Trajectory analysis using Markov chains

Master Thesis

Bas van de Kerkhof

November 26, 2014

Supervisors
dr. ir. M. Podt
dr. ir. H. Driessen
prof. dr. A.A. Stoorvogel
dr. P.K. Mandal

Assessment committee
prof. dr. A.A. Stoorvogel
dr. P.K. Mandal
dr. J.C.W. van Ommeren
dr. ir. M. Podt

UNIVERSITY OF TWENTE.

Hybrid Systems

Enschede

THALES

Sensors Development

System Engineering
Hengelo

Preface

The research presented in this report is carried out at Thales B.V. in Hengelo. This combined internship and final project is part of the master Applied Mathematics and took place between February 2014 and November 2014. During this period I investigated if it was possible to model and classify different types of trajectories. These trajectories belong to different targets which are tracked using a radar application.

I would like to thank Martin Podt, my supervisor at Thales, for the support during my period at Thales. Also thanks to my supervisors from the University of Twente, Anton Stoorvogel and Pranab Mandal. Apart from my direct supervisors I would also like to thank Perry Verveld, Hans Driessen and Linda Schippers for their help and support. Last but certainly not least I would like to thank Peter Vermaas, Sijmen de Bruijn and Tom Sniekers for all the support and friendship during the graduation period.

Bas van de Kerkhof, November 2014.

Abstract

On naval ships all over the world radars are used to detect and track different objects. The environment in which they do this has become much more complex over the last couple of years, e.g. increased population of ships, deception by hostile ships, piracy. So better methods for detecting and tracking different objects are desired and moreover better methods for classification of different targets. One of these methods is trajectory analysis.

The goal of this project was to study the suitability of Markov chains for trajectory analysis. More specifically, the research question was whether modeling of target trajectories using Markov chains can be used to distinguish a number of trajectories that are encountered in real-life situations. We focused on distinguishing a weave, consisting of straight lines and curves, from other trajectories that also consist of straight lines and curves.

The method we developed makes use of a hierarchical state diagram. On the highest level we have all possible trajectories we want to take into consideration. One level deeper we have the possible segments of each trajectory. Transitions between the different states are governed by stochastic Markov processes. We implemented the hierarchical state diagram in a particle filter which we then use to track a single object and classify its trajectory. The dynamical models used in the particle filter to predict the particles depend on the current state of each particle.

First we investigated the performance of our method when simulated measurements created by ourselves were used. Next we used GPS data loggings to see if our method performed well when 'real' data is used. Eventually we converted these GPS data loggings into radar measurements which we used to finetune the different settings of our method.

The output of our method when an object follows a weave trajectory can be distinguished from the output of our method when an object follows a non-weave trajectory, so our method is able to classify the different trajectories. In order to make it plausible that this classification can be done automatically we have modeled a simple detector.

Index terms — *Particle Filtering, Hybrid Systems, Multiple Model Filter, Markov Chains, Target Tracking, Radar Application, Trajectory Analysis*

Contents

1. Introduction

1.1 Background

On naval ships all over the world radars are used to detect and track different objects. The environment in which they do this has become much more complex over the last couple of years, e.g. increased population of ships, deception by hostile ships, piracy. So better methods for detecting and tracking different objects are desired and moreover better methods for classification of different targets.

In some situations simply detecting and tracking an object is not enough. More refined methods are needed to fulfill the needs of the naval ships. A method for classification of different objects uses constraints on the position of the objects to classify these objects. For example a ship can never sail on land, so if we detect an object on land it can not be a ship. This is a refinement of the already used methods. First the objective is to detect and track the object, next we want to know what kind of object we are tracking. Once we have determined, for example, that the object is in fact a ship, it does not tell us all its 'secrets'. We only know that it is a ship but not what kind of ship or what its intentions are etc. So we want an even better refinement of the classification. Therefore the question arises whether or not we can classify objects by looking at their motion patterns.

Different objects have their own characteristics. Not only the objects themselves are different from each other, also their actions and intentions can be different. For instance a fishing boat will systematically sail a pattern such that it makes sure it covers all of the fishing grounds. While a hostile boat performs certain maneuvers to mislead the object who is being attacked. If these patterns can be distinguished from another by the radar, it will significantly help the radar operator in his decision making process, because more detailed information is available.

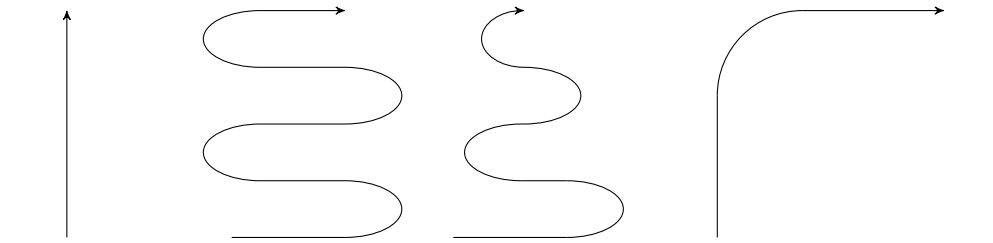
The recognition of different motion patterns, or trajectory analysis, is based on the idea that different object classes or different object intentions lead to different object trajectories. This means that analysis of trajectories could be used to refine object classification or to estimate the intent of an object based on the trajectory of that object, thereby improving situational awareness.

1.2 Problem description

The goal of this project is to study the suitability of Markov chains for trajectory analysis. More specifically, the research question is whether modeling of target trajectories using Markov chains can be used to distinguish a number of trajectories that are encountered in real-life situations. We will focus on refining the classification, that is distinguishing a weave, consisting of straight lines and curves, from other trajectories that also consist of straight lines and curves, see figure 1.1 for this kind of trajectories. This focus on distinguishing a weave comes from the fact that a weave pattern might be performed during an attack, thereby it can indicate suspicious behavior. A complicating factor is that not every weave is the same, which means that the proposed methods need to be robust for variations in trajectories.

During the tracking of the object we at the same time want to classify the trajectory of the object. This classification is needs to be made during the tracking of the object, since for example a ship needs to act fast after a threat is detected. If we first observe and track the object for a certain amount of time and thereafter say: 'The object followed a weave trajectory', we are already too late to take any action if necessary. If we are able to classify the object's trajectory fast and automatically, we support the radar operator in his decision making process. In this last case the radar operator does not have to manually investigate all objects which are being tracked, part of this is done automatically thereby improving situational awareness.

The proposed methods are new in the sense that they focus on classifying a trajectory automatically instead of a post-analysis of the trajectory. Until now the radar operator has to investigate the different tracks on the radar image and classify them separately. This is a complicating task



(a) Straight (b) Weave (c) Weave (d) Straight - curve - straight

Figure 1.1: Different trajectories consisting of straight lines and curves

requiring a high level of concentration from the radar operator. We try to 'outperform' the radar operator by classifying the trajectories automatically and as soon as possible. Thereby supporting the radar operator in his decision making process.

During this research we try to classify different trajectories. Similar research is already performed in different forms for example using the interacting multiple model algorithm or other multiple model algorithms, see [2], [3], [5], [8] and [9]. We try to improve these kind of methods by incorporating the classification of the trajectory into the particle filter. Using this method we try to develop a joint tracking and classification method. We want to use the models of the trajectories in the tracking application of the particle filter. This to reduce the errors of the velocity and position estimates.

Just like interacting multiple model filters, we simultaneously consider different dynamical models for the object's kinematics. But instead of using multiple filters in parallel, which each consider one dynamical model, we implemented all these models into a single particle filter and also incorporated Markov chains which govern the transitions between these models. Thereby making it a joint tracking and classification method.

1.3 Outline

In this report we treat the subject of trajectory analysis. In section 2 we introduce the used models and explain the implementation of Markov chains in a particle filter which is the basis of our method. In section 3 we investigate if our method is viable via the use of simulated radar measurements. Next in section 4 we use GPS data loggings for the finetuning of the settings of our method. In section 5 we continue with the finetuning of the settings of our method but now using radar measurements. In section 6 we develop a detector which uses the output of our method to classify if different trajectories are a weave trajectory or a non-weave trajectory. Section 7 provides an introduction on other methods for trajectory analysis. Eventually in section 8 we summarize and discuss the main results and look at the advantages and disadvantages of the different methods for trajectory analysis. A list of abbreviations and symbols is included on page 49. In the appendices A to D some extra results of the corresponding sections 3 to 6 are given.

2. Models

In this section we introduce the used models. Some theory about the models and implementation issues are given, eventually we further explain our approach to our research.

Our goal is to track an object and classify its trajectory simultaneously. We use a particle filter to estimate the object's state. A particle filter is a numerical approximation to the nonlinear Bayesian filtering problem. The particle filter in the current form became widely know due to the introduction of the resampling step in 1993, [7]. Until then the primary approach to solve the Bayesian filtering problem was the use of Kalman filters. The drawback of Kalman filters is that they can not handle nonlinearities. The systems we encounter during this research are nonlinear, therefore we use a particle filter. In this chapter we introduce the methods and models which we use in the particle filter algorithm.

2.1 Hybrid State

During our research we have a system which has a continuous state(vector), s_k, and the discrete state variables mode, m_k, and submode, sm_k. These last two variables are further explained in sections 2.3.1 and 2.3.2. All these variables together form the hybrid state of the system. Hybrid in the sense that the state vector contains both continuous and discrete variables. The continuous part of the state evolves according to a dynamic model:

$$s_{k+1} = f(s_k, m_k, sm_k, w_k), \tag{2.1}$$

where w_k is the process noise and k the time index. The discrete variables m_k and sm_k evolve according to a Markov process with transition matrices Π_m and Π_{sm_k} respectively. Also the measurements at each time step k, z_k, are related to s_k,

$$z_k = h(s_k, v_k), \tag{2.2}$$

with v_k the measurement noise. We assume that the probability density functions of the process noise, measurement noise and initial state, respectively p_{w_k}, p_{v_k} and p_{s_0,m_0,sm_0}, are known.

2.2 Dynamical Models

During our research we use three different dynamical models, each providing the evolution of the dynamics of s_k, the continuous part of the hybrid state of the system. The dynamic models are derived from [1], [10] and [11]. The three different dynamical models we have used are the nearly constant velocity model (NCV), the curvilinear motion model (CM) and the augmented coordinated turn model (ACT).

We use the three dynamical models in our particle filter simultaneously. The NCV-model and CM-model are used when the object follows a straight line. In early research we use the NCV-model, but it turned out that this model was insufficient for our purposes. Therefore in later research we used the CM-model, see section 4.2.2. The ACT-model is used when the object performs a turn.

2.2.1 Nearly Constant Velocity Model

The NCV-model is a discrete time process noise model. We define s_k, the continuous part of the hybrid state vector as follows:

$$s_k = \begin{bmatrix} x & v_x & y & v_y \end{bmatrix}'_k$$

Where x and y are the x- and y-position in m and v_x and v_y are the velocities in x- and y-direction in m/s. The process noise vector w_k is defined as

$$w_k = \begin{bmatrix} a_{x,k} & a_{y,k} \end{bmatrix}' \sim N(0, Q).$$

Furthermore we define

$$F = \begin{bmatrix} 1 & T & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & T \\ 0 & 0 & 0 & 1 \end{bmatrix}, G = \begin{bmatrix} \frac{1}{2}T^2 & 0 \\ T & 0 \\ 0 & \frac{1}{2}T^2 \\ 0 & T \end{bmatrix} \text{ and } Q = \begin{bmatrix} \sigma^2_{a_{x,k}} & 0 \\ 0 & \sigma^2_{a_{y,k}} \end{bmatrix},$$

with T the time step and $\sigma_{a_{x,k}}$ and $\sigma_{a_{y,k}}$ the process noise parameters in x- and y-direction at time k respectively. This leads to the evolution of s_k according to the NCV-model:

$$s_{k+1} = Fs_k + Gw_k.$$

2.2.2 Curvilinear Motion Model

The CM-model is similar to the NCV-model explained in section 2.2.1 except for the process noise covariance matrix Q. We use a rotation matrix to change the direction in which the process noise acts. In figure 2.1 we see a schematic two dimensional drawing to clarify the different directional components.

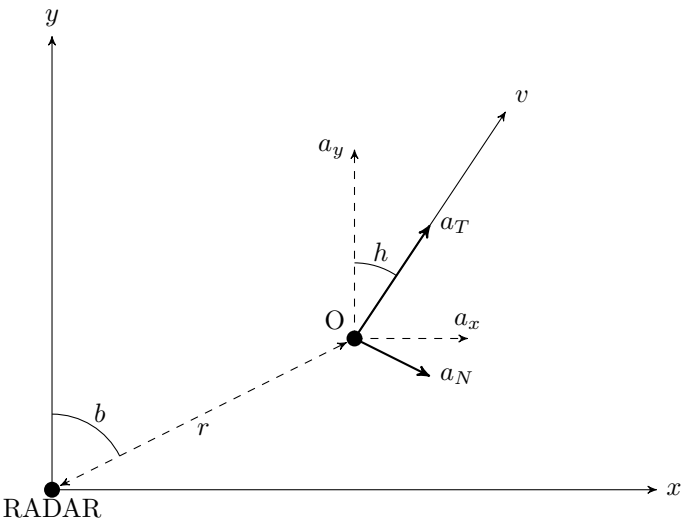


Figure 2.1: Two dimensional schematic representation of the directional components

In figure 2.1 we have the radar positioned at the bottom left corner and O as the object being tracked. This object has a velocity v and possible acceleration terms in x-, y-, tangential or normal direction, a_x, a_y, a_T or a_N respectively. The object has a range r, bearing b and heading h.

The difference between the NCV-model and the CM-model is that the process noise in the NCV-model acts in the x- and y-direction where in the CM-model the process noise acts in the tangential and normal direction. Therefore we define the process noise covariance matrix Q for the CM-model in the following way:

$$Q = RQ_{TN}R',$$

with $Q_{TN} = \begin{bmatrix} \sigma^2_{a_{T,k}} & 0 \\ 0 & \sigma^2_{a_{N,k}} \end{bmatrix}$, $R = \begin{bmatrix} \sin(h) & \cos(h) \\ \cos(h) & -\sin(h) \end{bmatrix}$ and $h = \arctan\left(\frac{v_x}{v_y}\right)$.[1]

We have the process noise parameters $\sigma_{a_{T,k}}$ and $\sigma_{a_{N,k}}$ in tangential and normal direction respectively.

Also note that the output of the CM-model is the same as the output of the NCV-model when $\sigma_x = \sigma_y = \sigma_T = \sigma_N$ holds. The advantage of the CM-model over the NCV-model is that the process noise is now dependent of the heading of the object, making the variance of the acceleration in moving direction controllable. Also we can use the process noise in the normal direction as some measure of heading uncertainty of the object.

[1] In the implementation of this model in Matlab, we use arctan 2 instead of arctan. This because of the fact that elements of arctan 2 lie in the closed interval $[-\pi, \pi]$ and elements of arctan lie in the closed interval $[-\pi/2, \pi/2]$. Since the heading of the object can lie in any quadrant we choose to use arctan 2 instead of arctan.

2.2.3 Augmented Coordinated Turn Model

To model the dynamics of an object while it is performing a turn we use the ACT-model. It is a non-linear model in which we add the angular velocity to s_k:

$$s_k = \begin{bmatrix} x & v_x & y & v_y & \omega \end{bmatrix}'_k,$$

with ω the angular velocity in rad/s. We also extend the process noise vector with a component for the angular velocity:

$$w_k = \begin{bmatrix} a_{x,k} & a_{y,k} & \alpha \end{bmatrix}' \sim N(0, Q).$$

This leads to the ACT-model:

$$s_{k+1} = F(s_k) + Gw_k,$$

in which $F(s_k) = \begin{bmatrix} x + \frac{v_x}{\omega}\sin(\omega T) - \frac{v_y}{\omega}(1-\cos(\omega T)) \\ v_x\cos(\omega T) - v_y\sin(\omega T) \\ y + \frac{v_x}{\omega}(1-\cos(\omega T)) + \frac{v_y}{\omega}\sin(\omega T) \\ v_x\sin(\omega T) + v_y\cos(\omega T) \\ \omega \end{bmatrix}$, $G = \begin{bmatrix} \frac{1}{2}T^2 & 0 & 0 \\ T & 0 & 0 \\ 0 & \frac{1}{2}T^2 & 0 \\ 0 & T & 0 \\ 0 & 0 & 1 \end{bmatrix}$ and $Q = \begin{bmatrix} \sigma_{a,x}^2 & 0 & 0 \\ 0 & \sigma_{a,y}^2 & 0 \\ 0 & 0 & \sigma_\alpha^2 \end{bmatrix}$.

We have the process noise parameters $\sigma_{a,x}$ and $\sigma_{a,y}$ in x- and y-direction and σ_α for the angular velocity component of the process noise.

In the ACT-model we let the process noise act in x- and y-direction since a turn can be entered in any direction independent of the heading of the object. Therefore the process noise should act in a direction independent of the heading of the object, so we choose to let the process noise act in x- and y-direction.

2.3 Markov chains

As said in section 1.2 the goal of this research is to investigate if different trajectories can be classified using Markov chains. A Markov chain is a process in which the state undergoes transitions depending on a transition probability matrix and the current state of the system.

We try to use Markov chains to model the hierarchical structure of the different trajectories and the different segments of each trajectory. Our idea is to use a Markov chain which describes the possible transitions between the different trajectories which are being considered, see figure 2.2. Our focus lies on distinguishing a weave from other trajectories. This weave consists of a certain sequence of straight lines and curves. The transitions between these segments is also modeled using a Markov chain. So this Markov chain is 'embedded' inside the Markov chain which governs the transitions between the different trajectories.

2.3.1 Mode

To identify the different trajectories we want to classify, we introduce the discrete variable mode, m_k. This discrete variable can take on four different values:

Mode, m	Meaning
1	Straight line
2	Left curve
3	Right curve
4	Weave

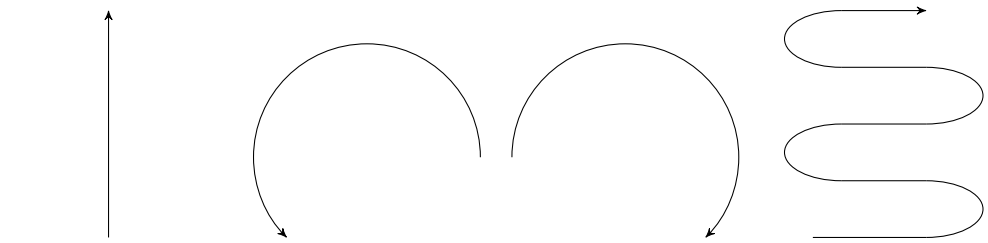
The corresponding trajectories are shown in figure 2.2.

The modes 1, 2 and 3 were chosen because these are the most common (parts of) trajectories any vessel will follow at a point in time. So these three modes cover the most basic trajectories. Because our focus lies on distinguishing a weave we also included the mode weave. This formulation leaves room for extension of the method since we can just add another mode which corresponds to a certain trajectory we also want to be able to distinguish.

The discrete variable mode evolves according to a first-order Markov chain with transition probability matrix Π_m, i.e.,

$$\mathrm{P}(m_{k+1} = j | m_k = i) = \text{i,j-th entry of } \Pi_m.$$

Hence Π_m is stationary over time.



(a) Straight, $m = 1$ (b) Left Curve, $m = 2$ (c) Right Curve, $m = 3$ (d) Weave, $m = 4$

Figure 2.2: Possible values for the discrete variable mode with the corresponding trajectories

The possible transitions are graphically represented in figure 2.3 and the transition probabilities, the entries of Π_m, are determined via simulations.

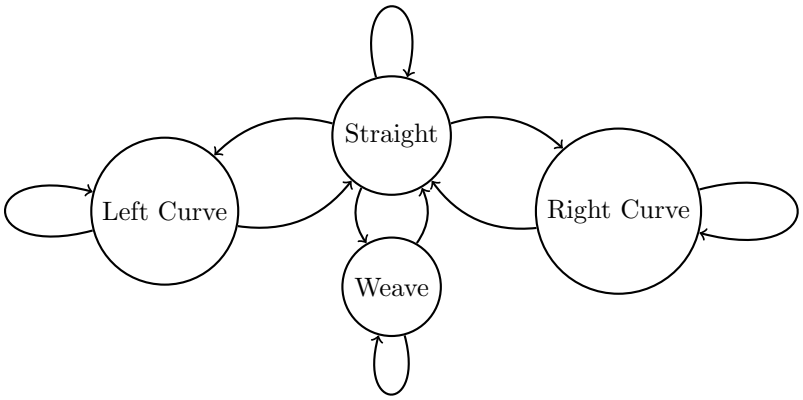


Figure 2.3: Schematic representation of the possible mode transitions

In early research we used different possible transitions, see appendix A. Eventually we found that the transitions shown in figure 2.3 give better results. The chosen transitions enable our method to distinguish a weave trajectory better from a straight-curve-straight trajectory, see section 4.2.3.

2.3.2 Submode

If we look closely at the weave trajectory, we see that it consists of straight lines and curves. It is even possible to segment all trajectories in straight lines and curves like the segmentation of a weave trajectory in figure 2.4.

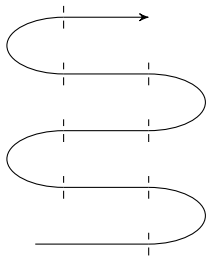


Figure 2.4: Segmentation of a weave trajectory.

By using this type of segmentation we can identify each trajectory by its specific sequence of straight lines and curves. To model this segmentation we use the discrete variable submode, sm_k. This discrete variable submode can take on four values:

Submode, sm_k	Meaning
1	Curve left, CL
2	Straight after curve left, SL
3	Curve right, CR
4	Straight after curve right, SR

Just like the mode, the submode evolves according to a first-order Markov chain but now with transition probability matrix Π_{sm_k}, i.e.,

$$\mathrm{P}(sm_{k+1} = j | sm_k = i) = \text{i,j-th entry of } \Pi_{sm_{k+1}}.$$

Hence Π_{sm_k} depends on the time index k. This is so since the entries of this matrix depend on time. In our research we only used the variable submode when the object is in the mode weave, since the sequence of submodes while in the other modes consist of only one value. The possible transitions of the submode are graphically represented in figure 2.5.

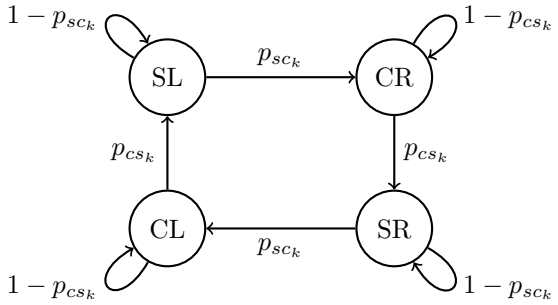
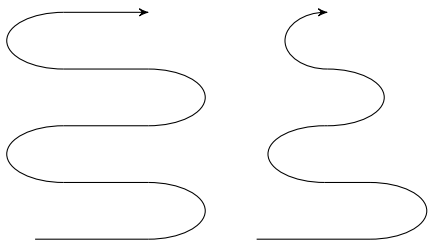


Figure 2.5: Schematic representation of the possible submode transitions

We explicitly modeled two different straight submodes, one after a curve left and one after a curve right. This to cover all possible segments which can occur. Each trajectory now has a unique sequence of submodes. If we would have modeled only one type of straight segment in the particle filter implementation we would always force some particles in a wrong direction. Because then transitions need to be allowed from a straight segment to both a curve left and a curve right. But when we use the two different straight segments we only allow transitions to one specific curve direction. Thereby making it a more distinct sequence to follow and less prone to errors. Especially for a weave trajectory we can now identify this trajectory as having consecutive straight and curve segments in which the curves alternate in direction.

The used modeling of the submodes also leaves room for extension of the method. Suppose we want to be able to distinguish a different trajectory. We can extend the variable mode to five possible values and define a new transition probability matrix Π_m. Next we also define a transition probability matrix for the submode, say Π_{sm2}, while in this new mode. The rest of the method remains the same and hence we have extended our method to also take this new trajectory in consideration while tracking and classifying an object's trajectory.

We aim for a method which is robust for variation in the different trajectories. When we look at weave trajectories, we see that there can be variations of the weave like the two variations in figure 2.6



(a) Weave (b) Weave

Figure 2.6: Two variations of a weave trajectory

Due to these variations, traditional Markov chains with stationary transition probability matrices are not sufficient for reaching our intended goal of a robust method for distinguishing a weave trajectory. Our model needs to be able to cope with the variations of weave trajectories. The variations can be characterized by the length of the straight segments and the angle of the curves. The question arises to what extent we call a trajectory a weave. To answer this question and model the variations we introduce the variables δ_k and θ_k. These variables contain information about the already traveled length of the current straight segment and the already traversed angle of the current curve respectively. We also define δ_{max} and θ_{max} and let these variables characterize the average weave trajectory we expect to encounter. These variables are used in determining the entries of the submode transition probability matrix Π_{sm_k}.

Delta

To model the length of the straight segments of a trajectory we use the variable δ_k which is defined as follows:

$$\delta_{k+1} = \begin{cases} 0 & \text{when beginning a straight segment,} \\ \delta_k + \sqrt{(x_{k+1} - x_k)^2 + (y_{k+1} - y_k)^2} & \text{while already on a straight segment.} \end{cases} \tag{2.3}$$

So δ_k represents the already traveled length of the current straight segment at time step k.

Theta

We model the traveled angle of the curve with the variable θ_k:

$$\theta_{k+1} = \begin{cases} 0 & \text{when entering a curve,} \\ \theta_k + \omega_k T & \text{while already in a curve.} \end{cases} \tag{2.4}$$

So θ_k represents the already traveled angle of the current curve at time step k.

As can be seen in figure 2.5 we have two probabilities, dependent of the time index k, which determine all the entries of Π_{sm_k}, namely the probability of a transition from a curve segment to a straight segment, p_{cs_k}, and the probability of a transition from a straight segment to a curve segment, p_{sc_k}. Since δ_{max} and θ_{max} characterize the average weave trajectory, we use these variables together with δ_k and θ_k for determining p_{sc_k} and p_{cs_k}. We use these variables because we want the probability of making a transition from a curve segment to a straight segment or from a straight segment to a curve segment larger when you stay in that particular submode, since we expect the object to follow the weave trajectory characterized by δ_{max} and θ_{max}.

To determine p_{sc_k} and p_{cs_k} we used two different methods, the min-method and the if-else-method. We chose to use these two methods since we expect them to determine p_{sc_k} and p_{cs_k} such that submode transitions are likely to occur at the same time the weave trajectory characterized by δ_{max} and θ_{max} makes a submode transition. We need to investigate if one method is more favorable over the other.

Min-method

$$p_{sc_k} = \min\left[\left(\frac{\delta_k}{\delta_{max}}\right)^n, 0.95\right],$$
$$p_{cs_k} = \min\left[\left(\frac{\theta_k}{\theta_{max}}\right)^n, 0.95\right].$$

Where n is a positive integer to be determined later on based on simulation outcomes.

If-else-method

$$p_{sc_k} = \begin{cases} p_1 & \text{if } \delta_k < \delta_{max}, \\ p_2 & \text{if } \delta_k \geq \delta_{max}. \end{cases}$$

$$p_{cs_k} = \begin{cases} p_1 & \text{if } \theta_k < \theta_{max}, \\ p_2 & \text{if } \theta_k \geq \theta_{max}. \end{cases}$$

Where p_1 and p_2 are probabilities with p_1 small and p_2 close to unity. These probabilities will be determined precisely using simulations. This if-else-method is much more 'strict' than the min-method in the sense that the transition probabilities are determined such that it is more likely to make submode transitions at the points were the average weave defined by δ_{max} and θ_{max} also makes submode transitions.

2.4 Integrating Markov chains in a particle filter

A particle filter is a numerical approximation to the nonlinear Bayesian filtering problem. A particle filter recursively computes an estimate of the posterior density of the system. This is the probability density function of the state of the system at time k given all measurements up to time k. The transitional density and the likelihood function play a role in this recursive computation, [4]. Since we use different dynamical models simultaneously in our particle filter, these densities depend on the current mode and submode. Therefore we modify a general particle filter by integrating the dependency on the (sub)mode and integrating the corresponding Markov chains.

In particle filtering equation (2.1) is often referred to as the dynamic model and equation (2.2) as the measurement model. These equations and the knowledge about the probability density functions of v_k and w_k define the densities $p(s_k, m_k, sm_k|s_{k-1}, m_{k-1}, sm_{k-1})$ and $p(z_k|s_k, m_k, sm_k)$ which are respectively the transitional density and the likelihood function. We use these densities in the recursive computation of the posterior density $p(s_k, m_k, sm_k|z_{1:k})$ or posterior. This posterior gives us information about the state given all measurement up to time k. It can be computed recursively from $p(s_{k-1}, m_{k-1}, sm_{k-1}|z_{1:k-1})$ in two steps, a prediction step and an update step. In the general form an analytical solution for this recursion does not exists. But a particle filter can numerically approximate the posterior using this recursion.

There are various particle filters which use different distributions to predict the random samples, or particles, which are used to estimate the posterior. The particle filter we use is a modified version of the so called Sampling Importance Resampling filter, or SIR filter. In this filter we predict our particles according to the transitional density which follows from the dynamic model (2.1). The dynamical models we used are explained in section 2.2.

We have modified the SIR filter by integrating the Markov chains described in section 2.3. The prediction of the particles must depend on the current mode and submode of each particle since different dynamical models are used in different (sub)modes. Therefore before we predict the particles, we first predict the new mode and if necessary the new submode of each particle by using the previous (sub)mode of that particle and the transition probability matrices Π_m and Π_{sm_k}. Next the particles can be predicted using the different dynamical models. After this prediction the variables δ_k and θ_k are updated accordingly.

We give the complete algorithm of the used particle filter below.

Particle filter algorithm

Input: $T, N, p_{s_0,m_0,sm_0}, p_{v_k}, p_{w_k}, \Pi_m, \Pi_{sm_k}$.

Initialization

Draw initial particles: $\{s_0^{(i)}, m_0^{(i)}, sm_0^{(i)}\}_{i=1}^N \sim p_{s_0,m_0,sm_0}$.

Set initial weights $q_0^{(i)}$ to $\frac{1}{N}$, $i = 1..N$

Uniformly distribute the particles over m and sm.

For all time steps $k = 1, 2, 3, ..., T$**:**

For all particles $i = 1, 2, ..., N$**:**

Predict $m_k^{(i)}$ and $sm_k^{(i)}$ using Π_m, $m_{k-1}^{(i)}$, $\Pi_{sm)k}$ and $sm_{k-1}^{(i)}$. *Markov chain aspect.*

Generate $w_{k-1}^{(i)} \sim p_{w_{k-1}}$.

Predict the particles: $s_k^{(i)} = f(s_{k-1}^{(i)}, m_k^{(i)}, sm_k^{(i)}, w_{k-1}^{(i)})$.

Update δ_k and θ_k accordingly using (2.3) and (2.4).

Update the weights: $q_k^{(i)} = q_{k-1}^{(i)} p(z_k|s_k^{(i)}, m_k^{(i)}, sm_k^{(i)})$.

end

Normalize the weights: $q_k^{(i)} = \frac{q_k^{(i)}}{\Sigma_{j=1}^N q_k^{(j)}}, i = 1..N$.

Resample
Generate new particles $\{\tilde{s}_k^{(i)}, \tilde{m}_k^{(i)}, \tilde{sm}_k^{(i)}\}_{i=1}^N$, such that:
$P(\{\tilde{s}_k^{(i)}, \tilde{m}_k^{(i)}, \tilde{sm}_k^{(i)}\} = \{s_k^{(j)}, m_k^{(j)}, sm_k^{(j)}\}) = q_k^{(j)}$.
Set the weights $\{\tilde{q}_k^{(i)}\}_{i=1}^N$ to $\frac{1}{N}$.

end

Output: $\{\tilde{s}_k^{(i)}, \tilde{m}_k^{(i)}, \tilde{sm}_k^{(i)}, \tilde{q}_k^{(i)}\}_{i=1}^N$ for all time steps $k = 1, 2, ..., T$.

2.4.1 Resampling

In our algorithm we resample at every time step. The idea of resampling is to duplicate particles which have a high weight and omit particles with a low weight. This technique is commonly used to overcome the problem of degenerating particles. With degenerated particles we mean particles which have a negligible weight, thereby they do not really contribute to the estimate of the state according to the particle filter. So these particles are not 'effective' and hence the effective sample size[2] decreases. Since we resample at every time step we expect no problems with degenerating particles.

On the other hand the choice to resample at every time step could cause some new problems. In our case it could lead to a low diversity of the particles among the different modes. For example when particles are predicted according to a certain mode and the new measurement is close to this prediction. It could be that after resampling we end up with particles only in that specific mode. Thereby the new prediction will mostly be done according to this mode while it could be that the object switches from mode during that time step. So the prediction in this time step could be completely off.

So in our case resampling is a trade off between the prevention of degenerating particles and the diversity of the particles among the different modes. Therefore we need to be aware of the possible occurrence of these problems.

2.4.2 Classification using particle filter output

To eventually classify the different trajectories we use the output of the particle filter, especially the variable mode. The question arises how the particle filter can make a classification using this variable. The idea is that using the variables mode and submode we predict the particles according to the different dynamical models. In the update step we use the new measurement to adjust the weights of the particles. The new measurement will probably lie closer to one of the predictions making the weights of this prediction larger with respect to the other predictions. Then after resampling we are likely to end up with more particles according to the prediction with a larger weight. The particle filter makes a 'choice' on which mode it finds more likely since the measurement fits better with the prediction according to that mode and therefore we can use this output of the particle filter to classify the objects trajectory.

2.5 Approach to our research

In the previous sections we have introduced the different models used in our research. The goal of our research is to develop a method which is able to distinguish different trajectories. From now on when we speak of 'our method' we mean the method developed during this research. Our method should specifically be able to distinguish a weave trajectory. This focus on distinguishing a weave trajectory comes from a naval standpoint. On naval ships radars are used to locate all the objects in the surrounding area. Some hostile ships perform a weave trajectory to mislead the radar operator who is tasked to classify the different trajectories of the radar image. Our method has to be able to automatically perform a fast classification of such a (hostile) weave trajectory, such that the radar operator can make a decision which actions need to be taken. So the classification needs to be done as soon as possible. This classification has to be performed using radar measurements since

[2]The effective sample size is often estimated by $N_{eff} = 1/\sum_{i=1}^{N}(q_k^{(i)})^2$

these measurements are used on the naval ships to locate the different objects in the surrounding area.

First in section 3 we investigate if our method is viable. Our method is tested using simulated radar measurements. These measurements are created by ourselves to see if our method works in the best possible conditions. Next in section 4 we try to find good settings for our method. With good settings we mean settings such that our method is able to distinguish different trajectories correctly without having to adjust these settings per case. So we try to come up with some general settings which lead to a good overall performance of our method without having to adjust these settings per situation. We try to find these settings using GPS measurements. We use GPS measurements in the finetuning of our method since these are much more accurate in comparison to radar measurements. In section 5 we investigate if the proposed general settings still hold when radar measurements are used instead of GPS measurements. Eventually in section 6 we develop a detector which can automatically distinguish a weave trajectory from a non-weave trajectory. Again we try to find general settings for our detector such that our method has a good overall performance when different trajectories are encountered without having to change these settings.

3. Results using simulated radar measurements

In section 2 we introduced the to be used method. To investigate if this method is viable and to get familiar with the different aspects of the method we have tested our method on simulated radar measurements created by ourselves. First we explain how we have created these measurements whereafter we look at the different aspects of the method. Some preliminary results are then given. More results from the investigation of the different aspects of the method are given in appendix A.

The results in this section are merely used to investigate if our method is viable. We will not go into great detail on all the aspects of our method, this will be done in later research. Since our main goal is to implement our method in a radar environment, we simulated radar measurements which we use while investigating the viability of our method.

3.1 Simulated measurements

Since our focus lies on distinguishing a weave trajectory, we have created such a weave trajectory ourselves. This weave trajectory is based on the NCV- and ACT-model. We chose an initial state and let this state evolve according to either the NCV- or the ACT-model in such a way that a weave trajectory was created. We used an angular velocity of $0.1 rad/s$ and set the process noise equal to zero. Next we converted the state into radar data using the following equations:

$$\begin{aligned} r_k &= \sqrt{x_k^2 + y_k^2}, \\ b_k &= \arctan(\frac{x_k}{y_k}), \\ d_k &= -\frac{x_k v_{x,k} + y_k v_{y,k}}{r_k}. \end{aligned} \tag{3.1}$$

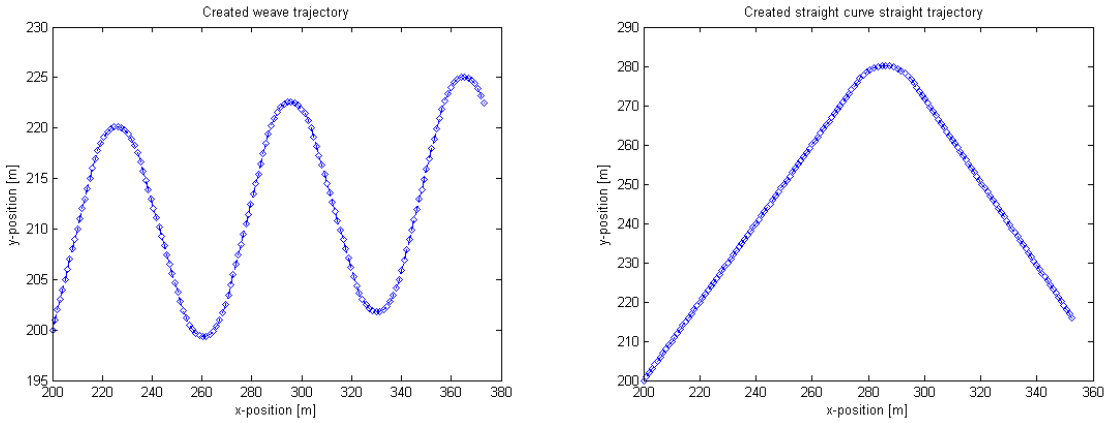
With r_k the range, b_k the bearing and d_k the Doppler velocity of the object at time k. We added some measurement noise to this data to create the radar measurements.

To obtain results we performed Monte-Carlo simulations. We used this technique since our method contains stochastic elements. To get representative results of our method and not by accident get one really good or really bad result we performed multiple Monte-Carlo runs. We also want to know if our method structurally performs well. The measurement noise was generated separately for each Monte-Carlo run. We also created a straight-curve-straight trajectory using the same dynamical models. The created trajectories are shown in figure 3.1. Both trajectories start at $x = 200m$ and $y = 200m$ and have initial velocities of $v_x = 1m/s$ and $v_y = 1m/s$.

3.2 Model settings

Using the created measurements we have investigated the influence of different settings of the models on our method. We looked at the following parameters:

- Number of particles
- Process noise
- Initial state
- Measurement noise
- Sampling time
- Mode transition probability matrix
- Submode transition probability matrix



(a) Weave trajectory

(b) Straight-curve-straight trajectory

Figure 3.1: Created trajectories

3.2.1 Number of particles

In particle filtering the particles are used to approximate the posterior density function. This posterior is better approximated when more particles are used. Unfortunately the use of more particles leads to an increased computation time of our method. So we need to find a number of particles which is large enough to approximate this posterior well, but we also do not want a too large number to keep the computation time small enough.

We also need to keep an eye on the number of particles per mode and per submode. It is possible that in some cases there is only a small number of particles in a certain mode. This can lead to an inaccurate representation of the posterior in that particular mode, which on one hand is undesirable. On the other hand when we encounter a scenario in which the object very clearly follows the trajectory corresponding to a certain mode. It is likely to get a small number of particles in the other modes which in this case is not unexpected and perhaps even not undesirable. But in general we want a large enough number of particles in each mode to represent the posterior in each mode correctly.

3.2.2 Process noise

The process noise influences the prediction of the particles. It is unlikely that the object precisely follows the used dynamical models, so some uncertainty is added in the form of process noise. The magnitude of the process noise should be chosen such that it is just large enough to cope with the deviation of the object with respect to the dynamical models, but not too large. If we choose the process noise too large it can happen that predictions according to different dynamical models have some overlap, see figure 3.2. Obviously this overlap between dynamical models is undesirable since for example when the process noise is set too large a curve can be predicted via the dynamical model for a straight line trajectory. So a wrong classification can be made while the object is being tracked correctly. So we need to set the process noise such that the different dynamical models are distinctive enough.

(a) High process noise creating overlap

(b) Low process noise, no overlap

Figure 3.2: Prediction via the dynamical models for a straight and a curve

3.2.3 Initial state

To investigate the limitations of our method we have looked at the performance of our method for different ranges of the object with respect to the radar. We performed simulations using different initial states for our created measurements. It turned out that our method can not handle trajectories which start close to the radar, within 10 - 20 meters. This is a direct consequence of the way we initialize the particles. We initialize the particles in a square around the first measurement, therefore if the first measurement is close to the radar we get a large interval for the possible bearing angles for these initial particles. So the predication of these particles is off, creating a larger error as a result of a 'bad startup' due to our method. But since we do not expect a hostile ship to be starting its trajectory this close to the radar we do not expect any startup problems like the one described.

Furthermore we see a larger error when the trajectory lies close to the radar at any point in time. This is the result of the particle cloud lying 'over the radar' creating predictions via 'wrong' bearing angles, creating a larger error.

Also the further away the object's trajectory with respect to the radar is, the larger the errors of our method. This is something we expected since the radar's absolute error due to the bearing error of the radar increases when an object is further away from the radar. The radar is less able to accurately track the object.

3.2.4 Measurement noise

The measurement noise is a measure of uncertainty of the radar. The values of the measurement noise can therefore differ per radar. We performed simulations using various magnitudes of measurement noise. These simulations show us smaller errors when the measurement noise is smaller. This is a logical outcome since the radar then has a smaller uncertainty and hence a larger accuracy.

3.2.5 Sampling time

The sampling time is the time between measurements. Normally this sampling time is determined by the radar system and can not be influenced. But for the purposes of investigating the effect of the sampling time on our method we performed simulation using various sampling times. It turned out that for the values we have used, namely $1s$, $2s$, $3s$ and $5s$, the method still performs well. Obviously the errors become larger when the time between measurements increases. Since the time between measurements is larger the influence of the process noise is also larger since the process noise depends on the time step, T. So the uncertainties on the prediction of the particles is larger due to the process noise, hence we expect to see larger errors.

3.2.6 Mode transition probability matrix

The mode transition probability matrix, Π_m, governs the transitions of the mode of each particle. The entries of this matrix directly influence the tracking and classification performance of our method. When we 'force' more particles in a certain mode by making the probabilities of transitioning to this particular mode larger, we see a larger fraction of particles this mode. If this mode is not the mode the real object is following, we see an increase in errors since a large number of particles is predicted according to the wrong mode. Not all these 'wrong' predictions are omitted in resampling, so errors increase. Small changes of the entries in this matrix immediately affect the outcome of our method.

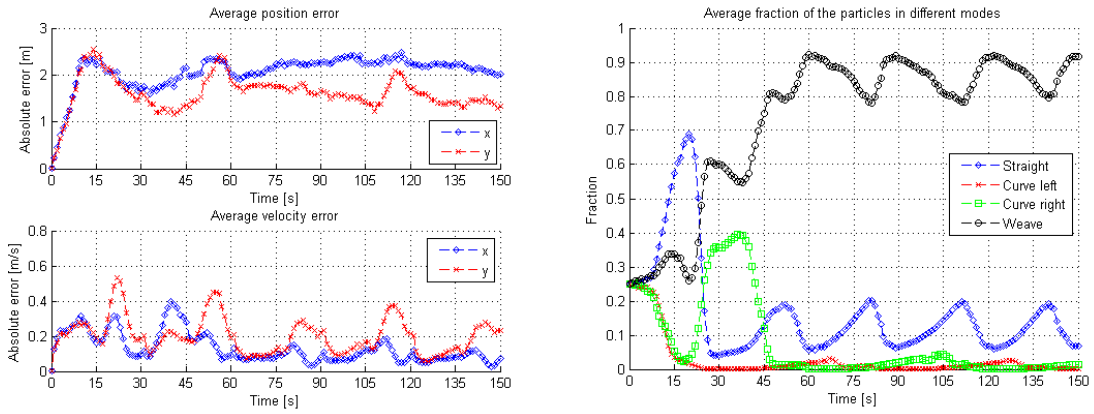
3.2.7 Submode transition probability matrix

In the same way the mode transition matrix governs the transitions of the mode of each particle, the submode transition probability matrix, Π_{sm_k}, governs the transitions of the submode of the particles when they are in the mode weave. This matrix is completely determined by p_{sc_k} and p_{cs_k}. These probabilities are determined by the min-method or by the if-else-method. We performed simulations using different parameters settings of both methods. We see that the classification is better when δ_{max} and θ_{max} are better estimations of the actual weave size. We also see that these variables can better be estimated too large then too small since otherwise an undesired transition is forced too soon. We got good results for the if-else-method when we chose p_1 in the order of

0.05 and p_2 in the order of 0.95. For the min-method we got good results for $n = 25$. But more simulations with varying values of these parameters need to be carried out using real measurements.

3.3 Preliminary conclusions

In figure 3.3 the results of the most promising simulation are presented. We used the weave trajectory of figure 3.1a as the true trajectory. We performed 100 Monte-Carlo runs in which the measurement noise was generated separately each run. Per run we used 6,500 particles, also we used the if-else-method to determine the entries of the submode transition probability matrix. We used $p_1 = 0.05$ and $p_2 = 0.95$ and slightly overestimated the values of δ_{max} and θ_{max}. The shown results are averages over all runs.



(a) Errors

(b) Classification

Figure 3.3: Errors and classification of most promising results using simulated radar measurements

When we look at the results in figure 3.3 we see after some 'startup period' a large fraction of particles in the mode weave. This is exactly what we want since the object is following a weave trajectory. Furthermore we see that the position error in both x- and y-direction is approximately $2m$. The error in velocity in both directions is approximately $0.2m/s$. We see some peaks in the errors of the velocity. These peaks come from the fact that the object performs a turn at these points. Therefore it is hard to accurately estimate the velocity of the object.

We see that our method has a good tracking and classification performance and therefore we can conclude that our method is viable. We need to further investigate the performance of our method when real measurements are used. During this further research we need to be aware of the used number of particles to represent the posterior correctly. The magnitude of the process noise should be investigated when real measurements are used since these real measurements are not that neatly defined as our simulated measurements. Our method can track and classify an object at different ranges with respect to the radar, but also has its limitations in the sense that this ability decreases when the object is too close or too far from the radar. Our method is also able to cope with different sampling times. The entries in the mode transition matrix should be determined by performing many simulations using real measurements since small changes in these entries already affect the outcome of the method. While determining the entries of the submode transition matrix we need to keep comparing the results for both the if-else-method and the min-method to see which method performs better. As for the estimations of δ_{max} and θ_{max} we can better overestimate than underestimate these values for a better classification.

4. Results using GPS measurements

The preliminary results of section 3 show us that our method is viable. Until now we have tested our method only using measurements created by ourselves. These measurements are very neatly defined and present the most favorable conditions for the method to work with. Unfortunately in real-life situations these measurements are often not that neatly defined. So to investigate if our method also works in these kind of real-life situations we used measurements obtained by Thales. These measurements are GPS data loggings of different vessels. We use this data in the finetuning of the setting of our method and in the investigation of the distinctiveness of the different dynamical models.

Eventually we want to implement our method in a radar environment. Unfortunately radar measurements have some disadvantages over GPS measurements like the accuracy and dependency of the measurement noise on the position of the object. But in real life we need to use a radar to obtain the measurements since not all vessels simply 'hand over' their correct GPS information. Since the theory behind our method does not change when different measurement types are used, we first try to finetune the settings of our method using GPS measurements. These settings should be chosen such that they work for a wide variety of trajectories and variations of trajectories since we do not want to have to establish these settings in each different situation. Due to the advantages of GPS measurements over radar measurements we first try to find these settings using the GPS measurements and in section 5 use these settings to see if they still hold when radar measurements are used.

4.1 GPS Measurements

GPS is the abbreviation of Global Positioning System. It is a satellite navigation system which provides the location of the object. GPS measurements have the advantage over radar measurements that the measurement noise of GPS measurements is independent of the position of the object, it acts in the x- and y-direction. Therefore this GPS measurements are useful in the finetuning of the settings of our method and determining the magnitude of the process noise parameters for the different dynamical models.

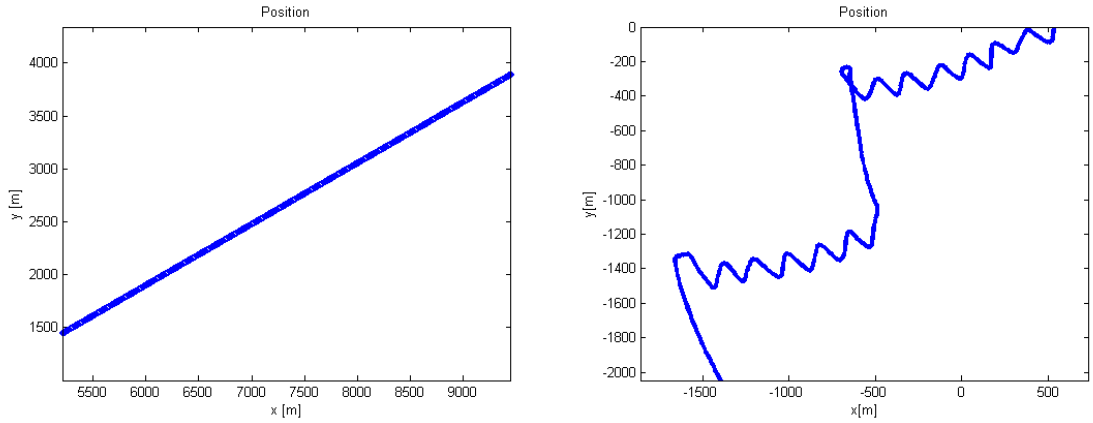
The trajectories we use as true trajectories in this section are shown in figure 4.1. During the finetuning we also used different trajectories. More details and results using these other trajectories are given in appendix B.

4.2 Tuning of the settings

To describe reality closely, the settings of our method should be chosen such that our method is first of all able to track an object correctly. We used the GPS data loggings to tune the settings such that we get some general settings for which our method has a good tracking performance for various trajectories. We do not want to have to establish new values for the parameters whenever a new scenario is encountered. Next we looked at the ability of our method to classify the different trajectories.

4.2.1 Measurement noise

For the tuning of the settings we first investigated the used measurements. We started by looking at the possible measurement noise settings. In most cases these settings are known since it is known what kind of GPS is used and what its reliability is. In our case we only have the knowledge that



(a) Trajectory 1 (b) Trajectory 2

Figure 4.1: Different trajectories obtained from GPS data loggings

the measurements are GPS data loggings but have no knowledge of the reliability of this GPS system.

We investigated the magnitude of the measurement noise by applying a moving average scheme on the measurements and comparing these results with the original measurements. We expect the vessels to follow a smooth pattern, so by investigating the differences between the original measurements and the measurements on which a moving average scheme is applied we try to find the magnitude of the measurement noise parameters. We found that the measurement noise parameters in both $x-$ and $y-$direction have a magnitude of $1m$, which is a magnitude which can be expected for GPS measurements.

4.2.2 Process noise

After we established these measurement noise parameters we investigated the tracking performance of our method. Because we first want to know if the used dynamical models are distinctive enough and result in a good tracking performance, we temporarily 'disabled' the mode weave. So we only allowed particles to be in the modes straight, left curve and right curve. The tracking performance is among others influenced by the process noise parameters, so we investigated the magnitude of these parameters.[1]

In the NCV-model we have two process noise parameters one in $x-$ and one in $y-$direction. In most literature both parameters have a value of $\frac{1}{3}$ times the maximal acceleration in that direction. When we looked at the measurements, we saw that some vessels decelerate before making a turn at such a rate that our method could not cope with this deceleration. These turns were entered in various headings. Therefore we could not simply adjust the magnitude of the process noise parameters of the NCV-model, since these act in the $x-$ and $y-$direction and are not dependent on the heading of the object. So we introduced the CM-model in which the process noise acts in normal and tangential direction, see section 2.2.2. Using this modification we could tune the process noise parameters such that our method could cope with this kind of deceleration. Now the process noise acts in the heading direction and is therefore better 'tunable' to cope with the deceleration. Eventually we found a good tracking performance using $\sigma_{a_T} = 2m/s^2$ and $\sigma_{a_N} = 1.754e-5m/s^2$. Using these magnitudes our method is able to cope with the deceleration before turns. The 'heading uncertainty', σ_{a_N}, is chosen small since we expect and see that the vessels follow nice straight lines.

The ACT-model has three process noise variables, one in $x-$direction, one in $y-$direction and one for the angular velocity. In this model the process noise parameters on the position act in

[1] One thing to note is that in the particle filter implementation we made sure that the dynamical models for a left curve and a right curve are really distinct. Since we update the angular velocity of a particle by adding a noise term to the old angular velocity it could happen that this angular velocity changes sign during such an update thereby creating some overlap between the two dynamical models. This is obviously undesirable since the particle can be 'considered' as being in a right (or left) curve but be predicted according to a left (or right) curve. To work around this problem and not change the distribution of the noise term which is added to the old angular velocity we give the particles of which the angular velocity changes sign a weight of 0. We also rescale the weights of the other particles such that the sum over all weights remains 1 which is necessary to use the posterior as a distribution.

the $x-$ and $y-$direction since a turn can hypothetically evolve in any direction since the angular velocity can increase or decrease making the curve more sharper or more straight. Therefore the process noise should act independent of the heading of the object, so we chose to still use the NCV-model when an object follows a curve. The process noise parameter for the angular velocity is set to $0.01 rad/s$ since we expect only small changes in the angular velocity while in a curve. The other process noise parameters are chosen as follows $\sigma_{a_x} = \sigma_{a_y} = 1 m/s^2$.

The magnitude of these settings is determined by performing simulations using different settings. It can be that these values are not optimal in the sense that an even better tracking performance can possibly be reached. But finding the optimal values would take even more simulations. Using these settings for the process noise parameters our method is able to track an object correctly. Therefore we choose to continue our research using these magnitudes.

After we got a satisfying tracking performance we 'enabled' the mode weave again. We looked if the method still had a good tracking performance, this was indeed the case. This was expected since the models used in the mode weave are the same as the models which we used in the finetuning of the tracking performance. Therefore the tracking performance of our method did not decrease.

4.2.3 Classification

Eventually we want to classify the trajectory based on the output of our method. For this classification we look at the fraction of particles in each mode per time step. In this way these fractions represent the probability our method gives to the object following the trajectory corresponding to that mode. To investigate which settings give the best results we performed many simulations using the different trajectories as true trajectories and different settings for our method.

Mode

We started by closely looking at the mode transition probability matrix, Π_m. This matrix should represent the (real-life) probabilities of making transitions between the different possible modes or trajectories. We first used a matrix in which the following transitions were possible, see figure 4.2.

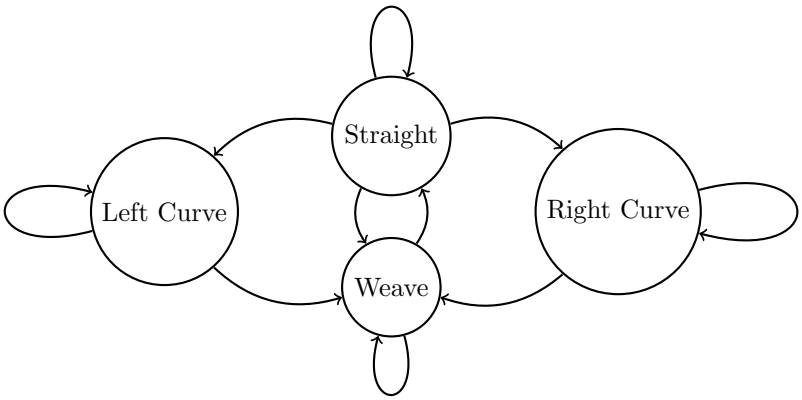


Figure 4.2: Schematic representation of possible mode transitions used in early research

We only allowed transitions out of the curve modes to go to the mode weave. But after reexamining this, we argued that transitions to the mode 'weave' should only be possible from the mode 'straight'. Modeling transitions in this way we think gives a better representation of the transitions in real life.

Using the old transitions a straight-curve-straight trajectory would at some point be classified as a weave trajectory since the particles can only get to the mode straight from a curve mode via the mode weave. Since we use the same dynamical models in all the modes we get that our method still is able to track the straight-curve-straight trajectory but at some point classifies it as a weave trajectory, which is undesirable.

Also when we compared results using the 'old' possible mode transitions and the 'new' possible mode transitions of figure 2.3, we found better classification performance using the 'new' possible

transitions when a straight-curve-straight trajectory was used in the simulations as the true trajectory. By performing additional simulations using different trajectories as true trajectories we chose Π_m in the following way:

$$\Pi_m = \begin{bmatrix} 0.97 & 0.01 & 0.01 & 0.01 \\ 0.35 & 0.65 & 0 & 0 \\ 0.35 & 0 & 0.65 & 0 \\ 0.03 & 0 & 0 & 0.97 \end{bmatrix}. \tag{4.1}$$

We used this matrix while finetuning the other settings.

Submode

From the preliminary results of section 3.3 we know that we can better overestimate than underestimate the variables δ_{max} and θ_{max}. To get a feeling on how large these variables should be, we investigated the measurements. We looked at the trajectories of of the GPS data loggings to see how the average weave encountered in the GPS data loggings looks like. The different trajectories obviously contained different sized weave trajectories. Eventually we set the variable δ_{max} to $500m$ since this is the maximal length of a straight segment in one of the weave trajectories. We chose θ_{max} to be $180°$ or πrad, this is a choice to define a weave trajectory as having curves of maximal πrad.

We choose these values in such a way that all the segments of the weave trajectories we investigated are smaller than the segments defined by δ_{max} and θ_{max}. This for the fact that we want to keep these variables the same while investigating different trajectories. In real life we do not have this kind of knowledge for each individual weave trajectory. So to cope with the absence of this knowledge we chose δ_{max} and θ_{max} large so we do not underestimate these variable, since we have already established that we can better overestimate than underestimate these variables. Remember that we try to find general settings such that our method has a good tracking and classification performance for various trajectories and variations in trajectories without having to change or re-establish these settings.

The transition probabilities in the submode transition probability matrix, Π_{sm_k}, are determined using two methods; the if-else-method and the min-method, see section 2.3.2. We have compared the different methods and varied the magnitude of δ_{max} and θ_{max}. These results can be found in appendix B.

From these simulations we conclude that the preliminary conclusions of section 3.3 regarding the submode still hold. A better estimation of δ_{max} and θ_{max} results in a better classification and overestimating these variables is better than underestimating these variables. Also the if-else-method gives better results and therefore we find this method more favorable.

Non-weave trajectory

To see how our method copes with trajectories that are not a weave or a straight line, we performed simulations using other non-weave trajectories. We specifically used a straight-curve-straight trajectory in which the angle of the curve is approximately πrad, see figure 4.3. It is important to note that the trajectory shown in this figure is simulated by ourselves and is not obtained from the GPS data loggings.

The curve of this trajectory is close to θ_{max}, possibly making it hard for our method to distinguish this non-weave trajectory from a weave trajectory. To obtain the results of figure 4.4 we used 10,000 particles and performed 10 Monte-Carlo runs in which the measurement noise was generated separately. We used the trajectory of figure 4.3 as the true trajectory. The shown result is the average over all runs.

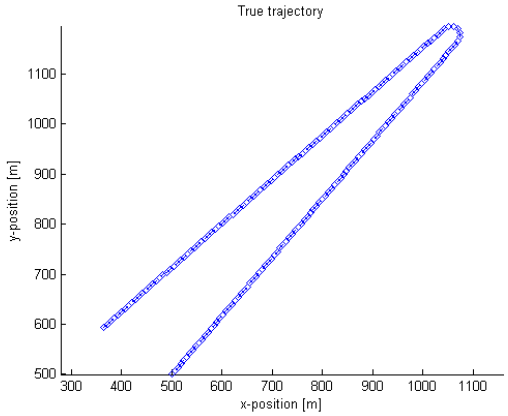
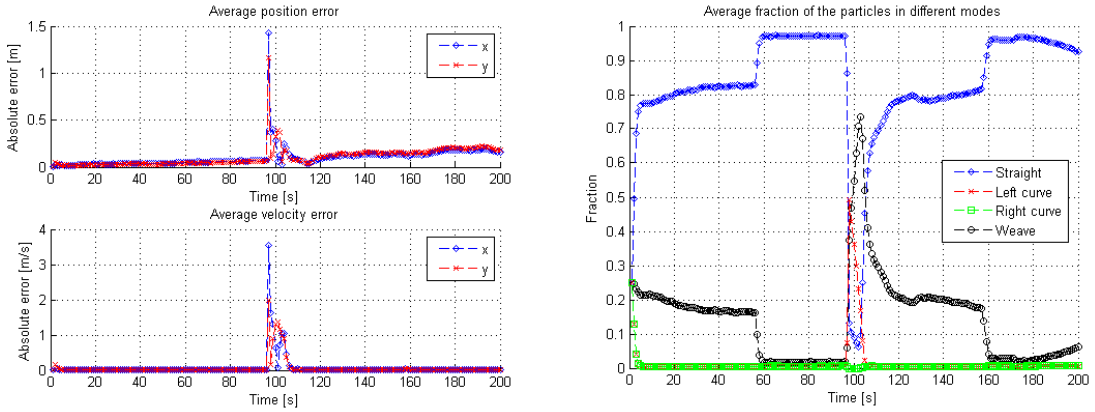


Figure 4.3: True trajectory for straight-curve-straight trajectory



(a) Errors

(b) Classification

Figure 4.4: Errors and classification for straight-curve-straight trajectory using Π_m as in (4.1)

We see that the errors in both velocity and position in all directions is very small. We see a peak in the errors at the first time step in which the object enters the left curve. These errors are due to the fact that our method needs some time steps to get a large fraction of particles in the correct mode such that the object is tracked correctly.

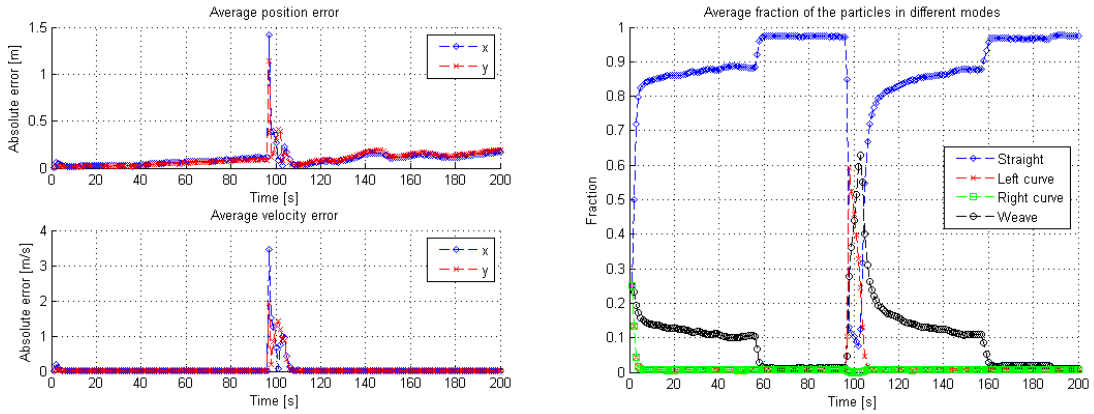
Figure 4.4b show us that our method gives a high probability of the trajectory being a weave when the object is in the curve of the trajectory. This is undesirable since we specifically stated that we want to classify this trajectory as a non-weave trajectory. But when we look at Π_m again, see (4.1), we see that this peak in the fraction of particles in the mode weave is explainable. The probability of a transition from the mode straight to any other mode is equal for these modes. But the probability of staying in the mode weave is significantly larger than the probability of staying in a curve mode. Since we use the same dynamical models in the different modes we are able to track the turn of the object in the mode weave as good as in any curve mode. Together this results in a higher fraction of particles in the mode weave while making a turn.

Since we were not yet satisfied with this result we again looked at the entries of Π_m. We made the probability of a transition from the mode straight to any of the curve modes larger than the transition of the mode straight to the mode weave. In real life it is more likely to come across a curve in comparison to a weave, therefore we choose the probability of a transition to a curve mode larger than a transition to the mode weave. We want to 'force' more particles in the curve modes. Also we slightly increased the probability of staying in a curve mode while in this curve

mode. These adjustments led to the following mode transition probability matrix:

$$\Pi_m = \begin{bmatrix} 0.97 & 0.012 & 0.012 & 0.006 \\ 0.34 & 0.66 & 0 & 0 \\ 0.34 & 0 & 0.66 & 0 \\ 0.03 & 0 & 0 & 0.97 \end{bmatrix}. \tag{4.2}$$

Using this Π_m we found better results. In figure 4.5 the results for a simulation in which we used the trajectory shown in figure 4.3 as the true trajectory. We used 10,000 particles and performed 10 Monte-Carlo runs in which the measurement noise was generated separately. The shown results are averages over all runs.



(a) Errors (b) Classification

Figure 4.5: Errors and classification for straight-curve-straight trajectory using Π_m as in (4.2)

The errors of figure 4.5a are of the same magnitude as the errors of figure 4.4a. In figure 4.5b we still see a peak of the fraction of particles in the mode weave, but it is smaller than in figure 4.4b and also the fraction of particles in the mode left curve is larger. We could try to reduce the peak in the fraction of particles in the mode weave for these kind of trajectories even more by making the probability of staying in one of the curve modes larger. But we chose not to do this since then we feel like these probabilities reflect the probabilities for real-life situations worse. It is likely that a turn of an object ends at some point and the object transitions to a straight segment. While when an object follows a weave trajectory it is likely to keep following that trajectory. Therefore the probability of staying in one of the curve modes is smaller than the probability of staying in the mode weave.

Also when we increase the probability of staying in one of the curve modes to the order of 0.97, we see that a weave trajectory is not classified as such anymore. It is classified as a sequence of curve and straight segments. Since more particles are now forced into the curve modes in comparison to the mode weave and the tracking performance in all modes is equal we see that the fraction of particles in the mode weave stays low. So our method's classification performance decreases when tracking a weave trajectory. Therefore we choose to keep Π_m as in (4.2).

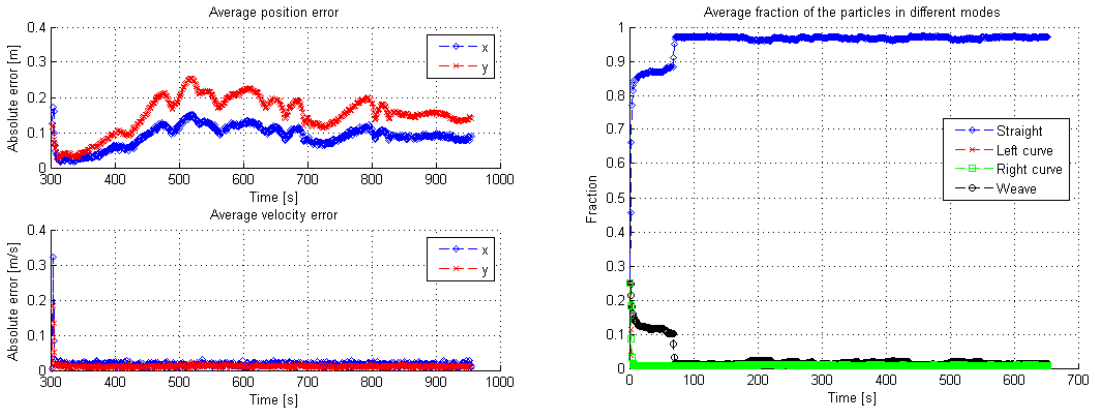
4.3 Final settings

Using the GPS data loggings we finetuned the settings of our method. We found general settings with which we can track and distinguish different types of trajectories using GPS measurements without having to adjust these settings per trajectory we encounter. We found the following settings:

- $\Pi_m = \begin{bmatrix} 0.97 & 0.012 & 0.012 & 0.006 \\ 0.34 & 0.66 & 0 & 0 \\ 0.34 & 0 & 0.66 & 0 \\ 0.03 & 0 & 0 & 0.97 \end{bmatrix}$

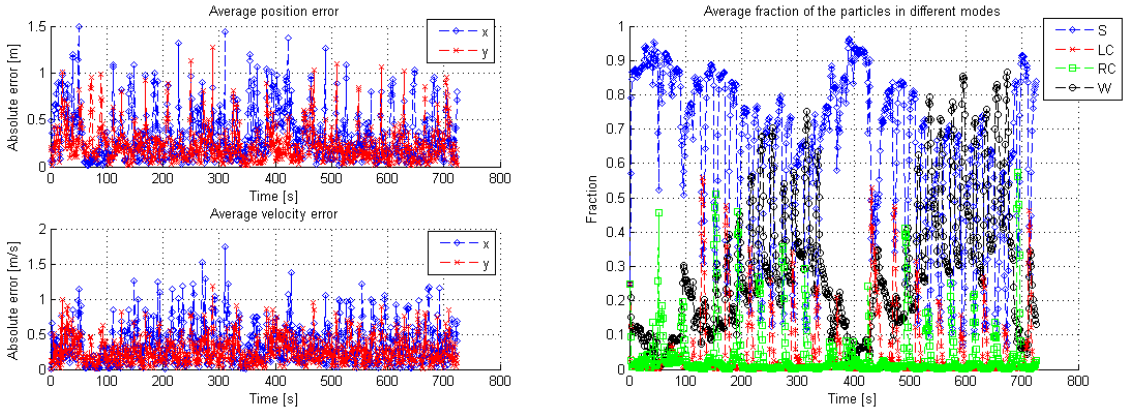
- $p_{sc_k} = \begin{cases} 0.05 & \text{if } \delta_k < \delta_{max}, \\ 0.95 & \text{if } \delta_k \geq \delta_{max}. \end{cases}$
- $p_{cs_k} = \begin{cases} 0.05 & \text{if } \theta_k < \theta_{max}, \\ 0.95 & \text{if } \theta_k \geq \theta_{max}. \end{cases}$
- $\delta_{max} = 500m$, $\theta_{max} = \pi rad$

To show the outcomes of our method using these 'final' settings we performed simulations using 10,000 particles and 10 MC-runs. For trajectory 1 the measurement noise was the same for all runs. For trajectory 2 the measurement noise was generated separately each run.[2] These results are shown in figure 4.6 and 4.7. More results using other trajectories can be found in appendix B.



(a) Errors (b) Classification

Figure 4.6: Errors and classification using trajectory 1 as true trajectory



(a) Errors (b) Classification

Figure 4.7: Errors and classification using trajectory 2 as true trajectory

Figure 4.6b shows the classification of trajectory 1. It clearly shows that this trajectory is most likely a straight trajectory which is indeed the case.

When we look at the errors in position and velocity of the figures 4.6 and 4.7, we see that the errors of figure 4.7 are clearly larger. The dynamical model used when the object follows a curve is more complex then the dynamical model when the object follows a straight line. Also the angular velocity has to be estimated in a curve. Therefore there is a larger uncertainty when estimating the position and velocity of the object while it follows a curve in comparison to when it follows a

[2]This difference in measurement noise generation is due to the format in which the measurements were available at Thales

straight line. In figure 4.6 the object only follows a straight line. Where in figure 4.7 the object also follows curves. Therefore the errors of figure 4.7 are larger than the errors of figure 4.6.

If we look at figure 4.7b we do not see such a clear classification. Our method starts well by giving large probability of the first part of the trajectory being a straight trajectory, which is indeed the case. Next a weave is entered, which we see in the classification output by an increase of the fraction of particles in the mode weave. But while this weave continues we see that the fraction of particles in the mode weave somewhat alternates with the fraction of particles in the mode straight. While in the best case scenario when the object follows a weave trajectory we do not want this alternating behavior but just a high fraction of particles in the mode weave.

This alternating behavior is explainable. Since we use the same dynamical models in the different modes we are able to track a straight segment in the mode weave as good as in the mode straight. So we expect to still see a large fraction of particles in the mode weave while the object follows a straight segment of a weave trajectory. But when we look at Π_m, see (4.2), we see that more particles are forced into the mode straight than into the other modes. This is so since the probability of transitioning from the mode straight to the mode weave is smaller than the probability of a transition from the mode weave to the mode straight. Also there are transitions possible from the curve modes to the mode straight and not to the mode weave. So when the object is following a straight segment in a weave it is more likely for the particles to transition to or stay in the mode straight than it is to transition to or stay in the mode weave since in both modes the tracking performance is exactly the same.

Perhaps the idea of more particles being forced into the mode straight becomes more clear when we look at the stationary distribution of the Markov process governed by Π_m. We computed that this stationary distribution is:

$$\begin{bmatrix}\pi_1 & \pi_2 & \pi_3 & \pi_4\end{bmatrix} = \begin{bmatrix}0.7870 & 0.0278 & 0.0278 & 0.1574\end{bmatrix}.$$

Hence we indeed see a larger fraction of particles being forced in the mode straight in the long run if we only look at the evolution of the Markov process. As said before, the tracking performance in each mode is equal so it is likely that this phenomenon is not negated by other aspects of our method as for example the resampling of the particles. Therefore we see this alternating behavior between the modes straight and weave while the object is following a weave.

The last part of the classification also shows the alternating behavior between the modes weave and straight while the object follows a weave trajectory. Right before this second weave the object follows a curve of more than πrad. In the classification plot of figure 4.7b we see a peak at this point of the fraction of particles in the mode left curve, which is good. We also see a (smaller) peak in of the fraction of particles in the mode weave. This peak is explained by the fact that at first this curve looks very much like the average curve of a weave defined by θ_{max}, which is πrad, we see an increase in the fraction of particles in the mode weave. But as the object continues further in a straight line trajectory we see that the fraction of particles in the mode weave decreases again and only starts to increase again when the object starts following a weave trajectory.

5. Results using radar measurements

Our research until now shows that our method is viable and that we have a method which has a good tracking performance when GPS measurements or simulated measurements created by ourselves are used. Our method is also able to distinguish different trajectories when GPS measurements are used. But as stated throughout this report we aim for a radar application which is able to track and classify the different trajectories. Therefore we convert the GPS data loggings into radar measurements using (3.1). Using this conversion we position the radar at the point $(x, y) = (0m, 0m)$. We used the same GPS data loggings which are used in section 4.

Throughout this section we see the GPS data loggings as the true trajectory. We convert these to radar data and add some measurement noise to create the corresponding radar measurements. The goal of this section is to find settings such that our method is able to track and classify various trajectories using only one specific set of settings. Since in real life we are not able to establish these settings for each situation, we want to find a set of settings such that our method performs well in general.

5.1 Radar measurements

Where a GPS 'measures' the $x-$ and $y-$position of the object, the radar we use measures the range, bearing angle and Doppler velocity of the object. Therefore we also have three measurement noise parameters. Due to the different type of measurements, we also get a different 'shape' of the measurement noise. To clarify this we have schematically drawn the measurement noise 'clouds' for a measurement of a weave trajectory using a GPS and a radar, see figure 5.1.

(a) Measurement noise cloud using GPS (b) Measurement noise cloud using radar

Figure 5.1: Schematic drawing of measurement noise cloud for different measurement types.

When using radar measurements the measurement noise can have a more significant influence on the output of the method since it can lie over a larger part of the trajectory. Perhaps even overlapping a (part of a) curve segment and a (part of a) straight segment, making it harder to distinguish these segments from another.

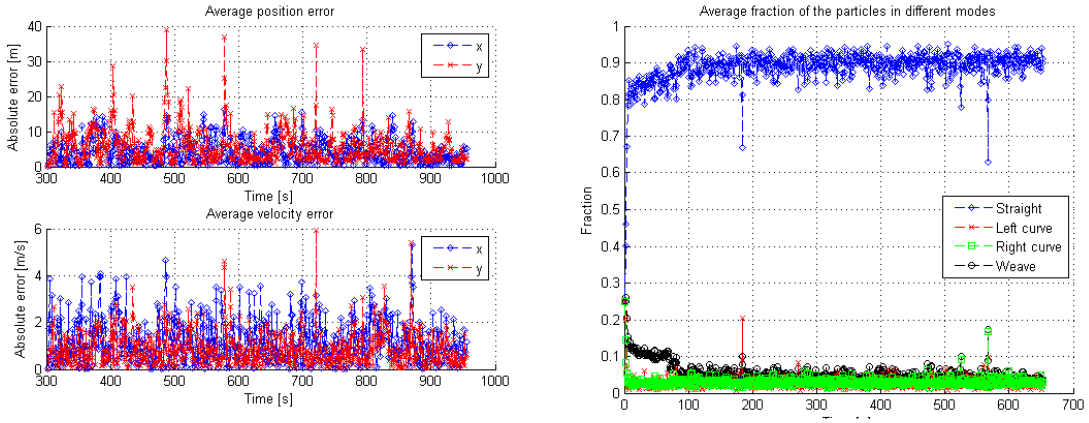
Also the effect of the measurement noise 'increases' when the range of the object increases. Of course the measurement noise parameters stay the same but due to the increasing range the bearing measurement has a larger absolute error, therefore creating a larger measurement noise 'cloud'.

The values of the measurement noise parameters can differ per radar since different radars can have different performances. We chose to use the following values for the measurement noise parameters: $\sigma_r = 10m$, $\sigma_b = 0.005rad$ and $\sigma_d = 3m/s$.

5.2 Results using final settings

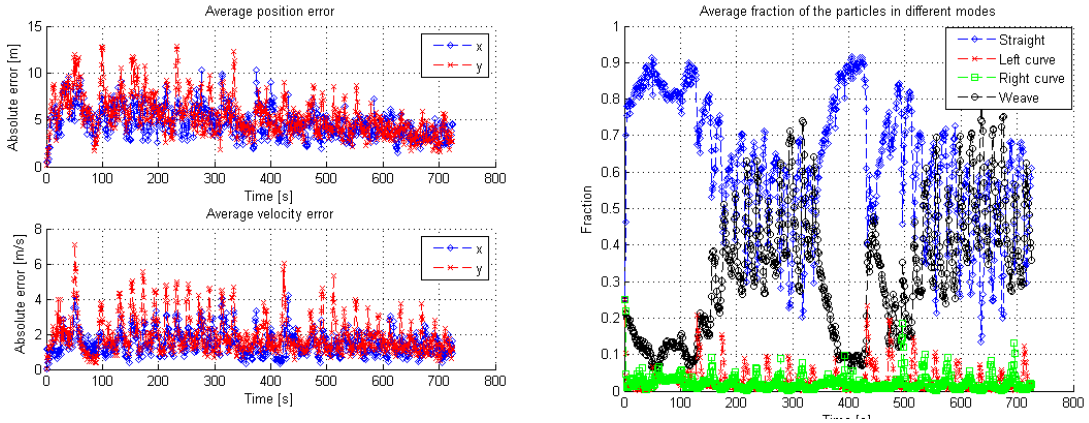
In section 4 we determined settings for which our method tracks and classifies different trajectories. These settings are determined using GPS data loggings. The conversion of these GPS measurements to radar measurements do not change the proposed method. The only difference is the type of measurements which are being used. This can of course influence the performance of the method. In this case we expect our method to have a better performance when GPS measurements are used compared to when radar measurements are used. As explained in section 5.1 the noise of radar measurements is more complex and depends on the position of the object. Where the noise of GPS measurements is independent of the position of the object. Therefore we expect that the radar measurements have a negative influence on the performance of our method in comparison to GPS measurements.

To investigate if our expectations about the performance of our method using radar measurements is true, we performed simulations using the settings of section 4.3. We performed simulations using 10,000 particles and 10 MC-runs. The results are shown in figures 5.2 and 5.3 and use respectively the trajectories of figure 4.1 as the true trajectories. For trajectory 1 the measurement noise was the same for all runs. For trajectory 2 the measurement noise was generated separately each run.[1] The shown results are averages over all runs. Results for other trajectories can be found in appendix C.



(a) Errors (b) Classification

Figure 5.2: Errors and classification using trajectory 1 as true trajectory



(a) Errors (b) Classification

Figure 5.3: Errors and classification using trajectory 2 as true trajectory

[1] This difference in measurement noise generation is due to the format in which the measurements were available at Thales

We see that both results have a significantly larger position and velocity error than the results of figures 4.6 and 4.7 where GPS measurements were used. For these last results we use radar measurement which result in a larger error as already explained.

Figure 5.2b shows the classification of trajectory 1. Just like the simulation outcome using GPS measurements, it clearly shows that this trajectory is most likely a straight trajectory which is indeed the case. We do see that the fraction of particles in the mode straight is bound between approximately 0.85 and 0.95 where the fraction of particles in the mode straight in figure 4.6b is almost everywhere around 0.98. So we do see some differences between the classification plots when GPS or radar measurements are used but these differences seem not significant.

When we look at figure 5.3b we see a similar classification plot as when GPS measurements were used, see figure 4.7b. We do see that the fraction of particles in the modes left curve and right curve is lower when radar measurements are used. Also the alternation between the modes weave and straight is bound between approximately 0.3 and 0.7 where this alternation is bound between 0.2 and 0.8 when GPS measurements are used.

It looks like the classification when radar measurements are used is less reactive in comparison to the classification when GPS measurements are used. This can be explained by the larger uncertainty of the radar measurements. The GPS measurements are more accurate and can therefore detect the different turns the object makes faster. So when using the GPS measurements the classification our method makes reacts faster in comparison to when radar measurements are used. But still when radar measurements are used we can distinguish the classification plots for the different trajectories.

5.3 Limitations

The results in section 5.2 show that we are able to distinguish a weave trajectory from a straight line trajectory using the proposed methods. To see to which extent our method is able to do so we investigated the limitations of our method which originate from the use of using radar measurements. As said before, radar measurements can influence our methods performance since the uncertainty of the radar measurements is larger then for GPS measurements and is also dependent on the position of the object.

To investigate the limitations of the use of radar measurements we rotated a weave segment of the trajectory of figure 4.1b and moved this trajectory in the x, y-plane to use it as the true trajectory during the simulations, see figure 5.4. The object starts at the leftmost point and ends at the point $(x, y) = (1000m, 1000m)$. For each of the simulations we used 10,000 particles and performed 10 Monte-Carlo runs for which the measurement noise was generated separately each run. The shown results are averages over all runs.

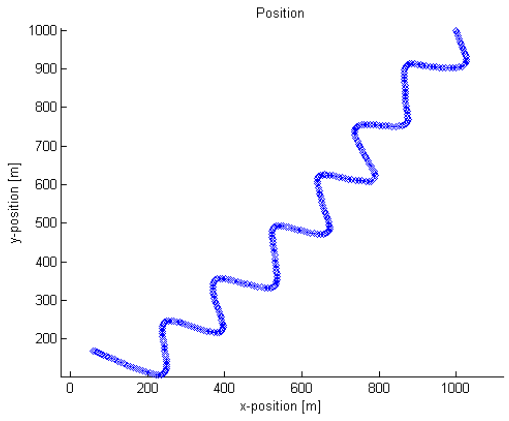


Figure 5.4: Weave trajectory

5.3.1 Range

The radar measurements we use have a fixed measurement noise parameter for the noise in bearing. Since this angle is fixed we see a larger 'absolute' uncertainty of the bearing measurement when

the range of the object increases. We expect that at some point our method is not able anymore to distinguish a weave trajectory from other trajectories due to this larger uncertainty.

For example suppose an object follows a straight line trajectory and for explanatory purposes only take the uncertainty in bearing into account. Since the bearing measurement noise has a Gaussian distribution, we know with a certainty of 99.7% that the measurements of this straight line lie between the two $3\sigma_b$-lines, see figure 5.5.

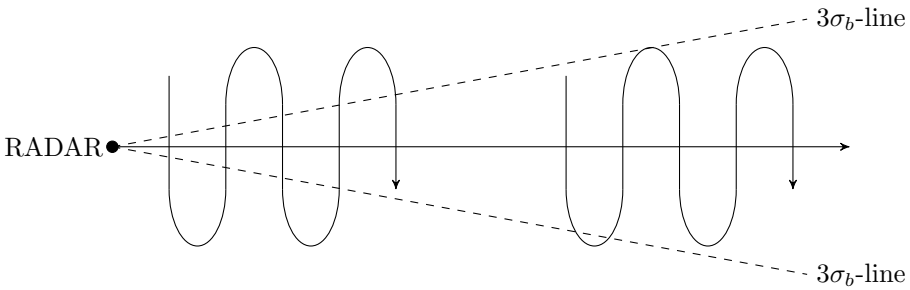
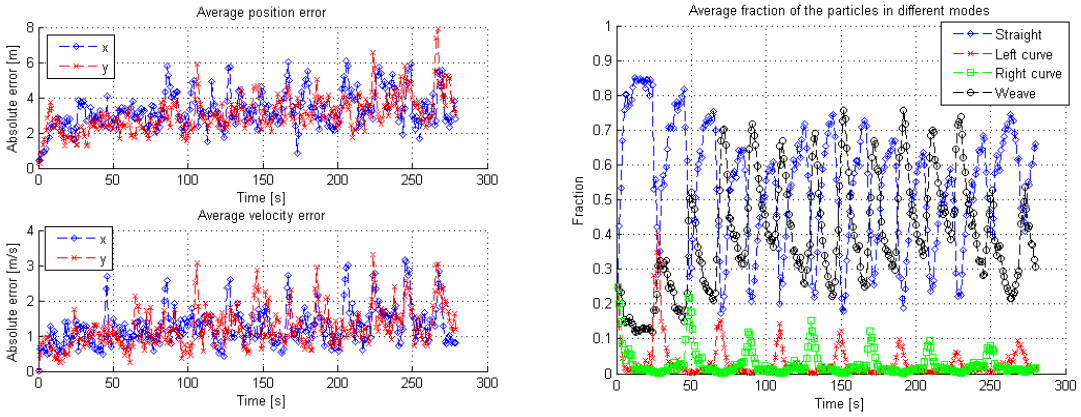


Figure 5.5: Schematic drawing of bearing measurement for straight line trajectory.

In this figure we have also drawn two weave trajectories, one close to the radar and further away from the radar. We see that (parts of) the close weave trajectory lie outside these $3\sigma_b$-lines, but almost the entire weave trajectory further away from the radar lies between the $3\sigma_b$-lines. This means that it could be that the measurements of the straight line trajectory follow this weave trajectory due to the uncertainty of the measurements. Which could lead to the straight line trajectory being classified as a weave trajectory. But also our method could find that the measurements of a weave trajectory lie within the measurement uncertainty of a straight line trajectory and therefore could classify a weave trajectory as a straight line trajectory. So we expect our method's ability to classify the different trajectories to become worse when the range of the object increases.

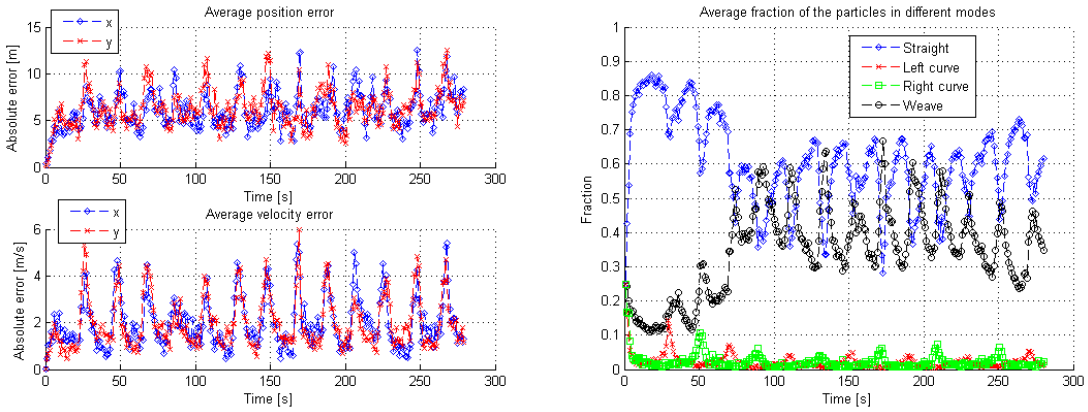
To see if our expectations about our method's classification ability do hold we calculated at what range an entire straight segment defined by δ_{max} lies between the $3\sigma_b$-lines for the used parameter values. This happens from a range around $16,700m$. We performed simulations in which we used the weave trajectory of figure 5.4 as true trajectory and translated this trajectory such that it ends at $(x, y) = (1000m, 1000m)$, $(2500m, 2500m)$, $(5000m, 5000m)$ and $(10000m, 10000m)$, see figures 5.6 to 5.9 respectively. With translated we mean that we shift the whole trajectory in the x, y-plane such that the endpoints of these trajectories lie on the specified points.



(a) Errors

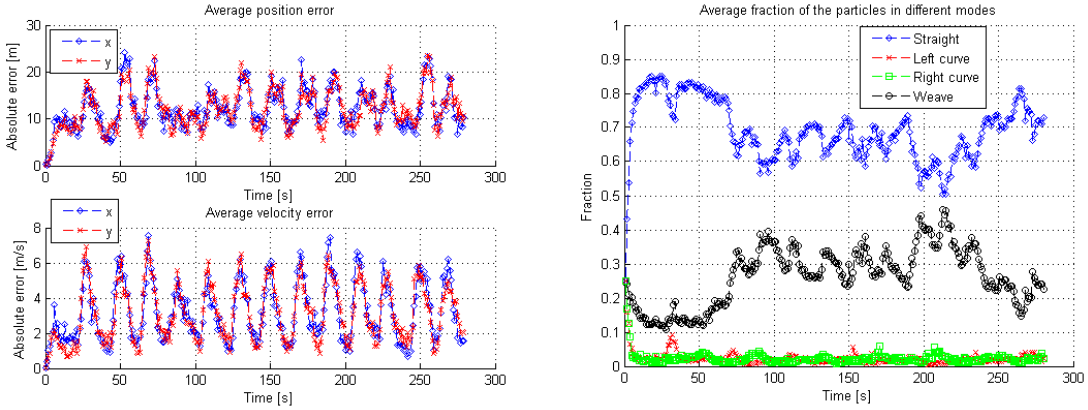
(b) Classification

Figure 5.6: Errors and classification for weave trajectory ending at $(x, y) = (1000m, 1000m)$



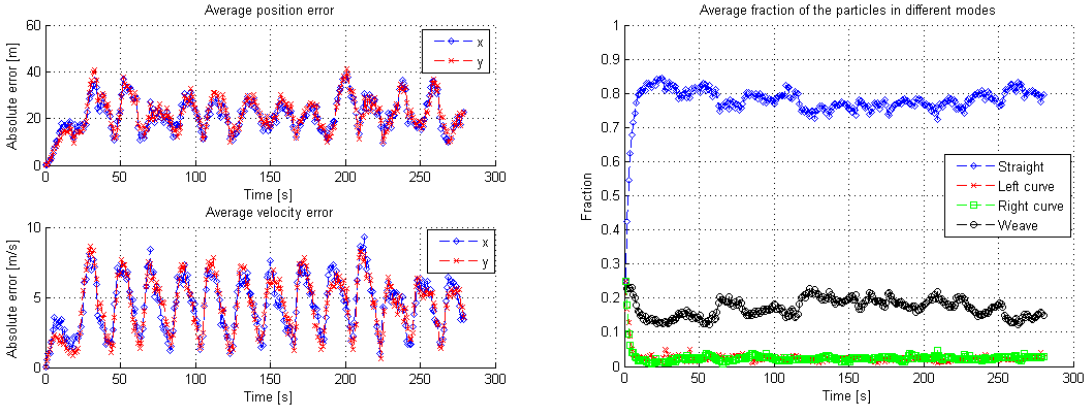
(a) Errors (b) Classification

Figure 5.7: Errors and classification for weave trajectory ending at $(x, y) = (2500m, 2500m)$



(a) Errors (b) Classification

Figure 5.8: Errors and classification for weave trajectory ending at $(x, y) = (5000m, 5000m)$



(a) Errors (b) Classification

Figure 5.9: Errors and classification for weave trajectory ending at $(x, y) = (10000m, 10000m)$

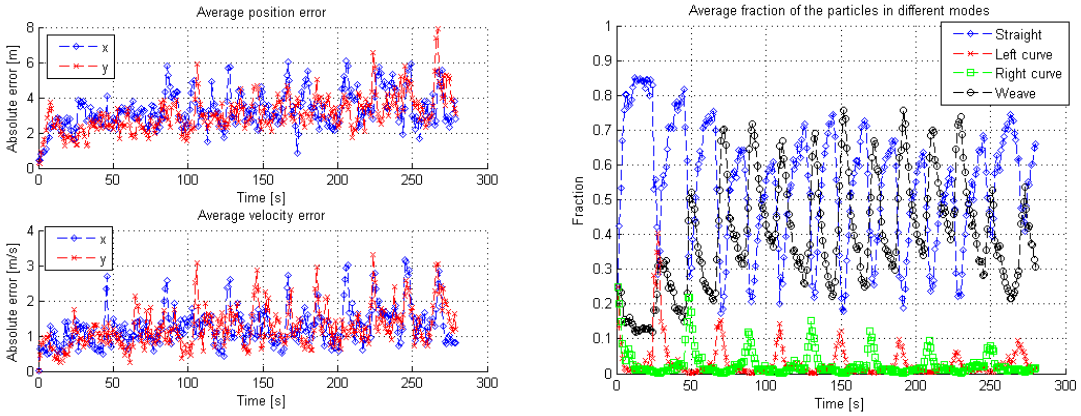
When we look at the figures 5.6 to 5.9 we indeed see that our method's ability to distinguish a weave trajectory becomes worse when the range of the object increases. Already before a range of $16.700m$ this ability becomes so bad that the classification plot of a weave trajectory, figure 5.9b, is similar to the classification plot of a straight line trajectory, figure 5.2b. We also see an increase of the errors when the range increases as a result of the larger measurement uncertainty.

When we look at the errors in position and especially velocity of the figures 5.6 to 5.9, we see that these errors have a somewhat sinusoidal form. The dynamical model used when the object follows a curve is more complex then the dynamical model used when the object follows a straight line. Also the angular velocity has to be estimated in a curve. Therefore there is a larger uncertainty when estimating the position and velocity of the object while it follows a curve in comparison to when it follows a straight line. This causes the errors in position en velocity to increase when the object follows a curve. When the object then follows a straight line these errors decrease again. Since a weave trajectory consists of a sequence of alternating straight lines and curves this increase and decease in position and velocity errors causes these plots to have a sinusoidal form.

5.3.2 Quadrants

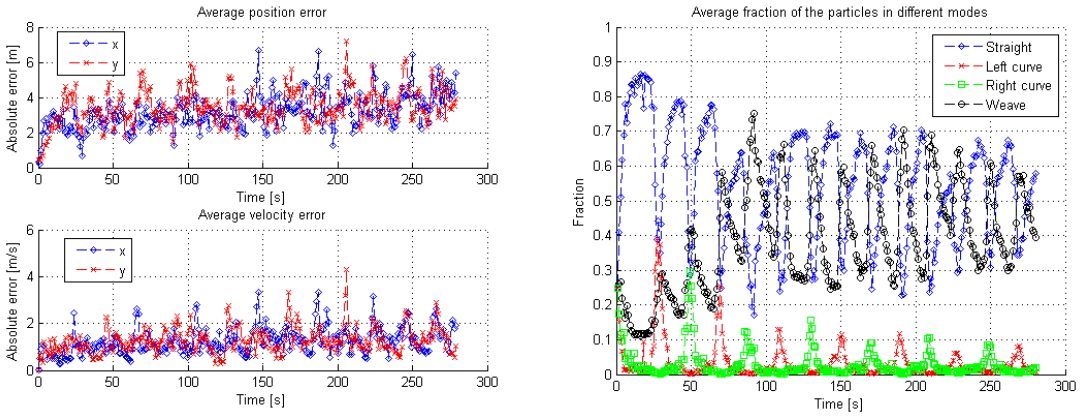
Another limitation we investigated is the quadrant in which the trajectory lies. We performed simulations in which we used the weave trajectory of figure 5.4 as true trajectory and rotated this trajectory around the radar such that the object still starts close to the radar and ends at $(x, y) = (1000m, 1000m)$, $(-1000m, 1000m)$, $(-1000m, -1000m)$ and $(1000m, -1000m)$, see figures 5.10 to 5.13 respectively.

When we look at the results in the figure figures 5.10 to 5.13 we see that the errors in all four quadrant are of the same magnitude. Also the classification plots are similar. So both the tracking performance and the classification performance is independent of the position of the object with respect to the radar at the same range with respect to the radar.



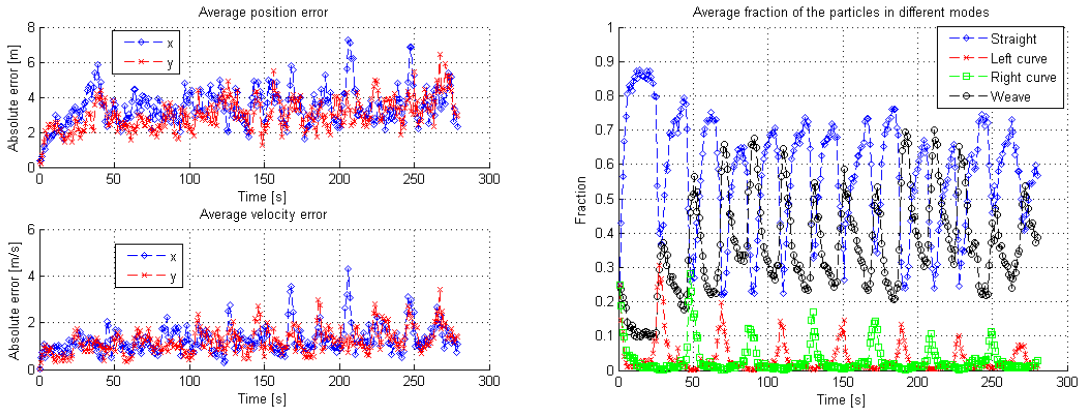
(a) Errors (b) Classification

Figure 5.10: Errors and classification for weave trajectory ending at $(x, y) = (1000m, 1000m)$



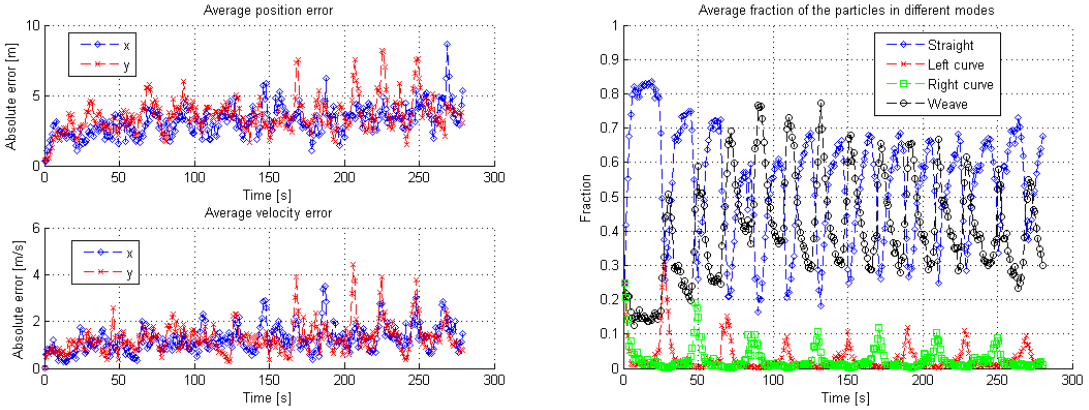
(a) Errors (b) Classification

Figure 5.11: Errors and classification for weave trajectory ending at $(x, y) = (-1000m, 1000m)$



(a) Errors (b) Classification

Figure 5.12: Errors and classification for weave trajectory ending at $(x, y) = (-1000m, -1000m)$



(a) Errors (b) Classification

Figure 5.13: Errors and classification for weave trajectory ending at $(x, y) = (1000m, -1000m)$

5.3.3 Orientation

The last limitation we investigated is the ability of our method to distinguish a weave trajectory when its overall direction differs, so for different orientations of the weave. The orientations of a weave trajectory we investigated are shown in figure 5.14.

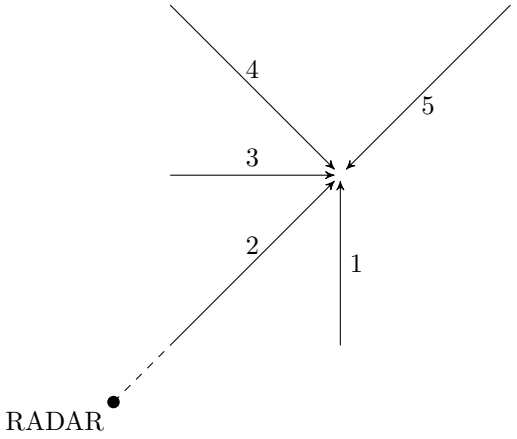
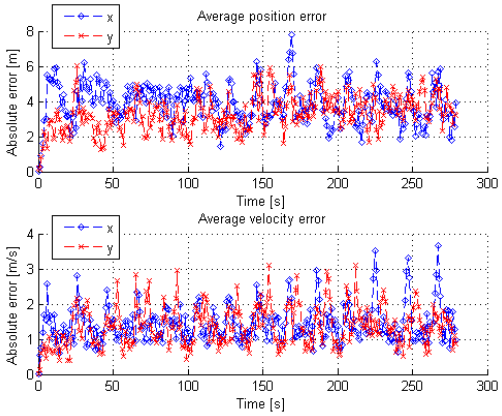


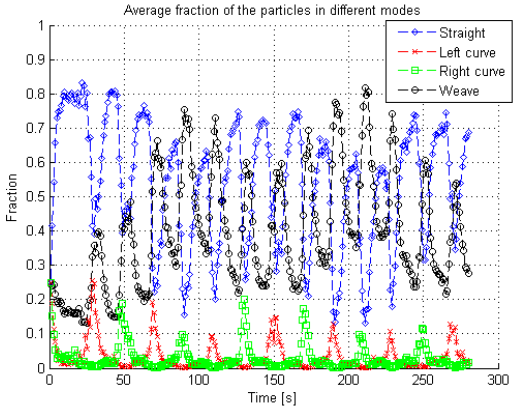
Figure 5.14: Schematic drawing of different overall orientations of weave trajectories.

For the true trajectory we used the weave trajectory of figure 5.4. We rotated this trajectory such that the orientations of figure 5.14 were investigated. All different trajectories ended at the point $(x, y) = (1000m, 1000m)$. The results using these different orientations are shown in figures 5.15 to 5.19.

When we look at the results of figures 5.15 to 5.19, we see that all figures except figure 5.19 show similar classification plots and the errors are of the same magnitude. The results in figure 5.19 show a slightly larger error and a slightly worse classification plot. This comes from the fact that the weave used as true trajectory for this orientation starts further away from the radar and ends at the point $(x, y) = (1000m, 1000m)$. Therefore it is expected to see a little worse classification and some larger errors as already stated in section 5.3.1. So we conclude that the orientation of the weave does not affect the tracking and classification ability of our method when these trajectories are at a similar range with respect to the radar.

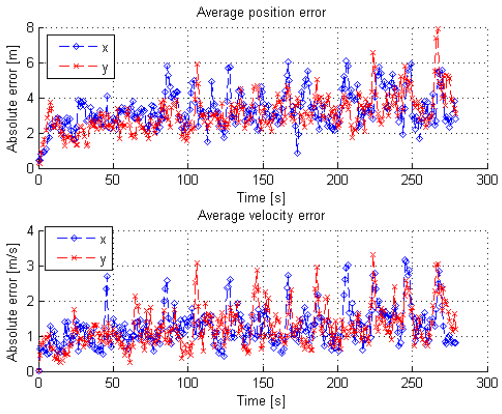


(a) Errors

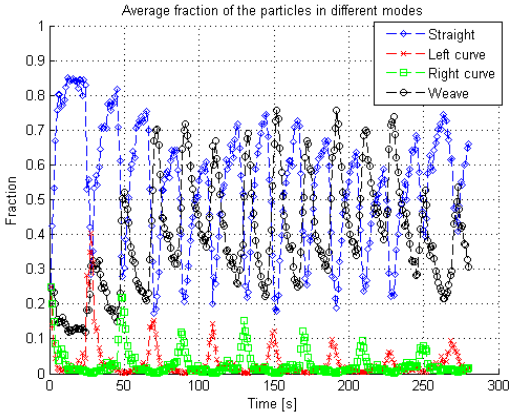


(b) Classification

Figure 5.15: Errors and classification using weave orientation 1

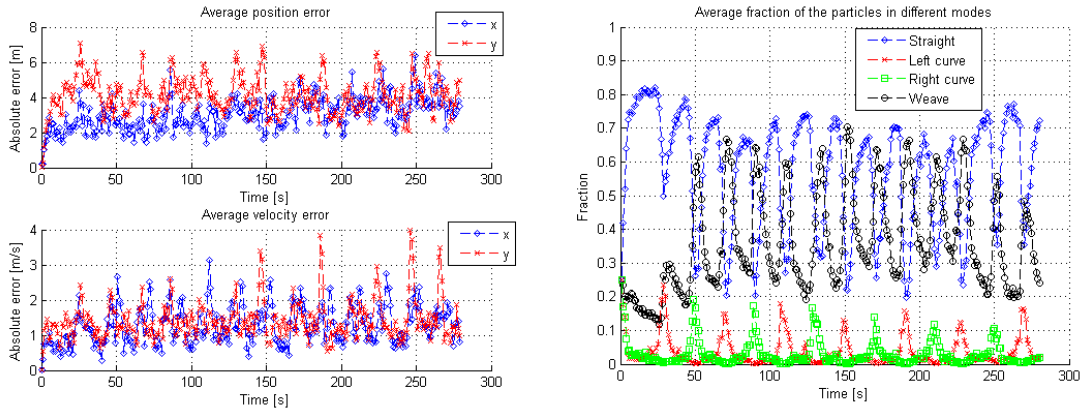


(a) Errors



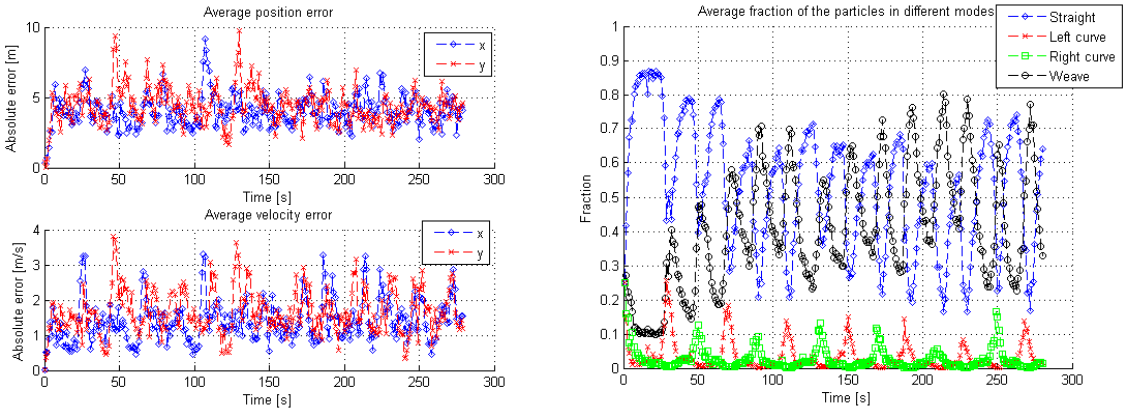
(b) Classification

Figure 5.16: Errors and classification using weave orientation 2



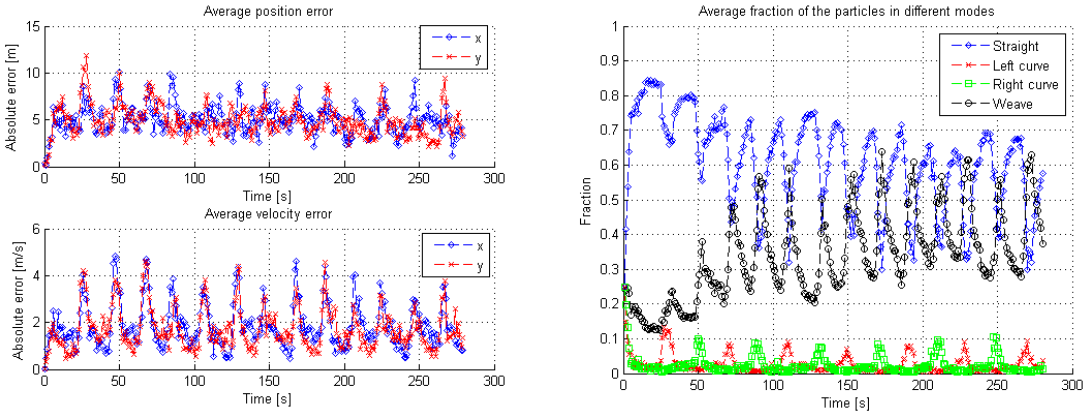
(a) Errors (b) Classification

Figure 5.17: Errors and classification using weave orientation 3



(a) Errors (b) Classification

Figure 5.18: Errors and classification using weave orientation 4



(a) Errors (b) Classification

Figure 5.19: Errors and classification using weave orientation 5

5.4 Preliminary conclusions

We can conclude that our method is able to distinguish weave trajectories from other trajectories when radar measurements are used in combination with settings which we do not have to adjust per trajectory in the sense that we can clearly distinguish the different classification plots from

each other. So in this sense our research question can be answered. But we also want to make it plausible that the classification can be done automatically at each time step. Therefore we have modeled a detector which is further explained in section 6.

Also we looked at the limitations of our method and can conclude that our method's tracking and classification performance gets worse when the range of the object with respect to the radar increases. The quadrant in which the trajectory lies and the orientation of the trajectory has no influence on the tracking and classification performance.

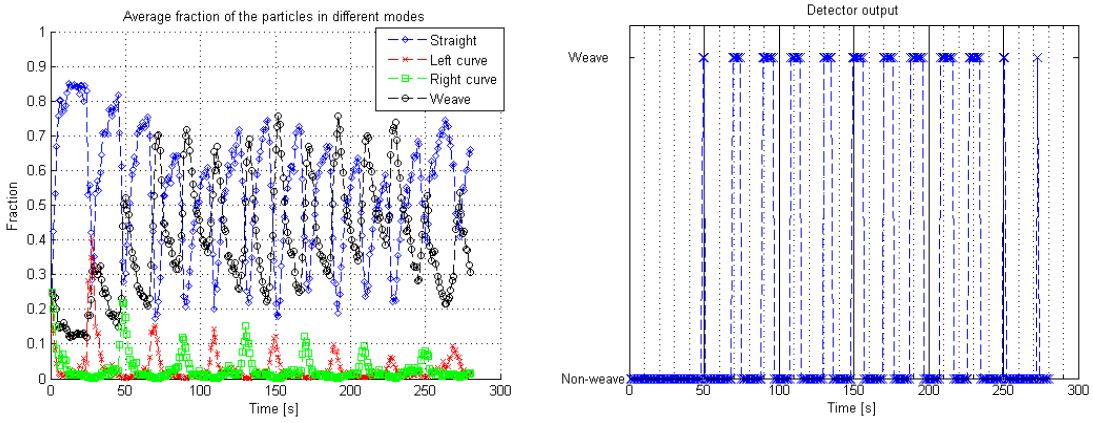
6. Detector

At this point in our research we are able to distinguish different classification plots when different trajectories are used as true trajectories in combination with radar measurements and with settings which do not have to be re-established for each trajectory, see section 5.2. But this classification is now done by looking at the plots after the simulation is completed. Since our goal was to help the radar operator in his decision making process, we want our method to track and classify the different trajectories simultaneously and as fast as possible. To make it plausible that this is actually possible we want to build a simple detector.

6.1 Simple detector

When we look at the classification plots in section 5 we can distinguish different types of classification plots but the classification itself is not immediately clear. The plots are hard to interpret by just looking at them. Therefore we want to build a detector which provides us with the most likely trajectory the object is following at each point in time.

First we used a very simple detector. The detector looks at the largest fraction of particles in the different modes and outputs that particular mode as being the most likely trajectory. When we look at figure 6.1a, where we see the classification plot, of a simulation using a weave trajectory as true trajectory, of figure 5.6 again, we already see that the fraction of particles in the mode weave is not always the largest fraction of particles. Hence when we look at the output of the proposed detector, figure 6.1b, we see that the detector classifies this trajectory as not being a weave for many time step, but we want to see a clear classification of the trajectory being a weave at all time steps.



(a) Classification

(b) Detector output

Figure 6.1: Classification and detector output using largest fraction of particles in the detector

So we conclude that a detector which looks at the highest fraction of particles does not work. Therefore we looked at a different detector. Eventually we used a detector which is based on applying a threshold on the average fraction of particles in the mode weave over the past N time steps.

6.2 Final detector

The idea behind looking at the average fraction of particles in the mode weave over the past N time steps comes from the current classification plots. We see an alternation between a large fraction of

particles in the mode weave and a large fraction of particles in the mode straight, as explained in section 4.3. If we take the average fraction of particles in the mode weave over some time steps we hope to end up with a plot in which we 'filtered out' this alternation and make it easier to apply a threshold on these averages.

The detector we have built uses the average fraction of particles in the mode weave over the past N time steps. So at time step k we look at the following:

$$\begin{aligned} AF(k) &= \begin{cases} \frac{1}{N} \sum_{i=k-N+1}^{k} F(i), \text{ if } k \geq N, \\ \frac{1}{k} \sum_{i=1}^{k} F(i), \text{ if } k < N. \end{cases} \\ \text{When } AF(k) &< \tau, \text{ no weave detected,} \\ \text{when } AF(k) &\geq \tau, \text{ weave detected.} \end{aligned}$$

In which N is the window length, k the current time step, $F(i)$ the fraction of particles in the mode weave at time i, $AF(k)$ the averaged fraction of particles in the mode weave at time k and τ the threshold for detecting a weave.

By looking at the average fraction of particles we expect our detector to be more robust. A weave trajectory should be followed for a longer period of time, therefore we expect the average fraction of particles over a most of this period to be larger then a certain threshold. The window length N should be chosen such that a weave is expected to last longer than N time steps. This to make the detector robust. If we would choose a window length which for example only covers a straight-curve-straight segment of a weave trajectory, we would get a lot of weave classifications for straight-curve-straight trajectories thereby giving false alarms. By choosing the window length large enough we will not classify such trajectories as being a weave trajectory and thereby decreasing the false alarm rate.

Also when we look at a part of the weave trajectory of N time steps we need to be able to classify that part of the trajectory as a weave trajectory ourselves since otherwise we would not need such a detector after all since we then are able to classify only parts of a weave trajectory already as a weave trajectory. We need to determine the threshold τ later on such that a weave is classified as such before the object has followed a weave trajectory for N time steps, since we want to outperform the radar operator. Taking all this in consideration we choose $N = 80$.

The threshold τ is determined by performing simulations using the different trajectories already used throughout this research. This threshold should be chosen such that a fast weave detection is possible and trajectories like a straight-curve-straight trajectory are not classified as a weave. Also we want to keep the threshold the same for all scenarios. We do not want to re-establish this threshold for every trajectory we encounter. We find a fast weave detection important since our detector should help the radar operator in his decision making process. So we need to be able to make a fast detection of a weave trajectory such that the radar operator can decide which actions need to be taken. When the radar operator has made this decision, the classification performance of that certain trajectory from that point on is not that important anymore since the radar operator already made a decision about which actions need to be taken. After performing different simulations we choose $\tau = 0.269$.

6.3 Results using final detector

Using $N = 80$ and $\tau = 0.269$ we performed simulations using the trajectories in figure 4.1 as true trajectories in combination with radar measurements. The results using a weave trajectory, trajectory 1 and trajectory 2 are shown in sections 6.3.1 to 6.3.3. The results for other trajectories can be found in appendix D.

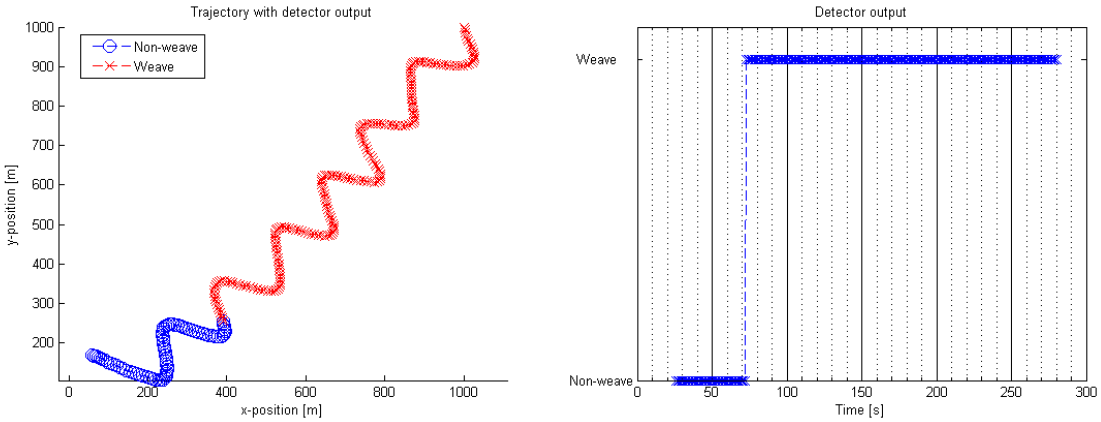
For these simulations we used 100,000 particles and performed only 1 Monte-Carlo run. In real life we also only have 'one run' to make our classification. Since we only have one run we have increased the number of particles to get a better estimation of the probability density functions. Also during our research it turned out that our method performs well in general. Therefore multiple MC-runs are not needed anymore. So we perform only 1 MC-run with a large number of particles for the simulations using the detector.

Since we initialize our particles uniformly over all modes we see that our method has a startup time. During this period the object is tracked but the classification is not yet reliable. Therefore

we do not take the fraction of particles in the mode weave in consideration during this startup time. We disregard the detector output during this period. For the simulations in this section we considered our method to have a startup time of 25 time steps.

6.3.1 Results for a weave trajectory

To investigate the performance of our detector when the object follows a weave trajectory we first looked at the weave trajectory of figure 5.4 as true trajectory, see figure 6.2. This trajectory starts at the leftmost point in figure 6.2a.



(a) Trajectory with detector output

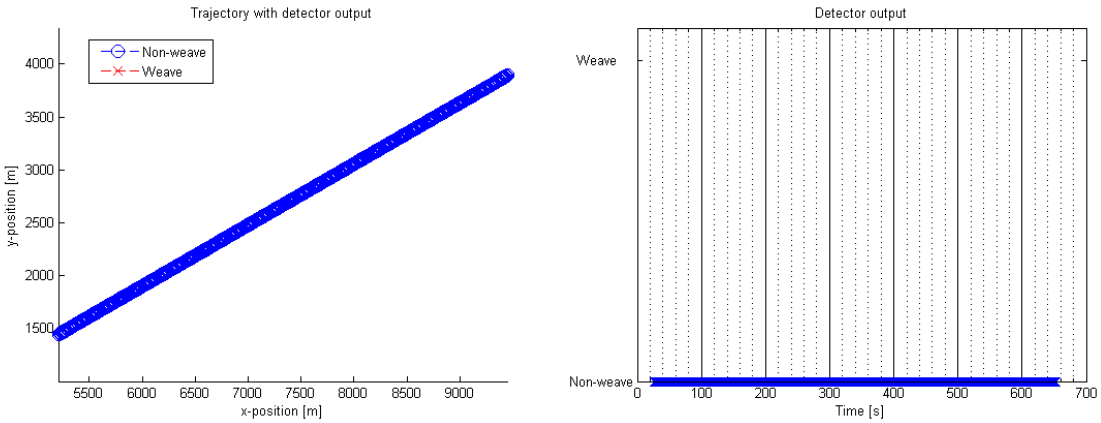
(b) Classification

Figure 6.2: Weave trajectory with detector output and classification

We see that our detector classifies this trajectory as a weave trajectory right after the object goes straight again after the third turn in alternating directions. We chose the window over which we average the particles to be 80 time steps. This window length was chosen such that we ourselves would classify a weave trajectory as such. Therefore we want our detector to classify this trajectory as a weave trajectory within 80 time steps, since we want to outperform the radar operator. It turns out that this is indeed the case, see figure 6.2b.

6.3.2 Results for trajectory 1

In figure 6.3 we see the classified trajectory and detector output for which trajectory 1 is used as true trajectory. The object follows a straight line trajectory. Our detector classifies this trajectory as a non-weave trajectory as can be seen in figure 6.3b.



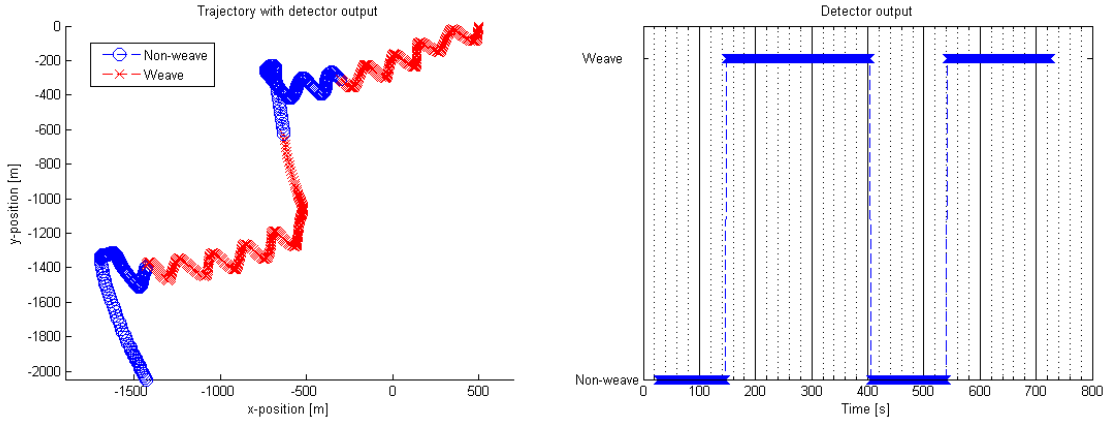
(a) Trajectory with detector output

(b) Classification

Figure 6.3: Trajectory 1 with detector output and classification

6.3.3 Results for trajectory 2

In figure 6.4 we see the classified trajectory and detector output for which trajectory 2 is used as true trajectory. The object starts by going straight for some time, then it makes two right turns and starts a weave trajectory. After some time following a weave trajectory the object goes straight again then makes a long left turn and starts following a weave trajectory again.



(a) Trajectory with detector output

(b) Classification

Figure 6.4: Trajectory 2 with detector output and classification

The first weave trajectory the object follows is classified as such by our detector very soon. This comes from the fact that right before this weave trajectory starts with a left turn, the object makes a right turn and goes straight for some time. Since these two turns have opposite directions we see that the fraction of particles in the mode weave already increases a bit. Therefore when the second turn of the weave trajectory is entered we see that our detector already classifies this trajectory as being a weave trajectory.

After this first weave trajectory ends we see that our method holds on to the classification of a weave trajectory while the object is already going straight for a longer period of time. This comes from the fact that we use the average fraction of particles in the mode weave over the last 80 time steps. In this way we take into account the history of the mode of the the particles. Therefore the detector holds on to this weave classification, it is not very reactive. But this is a phenomenon we do not find disturbing. Since the detector's intention is to help the radar operator in his decision making process, we need the detector to detect a weave trajectory fast. When a weave is detected, the radar operator makes a decision on which actions he needs to take. When the object then stops following a weave trajectory it initially does not matter that the detector classifies the straight line trajectory which is now followed as a weave trajectory, since the radar operator has already made a decision on which actions to take at that point in time. So we take this wrong classification for granted.

Also when the second weave trajectory is followed by the object the detector does classifies this as a weave trajectory. In this case the detector classifies this trajectory as a weave trajectory after the fourth turn of the object in alternating directions.

Remark

It is important to keep in mind that the detector we have build is only used to show that it is plausible for our method in combination with a detector to simultaneously track and classify an object. We came up with a simple detector which fulfills this purpose. Due to time constraints we did not focus on extending the analysis of this detector. Therefore a detailed analysis of the detector, including an analysis of the false and missed detection and the impact of them has not been performed. We have built this detector merely to show that simultaneous tracking and classification is possible without having to establish different settings for this detector for each different scenario or trajectory.

7. Other methods for trajectory analysis

For our research we investigated whether Markov chains can be used for trajectory analysis. During this research we also found literature about other methods which perhaps could be used for trajectory analysis. These methods have some similarities to the method we developed. Therefore we wanted to carry out some initial research on these methods to investigate whether these methods should be investigated thoroughly in later research as alternative methods for trajectory analysis.

The first method we investigated is stochastic context-free grammar. Grammar theory originated in computational linguistics to understand the structure of natural languages and is nowadays widely used in biology for the probabilistic modeling of RNA structures. Bluntly said grammar divides a sentence into words and those words are then again divided into letters. The grammar states how the letters, words and sentences are related to each other. It is this subdivision which perhaps could be used in trajectory analysis: a scenario (a sentence) is divided into trajectories (words) and these trajectories are again divided into straight and curve segments (letters).

The second method we investigated are hierarchical hidden Markov models. During our research on grammars we found some relations between different grammars and Markov models. In articles about grammars the use of hierarchical Markov models is also spoken of. Therefore we wanted to further investigate hierarchical hidden Markov models.

An introduction on grammars, following [17], together with an example, following [14], of the use of stochastic context-free grammars in trajectory analysis is given. Then we present some theory on hierarchical hidden Markov models, following [13]. Next some relations between different grammars and Markov models is presented.

Keep in mind that the work we present here is merely some initial research into both methods. It is not our intention to give a detailed analysis of these methods, we just want to present a general overview of these methods, their working and possible use for trajectory analysis.

7.1 Introduction on grammars

Grammars are used in the field of formal languages and computational linguistics. A formal language is a collection of strings having a certain predefined structure. The main questions of the theory of formal languages are:

- *String generation*: Given a language, how can we derive any string from this language?
- *String parsing*: Given a certain string and language, how can we tell if this string is part of this language?

The finite-dimensional models of languages that help answering these fundamental questions are called grammars. A *deterministic grammar* G is a four-tuple

$$G = (\mathcal{A}, \mathcal{E}, \Gamma, S_0),$$

where:

$\mathcal{A}$ is the alphabet (the set of terminal symbols of the grammar);

$\mathcal{E}$ is the set of non-terminal symbols of the grammar;

Γ is the finite set of grammatical production rules;

S_0 is the starting non-terminal.

The terminal symbols can be viewed as observation symbols which are generated from non-terminal symbols. A non-terminal symbol is a symbol which can not be observed and can be

thought of as an intermediary symbol which is eventually replaced by a single or combination of terminal symbols. In general, Γ is a partially defined function of the type $\Gamma : (\mathcal{A} \cup \mathcal{E})^* \rightarrow (\mathcal{A} \cup \mathcal{E})^*$, where $(\mathcal{A} \cup \mathcal{E})^*$ is the infinite set of all finite strings formed by concatenation of symbols from $\mathcal{A}$ and the empty string. We can see these production rules as a rewriting operation. In the rest of this section we will write non-terminal symbols as capital letters and terminal symbols using lower case letters. Using the definition of a deterministic grammar we can redefine a language in terms of its grammar $\mathcal{L} \triangleq \mathcal{L}(G)$.

So far, the formulation of the production rules is very general in the sense that no restrictions on this rewriting operation are given. The properties of the production rules are used by N. Chomsky to develop a hierarchy known as the Chomsky hierarchy of grammars:

- *Regular grammars (RG)*: Only production rules with one non-terminal on the left-hand side of the production rule and either one terminal or one terminal followed by a non-terminal on the right-hand side are allowed. So only production rules of the form $S \rightarrow a$ and $S \rightarrow aS$ are allowed. Regular grammars are also referred to as finite-state grammars.
- *Context-free grammars (CFG)*: The left hand side of the production rule must contain one non-terminal only and the right hand side may be any string. So it has the following form: $S \rightarrow \gamma$ with $\gamma \in (\mathcal{A} \cup \mathcal{E})^*$.
- *Context-sensitive grammars (CSG)*: Production rules of the form $\alpha_1 S \alpha_2 \rightarrow \alpha_1 \beta \alpha_2$ are allowed with $\alpha_1, \alpha_2 \in (\mathcal{A} \cup \mathcal{E})^*$ and β a non-empty string. So the allowed transformations of non-terminal S are dependent on its context α_1 and α_2.
- *Unrestricted grammars (UG)*: Any production rules of the form $\alpha_1 S \alpha_2 \rightarrow \gamma$ with $\alpha_1, \alpha_2, \gamma \in (\mathcal{A} \cup \mathcal{E})^*$.

This hierarchy of the grammars is used by Chomsky to classify the languages in terms of the grammars that can be used to define them: Context-sensitive language is a more restricted form of unrestricted language, context-free language is a special case of context-sensitive language and regular language is a special case of context-free language.

Some practical applications in which grammars could be used contain certain amounts of uncertainty that are often represented by probabilistic distributions. These factors require the extension of the grammars described above. We define a stochastic grammar. A *stochastic grammar* G_s is a five-tuple

$$G_s = (\mathcal{A}, \mathcal{E}, \Gamma, P_s, S_0),$$

where:

$\mathcal{A}$ is the alphabet (the set of terminal symbols of the grammar);

$\mathcal{E}$ is the set of non-terminal symbols of the grammar;

Γ is the finite set of grammatical production rules;

P_s is the set of probability distributions over the set of production rules Γ;

S_0 is the starting non-terminal.

Stochastic grammars are classified and analyzed on the basis of their underlying characteristic grammars. A characteristic grammar is obtained from the stochastic grammar by removing the probability distribution from the grammar definition. If the underlying characteristic grammar is an finite-state (or regular) grammar, the stochastic grammar is called a stochastic finite-state grammar (SFSG). If the characteristic grammar is a context-free grammar, the stochastic grammar is referred to as a stochastic context-free grammar (SCFG).

7.2 Example of the use of stochastic context-free grammar in trajectory analysis

In [14] a stochastic context-free grammar approach is used for intent inference. We present this example here, following [14].

In the tracking application the terminals of the grammar correspond to modes exhibited by the target. A target mode is defined as the direction of acceleration or orientation of the target at any

given time instant. The possible modes belong to the set $\Theta = \{0, \frac{\pi}{4}, \frac{\pi}{2}, \frac{3\pi}{4}, \pi, \frac{5\pi}{4}, \frac{3\pi}{2}, \frac{7\pi}{4}\}$, which represent the radial angular direction of acceleration of the target. Each mode is associated with a lower case alphabet; $\mathcal{A}_\Theta = \{a, b, c, d, e, f, g, h\}$. The terminal symbols correspond to modes of the target which are generated from non-terminal symbols. A non-terminal symbol can be thought of as an intermediary symbol which is eventually replaced by a single or combination of terminal symbols. They can be abstracted as high-level components of the shape of the trajectory that a target might travel.

The trajectory generation process begins with a unique starting non-terminal symbol S_0. A set of production rules Γ with associated probabilities of selection P_s define the manner in which non-terminal symbols produce other non-terminals and terminals. The set of all strings that can thus be generated by a grammar is called a language. Hence, each different trajectory which is being considered is a language with a specific grammar to model them. The tracking application's goal is to find which model is able to best explain the observed trajectory.

The trajectories which are being considered in [14] are a line trajectory, an arc trajectory and a rectangular trajectory. The line trajectory is modeled using a regular or finite-state grammar. It can be expressed as $\mathcal{L}_{line} = \{\alpha \in a^+\}$, where α represents any arbitrary string belonging to the language and a^+ represents an arbitrary number of symbols corresponding to direction $a \in \mathcal{A}_\Theta$.

The language of an arc can be expressed as $\mathcal{L}_{arc} = \{\alpha \in c^n a^+ g^n\}$, where α represents any arbitrary string belonging to the language and $a, c, g \in \mathcal{A}_\Theta$. These strings contain an equal number of 'upward' symbols and 'downward' symbols corresponding to the directions c and g. The middle of the string is formed by an arbitrary number of symbols corresponding to direction a.

The rectangles which are being considered are defined as trajectories comprising of three left ninety degrees turns but are not necessarily closed. However, at least two opposite sides of the rectangle have to be of equal length. The language of these rectangles can be expressed as $\mathcal{L}_{rectangle} = \{\alpha \in c^n a^+ g^n e^+\}$, where α represents any arbitrary string belonging to the language and $a, c, e, g \in \mathcal{A}_\Theta$. These strings contain an equal number of symbols corresponding to the directions c and g and an arbitrary number of symbols corresponding to directions a and e.

In the tracking application a base level tracker is used to estimate the target's position and velocity. Next a second-tier tracker is responsible for estimating the mode of the target using inputs from the base-level tracker. The output of this secondary sensor is an estimate of the actual target mode and is represented by a string of symbols, from $\mathcal{A}_\Theta$, corresponding to the different directions. A meta level tracker is used to find the most likely trajectory corresponding to the outputted string. This is related to string parsing: 'Given a certain string and language, how can we tell is this string is part of this language?' In this example the trajectories under consideration are represented by the languages $\mathcal{L}_{line}$, $\mathcal{L}_{arc}$ and $\mathcal{L}_{rectangle}$ and for each of these languages a grammar is constructed. The Early-Stolcke parser is used to compute the posterior model probabilities. It accepts a string of terminals as input and outputs a most likely production rule sequence that led to the generation of the observed symbols. In a formal language context this is called stochastic parsing, since we are parsing the language corresponding to the trajectory of the target. The Early-Stolcke parser is a top down parser and it builds a parse three that best describes a sequence of terminals, namely, the modes followed by the target. The Early-Stolcke parser is able to compute the likelihood of a string being generated by a certain grammar. So in this example the Early-Stolcke parser computes the likelihoods of a string of symbols which are observed being generated by the different trajectories which are being considered. Hence we are able to find the trajectory which is most likely to be the trajectory the target follows.

7.3 Hierarchical hidden Markov models

The method we developed is similar to a hierarchical hidden Markov model (HHMM). A hierarchical hidden Markov model is related to a 'normal' hidden Markov model (HMM). In a HHMM each state itself is again a HHMM, so hierarchical hidden Markov models are structured multi-level stochastic processes, just like our method.

Each state of a HHMM emits a sequence of symbols instead of a single symbol. Whenever a state in a HHMM is activated, it activates one of the substates of that state. This substate might also consist of substates of which one is activated etc. The activation of substates ends whenever a special state called a production state is reached. The production states are the only states which emit output symbols, which happens in the same way a 'normal' hidden Markov models emits output symbols. Hidden states which do not emit output symbols are called internal states.

When the production state has emitted a symbol, control returns to the state that activated the production state.

This hierarchical structure can be related to our method. We can see the production states as straight or curve segments. The higher levels can be seen as possible trajectories or parts of trajectories and at the top node the scenario which need to be classified itself.

Activation of a substate by an internal state is called a vertical transition. After such a vertical transition is completed, a horizontal transition occurs. This is a transition to a substate within the same level. When after a horizontal transition a terminal state is reached, control is returned to the substate in a higher level from which the last vertical transition originated. Note that multiple vertical transitions can occur before a sequence of production states is reached. Therefore the state at the top level produces sequences of observation symbols instead of a single observation symbol. See figure 7.1 for a schematic drawing of the structure of a hierarchical hidden Markov model.

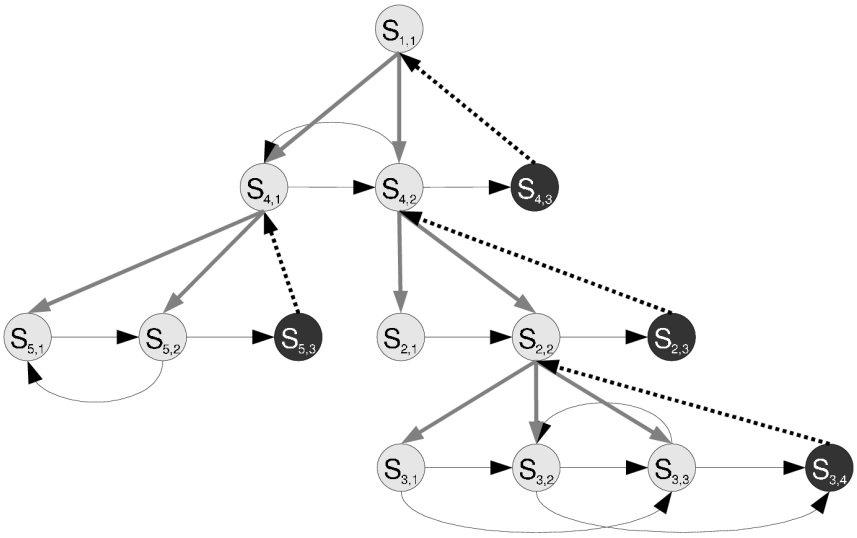


Figure 7.1: Schematic drawing of the structure of a hierarchical hidden Markov model

In figure 7.1 the vertical transitions are indicated by gray lines. Black lines represent the horizontal transitions. The internal states are indicated by light gray circles and the terminal states by dark gray circles. The production states are not shown in this figure.

It is important to note that every hierarchical hidden Markov model can be represented as a 'standard' single level hidden Markov model. The states of the HMM are the production states of the corresponding HHMM with a fully connected structure, i.e. there is a non-zero probability of moving from any of the states to any other state. The methods for estimating the parameters of a HHMM are more complex than for a HMM. However a HHMM exploits its hierarchical structure to solve certain problems more efficiently.

7.4 Relations between different grammars and Markov models

In this section we present some relations between different grammars and Markov models, following [17].

The grammars introduced in section 7.1 are related to Markov chains. Discrete-state discrete-time Markov chains are naturally considered the equivalent representations of stochastic finite state languages. A discrete-time Markov chain can be defined as stochastic timed automaton. However Markov chains do not accommodate for the alphabet of the grammar. Therefore, Markov chains can only capture the transition dynamics of the grammar, but do not address the generation and transformation aspects of the stochastic finite-state grammars. Hidden Markov models address this issue.

Generally speaking hidden Markov models are Markov chains which are indirectly observed trough a noisy process. To accommodate for the structural constraints of the hidden Markov model, the underlying Markov chain has to be converted to the Moore machine. Using this transformation

the hidden Markov model representation of a stochastic finite state grammar can be made which does accommodate the alphabet of the grammar.

Some context-free grammars can also be represented by a Markov chain. When a context-free grammar is non-self-embedding, it generates a finite-state language. A context-free grammar $G = (\mathcal{A}, \mathcal{E}, \Gamma, S_0)$ is *self-embedding* if there exists a non-terminal symbol $A \in \mathcal{E}$ such that a string $\alpha A \beta$, with α and β any non-empty strings of terminal and non-terminal symbols, can be derived from it in a finite number of derivation steps. So when a context-free grammar is non-self-embedding it can be represented by a Markov chain since it generates a finite-state language.

8. Conclusion & Discussion

In this chapter we summarize the results of the research we carried out and discuss these. Some advantages and disadvantages of the different methods for trajectory analysis are given. Also recommendations for further research are presented.

8.1 Conclusion

As said in the introduction the goal of this project is to study the suitability of Markov chains for trajectory analysis. More specifically, the research question is whether modeling of target trajectories using Markov chains can be used to distinguish a number of trajectories that are encountered in real-life situations within a radar environment. Where the focus lies on distinguishing a weave, consisting of straight lines and curves, from other trajectories that also consist of straight lines and curves. The proposed methods need to be robust for variations in trajectories. Also the classification of the type of trajectory the object is following must take place while tracking the the object.

At the end of our research we are able to track and classify different trajectories. Our method models all possible target trajectories as combinations of sub-trajectories trough a hierarchical state diagram. We use the two discrete variables mode and submode to identify the possible trajectories and possible sub-trajectories respectively. The transitions between the different modes and different submodes are governed by first-order Markov chains. These Markov chains are incorporated in a particle filter. Due to the stochastic aspect of these Markov chains our method is robust against some variations in the trajectories.

First we looked at the viability of our method by performing simulations using simulated radar measurements created by ourselves. These simulations gave us insight in the working of the different aspects of our method. Next we used GPS data loggings from Thales to investigate the performance of our method using GPS measurements. Eventually we determined the final settings of our method by performing simulations using radar measurements which were created by converting the GPS data loggings into radar measurements. The settings we determined are general in the sense that we do not have to re-establish these settings for different trajectories.

To automatically classify a trajectory as being a weave or a non-weave trajectory we have built a detector. Our focus on distinguishing a weave trajectory comes from the fact that this type of trajectory might be performed during an attack. The settings for the detector are determined by performing many simulations using different trajectories. Again the settings we determined are general in the sense that we do not have to re-establish these settings for different trajectories. The results look promising and demonstrate the feasibility of automatic weave detection.

8.2 Discussion

For determining the settings of our method we performed many simulations. We do not claim that these settings are optimal in the sense that we get the smallest possible errors and largest probability of detection. We do however state that we have found settings such that the results look promising and demonstrate the feasibility of automatic weave detection. Also we have found general settings which we do not have to re-establish for different trajectories. Due to time constraints we did not focus on optimizing the settings such that the errors were minimized etc. Also due to these time constraints we did not perform a detailed analysis of the detector, including an analysis of the false and missed detection and the impact of them.

Our research was carried out at Thales. Here we only had a limited number of real data loggings which we could use for our research. Therefore we could only investigate the performance of our method on a limited number of trajectories. So the robustness of our method regarding variations in trajectories is investigated, but only using this limited number of trajectories.

A last thing to note is that perhaps strictly speaking we did not use a first-order Markov chain which governs the transitions of the submode. For determining the entries of the submode transition probability matrix we make use of variables which keep track of the distance traveled in the current segment, namely δ_k and θ_k. These variables contain a form of history of the particle. Although the transition probabilities only depend on the current state, since δ_k and θ_k are contained in the state vector, they do implicitly depend on some form of history. Therefore strictly speaking we can not state that the submode transitions are governed by a first-order Markov chain, but these transitions do have a Markovian property. We did however state that the submode transitions are governed by a first order Markov chain since we feel like explaining this technicality earlier on would only confuse the reader.

Advantages and disadvantages of the different methods for trajectory analysis

In section 7 we introduced some other methods for trajectory analysis. We will now look at some advantages and disadvantages of these methods.

One motivation behind using stochastic context-free grammar models as for the arcs and rectangles in the given example is that such trajectories cannot be exclusively generated by Markov models. This implies that a Markov chain can generate a sample path that is an instance of a shape with some finite probability. However, it cannot do so with probability 1 unless the state space is extended artificially to the length of the trajectory. [1] In other words different targets can follow the same type of trajectory but at a different speed or scale. The recursive self-embedding property of a stochastic context-free grammar imparts scale-invariance to the trajectory analysis task. A grammar is able to generate all sizes of trajectories satisfying that certain shape. Therefore it is better able to cope with this scale-invariance in comparison to Markov models.

A disadvantage of using stochastic context-free grammars is that for the methods we found in the literature the shapes of the trajectories were orientation-dependent. So they were scale-invariant but orientation or rotation-dependent e.g. a upward facing arc or a line trajectory in the radial direction of $0rad$. So it seems like different languages with corresponding grammars are needed for different orientations of the same shape. Perhaps once a grammar for a certain shape or trajectory is constructed it can easily be rewritten to a different orientation. But no details on this were given in the articles we found.

Although it is stated in [13] that a hierarchical hidden Markov model is a simpler version of stochastic context-free grammar, we find that [15] disputes this statement. There is a huge explosion in the number of parameters for a hierarchical hidden Markov model with respect to a stochastic context-free grammar. A HHMM needs a initial distribution, all the internal hidden states must have a transition distribution and also an emission distribution for each hidden state which produces the output alphabet is required. The complexity of a HHMM is quadratic in the length of the observation but it has an exponentially larger number of parameters. When we look at the results shown in [15], we see that the HHMM parser requires a longer run-time compared to a parser for the stochastic context-free grammar.

Also as stated before we can represent every hierarchical Hidden Markov model as a 'normal' hidden Markov model. Therefore hierarchical hidden Markov models are also not scale-invariant. Making it a less favorable method over stochastic context-free grammars when different scales of shapes are being considered.

8.3 Recommendations for further research

Our research was carried out at Thales. Here we only had a limited number of real data loggings which we could use for our research and from a naval standpoint we focused on distinguishing a weave from other trajectories. We recommend further research on the performance of our method when more trajectories are being considered. Also research using more real data is recommended to further investigate our method's tracking and classification performance.

We recommend performing a detailed analysis of the detector regarding the false and missed detection probability and the impact of them.

The dynamical models used in the different modes are exactly the same. Therefore we see a alternation between fraction of particles in the modes straight and weave in the classification plots

[1] A formal proof of this assertion can be constructed using the pumping lemma for regular grammars.

when the object follows a weave trajectory. Perhaps it is possible to use different dynamical models in the modes straight and weave such that the classification becomes more clear.

In our research we stated that we can better overestimate the values of δ_{max} and θ_{max} rather than underestimating these values. Therefore we chose this values such that all segments of weave trajectories we encountered were smaller than these variables. We also concluded that better estimations of δ_{max} and θ_{max} reduced the errors of the position and velocity estimates and increased the classification performance. Therefore we recommend research on the possibility of implementing some form of automatic learning of the magnitudes of δ_{max} and θ_{max}. Since when it is possible to better estimate these parameters automatically for each weave trajectory which is encountered it will increase the method's tracking and classification performance.

If it is possible to implement the automatic learning of the variables δ_{max} and θ_{max} the method will be more robust for variations of the different trajectories. This robustness is probably also reached when a stochastic context-free grammar approach is used. As stated before a stochastic context-free grammar model is scale-invariant. It models the shape of the trajectory such that any scale of that trajectory can be generated. Therefore we recommend further research into the use of stochastic context-free grammar to see if this kind of method is more robust for variations of the different trajectories.

When the use of a stochastic context-free grammar is investigated, we recommend to also investigate the possibility of incorporating the models for the different trajectories into the tracking application. In our method we make use of a joint tracking and classification application. The models for the different trajectories are incorporated in the tracking application of our method. Where in the methods using stochastic context-free grammar we have seen so far the models for the different trajectories are separated from the tracking application. Therefore only a feedback loop from the detector using the stochastic context-free grammar models back to the tracking application is perhaps possible but a joint tracking and classification approach can not be achieved. Also the dependency of the different models on the orientation of the trajectories need to be investigated further when a stochastic context-free grammar approach is used.

Abbreviations & symbols

ACT-model	augmented coordinated turn model
$AF(k)$	averaged fraction of particles in the mode weave at time k
b_k	bearing at time k
CM-model	curvilinear motion model
d_k	Doppler velocity at time k
δ_k	already traveled length of the current straight segment at time k
δ_{max}	average length of a straight segment of a weave
$F(i)$	fraction of particles in the mode weave at time i
k	time index
m_k	discrete variable mode at time k
$m = 1$	straight line
$m = 2$	left curve
$m = 3$	right curve
$m = 4$	weave
MC-run	Monte-Carlo run
n	power of the function in the min-method
N	window length of final detector
NCV-model	nearly constant velocity model
p_{cs_k}	probability of a transition from a submode curve to a submode straight at time k
p_{sc_k}	probability of a transition from a submode straight to a submode curve at time k
p_1	probability to remain in the current submode in the if-else-method
p_2	probability to make a transition to another submode in the if-else-method
Π_m	mode transition probability matrix
Π_{sm_k}	submode transition probability matrix at time k
r_k	range at time k
sm_k	discrete variable submode at time k
$sm = 1$	curve left
$sm = 2$	straight after curve left
$sm = 3$	curve right
$sm = 4$	straight after curve right
σ_α	process noise parameter for angular velocity
$\sigma_{a_{x,k}}$	process noise parameter in x-direction at time k
$\sigma_{a_{y,k}}$	process noise parameter in y-direction at time k
$\sigma_{a_{T,k}}$	process noise parameter in tangential direction at time k
$\sigma_{a_{N,k}}$	process noise parameter in normal x-direction at time k
T	sampling time
τ	threshold for weave detection
θ_k	already traveled angle of the current curve at time k
θ_{max}	average angle of a curve segment of a weave
v_k	measurement noise at time k
v_x	velocity in x-direction
v_y	velocity in y-direction
w_k	process noise at time k
ω_k	angular velocity at time k
x	x-position
y	y-position

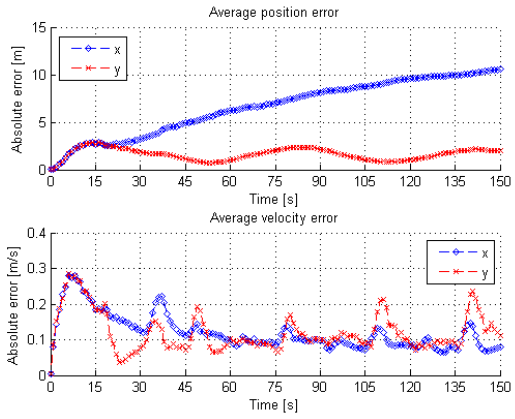
A. Extra results using simulated radar measurements

In this section we give some graphical results used to come to the preliminary results explained in section 3.

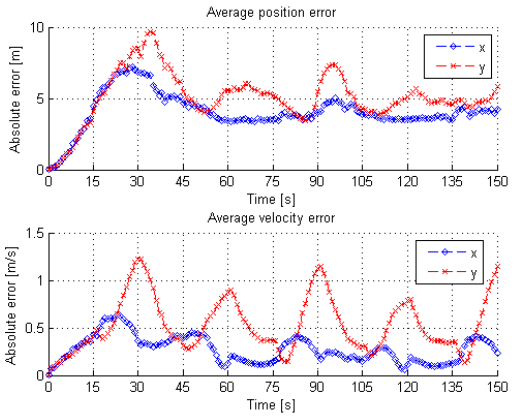
For all results in this appendix we performed 100 Monte-Carlo runs in which we generated the measurement noise separately for each run and 6,500 particles were used per run. The measurements were created to make the object follow a weave trajectory like figure 3.1a. The object starts at $x = 200m$ and $y = 200m$ and has initial velocities of $v_x = 1m/s$ and $v_y = 1m/s$. The shown results are averages over all runs.

Initial position

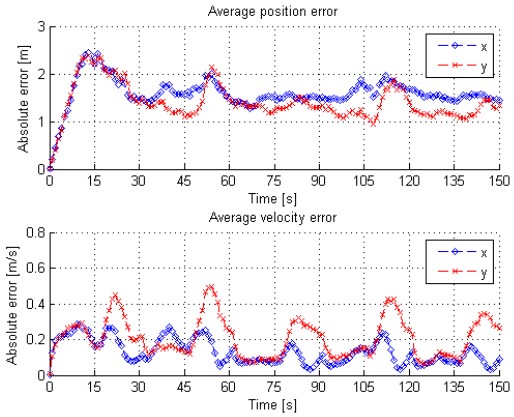
Different initial positions of the object result in different errors. If the object is too close to the radar the error increases due to the fact that the initial bearing angle can lie in a large interval. When the object is too far from the radar the error also increases due to the reduced performance of the radar itself. Average errors in position and velocity for different initial x- and y-position, or $s0_x$ and $s0_y$ respectively, are shown in figure A.1.



(a) $s0_x = 10m$, $s0_y = 10m$



(b) $s0_x = 5000m$, $s0_y = 5000m$



(c) $s0_x = 200m$, $s0_y = 200m$

Figure A.1: Errors of initial position comparison for a weave trajectory

We indeed see a larger position error when the object is too close or too far away as is the case for $s0_x = s0_y = 10m$ and $s0_x = s0_y = 5000m$ respectively. The error of the position for $s0_x = s0_y = 200m$ is smaller than the other cases.

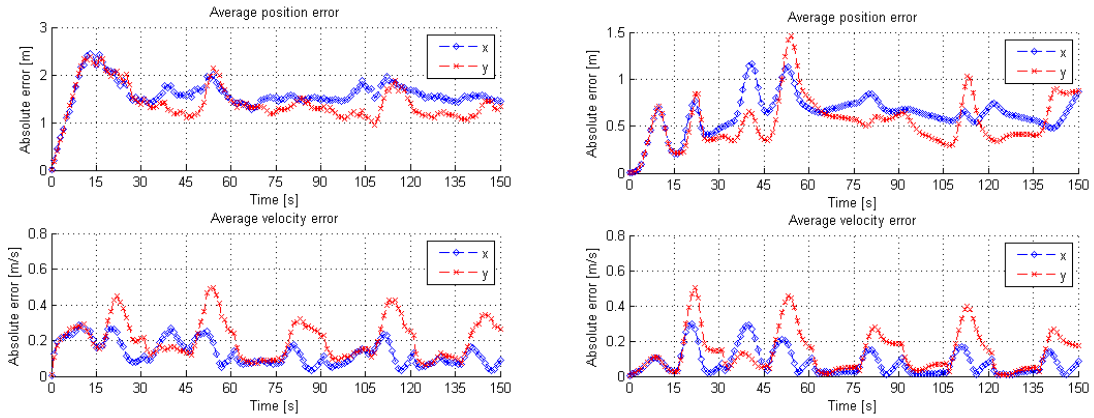
The velocity error is smaller when the object is closer to the radar. Also we see some peaks in the velocity errors. These peaks come from the object following a curve in the weave trajectory. The velocity is harder to estimate when the object is performing a curve, therefore we see an increase in the velocity errors in these curves.

In the case of $s0_x = s0_y = 10m$ we see that the position error is larger in the x-direction than in the y-direction. This comes from the fact that due to a large interval of initial bearing angles the different runs are widely distributed in the x-direction around the true trajectory. Therefore the position error is larger in x-direction.

Measurement noise

A radar has some specifics which give an indication of the radars performance. We model these specifics by the measurement noise. When the radar has a better performance we expect to see smaller errors.

We performed simulations where we used various magnitudes of measurement noise. Two of these simulation outcomes are shown in figure A.2.



(a) Normal measurement noise

(b) No measurement noise

Figure A.2: Errors of measurement noise comparison for a weave trajectory

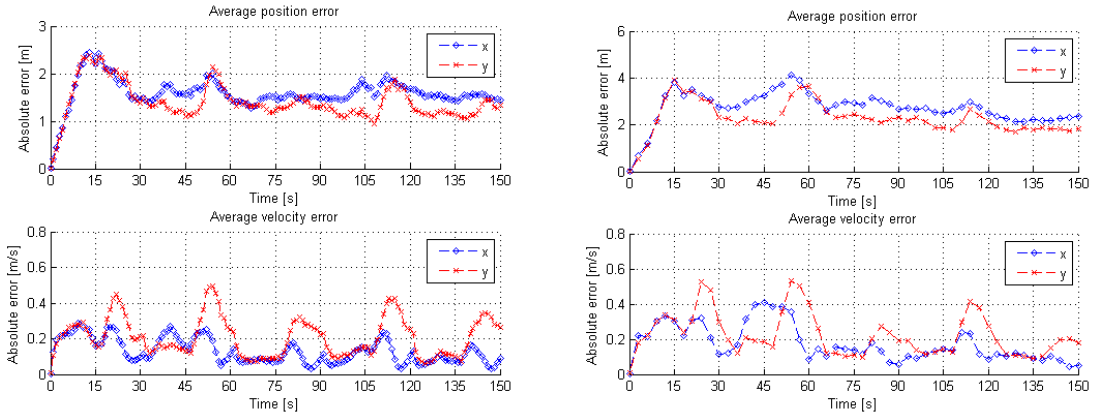
As expected we see smaller errors when no measurement noise is used in comparison to the case in which we do use measurement noise of a 'normal' magnitude.

Sampling time

The sampling time is a feature of the radar on which we normally have no influence. But to see if our method can cope with different sampling times we performed simulations in which we used different sampling times.

We expect to see a larger error in the position when a longer sampling time is used. This as an effect of the increased influence of the process noise on the output of our method. The outcomes for simulations with a sampling time of 1 and 3 seconds are shown in figure A.3.

When we look at these outcomes we indeed see a larger position error when a longer sampling time is used. But this error does not 'explode' so it looks like our method can cope with various sampling times.



(a) Sampling time of $1s$

(b) Sampling time of $3s$

Figure A.3: Errors of sampling time comparison for a weave trajectory

Mode transition probability matrix

During the first part of our research we defined the mode transition probability matrix, Π_m, in a different way compared to the defined Π_m in section 2.3.1. First we defined Π_m such that the transitions shown in figure A.4 were possible.

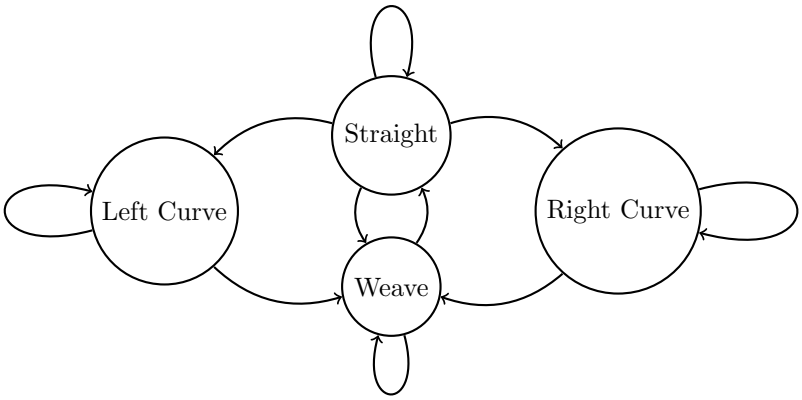


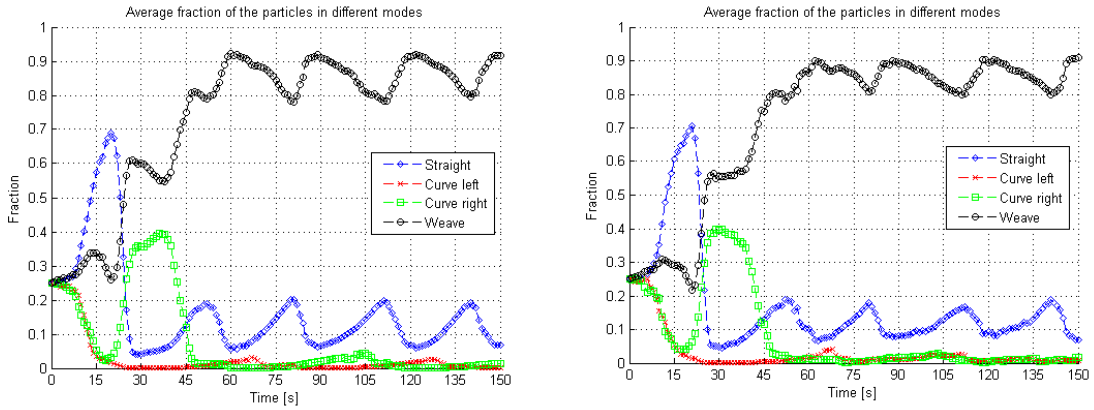
Figure A.4: Schematic representation of possible mode transitions used in early research

It is the case that the entries of Π_m do affect the outcome of our method directly since more particles are forced into certain modes due to these entries. We performed simulations using different mode transition probability matrices to investigate the effect of these changes in entries of Π_m. But since we used a significantly other Π_m in our later research we omit further outcomes of these simulations in this appendix.

Submode transition probability matrix

To determine the probabilities in the submode transition probability matrix we used two different method; the min-method and the if-else-method. We performed many simulations in which we used different settings for both methods to see which setting gave the best outcomes. For the min-method we varied n and used values between 1 and 25. For the if-else-method we varied p_1 between 0.01 and 0.075 and p_2 between 0.925 and 0.99. Also we varied the values of δ_{max} and θ_{max} and chose different values around the true value of these parameters. We compared the best outcomes of both methods. The results of the best outcomes of both methods are shown in figures A.5 and A.6.

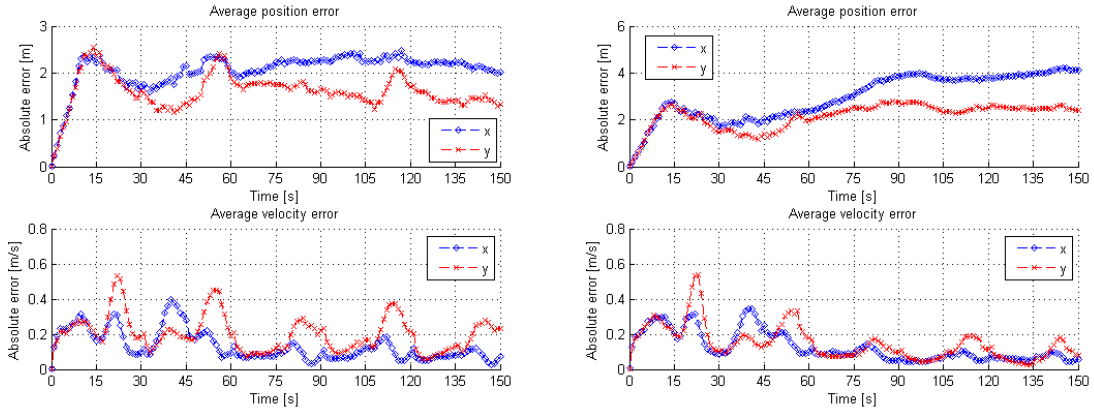
If we look at these results we see that both methods give almost the same classification. The fraction of particles in the different modes is comparable between both methods. But we see that



(a) If-else-method classification

(b) Min-method classification

Figure A.5: Classifications for a weave trajectory using different methods



(a) If-else-method errors

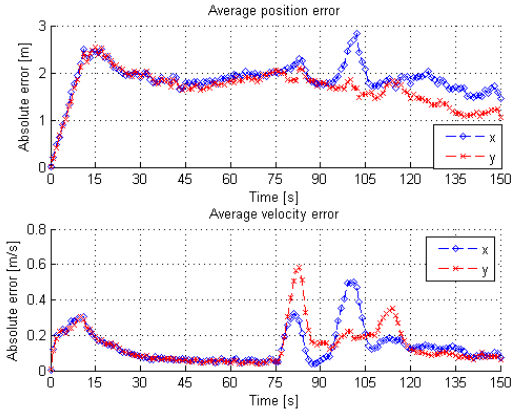
(b) Min-method errors

Figure A.6: Errors for a weave trajectory using different methods

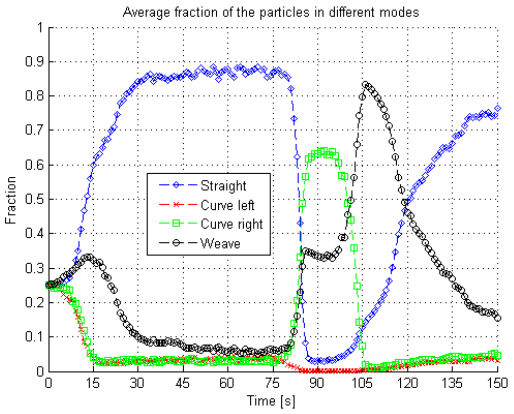
the error in position is smaller when we use the if-else-method. Therefore for now we prefer the if-else-method, but we need to compare both methods again when real measurements are used in our method.

Our research focuses on distinguishing a weave from other trajectories. So to see how our method performs when other trajectories are used which are not a weave we have created a straight-curve-straight trajectory as shown in figure 3.1b. We again compared the if-else-method with the min-method and the results are shown in figure A.7.

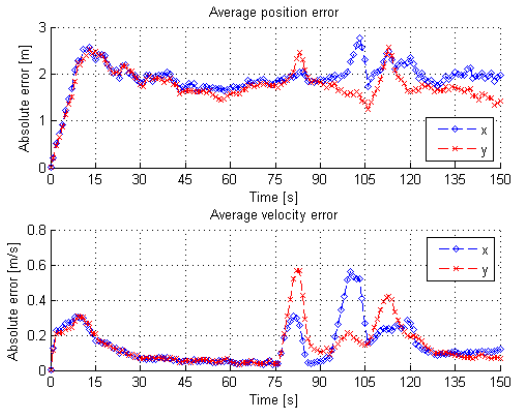
When we look at the results of figure A.7 we see that the magnitude of the errors for both methods is similar just as the classification of both methods. We also see that both methods have a high fraction of particles in the mode curve right when the object follows a curve. But we also see a high fraction of particles in the mode weave when this curve is followed for a longer period of time, so this looks like an undesirable result. On the other hand we can clearly distinguish the classification in figure A.7 of the classification in figure A.5. So also on this aspect our method looks viable.



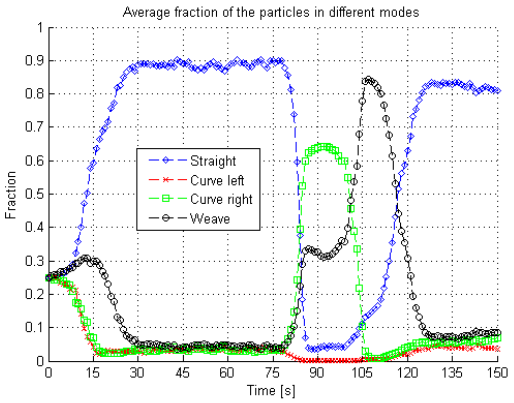
(a) If-else-method errors



(b) If-else-method classification



(c) Min-method errors



(d) Min-method classification

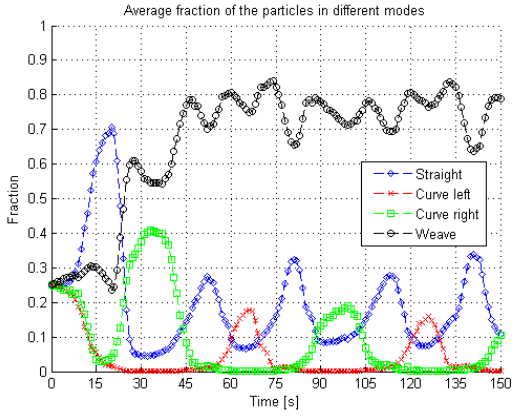
Figure A.7: Errors and classification for a straight-curve-straight trajectory using different methods

The determination of the probabilities in the min-method and the if-else-method makes use of the variables δ_{max} and θ_{max} which characterize the average weave trajectory. To see how we should choose these variables we performed simulations in which we varied the values of δ_{max} and θ_{max}.

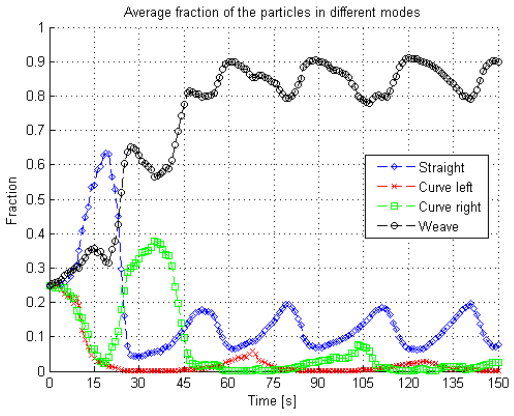
It turned out that a better estimation of δ_{max} and θ_{max} led to a better classification of a weave trajectory. This is an expected result since it is likely that the classification is better when we have a better knowledge on how the weave trajectory should look like.

To investigate the difference in classification outcomes between the if-else-method and the min-method we looked at the actual length of the straight and curve segments of a weave trajectory corresponding to the used measurements. Let the actual length of the straight segment be δ_{true} and the actual angle of the curve segment be θ_{true}. We used these variables in our estimations of δ_{max} and θ_{max} which we then used in different simulations to investigate the difference in classification. The results of some simulations are shown in figure A.8.

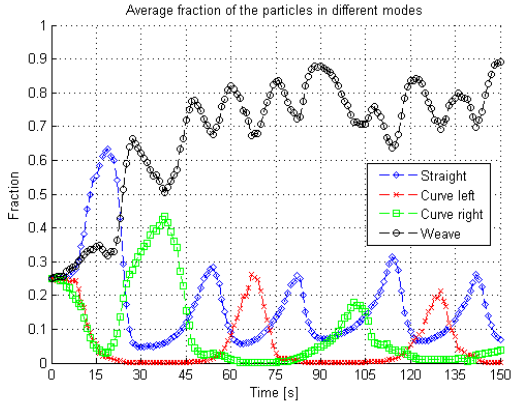
We see that we can better 'overestimate' δ_{max} and θ_{max} than 'underestimate' these variables. When these variables are too small, our method forces particles to make a transition from the current submode to another submode. This is so since the probability of making a transition out of the current submode to an other submode becomes large when the length of the straight and curve segments defined by δ_{max} and θ_{max} is reached. So our method feels like the segments of the measurements are too long to be a weave and therefore a worse classification is made. The errors of the simulations of figure A.8 are of the same magnitude and therefore omitted here.



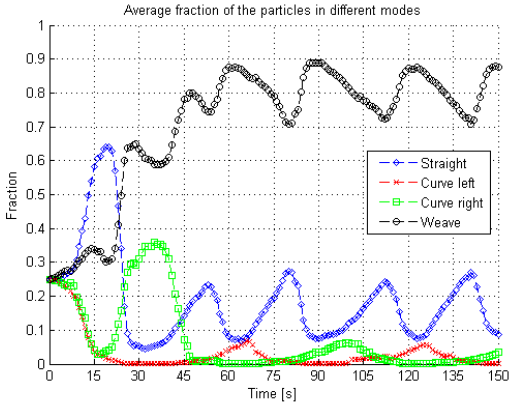
(a) $\delta max = 0.8\delta_{true}$ and $\theta max = \theta_{true}$



(b) $\delta max = 1.2\delta_{true}$ and $\theta max = \theta_{true}$



(c) $\delta max = \delta_{true}$ and $\theta max = 0.8\theta_{true}$

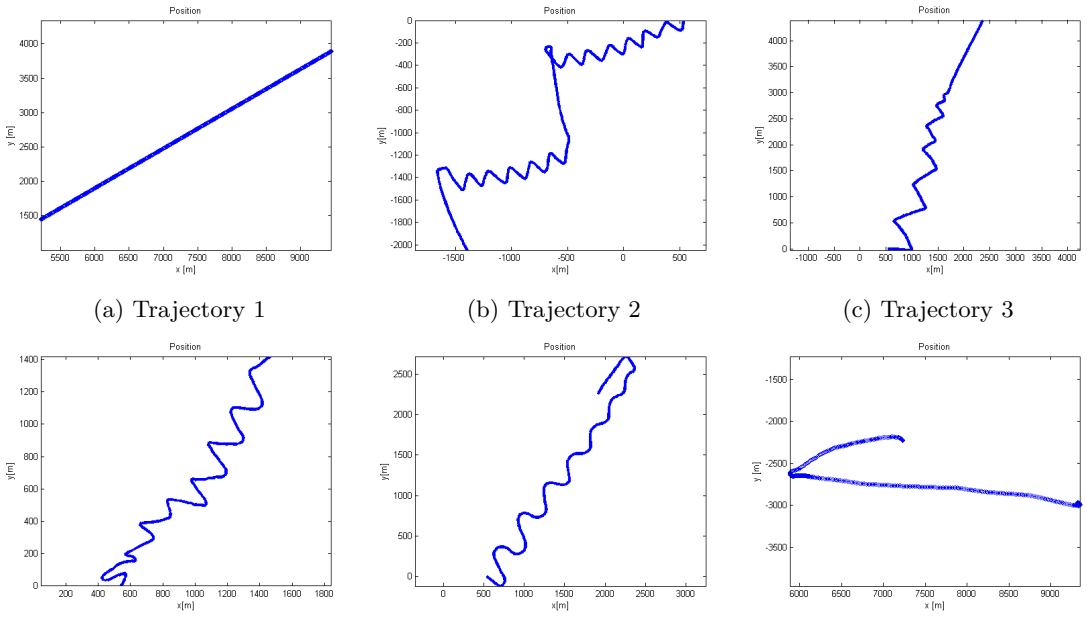


(d) $\delta max = \delta_{true}$ and $\theta max = 1.2\theta_{true}$

Figure A.8: Classifications of comparison of values for δmax and θ_{max}

B. Extra results using GPS measurements

In this section we give some extra results used while finetuning our method as is further explained in section 4. The trajectories we used are shown in figure B.1. Also more results using the final settings of section 4.3 in combination with GPS measurements are given.



(a) Trajectory 1 (b) Trajectory 2 (c) Trajectory 3

(d) Trajectory 4 (e) Trajectory 5 (f) Trajectory 6

Figure B.1: Different trajectories obtained from GPS data loggings

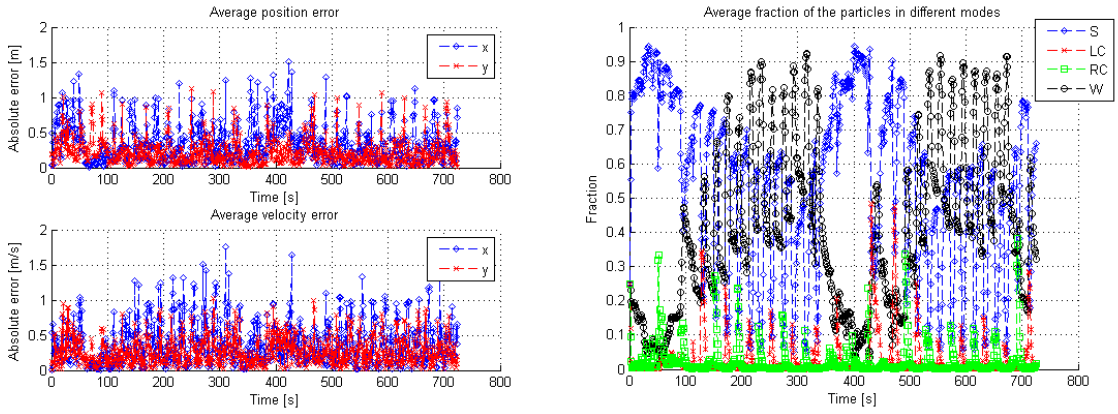
B.1 Submode transition probability matrix

The submode transition probability matrix provides the information about the transitions between the different submodes of the weave. We investigated two methods for determining the transition probabilities in Π_{sm_k}; the if-else-method and the min-method. To investigate if one method is favorable over the other method we performed simulations using the different trajectories of figure B.1 as true trajectories. For these simulations we used 10,000 particles and performed 10 Monte-Carlo runs. For trajectories 1 and 6 the measurement noise was the same for all runs. For trajectories 2, 3, 4 and 5 the measurement noise was generated separately each run.[1]. The shown results are averages over all runs.

In figures B.2 and B.3 the results of the simulations in which trajectory 2 is used as the true trajectory are shown. For both simulations we used $\delta_{max} = 500m$ and $\theta_{max} = \pi rad$. For the simulation of figure B.2 we used the if-else-method and for the simulation of figure B.3 we used the min-method for determining the entries of Π_{sm_k}. For the if-else-method we used $p_1 = 0.05$ and $p_2 = 0.95$. For the min-method we used $n = 25$.

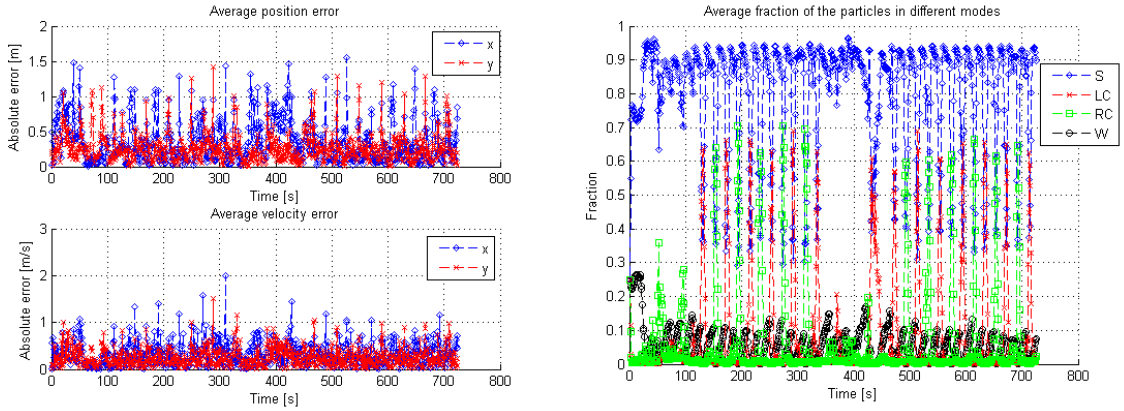
We see that the errors in figure B.2 and B.3 are of the same magnitude. This is also not surprising since we use the same dynamical models in all (sub)modes so the tracking performance

[1]This difference in measurement noise generation is due to the format in which the measurements were available at Thales



(a) Errors (b) Classification

Figure B.2: Errors and classification using the if-else method and trajectory 2 as true trajectory



(a) Errors (b) Classification

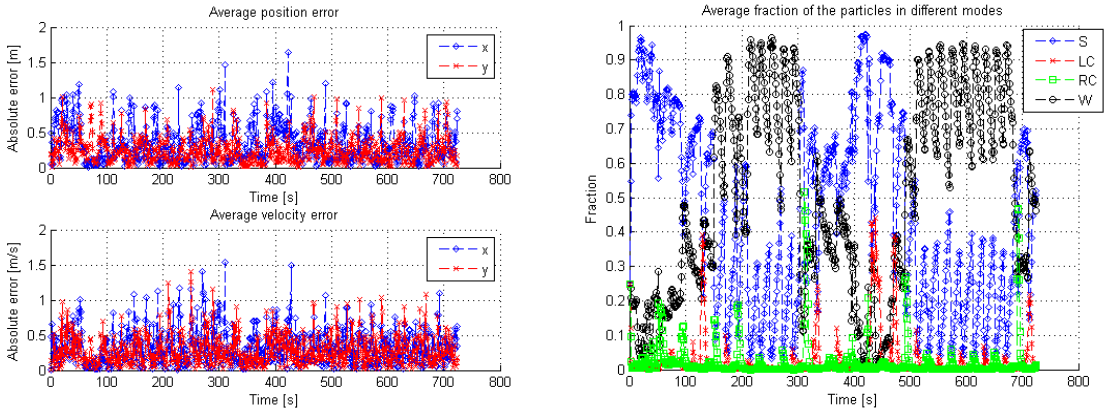
Figure B.3: Errors and classification using the min-method and trajectory 2 as true trajectory

should be equal in all (sub)modes.

When we look at the classification plots we clearly see that the min-method performs worse than the if-else-method. When the min-method is used, as is the case in figure B.3, the fraction of particles in the mode weave stays small. Where in figure B.2 when the if-else-method is used, we see that the fraction of particles in the mode weave is significantly larger. Therefore we find the if-else-method more favorable since this method has a better classification performance.

In figure B.2 we used the 'overestimated' values for δ_{max} and θ_{max}, namely $\delta_{max} = 500m$ and $\theta_{max} = \pi rad$. We also performed simulations using trajectory 2 as true trajectory using better estimates of δ_{max} and θ_{max}. The simulation outcomes when $\delta_{max} = 125m$ and $\theta_{max} = \frac{\pi}{2} rad$ in combination with the if-else-method is used are shown in figure B.4.

When we look at figure B.4 we see that the errors are of the same magnitude as the errors in figure B.2 for which we used the overestimated values for δ_{max} and θ_{max}. Again this comes from the fact that we use the same dynamical models in all (sub)modes so the tracking performance should be equal. But when we look at the classification plot of figure B.4 we clearly see a larger fraction of particles in the mode weave. So we see that when we use a better estimate of δ_{max} and θ_{max} we get a better classification performance. This conclusion also holds when we use the other trajectories of figure B.1 as the true trajectories.



(a) Errors (b) Classification

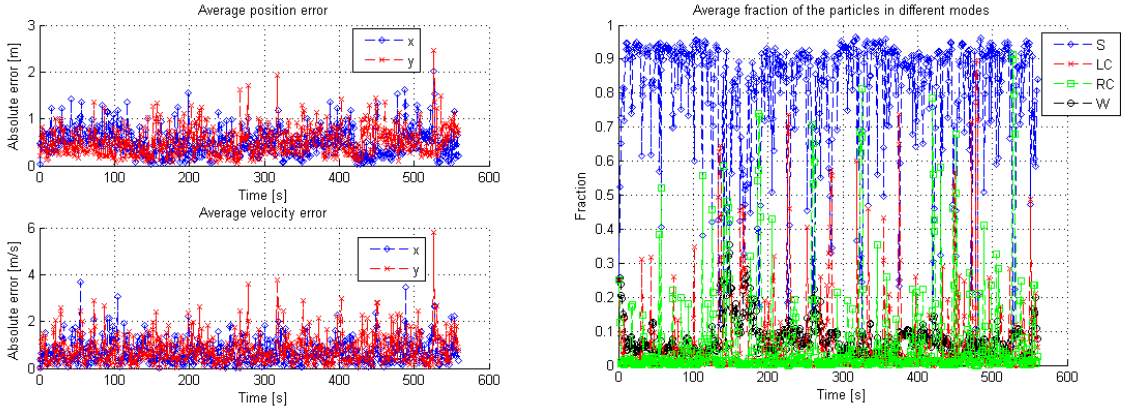
Figure B.4: Errors and classification using better estimates for δ_{max} and θ_{max}

B.2 Results using final settings

Using the final settings from section 4.3 we performed simulations to investigate the classification ability of our method using different trajectories in combination with GPS measurements. The different trajectories we used for the true trajectories in these simulation are shown in figure B.1. For each simulation we used 10,000 particles and performed 10 Monte-Carlo runs. For trajectories 1 and 6 the measurement noise was the same for all runs. For trajectories 2, 3, 4 and 5 the measurement noise was generated separately each run. The shown results are averages over all runs.

Results using trajectory 3 as true trajectory

In figure B.5 we see the errors and classification plot for which trajectory 3 is used as true trajectory. The object starts by going straight for some time, then it start to follow a weave trajectory.



(a) Errors (b) Classification

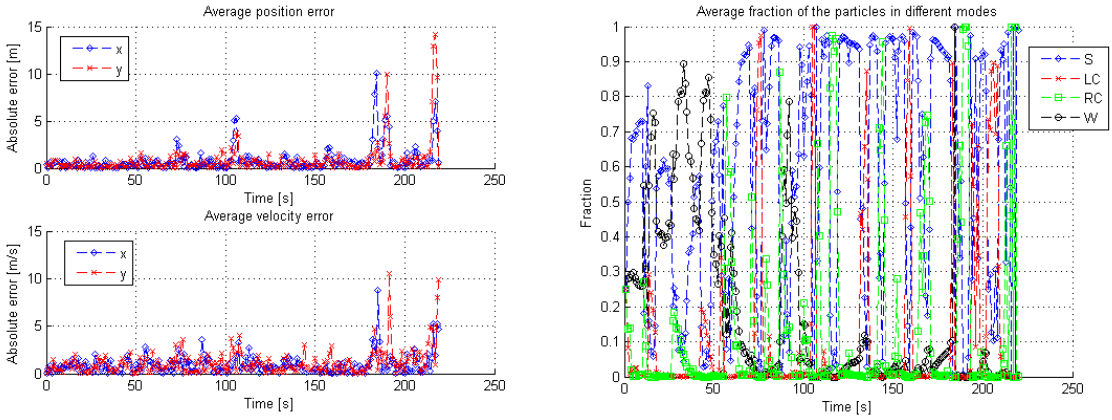
Figure B.5: Errors and classification using final settings and trajectory 3 as true trajectory

We see that the fraction of particles in the mode straight is high while the object follows a straight line trajectory. When the object starts to follow a weave trajectory we see that the fraction of particles in the mode weave does become a bit larger around 150 but not significantly. This is due to the fact that the object makes very sharp turns. It changes direction very fast making it hard for our method to pick up that the object is in fact performing a turn. In the used dynamical models a turn is a smooth curve in which the direction of the object's movement gradually changes. We see peaks of the fraction of particles in the modes left curve and right curve only for 1 or 2 time steps. After these time steps the object again follows a straight trajectory.

Since we use the same dynamical models in the different modes and more particles are forced into the mode straight we see a larger fraction of the particles in the mode straight during these straight segments. This in combination with the fast direction changes by the object and modeling assumption that a curve is smooth results in a small fraction of particles in the mode weave, thereby giving a bad classification.

Results using trajectory 4 as true trajectory

In figure B.6 we see the errors and classification plot for which trajectory 4 is used as true trajectory. The object moves in a weave-kind trajectory towards the origin.



(a) Errors (b) Classification

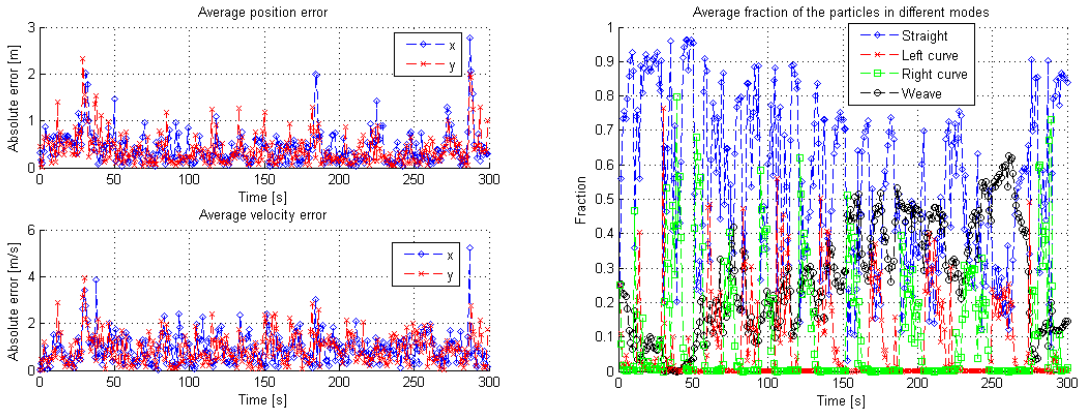
Figure B.6: Errors and classification using final settings and trajectory 4 as true trajectory

In the classification plot we see peaks of fractions of particles in the mode weave at the start of the trajectory. After approximately time step 60 we only see peaks of fractions of particles in the modes left curve and right curve alternating with peaks in of the fraction of particles in the mode straight. Just as when trajectory 3 was used as true trajectory we see that for some turns which the object makes it changes direction very fast and the curves are not smooth. Therefore our method is not able to 'pick up' the weave trajectory as good as we aim for. Also the shape of the end of the trajectory does not look that good like a weave anymore as the start of the trajectory, so the absence of peaks of the fraction of particles in the mode weave is explainable.

Results using trajectory 5 as true trajectory

In figure B.7 we see the errors and classification plot for which trajectory 5 is used as true trajectory. The object starts by going straight for some time, then it makes two right turns and gradually starts a weave trajectory.

In the classification plot we see that the fraction of particles in the mode weave does gets larger when the object follows a weave trajectory for some time. But this classification is less clear in comparison to the classification in figure 4.7 in which a weave trajectory is also followed at some time intervals.

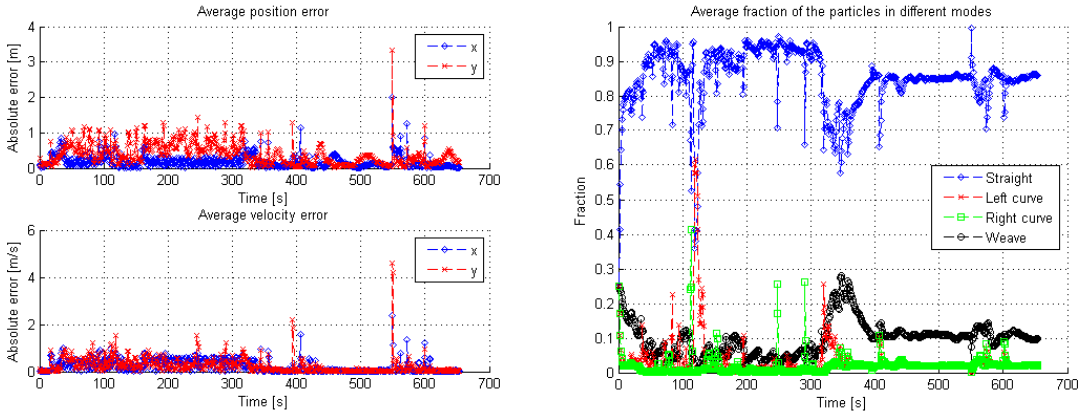


(a) Errors (b) Classification

Figure B.7: Errors and classification using final settings and trajectory 5 as true trajectory

Results using trajectory 6 as true trajectory

In figure B.8 we see the errors and classification plot for which trajectory 6 is used as true trajectory. The object moves in a straight line trajectory then makes a left turn, then it goes straight again and eventually lies still in the water.



(a) Errors (b) Classification

Figure B.8: Errors and classification using final settings and trajectory 6 as true trajectory

When we look at the classification plot we do see a peak in the fraction of particles in the mode left curve when the object if performing a left turn. We also see a small peak of the fraction of particles in the mode weave around time step 300. This peak originates from the movements the object makes when the object tries to lie still in the water. But overall the fraction of particles in the mode weave remains small thereby making this a good classification result since the object is indeed not following a weave trajectory.

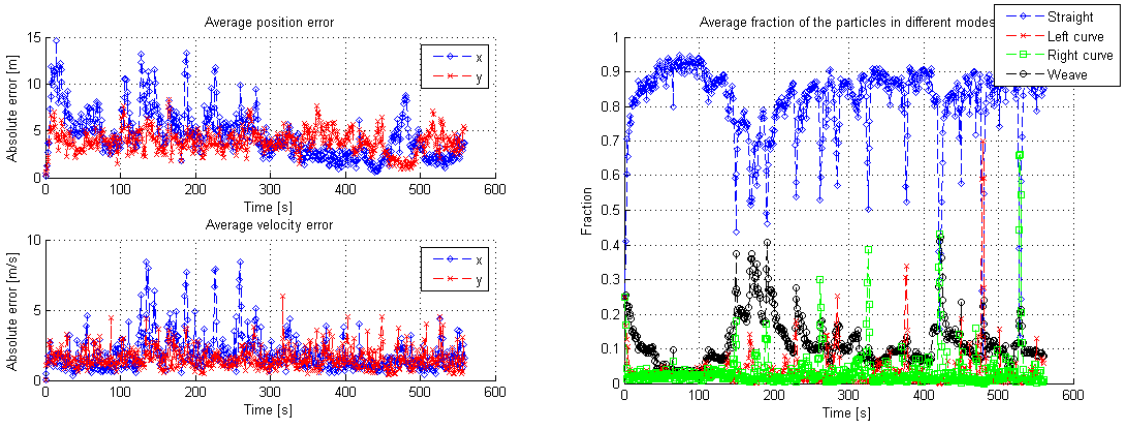
C. Extra results using radar measurements

In this section we give some extra results using the final settings of section 4.3 when radar measurements are used.

Using the final settings from section 4.3 we performed simulations to investigate the classification ability of our method using different trajectories while using radar measurements. The different trajectories we used for the true trajectories in these simulation are shown in figure B.1. We converted these trajectories to radar measurements. For each simulation we used 10,000 particles and performed 10 Monte-Carlo runs. For trajectories 1 and 6 the measurement noise was the same for all runs. For trajectories 2, 3, 4 and 5 the measurement noise was generated separately each run.[1]. The shown results are averages over all runs.

Results using trajectory 3 as true trajectory

In figure C.1 we see the errors and classification plot for which trajectory 3 is used as true trajectory. The object starts by going straight for some time, then it starts to follow a weave trajectory.



(a) Errors

(b) Classification

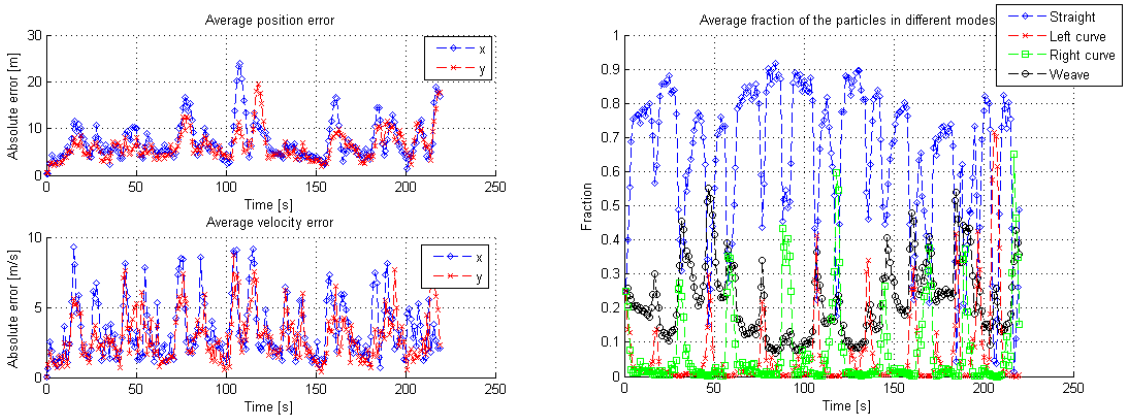
Figure C.1: Errors and classification using final settings and trajectory 3 as true trajectory

We see that the fraction of particles in the mode straight is high while the object follows a straight line trajectory. When the object starts to follow a weave trajectory we see that the fraction of particles in the mode weave does become larger but not significantly. This peak is larger and lasts longer in comparison to the same peak in the classification plot of figure B.5. This is explained by the fact that the classification using radar measurements is slower and less reactive than the classification using GPS measurements. Once our method in combination with radar measurements gives a higher probability to a certain mode is will maintain this higher probability for a longer time. When we use our method in combination with GPS measurements it does not maintain these high probabilities as long, it reacts faster.

Results using trajectory 4 as true trajectory

In figure C.2 we see the errors and classification plot for which trajectory 4 is used as true trajectory. The object moves in a weave-kind trajectory towards the radar.

[1]This difference in measurement noise generation is due to the format in which the measurements were available at Thales



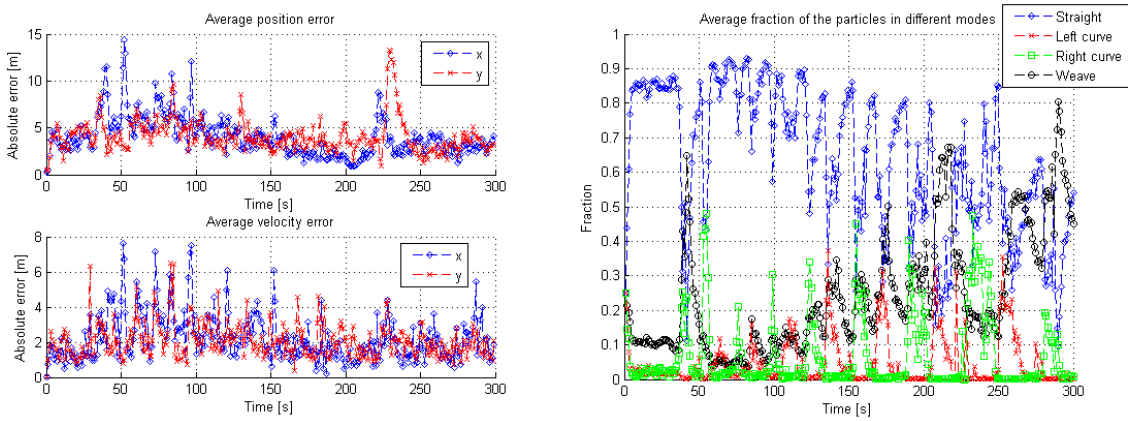
(a) Errors (b) Classification

Figure C.2: Errors and classification using final settings and trajectory 4 as true trajectory

When we look at the classification plot of figure C.2 we again see the less reactive classification when radar measurements are used. Just like the classification plot of figure B.6 we see an increase in the fraction of particles in the mode weave but this increase start later. Then this higher fraction only gradually decreases but when a turn is performed by the object this fraction increases again. So due to the less reactive classification a higher fraction of particles in the mode weave is maintained in comparison to when GPS measurements are used.

Results using trajectory 5 as true trajectory

In figure C.3 we see the errors and classification plot for which trajectory 5 is used as true trajectory. The object starts by going straight for some time, then it makes two right turns and gradually starts a weave trajectory.



(a) Errors (b) Classification

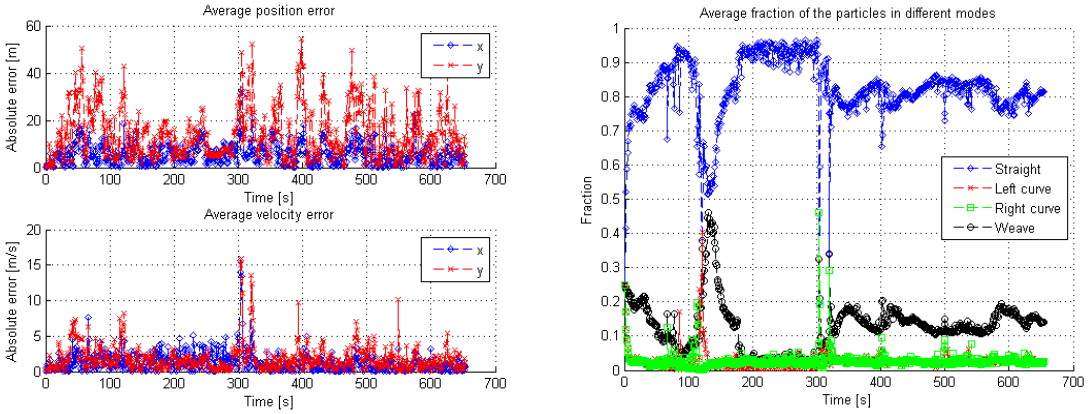
Figure C.3: Errors and classification using final settings and trajectory 5 as true trajectory

As expected we see larger errors in comparison with the errors in figure B.7 where GPS measurements were used.

The classification plot is comparable to the classification plot of figure B.7. Only the increase in fraction of particles in the mode weave starts later on and this fraction is smaller. It looks like the classification using radar measurements is slower and less reactive than the classification using GPS measurements.

Results using trajectory 6 as true trajectory

In figure C.4 we see the errors and classification plot for which trajectory 6 is used as true trajectory. The object moves in a straight line trajectory then makes a left turn, then it goes straight again and eventually lies still in the water.



(a) Errors

(b) Classification

Figure C.4: Errors and classification using final settings and trajectory 6 as true trajectory

We see that the errors in figure C.4 are larger than the errors in the figures C.1 to C.3. This comes from the fact that the trajectories used in those simulations lie closer to the radar, which result in smaller errors.

When we look at the classification plot we do see a peak in the fraction of particles in the mode weave. This is actually undesired since the object only performs one turn and we do not want to classify this as a weave trajectory. But on the other hand we can distinguish the classification plot of figure C.4 of those in the figures C.1 to C.3. So a different classification can be made.

We also see an increase in the fraction of particles in the mode weave from time step 300 till the end. This increase originates from the movements the object makes when the object tries to lie still in the water.

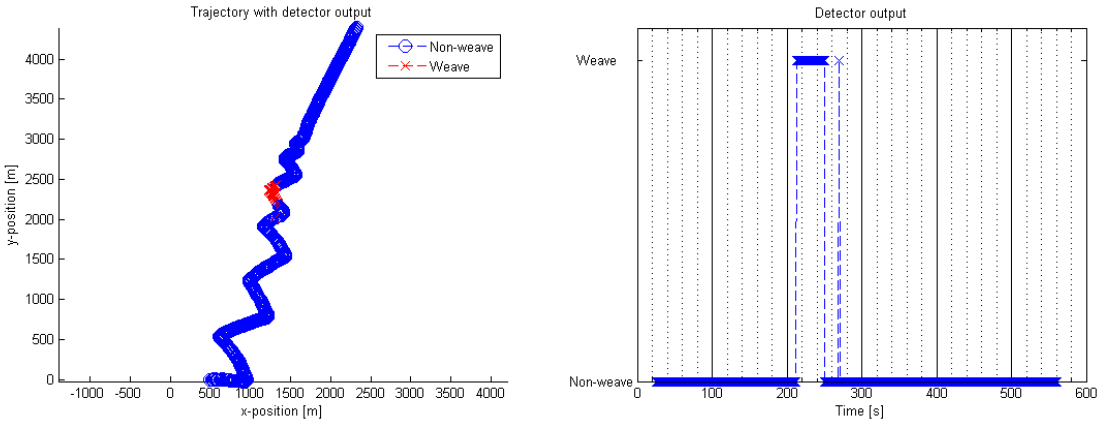
D. Extra results of the detector

In this section we give some extra results using the final detector as explained in section 6.

For the simulations presented in this section we have used 100,000 particles and performed only 1 Monte-Carlo run. In real life we also only have 'one run' to make our classification. Since we only have one run we have increased the number of particles to get a better estimation of the probability density functions. Also during our investigation it turned out that our method performs well in general.Therefore multiple MC-runs are not needed anymore. So we perform only 1 MC-run with a large number of particles for the simulations using the detector.

Results for trajectory 3

In figure D.1 we see the classified trajectory and detector output for which trajectory 2 is used as true trajectory. The object starts by going straight for some time, then it start to follow a weave trajectory.



(a) Trajectory with detector output

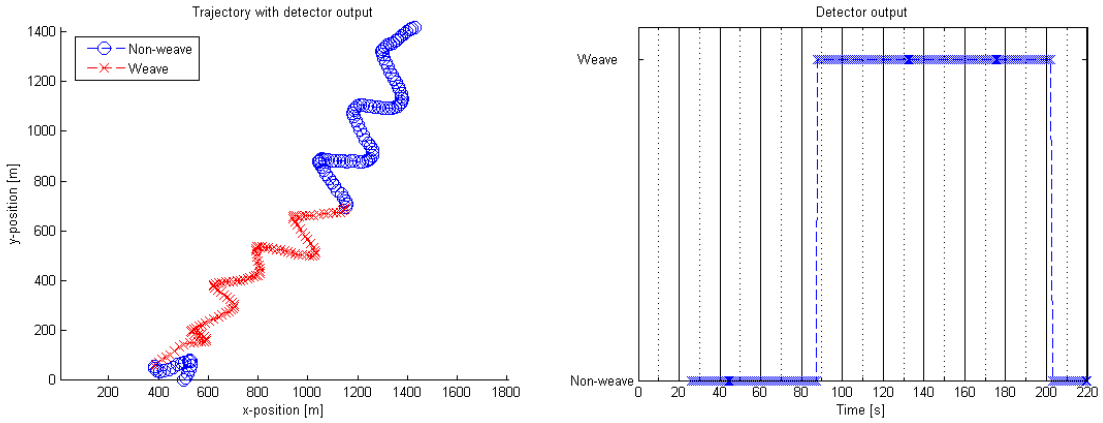
(b) Classification

Figure D.1: Trajectory 3 with detector output and classification

In figure D.1 we see that the detector only classifies this trajectory as a weave trajectory for a short period of time. This is due to the fact that the object makes very sharp turns. It changes direction very fast making it hard for our method to pick up that the object is in fact performing a turn. In the used dynamical models a turn is a smooth curve in which the direction of the object's movement gradually changes. Also since we use the same dynamical models in the different modes and more particles are forced into the mode straight due to the entries of Π_m we see a larger fraction of the particles in the mode straight during the straight segments of the weave trajectory. This in combination with the fast direction changes by the object and modeling assumption that a curve is smooth results in a small fraction of particles in the mode weave, making it hard for our detector to classify this trajectory correctly.

Results for trajectory 4

In figure D.2 we see the classified trajectory and detector output for which trajectory 4 is used as true trajectory. The object moves in a weave-kind trajectory towards the origin.



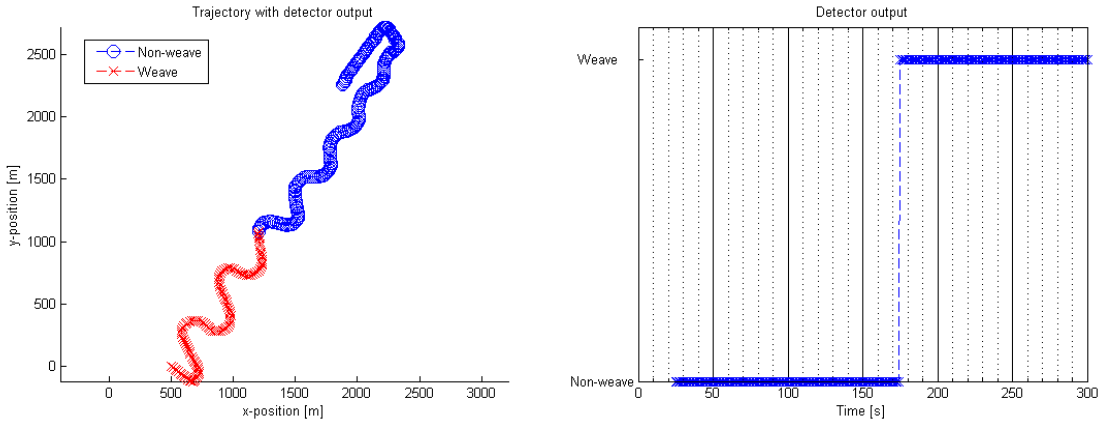
(a) Trajectory with detector output (b) Classification

Figure D.2: Trajectory 4 with detector output and classification

When we look at the detector output we do see that the classification as a weave trajectory is made. But just as when trajectory 3 was used as true trajectory we see that for some turns which the object makes it changes direction very fast. Therefore our method is not able to 'pick up' the weave trajectory as good as we aim for. The last part of the trajectory is classified as being a weave trajectory although the shape of this part of the trajectory does not look that good like a weave anymore. But as explained in section 6.3.3 we take this bad classification after a weave is followed for granted since the radar operator already decided on which actions to take.

Results for trajectory 5

In figure D.3 we see the classified trajectory and detector output for which trajectory 1 is used as true trajectory. The object starts by going straight for some time, then it makes two right turns and gradually starts a weave trajectory.



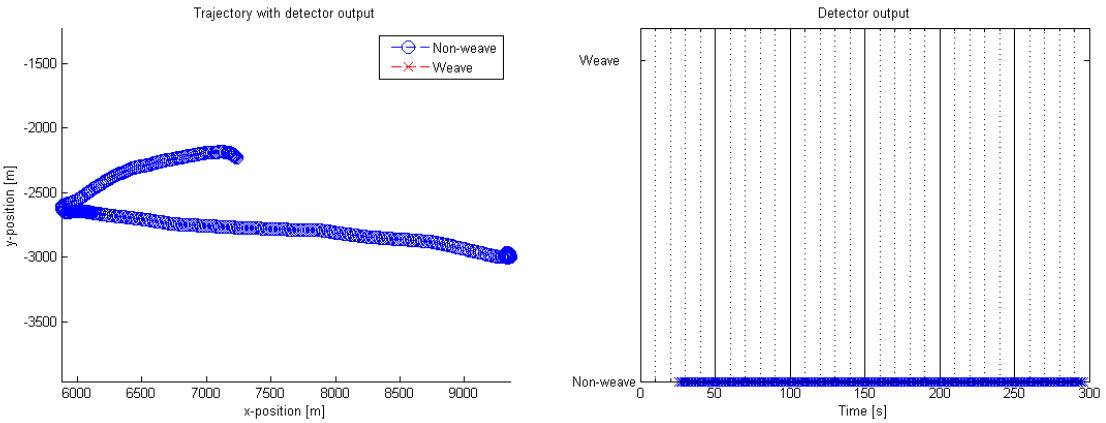
(a) Trajectory with detector output (b) Classification

Figure D.3: Trajectory 5 with detector output and classification

When we look at the detector output we see that the detector classifies this trajectory as a weave trajectory from time step 175. This is at a point where the object already follows a weave trajectory for a longer time but at the start of this weave trajectory it still is 'narrow'. After some time the weave trajectory becomes 'wider' and the detector indeed classifies this trajectory as a weave trajectory from that point on.

Results for trajectory 6

In figure D.4 we see the classified trajectory and detector output for which trajectory 6 is used as true trajectory. The object moves in a straight line trajectory then makes a left turn, then it goes straight again and eventually lies still in the water.



(a) Trajectory with detector output

(b) Classification

Figure D.4: Trajectory 6 with detector output and classification

When we look at the detector output we see that this trajectory is classified as a non-weave trajectory which is indeed the case.

Experiences at Thales

My internship was carried out at the company Thales in Hengelo. In this chapter I will tell something about my experiences at Thales and about the company itself.

My experiences

When I knew it was time for me to find an internship position I scheduled a meeting with my chairman. During this meeting we talked about the things I was looking for in an internship position and my preferences regarding the location of the position etc. After this meeting my chairman came up with several companies also including Thales. After reading into the different companies I chose to contact Thales for a possible internship position. I was invited to come to Thales and talk about the possibilities. At the end of this meeting we came up with an assignment which we all found suitable for the internship position, so my internship could begin.

Every morning I took my bike for a ride of approximately 25 minutes to get to Thales in Hengelo. On my first day I immideately had my picture taken for the security badge which gave me acces to the buildings. We got a small tour through some buildings and we went into the big radar tower at the backside of the facility. My desk was located in a room with 15 other people. They work in the same division of the company. On the work floor there was a calm atmosphere so everybody could concentrate on their work.

A typical workday for me started at 08.30h and ended at 17.00h. Most of the days I had lunch with Peter who started his internship at the same time as me.We then first went for a match of table tennis and then eat our lunch. On the workdays I did different things. Sometimes I had to program in Matlab, other days I was reading new information or I was working on my report. So there were several things to do on one day. Every week I either had a meeting with my supervisor to discuss my progress or a meeting with Peter, his supervisor and my supervisor to keep track of what we were both doing and see if we could help each other out. However if I had a problem I could just walk up to my supervisor and ask him for help.

At the end of the first week one of the younger employees of the division came to Peter and me and asked if we wanted to have a drink with some other younger employees of the division. They came up with the idea to have a drink every month with the younger employees of the division. We went to those drinks a couple of times and it was a nice way to meet these employees in an informal way.

During my internship, Thales was busy remodeling its facilities. Two completely new buildings were build of which one was the new office building, which even had a Starbucks bar inside it. We moved into this new building during the tenth week of my internship. In the new building nobody has its own desk anymore. You can sit at different desks in a certain area of the building, this is called ‘flexwerken’. I did not have much trouble adjusting to the new situation, but I can imagine that some people who already had their own desk for 20 years were less enthusiastic about this new way of working. However I did have trouble with my computer after we moved. This because suddenly I had no licenense anymore to work with Matlab. Luckily this problem was quickly solved, but the next problem became clear. I also had no license for a specific toolbox, so I still could not use Matlab the first couple of days in the new building. A couple days later this problem was also solved and I could finish my internship with my internship report.

Looking back at my internship period I am glad I chose Thales. It gave me insight in the structure and way of working at a company as large as Thales. At Thales Hengelo the facilities are up to date especially in the new office building. This enables the workers to do their jobs without being frustrated due to the limitations of old equipment. Also the atmosphere among co-workers is friendly and nice. All in all I would certainly recommend an internship position at Thales Hengelo.

About Thales

Thales Nederland B.V. was founded in 1922 as N.V. Hazemeyers Fabriek van Signaalapparaten. Trough the years it was called Hollands Signaalapparaten, Thomson CSF Signaal and since 2000 it is known as Thales Nederland. Thales Nederland is part of the Thales Group which is a worldwide organization. The company has about 2,000 employees working at the branches in Hengelo, Huizen, Delft and Eindhoven. Thales Nederland specializes in designing and producing professional electronics for defence and security applications, such as radar and communication systems. Moreover, Thales Nederland acts as a local point of contact for the complete portfolio of the Thales Group. Thales Nederland works in the fields of naval systems, land defence, the civil market (e.g. public transportation and national security), cryogenics and cyber security. In 2010 Thales Nederland generated about 600 million euro worth of sales, 80% of which is export.[1]

[1] For more information see: *https://www.thalesgroup.com/en/netherlands/about-us*

References

[1] S. Blackman, R. Popoli, *Design and analysis of modern tracking systems*. Norwoord: Artech House.

[2] H.A.P. Blom, Y. Bar-Shalom, *The Interacting Multiple Model Algorithm for Systems witch Markovian Switching Coefficients*. IEEE Transactions on Automatic Control, vol. 33, no. 8, August 1988 780 - 783.

[3] H.A.P. Blom, E.A. Bloem, *Particle filtering for stochastic hybrid systems*. 43rd IEEE Conference on Decision and Control, December 2004, 3221 - 3226.

[4] Y. Boers, H. Driessen, *Hybrid state estimation: a target tracking application*. THALES Nederland B.V., July 2002.

[5] Y. Boers, H. Driessen *A multiple model multiple hypothesis filter for Markovian switching systems*. Automatica 41, 2005, 709 - 716.

[6] M. Bootsveld, *Dealing with imperfect knowledge in sensor processing*. University of Twente, December 2012.

[7] F. Gustafsson, *Particle Filter Theory and Practice*. IEEE A&S systems magazine vol. 25, no. 7, July 2010, 53 - 81.

[8] E. Mazor, A. Averbuch, Y. Bar-Shalom, J. Dayan, *Interacting Multiple Model Methods in Target Tracking: A Survey*. IEEE transactions on aerospace and electronic systems vol. 34, no. 1, January 1998, 103 - 123.

[9] S. McGinnity, G.W. Irwin, *Multiple Model Bootstrap Filter for Maneuvering Target Tracking*. IEEE transactions on aerospace and electronic systems vol. 36, no. 3, July 2000, 1006 - 1012

[10] X. Rong Li, V.P. Jilkov, *Survey of Maneuvering Target Tracking. Part I: Dynamic Models*. IEEE transactions on aerospace and electronic systems vol. 39, no. 4, October 2003, 1333 - 1364.

[11] X. Yuan, C. Han, Z. Duan, M. Lei, *Comparison and Choice of Models in Tracking Target with Coordinated Turn Motion*. 7th International Conference on Information Fusion, 2005, 1497 - 1502

Used in the research on different methods for trajectory analysis

[12] M. Fanaswala, V. Krishnamurthy, *Spatio-Temporal Trajectory Models For Target Tracking*. 17th International Conference on Information Fusion, 2014, 1 - 8

[13] S. Fine, Y. Singer, N. Tishby, *The Hierarchical Hidden Markov Model: Analysis and Applications*. Machine learning, vol 32, 1998, 41 - 62

[14] V. Krishnamurthy, M. Fanaswala, *Intent inference via syntactic tracking*. Digital Signal Processing vol. 21, no. 5, September 2011, 648659

[15] T. Miller, W. Schuler, *An Empirical Evaluation of HHMM Parsing Time*. Proceedings of the Midwest Computational Linguistics, 2008

[16] N. Ueda, T. Sato, *Simplified Training Algorithm for Hierarchical Hidden Markov Models.* Electronics and Communications in Japan, part 3, vol. 87, no. 5, 2004

[17] N. Visnevski, V. Krishnamurthy, A. Wang, S. Haykin, *Syntactic Modeling and Signal Processing of Multifunction Radars: A Stochastic Context-Free Grammar Approach.* Proceedings of the IEEE, vol. 95, no 5, May 2007, 1000 - 1025