GEOMETRIC DEEP PARTICLE FILTER FOR MOTORCYCLE TRACKING: DEVELOPMENT OF INTELLIGENT TRAFFIC SYSTEM IN JAKARTA

Alexander A S Gunawan¹, Wisnu Jatmiko²

¹Bina Nusantara University, Mathematics Department, School of Computer Science, Jakarta, Indonesia
²Faculty of Computer Science, Universitas Indonesia, Depok, Indonesia

Emails: aagung@binus.edu, wisnuj@cs.ui.ac.id

Submitted: Oct. 4, 2014      Accepted: Jan. 20, 2015        Published: Mar. 1, 2015

Abstract - Intelligent Transportation Systems (ITS) is the combination of transportation systems with Information and Communication Technology (ICT). In Jakarta traffic, there is unique issue that does not arise in developed countries: very large number of motorcycles. Nevertheless, the enabling technologies for the detection, measurement, recording, and information distribution of motorcycle have not been fully developed in the existing researches. With the above considerations, we establish research which aimed to develop enabling technology especially in here for tracking motorcycle using camera.

This paper is presented our proposed tracker which called as Geometric Deep Particle Filter (GDPF) for tracking motorcycle using camera. The tracker is inspired by human visual perception which has nonretinotopic nature. Based on particle filter approach, our goal is to improve the transition model in order to overcome motorcycle maneuver. We will exploit this curved nature of the state space using geometric computing theory, such as Lie groups, and Lie algebras. A number of experiments have been conducted for this research, and it has been found that GDPF has achieved certain degree of success in object tracking.

Index terms: visual tracking, motorcycle, nonretinotopic, particle filter, deep learning, geometric computing, affine transformation.
I. INTRODUCTION

Many Asian megacities are suffered with traffic congestion which tends to resist various solution initiated by different governments. Jakarta as one of Asian megacities has experienced the traffic congestion for many years and became extremely worse today. Recently, the Indonesia Ministry of Economic Coordination asserted that the Jakarta inhabitants spend about 60% of their travel time in the traffic [1]. And the transportation cost has reached about 30 percent of their ordinary incomes [2]. Furthermore the average speed of a vehicle in Jakarta traffic drops to ±20 km/h in 2010 from ±26 km/h in 2002. Jakarta Traffic Management Center stated that the number of the private cars increased about 4 percent in a year since 2009 to 2,115,786 vehicles. During the same duration, the public vehicles increased about 1.5 percent to 859,692 vehicles, and the motorcycles significantly increased about 11.1 percent to 7,516,536 vehicles[1]. Furthermore the number of the motorcycles has increased by a factor of three from 2002 to 2007[3]. Thus the traffics have dominant motorcycles, which do not have clear vehicular lane in driving. Therefore, there is big interest to reduce the congestion problem in Jakarta. To solve the congestion problem, it is obviously indispensable to build an efficient and high capacity mass transportation system [4]. Nevertheless in megacities like Jakarta, there is a great barrier to build new infrastructure of mass transportation system due to space constraints. Thus, it is critical to optimize the existing infrastructure. For this purpose, we can utilize information technology to optimize the transportation system. Combination of the transportation system and the information technology system is called as Intelligent Traffic System (ITS). By interaction of the two systems, the transportation can be managed more efficiently. In ITS, information technology is connected with the transportation system through enabling technologies[5](see Figure 1). More detail, the enabling technologies including data acquisition, data processing, data communication, information distribution, and information utilization are tabulated in Table 1.
Table 1: ITS enabling technologies [5]

<table>
<thead>
<tr>
<th>ITS Enabling Tech</th>
<th>Infrastructure Side</th>
<th>Vehicle Side</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data acquisition</td>
<td>Traffic detectors</td>
<td>Automatic vehicle detection</td>
</tr>
<tr>
<td></td>
<td>Weather monitors</td>
<td>Weigh-in-motion</td>
</tr>
<tr>
<td>Data processing</td>
<td>Data fusion</td>
<td>Global positioning system</td>
</tr>
<tr>
<td></td>
<td>Automatic vehicle detection</td>
<td>Digital map</td>
</tr>
<tr>
<td>Data communications</td>
<td>Stationary communications</td>
<td>Mobile communications</td>
</tr>
<tr>
<td></td>
<td>Fiber optics</td>
<td>Dedicated short-range communications</td>
</tr>
<tr>
<td>Information distribution</td>
<td>Variable message sign</td>
<td>Highway advisory radio</td>
</tr>
<tr>
<td></td>
<td>Internet</td>
<td>Radio data system/traffic message channel</td>
</tr>
<tr>
<td>Information utilization</td>
<td>Ramp metering</td>
<td>Route guidance</td>
</tr>
<tr>
<td></td>
<td>Urban traffic control</td>
<td>Crash avoidance</td>
</tr>
</tbody>
</table>

One of ITS applications is the floating vehicle technique (FVT), which mainly use GPS as vehicle sensors to provide a real-time traffic data [6]. The FVT framework can be seen in Figure 2.

The proposed system in our research consists of traffic sensors that are video camera and GPS in smart phone and traffic verifier that is Twitter (see Figure 3). The system is based on non intrusive technique [7] by using remote observations, which includes wired and wireless sensors.
In our proposed integrated ITS architecture, there are 2 traffic sensors and 1 traffic verifier (see Figure 3). Smartphone with location data acquisition through GPS, require users to act as agents and Twitter requires dedicated volunteers to feed information of traffic data. But, video cameras through CCTV require no users or volunteers, but collected video camera must be processed sophisticatedly to extract useful traffic data.

In summary, the integrated ITS which is under construction at Faculty of Computer Science, University of Indonesia will work as follow [7]:

- Activated mobile agents through GPS in smart phone will send location coordinates (i.e. longitude and latitude), then server will detect whether the mobile agent is inside a certain zone (called as VDZ) as it moves along a road.
- Video cameras from the CCTV, provided by Jasa Marga and TMC – Jakarta’s Police department will be processed by a Beagle Board to execute vehicle classification, tracking and speed estimation.
- Traffic situation is extracted from TMC’s Twitter data.
- Finally, all data from three traffic sensors is processed to give a useful traffic advice to the user in the server.

Society is rapidly accepting the use of cameras in a wide variety of locations and applications such as traffic surveillance. In this paper, we focus on video camera as non-intrusive sensor through CCTV which is shown in Figure 3. In general, visual traffic surveillance systems with
the ability to detect, track, count, and classify vehicles can be utilized in obtaining better traffic control in ITS. Nevertheless, visual traffic surveillance using computer vision techniques must cope with a number of challenges in handling the tasks, for example: cars in general have similar shapes and sizes, but in video camera can be seen very different due to different environment (e.g. varying weather and quality of light) and 3D deformation (e.g. different angles of view). Therefore, visual vehicle classification poses a difficult problem because of high intraclass variation and relatively low interclass variation [8]. Thus the main step in visual traffic surveillance is to solve the observation model of the visual object, because in urban ITS, it is common to use broader categories of road user such as people, car, van, bus and motorcycle. In this paper, we proposed deep learning [9] in creating the observation model based on appearance of the object in video camera.

Due to various social and economic factors, motorcycle is becoming increasingly popular in Jakarta. Thus the performance of visual traffic surveillance has to be increased up to motorcycle detection and tracking in our study to optimize Jakarta’s transportation system. Unfortunately, majority of the existing research are tailored to tackle the condition of traffic in advanced countries, where the number of motorcycles is not too significant. By considering the uniqueness of the present traffic and limitation of the existing research, we devise to extend the tracking algorithm to take into account realities found in the Jakarta urban traffic. As the first step in our project, the enabling technology based on video camera is developed for tracking vehicles which take into account motorcycle by increasing the tracking performance using deep learning [9] and geometric particle filter [10].

II. MOTORCYCLE TRACKING BASED ON CAMERA

Intelligent transportation systems (ITS) have attracted huge research attention in vehicle detection, tracking, and recognition. In light of the availability of low-cost camera, computer vision has become a promising base technology for traffic surveillance systems. Furthermore vision sensors provide more information than the conventional sensors widely used in ITS. Thus main attention in developing enabling technologies for ITS is now focused on vision-based traffic surveillance systems.
CCTV is one of the tools that can be used to capture visual data from daily traffic events. Information from captured video of CCTV can give us real-time information about the traffic situations, such as number of vehicles, vehicles speed and their behavior. This research uses CCTV as its source of traffic information, provided by Jasa Marga and TMC – Jakarta’s Police department. In several major cities of Indonesia, CCTVs have been installed in many intersections, in order to monitor traffic condition. Currently, CCTVs installation in various cities in Indonesia, especially in Jakarta are not equipped with video processing, thus it is only used for traffic surveillance. Therefore, video processing will make a better use of many CCTVs that have been installed on Jakarta streets[11].

As stated in introduction, the number of motorcycles increases significantly in recent years and become dominant transportation mode in Jakarta. Thus traffic sensors are needed to obtain better tracks of motorcycles to evaluate their impact on crashes and traffic flow behavior. Historically, the effort to involve motorcycles in traffic model is based on fluid flow assumptions. Tracking and counting motorcycles was a low priority and actually ignored for long time in advanced countries. As a result, there have been little efforts by industry to handle the issue of tracking and counting motorcycles. Thus, most commercially available systems are unable to accurately capture motorcycle traffic[12]. The main reasons why motorcycles are difficult to handle is their small size and various appearances on road. The problem is further worsened by behavior of motorcyclists. Existing methods in computer vision exhibit difficulties in tracking motorcycles that travel not in vehicular lane or close behind each other. They also have difficulty in distinguishing group of motorcycles from passenger vehicles. The design of most traffic monitoring equipment assumes that vehicles travel one by one in certain distance and single vehicular lane. This is not true for motorcycles. Rather, it is common for motorcyclists to ride closely spaced in group and travel not in single vehicular lane. These behaviors will confuse most traffic monitoring equipments. Alternatively, some researchers conduct manual counting and classification, but these efforts are labor intensive and lead to inaccurate data. Finally, we can simplify the difficulty to handle motorcycle into two categories, first due to its appearance (e.g. occlusion, 3D deformation) and second due to its dynamics (e.g. maneuver). Based on this conclusion, we propose an approach to increase the tracking performance which take into account motorcycle by using deep learning [9] to handle its appearance problems and geometric particle filter[10] to solve its dynamic motions.
Over the past years, several works were performed in motorcycle tracking on public roads. There is a survey in this subject [13]. The survey focused on on-road detection when motorcycles approach the research vehicle and the sensor is attached on its rear side. The paper discusses the motorcycle detection problems using different sensors and also types of sensor for vision-based motorcycle detection. It is also described a critical overview of the methods of detection and tracking motorcycle using moving camera.

Paper[14] proposes a vision-based motorcycle monitoring system to detect and track motorcycles for data collection purposes. The system can handle occlusion problems using segmentation method and its features such as visual length, visual width, and pixel ratio. To verify motorcycles, it is used helmet detection. The result of this paper is extended in paper [15], which was evaluated using video data, collected indifferent weather and lighting conditions. Experimental results show the average of correct motorcycle detection rate is greater than 90% under various weather conditions.

Duan et al presents a real-time on-road lane change assistant that can identify motorcycles [16]. The information used to detect vehicles during the daytime includes symmetry, color, shadow, geometric features. And classification is done using Support Vector Machines (SVMs). A field tests on different road provided motorcycle detection rates of over 90%. No mention was stated in extending research to motorcycle tracking.

Chiverton [17] proposed a system for the classification and tracking of motorcycle riders with and without helmets. The system uses support vector machines trained on histograms. The trained classifier is incorporated into a tracking system where motorcycle riders are automatically segmented from video data using background subtraction. Tests show the system can accurately classify whether riders are wearing helmets or not on static photographs.

Due to the large number of existing motorcycle in Brazil, paper [18] have goals to study and implement some methods for automatic detection of motorcycles on public roads. For feature extraction of motorcycle images, it is used the algorithms SURF, HAAR, HOG and LBP as descriptors. And for image classification, it is used Multilayer Perceptron, Support Vector Machines and Radial-Bases Function Networks as classifier. Finally, it can be concluded that SVM combined with LBP proved to be robust enough for motorcycle detection.

Different to papers in [14, 15, 16, 17 and 18] which are mainly focused on motorcycle detection and classification, the research of Nguyen et al [19] have been emphasized to increase tracking
performance using particle filter. The paper overcomes the limitations of particle filter for maintaining the multi-modal target distribution using multi-modal Particle filter (MPF). Multi-modal target distribution may arise due to occlusion, clutter or the presence of multiple objects.

III. PROPOSED SOLUTION

In general, a common visual object tracking system is composed of four parts: object initialization, search mechanism, observation model and transition model, which are explained as follow:

- Object initialization. The object initialization can be done manually or automatically. Manual initialization is performed by annotating object locations with bounding boxes. While automatic initialization is usually done by object detectors.

- Search Mechanism. Deterministic or stochastic approaches can been used to estimate the state of the target objects. In deterministic approach, the tracking problem is considered as an optimization problem. By assuming the objective function is differentiable with respect to the motion parameters, gradient descent methods can be used to solve the tracking problem. On the other hand, stochastic algorithms such as particle filters have been used broadly because this approach does not need to make assumptions about the differentiability of motion parameter.

- Observation model. The goal of observation model is to describe the “thing” that we are tracking. This generally consists of two steps: choosing visual features and building models as object representation based on the chosen features. For comprehensive explanation of this subject can be looked in [20].

- Transition Model. The location of a target object can be represented by several parameters called as hidden states in certain transformation space. Transition model will represent the dynamics between hidden states in this space. Commonly, it is formulated as a dynamic state estimation problem, that is:

$$x_t = f(x_{t-1}, v_t)$$
$$z_t = h(x_t, w_t)$$

(1)
where $x_t$ is the current hidden state at $t$, $f$ is the state dynamic function, $v_t$ is the dynamic process noise, $z_t$ is the current observation, $h$ denotes the measurement function, and $w_t$ is the measurement noise.

In order to design the four parts of visual object tracking system, we study human visual perception and get inspiration for improving the existing system. The key to reverse engineering in human visual perception is the understanding how to represent and process the visual information in the brain. Human visual processing is often assumed to be retinotopic, which means the visible object are projected from photo-receptors in the retina to neurons have similarity as mapping from real object to appearance in a digital image[21]. This is analogous to a computer keyboard, which a physical key press is mapped to an ASCII character. Illustration of retinotopic mapping can be seen in Figure 4.

![Retinotopic mapping of the visual field][22]

While it is clear that geometry and human visual perception are closely related, it is very little understanding beyond its retinotopic association. Nevertheless, a recent study on human vision [23] shows that the representation in higher visual areas of the visual cortex occurs in a nonretinotopic manner. This means visual perception creates dynamic layers for each moving object in the scene. Therefore, human visual processing always maintains the identity of observed objects across space and time. This representation suggests that the appearance of the objects and their perceived locations are marginal independent. The further results related to attention stated that nonretinotopic integration occurs in all human visual processing. However for complex motion structures, the integration depends strongly on attention by combining different, independent motion trajectories [24]. Nonretinotopic integration is illustrated in Figure 5.
Based on the study of human visual perception, we propose a new tracker that represents nonretinotopic integration in human visual processing. The initial idea of this tracker can be found in [25]. Many factors [26] can affect the tracking robustness, which cause visual object tracking to be greatly challenging, including (i) real-time requirements; (ii) object dynamic motions (e.g., fast motion, blur motion, in-plane rotation, maneuver); (iii) object appearance variations (e.g., partial or full occlusion, deformation, scale variation, out-of-view); (iv) different environment (e.g., illumination variation, background clutters); (v) low quality camera sensors (e.g., low frame rate, low resolution, and color distortion). These challenges can lead to tracking deterioration and failures.

In order for the tracker to be robust in facing unpredictable noises, it is laid on Bayesian approach for object tracking. There are two main components in nonretinotopic integration concept: (i) Retinotopic space, representing visual appearance through visual observation, and (ii) Nonretinotopic manifold, representing object identification through object motion knowledge. Through Bayesian approach, the nonretinotopic nature can be modeled by building two submodels, which marginally independent, that is:

1. Observation model, representing retinotopic space in handling visual appearances.
2. Transition model, modeling nonretinotopic manifold in estimating object motion dynamics.

Attention in our nonretinotopic nature has a role to govern the relationships between retinotopic space and nonretinotopic manifold for handling the visual world based on its complexity. Furthermore, the proposed Bayesian model will mimic how to handle the relationships between observation model and transition model as is done by the attention in human vision.
processing. The proposed solution is described in the next subsection, which starts with Bayesian model for object tracking.

A. Tracking as Bayesian Problem

The visual tracking problem of moving objects can be modeled as first-order Markov process where we want to estimate the hidden state $x_i$ from the observations set in discrete time $t$. The main purpose of the tracker is to estimate the distribution of hidden state $x_i$, which represents the evolution of object dynamics, when given observation set of the target $z_{1:t-1}$. Based on Bayes' theorem, object tracking problems can be described [27] as follows:

Prediction:

$$p(x_{t|t} | z_{1:t-1}) = p(x_t | x_{t-1}) p(x_{t-1} | z_{1:t-1})$$  \hspace{1cm} (2)

Update:

$$p(x_{t|t} | z_{1:t}) = \frac{p(z_t | x_{t|t}, z_{1:t-1}) \ p(x_t | z_{1:t-1})}{p(z_t | z_{1:t-1})}$$  \hspace{1cm} (3)

The above equations give Bayesian solution of estimation problem in a recursive form. The above recursive Bayesian solution provides the posterior conditional distribution $p(x_t | z_{t})$ at time $t$, given set of measurements from $z_t$ to $z_i$. For practical reason, the above prediction and update stages are simplified under several assumptions. First, the measurements at a given time are conditionally independent of measurements taken at other times, i.e., the measurements at time $t$ are independent to the other measurements at times $\leq t-1$, and they depend only on the current states of objects via $x_i$ and not on its entire hidden state sequence, thus the measurement likelihood $p(z_t | x_{t|t}, z_{1:t-1})$ is simplified to be $p(z_t | x_t)$. Second, based on real world system, it is assumed that the system follow first-order Markov process where the present hidden state dependsonly on the last state, that is $p(x_{t} | x_{t-1}) = p(x_t | x_{t-1})$.

The main purpose in object tracking is to estimate the distribution $p(x_t | z_{1:t})$. This knowledge can be derived from $p(x_{t|t} | z_{1:t})$ in equation (3) by integrating the distribution over all previous hidden states. After the integration, the prediction and update stages in equations (2) and (3) become [28]:

Prediction:

$$p(x_t | z_{1:t-1}) = \int p(x_t | x_{t-1}) p(x_{t-1} | z_{1:t-1}) \ dx_{t-1}$$  \hspace{1cm} (4)
Update:

\[ p(x_t \mid z_{1:t}) = \frac{p(z_t \mid x_t) \ p(x_t \mid z_{1:t-1})}{p(z_t \mid z_{1:t-1})} \]  

(5)

The denominator in equation (5) \( p(z_t \mid z_{1:t-1}) \) is the normalization factor that ensures that the probability distribution \( p(x_t \mid z_{1:t}) \) sums up to 1, in order to satisfy the axioms of probability. The integral in equation (4) is called as the Chapman-Kolmogorov equation. The solution of the integral gives the predicted state \( p(x_t \mid z_{1:t-1}) \). After receiving the measurement \( z_t \) at time \( t \), the predicted state is corrected by likelihood factor \( p(z_t \mid x_t) \) and finally renormalized by \( p(z_t \mid z_{1:t-1}) \).

The graphical model of Bayesian model of tracking problem can be seen in Figure 6.

![Figure 6: Bayesian model of tracking problem](image)

Solving the recursive Bayesian solution in equations (4) and (5) is at the core of solving object tracking problems. The main difficulty is how to calculate the integration in the Chapman-Kolmogorov equation. There are two mainstream approaches for solving the integral. First, the involved distributions in the equation (4) are assumed as Gaussian. This approach will lead to renown algorithm called as Kalman filter. Second, the integration in the Chapman-Kolmogorov equation is solved numerically using Monte Carlo method. This numerical approach is called as Particle filter, which gives discrete approximation to the exact model posterior \( p(x_t \mid z_{1:t}) \), rather than the optimal solution to an approximate model like Kalman filter. The Particle filter will explain detailed in the next subsection.

B. Particle Filter as Numerical Implementation
Particle filter was devised to numerically implement the recursive Bayesian solution in equations (4) and (5) which approximates the posterior distribution \( p(x_i | z_{1:t}) \) using a finite set of weighted samples or particles. It has been used by many researchers to solve nonlinear and non-Gaussian system in Bayesian estimation.

The fundamental idea behind the particle filter is Monte Carlo simulation [29], in which the posterior density is approximated by a set of particles with associated weights. Particle filter are most frequently formulated as sequential importance sampling (SIS) methods. In SIS method, we first define important density \( q(x_t | x_{t-1}, z_t) \). The importance density can be thought as scaled version of \( p(x_t | z_{1:t}) \) with a different scaling factor at each \( x_t \). However the importance density should be a known and easily sampled probability distribution function. Then, SIS involves drawing particles from an importance density, such that particles of the state \( x_t \) are obtained by predicting particles in time \( t \) from particles in \( t-1 \) and current time measurement \( z_t \).

![Dynamic Bayesian Network Graph of Particle Filter](image)

The graphical model of particle filter can be seen in Figure 7. Given \( n \) particles \( x_i^1, ..., x_i^n \), and their weights \( w_{i1}^1, ..., w_{in}^n \), it can be calculated weights representing the posterior distribution at time \( t \) by:

\[
w_i^{(t)} = w_i^{(t-1)} \frac{p(x_i^{(t)} | x_{i-1}^{(t-1)})}{q(x_i^{(t)} | x_{i-1}^{(t-1)}, z_t)} p(z_t | x_i^{(t)}) \quad i = 1...n
\]

And the posterior distribution is:

\[
p(x_t | z_{1:t}) = \sum_{i=1}^{n} w_i^{(t)} \delta(x_t - x_i^{(t)})
\]

There are several practical problems in the SIS particle filter procedure. First, there is resampling process based on particle weight in equation (6) to generate a new particles set. The resample step
will lower the particle number with low weight and expand the number of high weight particles. Finally, it is required to determine the likelihood function \( p(z_t \mid x_t) \), transition density \( p(x_t \mid x_{t-1}) \) and importance density \( q(x_t \mid x_{t-1}, z_t) \). A suitable importance density is usually identified based on analytical approach which considering trade-off between accuracy and computation load. There are three main type of particle filters based on their importance density: bootstrap, optimal and auxiliary[29].

**B.1 Bootstrap Particle Filter**

The easiest particle filter to implement is the bootstrap particle filter (BPF). In the BPF, the transition density is selected as the importance density, that is:

\[
q(x_t \mid x_{t-1}, z_t) = p(x_t \mid x_{t-1})
\] (8)

For this choice of importance density, the weight update equation becomes:

\[
\tilde{w}_t^{(i)} = w_{t-1}^{(i)} p(z_t \mid x_t^{(i)})
\] (9)

The BPF has the distinctive feature that the incremental weights do not depend on the past trajectory of the particles but only on the conditional likelihood of the observation \( p(z_t \mid x_t) \). For the BPF, sampling is very straightforward with the state transition equation used to predict new particles and is followed by the resample and move steps. BPF has been employed by many researchers in visual object tracking and it is called as condensation algorithm (Conditional Density Propagation)[30]. We have implemented this particle filter approach, and all experiment results in next section based on this approach.

**B.2 Optimal Particle Filter**

It must be considered that for the BPF algorithm, the importance density does not take into account the current measurement \( z_t \). Although BPF is easy to implement, it has some weakness related to its robustness in facing unpredictable noises. To overcome the problems of the BPF, we need to choose the importance density more careful. The optimal choice for an importance density is to choose one that minimizes the variance of the weights. Optimal particle filter (OPF) can be achieved by selecting important density as:

\[
q(x_t \mid x_{t-1}, z_t) = p(x_t \mid x_{t-1}^{(i)}, z_t)
\] (10)

Thus, the importance weight update becomes:
However, OPF suffers from two main drawbacks. First, we must be able to make sampling from important function, given current measurement \( p(x_t \mid x_{t-1}, z_t) \). Second, we have to calculate predictive likelihood \( p(z_t \mid x_{t-1}) \). It means we must be able to predict current measurement based on last hidden states. Nevertheless, there are several appealing approaches where OPF can be used. The first approach is multiple model particle filters (MMPF)[31], which based on the assumption that hidden state \( x_t \) is a member of a finite set. Initially, implementation of MMPF is mainly in object estimation based on radar applications, but recently there is implementation in visual tracking for bicycles [32]. The second approach is based on the assumption \( p(x_t \mid x_{t-1}, z_t) \) is approximated by Gaussian distribution. In [33], optimal importance function is obtained explicitly by approximating using local linearization of the measurement function based on Taylor expansion.

**B.3 Auxiliary Particle Filter**

The auxiliary particle filter (APF) is a particle filter that augment the last hidden state \( x_{t-1} \) with the auxiliary variable \( \xi \) in order to increase accuracy of predictive likelihood \( p(z_t \mid x_{t-1}) \). APF also attempts to imitate the optimal particle filter by performing the resampling step at time \( t-1 \) using current observation. In APF, important density used to draw samples is chosen as factorized form:

\[
q(x_t, \xi \mid z_t) \propto q(\xi \mid z_t) q(x_t \mid \xi, z_t)
\]

where:

\[
q(\xi \mid z_t) = w_{n-1}^{(i)} p(z_t \mid \mu_t)
\]

\[
q(x_t \mid \xi, z_t) = p(x_t \mid x_{t-1})
\]

while \( \mu_t \) is expectation value associated with \( p(x_t \mid x_{t-1}) \).

Then the importance weight update becomes become:

\[
\tilde{w}_t^{(i)} = \frac{p(z_t \mid x_t^{(i)})}{w_t^{(i)}}
\]

Since the publication of Liu-West filter[34], APF is broadly used to estimate parameters in econometrics. Kwon et al [35] is used the Liu-West filter in visual object tracking by assigning the dynamic parameters as auxiliary variables. Based on our experiment, APF is not too suitable
for visual object tracking. Because the dynamic parameters frequently change abruptly, thus they
can not play a role as auxiliary variable to increase accuracy of predictive likelihood.

C. Deep Learning as Observation Model

Deep learning is computational implementation of theories of brain development about
neocortical development, proposed by cognitive neuroscientists. Deep learning can be seen as
extension of neural networks, based on the understanding that the neocortex of the brain is a
hierarchy of filters where each layer captures some of the information[36]. The result of the
multilayer process is a self-organizing stack of raw data well-tuned to their environment.

Deep learning hypothesizes that hierarchy of intermediate representations are needed in learning
high-level representations of data. In the visual case, the first level of representation could be
Gabor-like filters, the second level could be line and corner detectors, and higher level
representations could be objects and concepts [36]. Recently, deep learning architectures have
been used successfully to solve several complicated tasks, such as image classification[37].
The goal of observation model is to create a representation of “thing” that we are tracking. Based
on our study, deep learning is a prominent candidate based on recent results of machine learning.
Thus we accommodate deep learning techniques for creating effective object representation
which can extract image features automatically. In our proposed solution, we will use deep
learning [9] to handle its appearance problems in visual object tracking.

Furthermore based on deep learning techniques, the developed observation model is divided into
two steps: offline training step and online tracking step. The offline training stage is carried out
by training one kind of deep learning, that is stacked denoising autoencoder (SDAE) with
auxiliary image data [38]. During the online tracking process, an additional classification layer
based on Extreme Learning Machine (ELM) is added to the encoder part of SDAE.

Given a set of observed images $I = \{I_t, ..., I^n\}$. If the state variable $\bar{x}_t$ describes the location of the
target object at time $t$, then observation model is aimed to estimate the value of the state variable
$\bar{x}_t$ based on observed images. The aim of observation model is to measure the observation
likelihood of each particle. As stated above, the observation model consists of offline training
step and online tracking step, described in next subsection.
C.1 Offline Training Step

For offline training, it is used the tiny images dataset\cite{39} as auxiliary data. The dataset covers many of the objects and scenes found in the real world which collected from the web using seven search engines. In our research, from 80 million tiny images each of size 32×32, it is sampled 1 million images randomly and converted to grayscale. Consequently, each image will be represented by a vector of 1024 dimensions corresponding to 1024 pixels. The feature value of each dimension is then scaled linearly to the range \([0, 1]\).

The first learning process of SDAE is to recover a corrupted version data sample by introducing several types of noises. Thus in first layer of SDAE contains a hidden layer with more neurons than the input neurons, in order to create robust features. The subsequent layers have fewer neurons than the input’s for building filter hierarchy, which each layer extracts some of specific features. The whole structure of the SDAE is depicted in Figure 8.

![Figure 8: Stacked denoising autoencoder (SDAE)\cite{38}](image)

C.2 Online Tracking Step

It is clear that the learning speed of feed-forward neural networks is in general far slower than required in visual object tracking \cite{40}. The key reasons of these characteristics are: (1) neural networks employ the slow gradient-based learning algorithms, and (2) all parameters of the networks are tuned iteratively using gradient-based algorithms. Therefore, we devise for our tracker to use extreme learning machine (ELM) in the online tracking step. ELM can learn many
times faster than conventional feed-forward neural networks[40]. ELM will be added to the encoder part of the trained in offline step as he additional classification layer. 

The main different of our implementation comparing to other generic ELMs is in the input layer. In our tracker, it is utilized in the end result of offline training step. On the other hand, the other ELMs use a set of random number for input layer. The overall network architecture is shown in Figure 9.

![Figure 9: Network architecture for online tracking](image)

Conventional ELM tends to generate over-fitting model and its performance is deteriorated when outliers present in the dataset. The problems can be solved using regularization techniques and thus make ELM more reliable. For the case where the number of training samples is enormous, Huang et al [41] gives the alternative regularization solution, that is:

$$\beta = \left( \frac{1}{c} + H^T H \right)^{-1} H^T T$$

(14)

where $\beta$ is neuron weight in additional classification layer, $I$ is identity matrix, $H$ is the end result matrix of offline training, and $T$ is target value (positive or negative value). In addition, coefficient $c$ is regularization factor and is set $1e6$ in the research.

D. Geometric Computing in Transition Model

Over the past decade, there has been a big development in the understanding of the geometry in computer vision. The theory and implementation have now achieved maturity level where excellent results can be reached for unsolved problems in a decade ago. In visual object tracking, it is well-known that the evolution of hidden states lives certain transformation space, which is not a vector space. The transformation space is a curved space possessing interesting structure as
a Lie group. We will exploit this curved nature of the state space using geometric computing theory, such as Lie groups, Lie algebras and geometric algebra.

The common approach to model the evolution of state variables is by choosing a set of local coordinates and applying existing vector space methods. The approach frequently produces results that depend on the chosen local coordinates. The performance of such local coordinate-based approaches actually depends on whether the underlying geometry of the transformation space is taken into account precisely. However, this performance deterioration can be seen only when we work around the extremes and unusual cases. Our interest in motorcycle tracking gives motivation to utilize the geometric approaches, since motorcycle dynamic motions have many unpredictable maneuvers.

Given these considerations, we initially focus to regard visual object tracking as Bayesian filtering problem on 2D affine transformation space $\text{Aff}(2)$. The approach that we adopt for our proposed solution is realized by particle filtering of state equation on Lie group $\text{Aff}(2)$. The state equation on Lie group can be thought geometrically well defined comparing its version on vector space.

**D.1 Particle Filtering on the Affine Group**

Given object template coordinate point $p=(p_x, p_y)^T$. The 2D affine transformation of the object coordinates is implemented by multiplying its homogeneous coordinates $\tilde{p}=(p_x, p_y, 1)^T$ with transformation matrix $\begin{bmatrix} G & t \\ 0 & 1 \end{bmatrix}$, where $G$ is an invertible $2 \times 2$ real matrix and $t$ is a $\mathbb{R}^2$ translation vector. This matrix possesses interesting structure as a Lie group and is called as 2D affine group $\text{Aff}(2)$.

The power of Lie group structures comes from combining the algebraic and topological properties through differentiability requirements. The definition of Lie group $G$ is a group which is a differentiable manifold with smooth product and inverse group operations. And Lie algebra $g$ associated with Lie group $G$ is defined as a tangent vector space at the identity element of $G$. A Lie group $G$ and its Lie algebra $g$ can be related via the exponential map $\exp: g \rightarrow G$ and its inverse, $\text{Log}: G \rightarrow g$ (see Figure 10). The 2D affine group $\text{Aff}(2)$, is associated with its Lie algebra $\text{aff}(2)$.
represented as \[ \begin{bmatrix} U & v \\ 0 & 0 \end{bmatrix} \], where \( U \) is 2\times2 real matrix and \( v \) is a \( \mathbb{R}^2 \). A detailed explanation of Lie groups and Lie algebras can be found in [42].

![Figure 10: Relation of Lie group and Lie algebra](Source: opticalengineering.spiedigitallibrary.org)

Dynamic state estimation problem represented equation (1) consists of: state equation and measurement equation. The state equation on the affine group \( \text{Aff}(2) \) for a left-invariant system can be written as:

\[
\frac{dX}{dt} = X.A(X) \ dt + X \sum_{i=1}^{6} b_i(X) E_i \ dw_i
\]

(15)

where \( X \in \text{Aff}(2) \) is the state, the maps \( A: \text{Aff}(2) \to \mathbb{R}^2 \) and \( b_i: \text{Aff}(2) \to \mathbb{R} \) are possibly nonlinear, \( dw_i \in \mathbb{R} \) denote the Wiener process noise, and \( E_i \) are the basis elements of \( \mathbb{R}^2 \) represented by the following matrixes:

\[
\begin{align*}
E_1 &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}; & \\
E_2 &= \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}; & \\
E_3 &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}; & \\
E_4 &= \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}; & \\
E_5 &= \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}; & \\
E_6 &= \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}
\end{align*}
\]

(16)

Each geometric transformation mode corresponds to each \( E_i \) is shown in Figure 11.
The continuous state equation on the affine group (15) is usually discretized using the first order exponential Euler discretization as following:

\[ X_k = X_{k-1}. \exp \left( A(X, t) \Delta t + dW_k \sqrt{\Delta t} \right) \]  

(17)

where \( dW_k \) represents the Wiener process noise on \( \text{aff}(2) \) with a covariance \( P \in \mathbb{R}^{6 \times 6} \), that is:

\[ dW_k = \sum_{i=1}^6 \epsilon_{i,k} E_i \]

with \( \epsilon_k = (\epsilon_{x,1}, \ldots, \epsilon_{x,6}) \) are a six dimensional Gaussian noise sampled from \( N(0, P) \). Then the measurement equation on the affine group can also be represented in the discrete term as:

\[ y_i = g(X_k) + n_k \]  

(18)

while \( g: X_k \rightarrow \mathbb{R}^{N_y} \) is a nonlinear function and \( n_k \) is a Gaussian noise with a covariance \( R \in \mathbb{N}_{x} \times \mathbb{N}_{y} \) that is \( n_k \sim N(0, R) \).

It is performed manual initialization to the initial object template by annotating object locations with bounding boxes. If \( X_k \in \text{Aff}(2) \) represents the 2D affine transformation of the object template in the \( k^{th} \) frame with respect to initial object template, then our task is to estimate \( X_k \) in certain discrete time interval. Such a visual tracking task can be done by applying particle filter procedure on \( \text{Aff}(2) \).

The term \( A(X, t) \in \text{aff}(2) \) in (17) is the state dynamics on Lie group which determines the particle propagation. The simplest and common choice for the state dynamics is a random walk model, that is: \( A(X, t) = 0 \). This choice obviously does not exploit the benefit of geometric approach. Though using geometric approach, we have taken into account curved nature of the state space. This means all related motion parameter is now easier to manage and then more accurate dynamic
motion model can be built. Furthermore, a more effective way to enhance tracking performance is to build a more appropriate state dynamics model.

In our initial experiment, we create the state dynamics model by using the first-order autoregressive (AR) process on \( \text{Aff}(2) \). The state equation with the state dynamics based on the AR process on \( \text{Aff}(2) \) can be represented as:

\[
X_k = X_{k-1} \exp \left( A_{k-1} \Delta t + dW_k \sqrt{\Delta t} \right)
\]

\[
A_{k-1} = a \log \left( X_{k-2} \omega X_{k-1} \right)
\]

(19)

where \( a \) is the AR process parameter. This AR-based state dynamics model can be understood as an infinitesimal constant velocity model. Based our experiment, it can be seen better performance in tracking by employing this AR model.

We now write our proposed visual tracking algorithm described so far as following:

### Table 2: Geometric Deep Particle Filter (GDPF) Algorithm

<table>
<thead>
<tr>
<th>A. Initialization</th>
<th>For i=1, ..., N, set ( X_i^{(0)} = I ) and ( A_i^{(0)} = 0 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Set time ( k = 0 )</td>
<td></td>
</tr>
<tr>
<td>2. Set number of particles as ( N )</td>
<td></td>
</tr>
<tr>
<td>3. Set the initial state ( X ) and dynamics model ( A(X) )</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>B. Sequential importance sampling</th>
<th>For every particle ( i=1, ..., N )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Set ( k = k + 1 )</td>
<td></td>
</tr>
</tbody>
</table>
| 2. Draw six dimensional Gaussian noise \( \epsilon_k = (\epsilon_{k,1}, ..., \epsilon_{k,6}) \) | \( \epsilon_k \sim N(0, P) \)
| 3. Draw Wiener process noise \( dW_k \) | \( P : 6 \times 6 \) cov matrix
| 4. Draw new particles using geometric computing | \( dW_k = \sum_{i=1}^{6} \epsilon_{k, i} E_i \)
| 5. Generate unnormalized importance weights using deep learning | \( X_k = X_{k-1} \exp \left( A_{k-1} \Delta t + dW_k \sqrt{\Delta t} \right) \)
| 6. Normalize the weights | \( A_{k-1} = a \log \left( A_{k-2} A_{k-1} \right) \)
| \( \tilde{w}_k^{(i)} = w_k^{(i)} / \sum \tilde{w}_k^{(i)} \) | \( \tilde{w}_k^{(i)} \equiv w_k^{(i)} p(z_k | x_k^{(i)}) \)

<table>
<thead>
<tr>
<th>C. Resampling</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1. For ( i=1, ..., N ), resample from ( X_k^{(i)} ) according to normalized weight ( w_k^{(i)} )</td>
<td></td>
</tr>
<tr>
<td>2. Time step and initialize weight ( X_k^{(i)} \rightarrow X_{k-1}^{(i)} ) ( w_k^{(i)} = \frac{1}{n} )</td>
<td></td>
</tr>
</tbody>
</table>

| D. Go to the importance sampling step (B) | |
IV. EXPERIMENTAL RESULTS

In order to evaluate the proposed solution, called Geometric Deep Particle Filter (GDPF), it is done the experiments using several video dataset in order to track the target. The experiments are implemented on Intel i3 2.53 [GHz] CPU (without GPU) and 2 [GB] RAM. The experiments are done using 1000 particles and facing to various challenges in ten video sequences that is illumination variation, partial or fully occlusions, 3D object deformation, fast motion, image blur and out-of-view. The used dataset are: woman [38], car4, davidin [43], person, partialoc, fulloc [44], cube, vase [33], ballocc, motorcycle (own generated video). For initialization, the rectangular box on target object in first frame is chosen carefully to represent all features of the target object. And the diagonal covariance matrix is adjusted with suitable values based on target object dynamics. Finally, we empirically compare our proposed solution with Incremental Visual Tracker (IVT) [43] and Deep Learning Tracker (DLT) [38] for tracking performance. IVT, which based on principal component analysis (PCA), is regarded as representation of common visual trackers that based in specific feature.

A. Accuracy

Except motorcycle video dataset, all video sequences can be track well by GDPF. Thus, the tracking problem of GDPF tracker is related to 3D object deformations which appearing in motorcycle dataset. For DLT, there are three dataset cannot track well, that is: cube, vase and of course motorcycle. DLT just implements Brownian motion for state dynamics model. Thus it is failed to track when facing fast and unpredictable motions. This is the main weakness of DLT comparing to GDPF. And it explains why DLT cannot track cube and vase dataset well. Both trackers have confidence threshold value. And if the maximum confidence of all particles in one frame is below the threshold, the whole deep learning will be tuned again.

On the other hand, the tracking performance of IVT is poor. It cannot track all frames in video sequences well. Based on experiment results, it can be seen IVT only suitable for certain target objects. The IVT suitable characteristics of target object are slow motion (in partialoc) and relative permanent like face (in davidin). In order to compare the accuracy of trackers, the tracking performance of each sequence is listed in Table 3.
Table 3: Comparison of accuracy on 10 video sequences

<table>
<thead>
<tr>
<th>DATA SET</th>
<th>TRACKING PERFORMANCE</th>
<th>#FRAME</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>DLT</td>
<td>IVT</td>
</tr>
<tr>
<td>WOMAN</td>
<td>✓</td>
<td>x</td>
</tr>
<tr>
<td>CAR4</td>
<td>✓</td>
<td>x</td>
</tr>
<tr>
<td>DAVIDIN</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>PERSON</td>
<td>✓</td>
<td>x</td>
</tr>
<tr>
<td>PARTIALOCC</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>FULLOCC</td>
<td>✓</td>
<td>x</td>
</tr>
<tr>
<td>BALLOCC</td>
<td>✓</td>
<td>x</td>
</tr>
<tr>
<td>CUBE</td>
<td>✓</td>
<td>x</td>
</tr>
<tr>
<td>VASE</td>
<td>✓</td>
<td>x</td>
</tr>
<tr>
<td>MOTORCYCLE</td>
<td>x</td>
<td>x</td>
</tr>
</tbody>
</table>

Figure 12 illustrates the performance of GDPF in facing partial occlusion (frame #84 – frame #180) using woman dataset. As shown in that sequence of figures, GDPF can overcome this partial occlusion challenge.

Figure 13 shows the performance of GDPF in handling illumination variation and soft 3D object deformation (frame #405 – frame #500) using davidin dataset. As shown in that figure, GDPF can overcome this object deformation challenge.
Figure 13: Illumination and soft 3D object deformation in davidin dataset[43]

Figure 14 shows the performance of GDPF in handling full occlusion (frame #126 – frame #200) using fullocc dataset. As shown in that figure, GDPF can overcome this full occlusion challenge.

Figure 14: Full occlusion in fullocc dataset[44]

Furthermore we would like to show the failed tracking of DLT in vase dataset due to fast and unpredictable motions. This sequence of images can give an insight why one tracker cannot track well due to not implementing state dynamics model properly. Figure 15 shows the performance of DLT in handling fast motions (frame #20 – frame #90) using vase dataset. As shown in that figure, DLT cannot pass the fast motions challenge.
Finally we also want to show the weakness of proposed solution GDPF tracker in facing hard 3D object deformation using motorcycle dataset. We plan to overcome this weakness by employing pose estimation via geometric algebra in future development. Figure 16 shows the performance of GDPF in handling hard 3D deformation (frame #60 – frame #200) using motorcycle dataset. As shown in that figure, GDPF cannot pass the hard 3D deformation challenge.
V. FUTURE DEVELOPMENT

Based our experiment results, we still face two main problems, one caused by the appearance of the video data (e.g., occlusion, 3D deformation) and another due to dynamic movement of the motorcycle (e.g., maneuvers, rapid movement). In this section, we would like to propose some possibilities to improve the tracker in order to solve those problems.

A. Interacting Multiple Model

In the proposed solution section, it has been described about optimal particle filter (OPF). The most promising of OPF implementation is multiple model particle filter (MMPF). We wish to implement MMPF using geometric approach to increase tracking performance in facing hard maneuvering object.

The Interacting Multiple Model (IMM) [45] is one type of adaptive estimators which is based on the assumption that a finite number of models are required to characterize the target motion at all times. The IMM characterizes a target as behaving according to one of M modes, for example: stopped, moving with constant velocity, or accelerating. Each mode has an associated probability. Switching rates between modes are assigned a priori. When new data comes in, mode probabilities adjust based on agreement with measurements. The goal of IMM is to correctly estimate mode probabilities to minimize tracking error.

IMM actually is an estimator used in radar tracking, as explained by Bar Shalom et al [45]. In general, the IMM algorithm consists of a set of mode filter modules which work together to deliver the optimal state estimation. The individual mode filters commonly are Kalman Filters (KF) or Extended Kalman Filters (EKF). Recently, particle filter (PF) has been introduced. Particle filter, by using many random samples to directly approximate the probability density function of a state distribution, can deal with any nonlinearity in the dynamics and measurements. Furthermore, the assumption that the noises have to be Gaussian can be neglected. Combining particle filter with the IMM approach, as introduced by Ristic et al [31] is a promising approach. The combination is called as Multiple Model Particle Filter (MMPF).

Techniques for tracking maneuvering targets have been used in many tracking and surveillance systems like in motorcycle tracking. In particular, tracking a maneuvering target using multiple models can provide better performance than using a single model. Many variant of multiple
model techniques to track a maneuvering target have been proposed in the literature [46]. However, the common implementation of multiple model techniques is interacting multiple model (IMM) categorized as second generation multiple model. The first generation is characterized by the fact that each of its elemental filters operates individually and independently of all elemental filters. Its advantage over many non-MM approaches stems from its superior output processing of results from elemental filters to generate the overall estimate. The second generation inherits the first generation’s superior output processing, but its elemental filters work together as a team via effective internal cooperation, rather than work independently as in the first generation. The architecture of the IMM algorithm with three models is illustrated in Figure 17.

![Architecture of IMM estimation algorithm with three models](image)

Figure 17: Architecture of IMM estimation algorithm with three models[46].

A.1 Driving patterns

To create individual mode filters in IMM, we have to consider driving pattern of target object. The multi-model structure in ground target tracking is a result of driving patterns, varying terrain conditions, road segment constraints, and road intersections [47]. In Figure 18, depicts the various driving patterns of a vehicle: straight line and curve, cut-in/out, u-turn, and interchange[48]. All of these patterns can be designed by a combination of a constant velocity rectilinear motion, a constant acceleration rectilinear motion, a constant angular velocity curvilinear motion, and a constant angular acceleration curvilinear motion. These typical driving patterns are described briefly as follows:

a) Straight line and curve: In this situation, the vehicle follows straight lines and curves on a curved road.
b) Cut-in/out: This situation indicates a maneuvering vehicle cuts in (or out) to (or from) the lane. In this situation, three surrounding vehicles are assumed: one in front, one to the left, and one to the right. In this case, the target vehicle changes its motion from a rectilinear motion to a curvilinear motion and then back to a rectilinear motion.

c) U-turn: This situation occurs when the target vehicle changes its driving direction by 180°.

d) Interchange: In this situation the target vehicle undergoes a 3-dimensional motion. The target vehicle moves rectilinearly, then drives a uniform circular turning of up to 270° with a constant yaw rate, and finally converts to a rectilinear motion.

![Diagram of various driving patterns](image)

Figure 18: Various driving patterns of vehicles[48].

**B. Pose Estimation via Geometric Algebra**

As stated in experiment results, our proposed tracker does not overcome hard 3D object deformation in motorcycle dataset. One solution to solve this problem is extending state dynamics with orientations. The combination of position and orientation is referred to as the pose of an object. The specific task of determining the pose of an object in an image (or stereo images, image sequence) is referred to as pose estimation. This information can then be used, for example, to allow a robot to manipulate an object or to avoid moving into the object. The pose estimation problem can be solved in different ways depending on the image sensor configuration, and choice of methodology.

If a calibrated camera is known, then a mapping from 3D points in the scene and 2D points in the image. Furthermore while the geometry of the object is known, the projected image of the object
on the image is a well-known function of the object's pose. Once a set of control points on the object, typically corners or other feature points, has been identified it is then possible to solve the pose transformation from a set of equations which relate the 3D coordinates of the points with their 2D image coordinates.

Rosenhahn et al [49] explain 3D pose estimation through geometric algebra. For detailed explanation of geometric algebra, it can be referred Dorst et al [50]. The pose estimation problem is modeled on the base of several geometric constraint equations. In that way, the projective geometric aspect of pose estimation is only implicitly represented. Thus, pose estimation becomes a pure kinematic problem. The dynamic measurements of these constraints are either points or lines. The motor algebra is used to introduce constraint equations, which keep a natural distance measurement.

Rosenhahn et al [51] apply a novel pose estimation algorithm to the tracking problem based on geometric constraint equations model [49]. To estimate the motion in object tracking, it is necessary to establish the correspondences between the image features and the object features. If a correspondence can be assumed, it can be used to estimate the pose and vice versa, the error of the pose can be used to evaluate the correspondences. For tracking, the error measures of the algorithm are used to characterize the quality of an estimated pose. The key idea of the tracking algorithm is random start local search. An interaction between tracking and pose estimation problem can be visualized in:

![Tracking Diagram](image)

**Figure 19:** The interaction of tracking and pose estimation problem [52]

C. Fuzzy Control Based as Attention Switching Model

In proposed solution section, it has been explained about the role of attention in the nonretinotopic integration. The nonretinotopic integration occurs in all human visual processing. However for complex motion structures, the integration depends strongly on attention by
combining different, independent motion trajectories [24]. We plan to implement this feature for increasing tracking effectiveness and thus decreasing computation time. The key point to implement attention switching model is to use a fuzzy controller, which could imitate the thoughts of human beings in some degree. The fuzzy controller can be designed to detect the object behavior by comparing the result of observation model and transition model. The similar idea has been employ for detecting maneuvering object [53].

A fuzzy controller can be regarded as a nonlinear static function that maps controller inputs onto controller outputs (look at Figure 20). A controller is used to control the system. The system has a desired response that must be maintained under whatever inputs are received. The inputs to the system can change the state of the system, which causes a change in response. The task of the controller is then to take corrective action by providing a set of inputs that ensures the desired response.

As described in Figure 20, a fuzzy controller consists of four main components, which are integral to the operation of the controller:

a) Fuzzy rule base: The rule base, or knowledge base, contains the fuzzy rules that represent the knowledge and experience of a human expert about the system. These rules express a nonlinear control strategy for the system.

b) Condition interface (fuzzifier): The fuzzifier receives the actual outputs of the system, and transforms these non-fuzzy values into membership degrees to the corresponding fuzzy sets.
c) Action interface (defuzzifier): The action interface defuzzifies the outcome of the inference engine to produce a non-fuzzy value to represent the actual control function to be applied to the system.

d) Fuzzy controller (inference engine): The inference engine performs inferring upon fuzzified inputs to produce a fuzzy output.

VI. CONCLUSIONS

This paper is presented our proposed tracker which called as Geometric Deep Particle Filter (GDPF). This tracker is inspired by human visual perception which has nonretinotopic nature. A number of experiments have been conducted for this research, and it has been found that GDPF has achieved certain degree of success in object tracking. There are three main conclusions:

a. Dynamic model is crucial for object tracking. Then we would like to extend our approach using geometric computing for modeling state dynamics to multiple-model particle filter (MMPF). The goal of enhanced algorithm is to be able track hard maneuvering object like in motorcycle tracking.

b. Object orientation is important information for object tracking. State dynamics should take into account the orientation, beside only position.

c. Attention switching model should be integrated in GDPF, in order to increase tracking effectiveness and thus decrease computation time. We plan to implement this idea using fuzzy based controller.

REFERENCES


