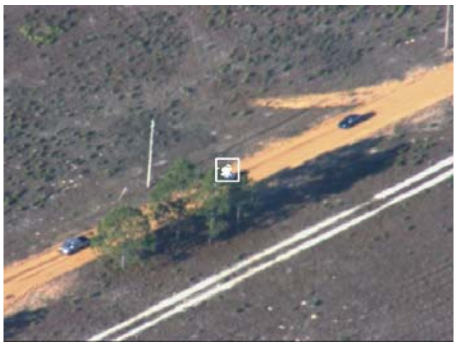
In this application, the color histogram is utilized for likelihood calculation [53]. The Bhattacharyya coefficient of the color distribution is calculated as a likelihood measure. In the figures, the particles are shown with small white dots, which are surrounded by a white rectangle. It can be observed in Figure 9.33 that at frame 740 the occlusion starts. Until this frame the particles are sampled around the tracked vehicle.

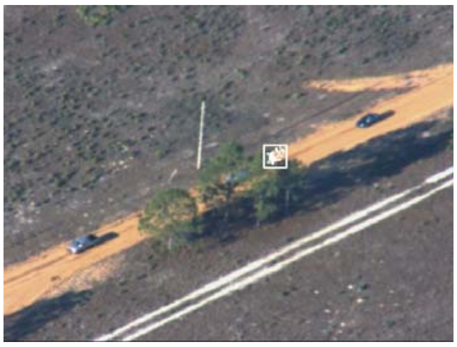
The Condensation algorithm continues to resample after the occlusion occurs. As a result, the particles stay re-sampled on the road color distribution, which is more close to the target color distribution than the tree's color distribution. The Condensation has no mechanism to overcome this situation and loses the vehicle, fails tracking further.

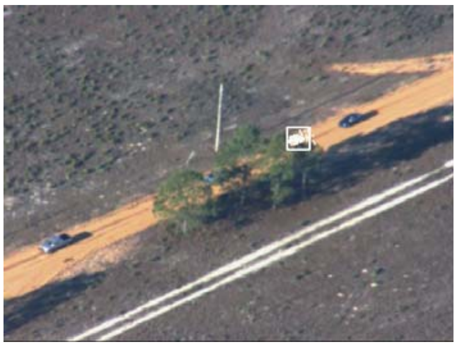
On the other hand, the DEMC Particle Filter detects occlusion at frame 740. After that point, it stops re-sampling and the particles get scattered by the noise added in each iteration. This behavior corresponds a search mechanism, which continues until a sufficiently close color distribution is observed on the image. This happens at frames 780 and 820. The DEMC Particle Filter continues its normal operation until it detects a new occlusion.

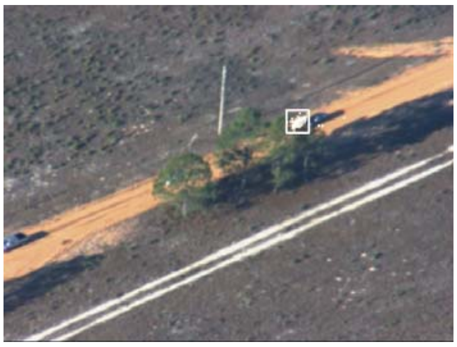

(a) Video Occlusion 2 Frame 700

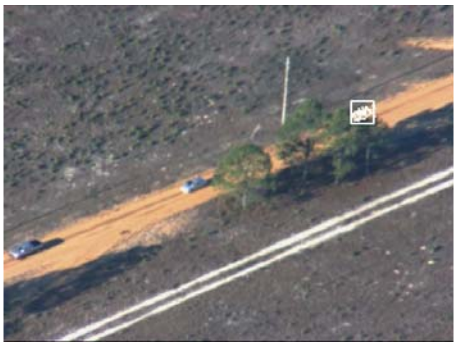

(b) Video Occlusion 2 Frame 720


(c) Video Occlusion 2 Frame 740


(d) Video Occlusion 2 Frame 760


(e) Video Occlusion 2 Frame 780


(f) Video Occlusion 2 Frame 800


(g) Video Occlusion 2 Frame 820

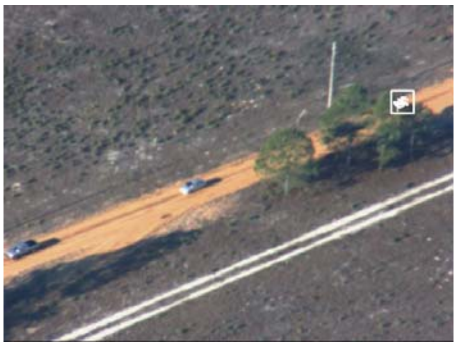
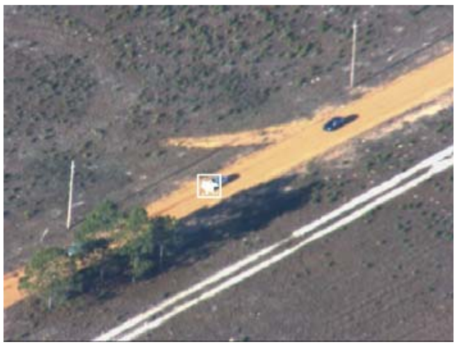
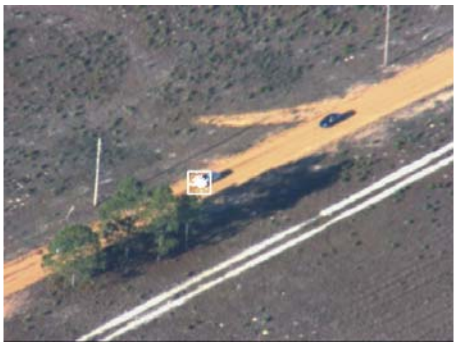
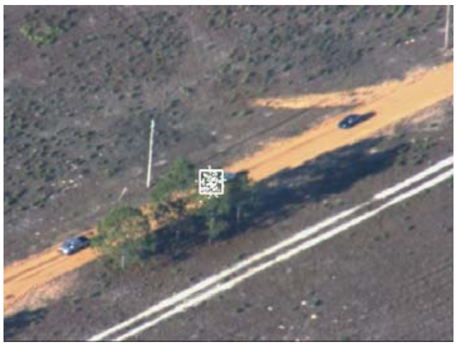
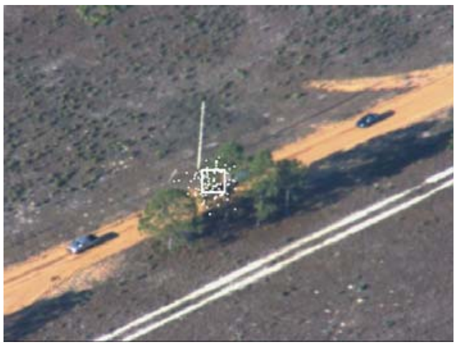

(h) Video Occlusion 2 Frame 840

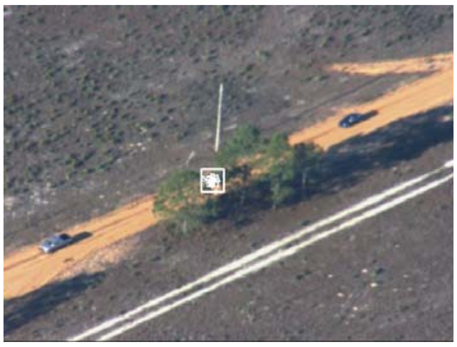
Figure 9.33 : Test Video Occlusion 2 (without Occlusion Detection) (a) Frame 700 (b) Frame 720 (c) Frame 740 (d) Frame 760 (e) Frame 780 (f) Frame 800 (g) Frame 820 (h) Frame 840

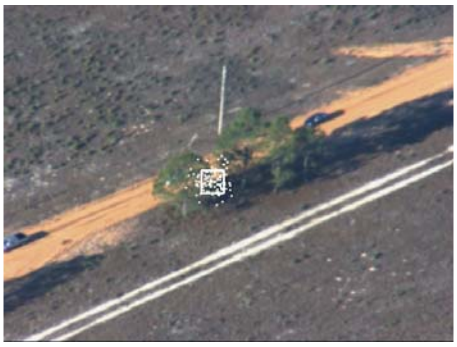

(a) Video Occlusion 2 Frame 700

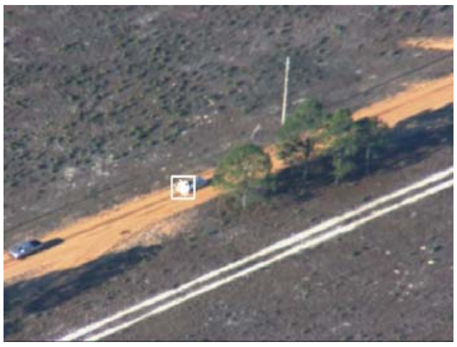

(b) Video Occlusion 2 Frame 720


(c) Video Occlusion 2 Frame 740


(d) Video Occlusion 2 Frame 760


(e) Video Occlusion 2 Frame 780


(f) Video Occlusion 2 Frame 800


(g) Video Occlusion 2 Frame 820

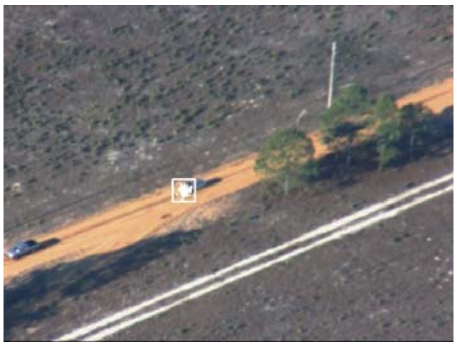

(h) Video Occlusion 2 Frame 840

Figure 9.34 : Test Video Occlusion 2 (with Occlusion Detection) (a) Frame 700 (b) Frame 720 (c) Frame 740 (d) Frame 760 (e) Frame 780 (f) Frame 800 (g) Frame 820 (h) Frame 840

10. CONCLUDING REMARKS

We have proposed a new algorithm DEMC Particle Filter in order to overcome the drawbacks of standard Condensation algorithm. We implemented the DEMC Particle Filter for 3D license plate tracking in video sequences. We compared the performance and the computation complexity of DEMC Particle Filter, Auxiliary Particle Filter, Condensation Algorithm and Genetic Condensation Algorithm. In the next sections, we will give a summary for the results, contributions and give suggestions for future work.

10.1 Summary of Results

The main deficiency of the Condensation algorithm is the sample impoverishment, which increases significantly the number of samples to achieve sufficient tracking performance. We may conclude that the meaningful samples, which represent the prior and posterior probability distributions, are needed. Condensation algorithm's re-sampling approach can be improved by different approaches. One of the approaches, that have been presented, is the Genetic Condensation algorithm, which utilizes genetic generations in order to improve the final sample set.

DE optimization tries to update all the samples once, by the means of combining the most probable samples. DEMC sampling approach does not spoil the multi-modal nature of the probability distribution. Instead, the mechanism of the DEMC sampling preserves the peaks of the probability distribution.

DEMC Particle Filter has a drawback of computational complexity compared to standard Condensation algorithm, even though it can achieve better tracking performance of Condensation algorithm with same computation load.

10.2 Summary of Contributions

- A new 3D dynamic system model is built for tracking license plates.

- The prediction of the license plate position on 2D is modeled as a projection of 3D points.
- Instead of working on 2D and suffering from strong perspective, an alternative approach has been studied.
- In order to estimate a 3D dynamic model, Condensation algorithm is utilized.
- Deficiencies of the Condensation algorithm has been studied.
- A new algorithm DEMC Particle Filter is proposed and its performance is evaluated. DEMC Particle Filter is superior compared to the Auxiliary Particle Filter, Condensation Algorithm and Genetic Condensation Algorithm with the same number of likelihood calculations.

10.3 Suggestion for Future Research

As a result of tracking, we obtain a set of coordinates of the license plate, from which the license plate can be extracted. These extracted license plate images can be used for recognition purposes. Recognition from multiple images will increase the performance of the recognition results.

The extracted license plate images can also be combined in order to get resolution increase with super resolution techniques. Recognition from higher resolution will improve the recognition performance.

Next opportunity of the tracking is detecting temporary occlusions of the license plate and eliminating these frames from the complete tracking sequence. Occlusion detection will stop recognition process when license plate is occluded, which can cause severe recognition errors if not avoided.

In our approach likelihood calculation is performed by simple template matching. Different features can be utilized for increasing the likelihood calculation performance. The techniques such as SIFT (Scale-invariant feature transform), GLOH (Gradient Location and Orientation Histogram) and SURF (Speeded Up Robust Features) can be utilized in the likelihood calculation process.

On the other hand, we have the 3D motion pattern of the license plate, so the vehicle, from which we can deduce the vehicles speed and illegal behavior.

Nevertheless, the implementation of our license plate tracking application can be improved in order to consume less system resources. This can be done by optimizing the c source code by applying better programming experience.

REFERENCES

[1] **Dubey, P.**, 2005. Heuristic approach for license plate detection, *IEEE Conference on Advanced Video and Signal Based Surveillance*, Como, Italy, September 2005, 366 - 370.

[2] **Lee, H. J., Chen, S. Y. and Wang, S. Z.**, 2004. Extraction and recognition of license plates of motorcycles and vehicles on highways, *Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference*, Aug. 2004, 356 - 359.

[3] **Hsieh, C. T., Juan, Y. S. and Hung, K. M.**, 2005. Multiple license plate detection for complex background, *Pattern Recognition, 2004. Advanced Information Networking and Applications, 2005. AINA 2005. 19th International Conference*, March. 2005, 389 - 392.

[4] **Jia, W., Zhang, H. and Piccardi, X. H.**, 2005. Mean shift for accurate license plate localization, *Intelligent Transportation Systems*, Sept.. 2005, 566 - 571.

[5] **Jun, X., Sidan, D., Duntang, G. and Qinhong, S.**, 2004. Locating car license plate under various illumination conditions using genetic algorithm, *Signal Processing, 2004. Proceedings. ICSP '04. 7th International Conference*, Aug. 2004, 2502 - 2505.

[6] **S. K. Kim, D. W. Kim, H. J. Kim**, 1996. A recognition of vehicle license plate using a genetic algorithm based segmentation, *Image Processing ICIP96 .International Conference*, Sept.. 1996, 661 - 664.

[7] **Zayed, M., Boonaert, J. and Bayart, M.**, 2004. License plate tracking for car following with a single camera, *Intelligent Transportation Systems 2004*, Oct. 2004, 719 - 724.

[8] **Vacchetti, L., Lepetit, V. and Fua, P.**, 2004. Combining edge and texture information for real-time accurate 3D camera tracking, *Mixed and Augmented Reality, 2004. ISMAR 2004*, Nov. 2004, 48 - 56.

[9] **Ponsa, D., Lopez, A., Serrat, J., Lumbreras, F. and Graf, T.**, 2005. Multiple vehicle 3D tracking using an unscented Kalman Filter, *Transportation Systems IEEE 2005*, Sept.. 2005, 1108 - 1113.

[10] **Chang, W. Y., Chen, C. S. and Hung, Y. P.**, 2005. Appearance-guided particle filtering for articulated hand tracking, *Computer Vision and Pattern Recognition,. CVPR 2005*, June 2005, 235 - 242.

[11] **Lee, H., Kim, D. and Bang, S. Y.**, 2003. Real-time automatic vehicle management system using vehicle tracking and car plate number identification, *Multimedia and Expo, 2003. ICME '03,* July 2003, 353 - 356.

[12] **Anagnostopoulos, C., Tsekouras, I., Kouzas, G., Loumos, G. and Kayafas, V.**, 2005. Using sliding concentric windows for license plate segmentation and processing, *Signal Processing Systems Design and Implementation,* Nov. 2005, 337 - 342.

[13] **Isard, M. and Blake, A.**, 1998. A mixed-state condensation tracker with automatic model-switching, *Computer Vision, 1998. Sixth International Conference,* Jan. 1998, 107 - 112

[14] **MacCormick, J. and Blake, A.**, 1998. A probabilistic contour discriminant for object localisation, *Computer Vision, 1998. Sixth International Conference,* Jan. 1998, 390 - 395.

[15] **Baoxin, L. and Chellappa, R.**, 2000. Simultaneous tracking and verification via sequential posterior estimation, *Computer Vision and Pattern Recognition IEEE 2000,* Oct. 2000, 110 - 117.

[16] **Philomin, V., Duraiswami, R. and Davis L.**, 2000. Pedestrian tracking from a moving vehicle, Intelligent Vehicles, *Intelligent Vehicles Symposium, 2000 IEEE,* Oct. 2000, 350 - 355.

[17] **Zhu, Y. and Zhi-Qiang, L.**, 2005. Genetic CONDENSATION for motion tracking In: Machine Learning and Cybernetics, *Machine Learning and Cybernetics, 2005,* Aug. 2005, 5542 - 5547.

[18] **Le, L., Xiang-Tian, D. and Hager G.**, 2004. A Particle Filter without Dynamics for Robust 3D Face Tracking, *Computer Vision and Pattern Recognition Workshop 2004,* June 2004, 70 - 73.

[19] **Yalcin, I. K. and Gokmen, M.**, 2005. License Plate Tracking from Monocular Camera View by Condensation Algorithm, *ICIC 2005 Heife China Advances in Intelligent Computing LNCS,* June 2005, 860 - 869.

[20] **Yalcin, I. K. and Gokmen, M.**, 2006. Integrating Differential Evolution and Condensation Algorithms for License Plate Tracking, *Pattern Recognition, 2006. ICPR 2006. 18th International Conference*, Aug. 2006, 658 - 661.

[21] **Trucco, E. and Verri, A.**, 1998. *Introductory Techniques for 3-D Computer Vision,* Prentice Hall, New York.

[22] **Faugeras, O.**, 1993. *Three Dimensional Computer Vision,* MIT Press, New York.

[23] **Kalman, R. E.**, 1960. A New Approach to Linear Filtering and Prediction Problems *Transaction of the ASME—Journal of Basic Engineering*, 82, 35 - 45.

[24] **Fearnhead, P.**, 1998. Sequential Monte Carlo methods in filter theory*, Phd Thesis,* University of Oxford

[25] **Ko, M. A. and Kim, Y. M.**, 2004. A simple OCR method from strong perspective view*, Proceedings of the 33rd Applied Imagery Pattern Recognition Workshop (AIPR'04)*, 2004, 1550.

[26] **Cui, Y., Huang, Q.**, 1998. Extracting characters of license plates from video sequences, *Machine Vision and Applications (1998)*, 10, 308–320

[27] **Hsien-Huang, P. W.**, 2006. License Plate Extraction in Low Resolution Video*, Proceedings of the 18th International Conference on Pattern Recognition (ICPR'06)*, 2006, 824 - 827.

[28] **Barkat, M.**, 2005. *Signal detection and estimation,* Artech House, London.

[29] **Buhmann, M. D. and Iserles, A.**, 1997. *Approximation Theory and Optimization,* Cambridge University Press, Cambridge.

[30] **Bar-Shalom, Y., Li, X. R. and Kirubarajan, T.**, 2001. *Estimation with applications to tracking and navigation,* John Wiley & Sons, Inc, Canada.

[31] **Kay, S. M.**, 1993. *Fundamentals of Statistical Signal Processing: Estimation Theory,* PrenticeHall PTR, Henceforth.

[32] **Brookner, E.**, 1998. *Tracking and Kalman Filtering Made Easy,* John Wiley & Sons, Inc, New York.

[33] **Chong, E. K. P. and Zak, S. H.**, 2001. *An Introduction to Optimization,* John Wiley & Sons, Inc, New York.

[34] **Sarker, R., Mohammadian, M. and Yao, X.**, 2003. *Evolutionary Optimization,* Kluwer Academic Publishers, New York.

[35] **Najim, K., Ikonen, E. and Daoud, A. K.**, 2004. *Stochastic Processes Estimation, Optimization & Analysis,* Kogan Page Science, London.

[36] **Bonnans, J. F., Gilbert, J. C., Lemarechal, C. and Sagastizabal, C. A.**, 1997. *Numerical Optimization Theoretical and Practical Aspects,* Springer, Berlin.

[37] **Spall, J. C.**, 2003. *Introduction to Stochastic Search and Optimization,* Wiley-Interscience John Wiley & Sons, Inc, New Jersey.

[38] **Marti, K.**, 2005. *Stochastic Optimization Methods,* Springer, Berlin.

[39] **Fishman, G. S.**, 1996. *Monte Carlo Concepts, Algorithms and Applications,* Springer, Berlin.

[40] **Sakawa, M.**, 2002. *Genetic Algorithms and Fuzzy Multiobjective Optimization,* Kluwer Academic Publishers, Massachusetts.

[41] **Price, K. V., Storn, R. M. and Lampinen, J. A.**, 2005. *Differential Evolution a Practical Approach to Global Optimization,* Springer, Berlin.

[42] **Feoktistov, V.**, 2006. *Differential Evolution In Search Of Solutions,* Springer, Berlin.

[43] **Dobbert, T.**, 2005. *Matchmoving: The Invisible Art of Camera Tracking,* Sybex, Alameda.

[44] **Merklinger, H. M.**, 2007. *Focusing the View Camera,* Published by the author, Pdf only.

[45] **Wegener, I., translated from the German by Pruim, R.**, 2005. *Complexity Theory Exploring the Limits of Efficient Algorithms,* Springer, Berlin

[46] **Ter Braak**, **C. J. F.**, 2006. A Markov Chain Monte Carlo version of the genetic algorithm Differential Evolution: easy Bayesian computing for real parameter spaces, in *Statistial Computing 2006,* Springer, **16,** pp. 239-249

[47] **Strens, M., Bernhardt, M. and Everett, N.**, 2002. Markov Chain Monte Carlo Sampling using Direct Search Optimization *Proceedings of the Nineteenth International Conference on Machine Learning*, July 2002, 602 - 609.

[48] **Arulampalam, S., Maskell, S., Gordon, N. and Clapp, T.**, 2002. A Tutorial on Particle Filters for On-line Non-linear/Non-Gaussian Bayesian Tracking, *IEEE Transactions on Signal Processing*, 50, 174 - 188.

[49] **Doucet, A.**, 1998: On Sequential Simulation-Based Methods for Bayesian Filtering, *Signal Processing Group Department of Engineering University,* **CUED/F-INFENG/TR. 310,** Cambridge University

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

[51] **Bolic, M., Djuric, P. M. and Hong, S.**, 2004. Resampling Algorithms for Particle Filters: A Computational Complexity Perspective, *EURASIP Journal of Applied Signal Processing*, 15, 2267-2277.

[52] **Karlsson, R. and Bergman, N.**, 2000. Auxiliary particle filters for tracking a maneuvering target, *IEEE Conference on Decision and Control,* Sydney, Australia, December 2000, 3891 - 3895.

[53] **Hebert, M.**, 2007. Video Verification of Identity (VIVID) Project Homepage, <http://www.cs.cmu.edu/~hebert/vivid.html>.

CURRICULUM VITAE

İlhan Kubilay YALÇIN was born in İstanbul in 1976. He got the Bachelor of Science degree from Istanbul Technical University Electrics and Electronics Engineering Faculty, Electronics and Communication Engineering Department in 1998. At the same year, he was accepted to the Master of Science program in the Computer Science Department of Gerhard Merchator University in Germany and graduated in 2000. In 2000 fall semester, he started his Ph.D. education at Istanbul Technical University Computer Engineering department. Currently, he is working as senior researcher in TUBITAK Marmara Research Center, Information Technologies Institute in Gebze, Kocaeli.