State estimation on non-linear Autonomous Guided Vehicles

Damianos Tranos

Kongens Lyngby 2016

Technical University of Denmark
Department of Applied Mathematics and Computer Science
Section for Dynamical Systems
Richard Petersens Plads, building 324,
2800 Kongens Lyngby, Denmark
Phone +45 45 25 30 31
compute@compute.dtu.dk
www.compute.dtu.dk

Technical University of Denmark
Department of Electrical Engineering
Automation and Control
Elektrovej, building 326,
2800 Kongens Lyngby, Denmark
Phone +45 45 25 38 00
elektro@elektro.dtu.dk
www.elektro.dtu.dk

Abstract

This thesis deals with the application of nonlinear state estimation techniques for the localization of an Autonomous Guided Vehicle, operating in an orchard environment. Because GPS availability is not guaranteed, the localization relies solely on odometry and measurements taken by a laser scanner.

Suitable models are derived to generate the vehicle's motion, the sensor measurements and the environment; they are then used for the construction of a simulation platform. Subsequently, three contemporary nonlinear filters are studied as possible solutions, the Unscented Kalman Filter, the Particle Filter and the Unscented Particle Filter.

These filters are implemented in the simulation platform and their ability to localize the vehicle is assessed, with respect to estimation accuracy and computational effort, in two scenarios; one in which the environment is modeled correctly and another where there exist discrepancies between the environment and the model.

Preface

This thesis was prepared at the Department of Applied Mathematics and Computer Science and the Department of Electrical Engineering, the Technical University of Denmark, as part of the requirement for the degree of M.Sc. in Electrical Engineering.

I would like to express my gratitude to my supervisors, Associate Professors Niels Kjølstad Poulsen and Ole Ravn both for their earnest support, insightful advice and pleasant attitude throughout the project.

I would also like to thank my friends and fellow students at DTU, Robert Miklos, Leon Nagel and Rasmus Steffensen for making my university experience as enjoyable as it has been and supporting me throughout my studies.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

In 2012, the Danish Agriculture & Food Council released their facts and figures booklet where they reported 26560 square kilometers of arable land, more than 61.4% of the country's total area. It is not surprising therefore, that the agriculture and food sector represents 20% of the total Danish commodity exports, accounting for more than 16 billion Euro. Despite this, agricultural tasks are in general physically demanding, time-consuming and repetitive.

A significant subset of these tasks are harvesting, cattle feeding and tree spraying activities which require the workers to be mobile and travel large distances. The workers must also be able to navigate their environment safely, without crushing into trees or driving off the field. They must therefore exhibit some measure of autonomy as they go about their predetermined tasks. This calls for the application of automation technology.

A machine which fulfils the above requirements qualifies as an Autonomous Guided Vehicle (AGV) or mobile robot. It is essentially any vehicle which is equipped with proprioceptive and exteroceptive sensors and is able to accurately identify its position in an arbitrary reference frame (localize), using these sensors. This feature is an irreplaceable prerequisite for autonomy, for if the vehicle

cannot localize itself, it cannot make any decisions as to where and how to move or what to do.

Interestingly enough, no perfect solution exists for the localization problem despite the attention it has received from the robotics community. This is because even though a vast selection of sensors are available, they are all corrupted with statistical noise and prone to failure. A powerful method which can exploit a time series of these noisy measurements to produce an estimate of the vehicle's position, is the Kalman filter [18]. This process is commonly referred to as state estimation or filtering.

There is a caveat however which precludes the Kalman filter from being the general solution to the localization problem; its results are valid only for linear systems and uncertainties that are normally distributed. Since an AGV is a physical system, any reasonable model hoping to capture its full operation will necessarily be nonlinear. Even if this were not the case, the environment it operates in is rife with nonlinearities. This necessiates the use of nonlinear state estimation methods.

1.2 Problem Description

The task of this project is to investigate the state of the art nonlinear filtering methods available and apply them for the solution of the localization problem of an Autonomous Guided Vehicle in an orchard environment given by Figure 1.1.

The system considered, is a HAKO tractor (Figure 1.2), which is equipped with encoders on the drive shaft and front wheels to measure the position and orientation through odometry. The tractor also comes equipped with a gyro to measure its heading and a Real Time Kinematic (RTK) GPS, to measure its absolute position. However, the RTK GPS is sensitive to shadowing from surrounding trees and bushes in the orchard which degrades the localization [23]. For this reason, the tractor is outfitted with a SICK LMS-200 laser scanner which can be used to measure the tree rows in the orchard. Figure 1.3 shows two examples of how these rows look like.

The problem of localization in such an environment has been previously considered in [10]. However, the focus was in the application of filters that relied on linearization, such as the established Extended Kalman (EKF), the newer Unscented Kalman Filter (UKF) as well as the Second-order Divided-Difference filter (DD2).

Figure 1.1: The orchard environment in which the AGV will be operating [2].

Figure 1.2: The HAKO tractor [10, 22].

Figure 1.3: Images of the tree rows in the orchard. The vegetation is in general quite dense with large gaps occurring only when trees have been removed [22].

Therefore, this thesis contributes to this problem through the application of not only the UKF, but also of a class of algorithms known as Particle Filters (PF) [6, 26] which rely on Monte Carlo methods to approximate any arbitrary distributions regardless of nonlinearities.

The overall objectives of the project are as follows.

- To construct a simulation environment which replicates the scenario presented in [10, 22] in MATLAB.
- To study and successfully implement the Unscented Kalman and the Particle Filter for this problem.
- To investigate nonlinear filtering and identify a method with the potential of outperforming the two previous ones.
- To assess the performance of any implemented filters through a simulation study.

1.3 Outline of the Thesis

Chapter 2 deals with the development of the kinematic model of the HAKO tractor and discusses the odometry related errors. The path-planning and control of the tractor follow subsequently, before dealing with the modeling of the laser scanner.

Chapter 3 defines the state estimation problem and presents the Bayesian estimator which is the optimal but infeasible solution. Thereafter, the Extended Kalman, Unscented Kalman and Particle Filters are presented. Finally, the Unscented Particle Filter, a combination of the later two methods, is elaborated.

Chapter 4 considers the application of these filters to the localization problem and their performance assessment. Two cases of the problem are considered; one where the environment information is correct and one where there are discrepancies.

Chapter 5 offers a summary of the work and a concluding discussion on the results as well as suggestions for future problems and approaches.

Chapter 2

System Modeling

This chapter deals with the mathematical modeling necessary to simulate the HAKO in the orchard as well as implement the aforementioned filtering techniques. The HAKO is a continuous dynamical system whose dynamics are most accurately represented by stochastic differential equations. However, since the objective is to implement a state estimation algorithm which will be carried out by a digital computer, it is reasonable to consider a discrete model. The mathematical models used in state estimation applications are generally of the form:

$$x_k = f_{k-1}(x_{k-1}, u_{k-1}, w_{k-1}) \qquad x \in \mathbb{R}^n, k \in \mathbb{N} \tag{2.1}$$

$$y_k = h_k(x_k, e_k) \qquad y \in \mathbb{R}^m \tag{2.2}$$

Here, x_k is the state vector and contains the variables that one might be interested in estimating. The evolution of the state through the sample time k is governed by the nonlinear time-varying stochastic difference equation f_k (also called process function). There may also exist a deterministic variable vector u_k which typically represents control inputs or known disturbances as well as an independently, identically distributed (i.i.d.) stochastic vector w_k whose role is to encapsulate unmodelled dynamics or represent uncertainty in general.

Equation (2.2), referred to frequently as the measurement function, represents the relationship between the states and the available measurements y_k. The (i.i.d.) stochastic vector e_k serves to model the uncertainty due to stochastic disturbances which affect the measurement, such as sensor noise.

2.1 Robot Kinematics

For the localization of an AGV, the states must relate to the vehicle's absolute position. The orchard environment considered is relatively flat and spans an area no larger than a square kilometer, leading to the curvature of the Earth being negligible. Therefore, it can be reasonably treated as two dimensional plane so that the three states of interest are the position of the HAKO on the x axis (x_k) and on the y axis (y_k), which are relative to some arbitrary global reference frame, as well as the orientation of the vehicle θ_k shown in Figure 2.1.

$$\mathbf{x}_k = \begin{bmatrix} x_k \\ y_k \\ \theta_k \end{bmatrix} \tag{2.3}$$

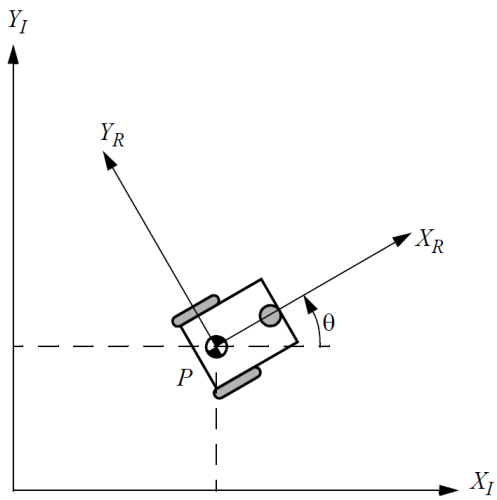

Figure 2.1: A global reference frame and a mobile robot local reference frame[27].

The state vector is given in (2.3). The orientation is defined as the angle between the global and the local x axes. By convention, the local x axis is pointing towards the direction the robot is facing so that it is colinear with its forward velocity vector. With this representation, it is possible to relate the local reference frame to the global one using the rotation matrix (2.4):

$$R(\theta_k) = \begin{bmatrix} cos(\theta_k) & -sin(\theta_k) & 0 \\ sin(\theta_k) & cos(\theta_k) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.4}$$

which follows directly if one considers the projection of the global coordinates to the local ones. Moreover, if the origins of the two frames have a distance d then one can instead use the more general homogeneous transformation matrix [29]:

$$\begin{bmatrix} \mathbf{x}_k^W \\ 1 \end{bmatrix} = \begin{bmatrix} R(\theta_k) & \mathbf{d}_{3\times1} \\ 0_{1\times3} & 1 \end{bmatrix} \begin{bmatrix} \mathbf{x}_k^R \\ 1 \end{bmatrix}. \tag{2.5}$$

And its inverse:

$$\begin{bmatrix} \mathbf{x}_k^R \\ 1 \end{bmatrix} = \begin{bmatrix} R^{-1}(\theta_k) & -R^{-1}(\theta_k)\mathbf{d}_{3\times1} \\ 0_{1\times3} & 1 \end{bmatrix} \begin{bmatrix} \mathbf{x}_k^W \\ 1 \end{bmatrix} \tag{2.6}$$

so as to readily transform the state vector between the two reference frames. Note that in (2.6) the fact that $R^{-1} = R^T$ can be exploited to simplify the computation of the inverse transformation.

With the definition of the global and local coordinate systems and the robot state vector, the next step is to consider the steering and driving method of the vehicle. The HAKO tractor has two fixed standard wheels on its rear axle. They are connected through a gearbox to the engine and thus are used for driving. The front axle of the vehicle follows the Ackermann steering principle depicted in Figure 2.2.

The characteristic property of this steering method is that the two front wheels rotate by different angles when the vehicle follows a curved path, so that the virtual lines extending from each wheel intersect at a certain point P which is defined as the robot's instantaneous center of rotation (ICR) [27]. As shown in the figure, it is possible to reduce the model further to that of a tricycle while

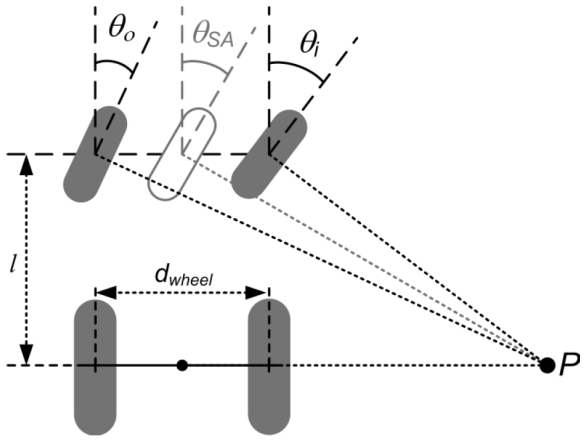

Figure 2.2: Geometry of the Ackermann steering principle.[3]

preserving the same maneuverability. With this tricycle model, the control input vector becomes:

$$\mathbf{u}_k = \begin{bmatrix} v_k \\ \theta_{SA,k} \end{bmatrix} \tag{2.7}$$

with v_k being the distance traveled in one sample, a quantity which can be inferred directly from the encoder at the gearbox by measuring revolutions per minute. The steering angle θ_{SA} can be related to the angle of either steered wheel using (2.8):

$$\theta_{SA} = arccot\left(\frac{d}{2l} + cot(\theta_i)\right) = arccot\left(cot(\theta_o) - \frac{d}{2l}\right) \tag{2.8}$$

with θ_i and θ_o being the angles of the inner and outer wheels with respect to the ICR, d being the distance between the rear wheels and l being the distance between the front and rear axles.

The kinematic model for the HAKO can then be given by (2.9). The new state vector at sample k is equal to the previous state plus the contribution from the control input. The forward velocity affects the position in both the x and the y

axis, based on the orientation of the vehicle.

$$\mathbf{x}_k = \mathbf{x}_{k-1} + \begin{bmatrix} v_{k-1} cos\left(\theta_{k-1} + \frac{\Delta\theta_{k-1}}{2}\right) \\ v_{k-1} sin\left(\theta_{k-1} + \frac{\Delta\theta_{k-1}}{2}\right) \\ \Delta\theta_{k-1} \end{bmatrix} \tag{2.9}$$

The evolution of the orientation is denoted by $\Delta\theta$ and follows from considering the geometry in Figure 2.3. Here it is shown that the tricycle takes a turn and forms an arc L in one sample time. The equation for the arc of a circle with radius q is known to be:

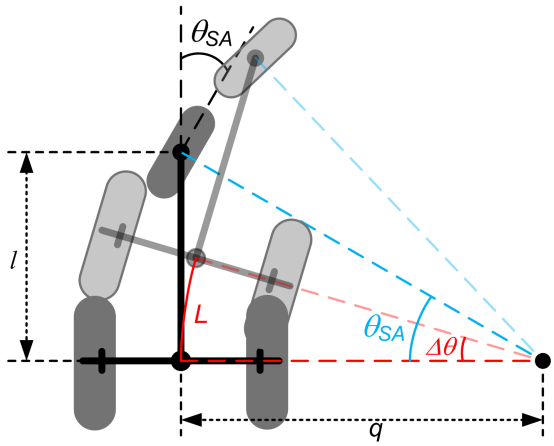

Figure 2.3: Trycicle moving on an arc L [22].

$$L = q\Delta\theta \tag{2.10}$$

It also follows from the definition of the tangent for an orthogonal triangle that:

$$tan(\theta_{SA}) = \frac{l}{q} \tag{2.11}$$

By substitution of (2.10) into (2.11) and carrying out simple algebra, one arrives

at the result:

$$\Delta\theta = \frac{Ltan(\theta_{SA})}{l} \tag{2.12}$$

Finally, if L is sufficiently small, it is approximated by the distance traveled at each sample k; the forward velocity v_k, leading to:

$$\Delta\theta_k = \frac{v_k tan(\theta_{SA,k})}{l} \tag{2.13}$$

This completes the formulation of the kinematic model. This model is suitable primarily when the vehicle operates at low speeds due to the fact that velocities and applied forces (i.e. the dynamics) are not considered. Disturbances that could be attributed to terrain anomalies such as bumps or inclines and the viscous friction at the contacts between the wheels and the ground are also not explicitly modeled. Lastly, note that the chosen kinematic model makes the Markov assumption; only the previous state and the inputs are needed to determine the current state vector. This is an assumption that does not hold in practice as there may be an undetected obstacle or change in the vehicle which leads to a different behavior.

A justification for these omissions is that the model approximates the behavior of the vehicle to a reasonable degree while being simple and easy to implement in state estimation algorithms. Disturbances and unmodeled effects can then be included in a probabilistic manner by augmenting (2.9) with a stochastic component; the random variable w_k as mentioned in (2.1).

It is intuitively sound to include this vector additively to the input vector so that:

$$\hat{\mathbf{u}}_k = \begin{bmatrix} v_k \\ \theta_{SA,k} \end{bmatrix} + \mathbf{w}_k \tag{2.14}$$

This new $\hat{u}_k$ represents the measurement of the forward velocity and steering angle from the relevant encoders, corrupted by sensor noise and the aforementioned possible disturbances. The vector w_k is assumed to be normally distributed with zero mean and covariance R_1 ($\mathbf{w}_k \sim N(0, R_1)$). There are multiple sources of uncertainty, hence the central limit theorem makes the normal distribution a good candidate as long as it contains the support of the true distribution of the control signals (i.e. the R_1 matrix is sufficiently large). In fact, as will be

further discussed in later chapters, the R_1 matrix can be loosely regarded as a tuning parameter which reflects the engineer's confidence in the model in the context of state estimation.

For the purposes of this model, R_1 will be the diagonal, time varying matrix:

$$R_{1,k} = \begin{bmatrix} \sigma_{v,k}^2 & 0 \\ 0 & \sigma_{\theta,SA,k}^2 \end{bmatrix} \tag{2.15}$$

which assumes that there is no correlation between the deviation of the forward velocity and the steering angle. If such a correlation is known from experimental data or modeling, it can be included in the off-diagonal terms. The term $\sigma_{v,k}$ is the standard deviation of the forward velocity and is given by [20]:

$$\sigma_{v,k}^2 = k_v^2 |v_k| \tag{2.16}$$

where k_v is a constant representing the standard deviation occuring from one meter of travel. Equation (2.16) states that the more the vehicle travels each sample, the larger the variance grows, which is intuitive. In a similar manner, $\sigma_{\theta,SA,k}$ follows from (2.13):

$$\sigma_{\theta,SA,k}^2 = \left(\frac{tan(k_A + k_{SA}|\theta_{SA,k}|)}{l} \right)^2 |v_k| \tag{2.17}$$

with k_A being corresponding to the standard deviation of the angle from one meter of travel and k_{SA} is a constant affecting the contribution of turning to the error.

2.2 Path Planning and Control

Having acquired the model of the tractor, it is now desired to have it navigate in the orchard. To do so, one needs to specify the control inputs so that the tractor follows a specified path or trajectory. A trajectory q is a mapping from time to the x-y plane which satisfies any arbitrary requirement $q(t) = f(x_t, y_t)$ (i.e. one can impose that the tractor is located at a certain point at a certain time). A naive method of specifying a trajectory would be as a combination of predetermined line and circular arc segments. However, a more general and

user-friendly method involves specifying a set of via points which the robot should pass through. The task then simplifies to computing a set of high order polynomial functions of time which smoothly connect these points.

Typically these polynomials are chosen so that they are quintic (fifth-order)[4]. This is due to the fact that lower order polynomials do not allow the specification of constraints on acceleration which may be required as part of the mission or to prevent discontinuities in the acceleration. This may be important because an impulsive jerk (defined as the derivatives of the acceleration) can excite the vibration modes of the vehicle [29].

The equation for the quintic polynomial that describes the position of the vehicle on one axis is given in (2.18).

$$q(t) = c_0 + c_1 t + c_2 t^2 + c_3 t^3 + c_4 t^4 + c_5 t^5 \tag{2.18}$$

with c_i being the defining constants of the polynomial. Then the velocity and acceleration follow from differentiation:

$$\dot{q}(t) = c_1 + 2c_2 t + 3c_3 t^2 + 4c_4 t^3 + 5c_5 t^4 \tag{2.19}$$

$$\ddot{q}(t) = 2c_2 + 6c_3 t + 12c_4 t^2 + 20c_5 t^3 \tag{2.20}$$

If one were to differentiate (2.20) once more, one would acquire the jerk. The resulting equation will be of a second order (provided $c_5 \neq 0$) and thus parabolic with no discontinuities. Equations (2.18-2.20) collectively describe the position, velocity and acceleration at one point. Therefore there will be six equations for the two points which are to be connected, and together they form a linear system (2.21) which is exactly determined.

$$\begin{bmatrix} 1 & t_0 & t_0^2 & t_0^3 & t_0^4 & t_0^5 \\ 0 & 1 & 2t_0 & 3t_0^2 & 4t_0^3 & 5t_0^4 \\ 0 & 0 & 2 & 6t_0 & 12t_0^2 & 20t_0^3 \\ 1 & t_f & t_f^2 & t_f^3 & t_f^4 & t_f^5 \\ 0 & 1 & 2t_f & 3t_f^2 & 4t_f^3 & 5t_f^4 \\ 0 & 0 & 2 & 6t_f & 12t_f^2 & 20t_f^3 \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \\ c_4 \\ c_5 \end{bmatrix} = \begin{bmatrix} q(t_0) \\ \dot{q}(t_0) \\ \ddot{q}(t_0) \\ q(t_f) \\ \dot{q}(t_f) \\ \ddot{q}(t_f) \end{bmatrix} \tag{2.21}$$

Here, t_0 and t_f denote the times when the polynomial must satisfy the initial and end points respectively. Then it is a matter of solving for the constants

c_i which is a straightforward and computationally efficient process in software packages such as MATLAB. One linear system has to be solved for each degree of freedom on which a trajectory is required. This approach can then be extended to handle multiple via points in a sequence by ensuring that the end conditions ($q(t_f)$, $\dot{q}(t_f)$ and $\ddot{q}(t_f)$) of a move are the initial conditions ($q(t_0)$, $\dot{q}(t_0)$ and $\ddot{q}(t_0)$) of the subsequent one. This leads to one trajectory which satisfies all the via points as was defined in the beginning of the section.

Having generated the path, the task is then to have the vehicle follow it. To this end, a simple reference tracking feedback controller will be designed based on the derived kinematic model (2.9). The objective of this controller is to bound the distance between the vehicle and a goal (carrot point) which travels on the path at a constant speed (Figure 2.4). Such a controller is commonly referred to as Pure Pursuit [4]

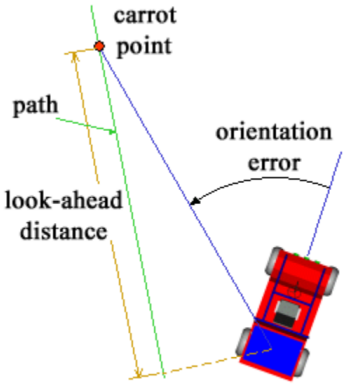

Figure 2.4: Vehicle in pursuit of the carrot point [21].

The controller is given below from the equations (2.22-2.24):

$$e_v(k) = \sqrt{(x_{cp}(k) - x_k)^2 + (y_{cp}(k) - y_k)^2} - d \tag{2.22}$$

$$v_k = K_p e_v(k) + K_i \sum_{n=t_0}^{k} e_v(n) T_s \tag{2.23}$$

$$\theta_{SA} = K_{\theta_{SA}}(arctan(\frac{y_{cp}(k) - y_k}{x_{cp}(k) - x_k}) - \theta_k) \tag{2.24}$$

Firstly, the error is computed as the Euclidean distance between the carrot point

and the vehicle's position, minus a chosen distance d which the vehicle should maintain as it pursuits the goal. Secondly, a discrete PI (Proportional-Integral) control action is assigned to the forward velocity using this error with K_p and K_i being tuning parameters for the controller (note that T_s is the sampling time). The integral term is required if one is interested in eliminating steady state error. Otherwise, setting both d and K_i to zero leads to a simpler controller (less tuning parameters) and still satisfies the objective of following the carrot point at a fixed distance. Finally, in (2.24) the steering angle is determined proportionally to the deviation between the vehicle's orientation and the orientation required for it to face towards the carrot point.

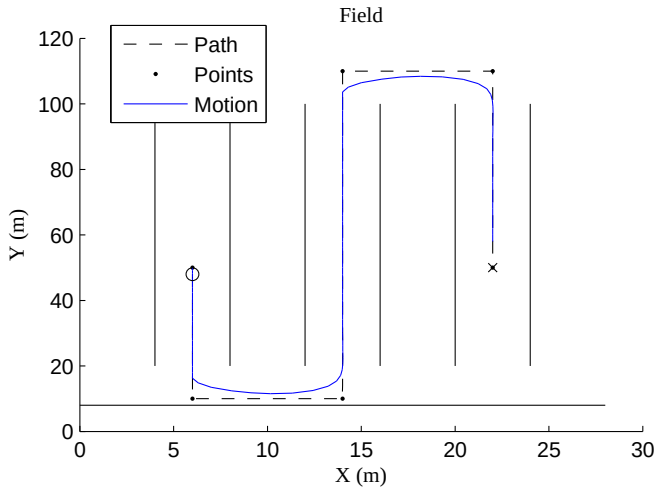

Figure 2.5: Generated path and motion of the vehicle. The robot starts at the small circle [6,50] and drives towards the cross [22,50].

With these tools, the robot can now be simulated as it drives in a replica of the orchard environment. The resulting motion is shown in Figure 2.5. The solid black lines represent the trees/bushes in the orchard which will be discussed in the next section. The path is generated using the MATLAB function *mstraj* from the Robotics Toolbox[4], using six via points constrained so that the velocity is constant at 2 m/s. Then the controller parameters are tuned to the values $d = 0$, $K_p = 0.2$, $K_i = 0$ and $K_{\theta_{SA}} = 0.8$, leading to the over-damped motion shown in the figure. The reasoning behind this path assignment and controller tuning was to use the least number of via points while generating a realistic driving motion for a four wheeled vehicle. It is important to note that, the true states are being used for the feedback so as to decouple the controller performance from the state estimation algorithm.

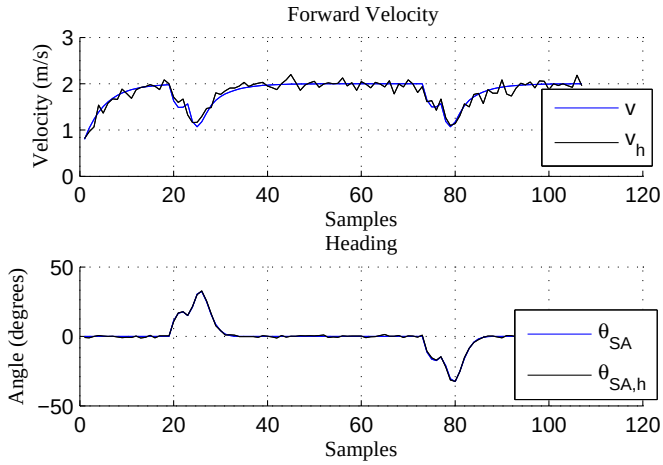

Figure 2.6: Time series plots of the control inputs and their estimates from the encoders.

The control inputs are given in Figure 2.6. It can be seen that the vehicle indeed drives at a constant speed of 2 m/s excluding the two turning sections where the speed is reduced to allow the vehicle to perform the necessary maneuver. These inputs are reasonable in the absence of knowledge of the limitations on the HAKO's maximum velocity and steering angle rates.

Along with the true control inputs, time series plots of their measurements from the encoders are also presented. These correspond to the input vector in (2.14 and it is these data which will be used in the state estimation algorithms. The standard deviation parameters for the process noise are chosen to be $k_v = 0.1$, $k_A = 0.05$ and $k_{SA} = 0.1$. One can make a direct estimate of the vehicle's position history using these control inputs, a process referred to as dead-reckoning. This estimate is plotted against the actual motion in Figure 2.7.

As can be seen in the figure, the estimation degrades rapidly due to the error accumulation. This result is inevitable; no matter how small the process noise is at each sample, as time tends to infinity the estimate will drift from the true position. In this case, the process noise was tuned to be larger than in [22] and [10], so as to meaningfully stress the state-estimation algorithms considered.

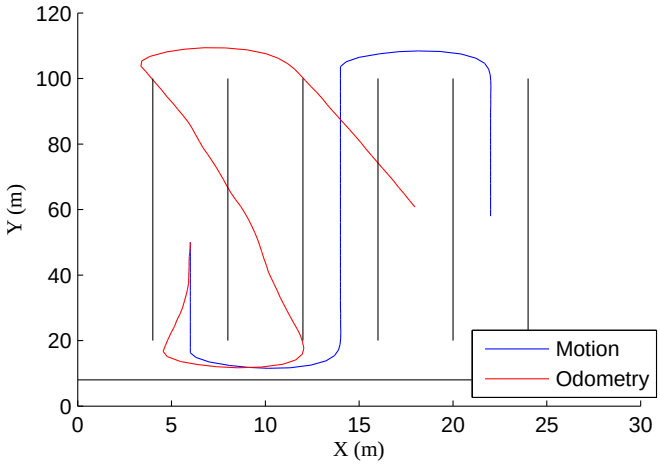

Figure 2.7: Robot motion vs estimate based on the odometry.

2.3 Laser Scanner and Orchard

In the previous section, it was shown that dead-reckoning leads to large errors over time. However, each individual measurement is not so noisy as to give no meaningful information regarding the vehicle's pose. This can be argued from the fact that, in Figure 2.7, the first few state estimates were very close to the true ones. Thus, if these estimates could be corrected using information from a different source, the estimation quality would improve. For the HAKO tractor, such a source would be the distance measurements of the tree row positions given by the laser scanner.

In order for the vehicle to make use of this information, it needs to have a representation of its environment so as to be able to relate the measurements to its position relative to the global reference frame. This representation is referred to as a map [31] and there exist two approaches to creating one. The first is to partition the space into sufficiently small blocks so that they form a grid. Each block is associated with an occupancy scalar variable which is equal to one if the space is occupied and zero otherwise. This is called a grid map. The second approach, is to identify a set of features or landmarks (typically points or geometric primitives such as lines and circles) and record their position with respect to a reference frame. Such a map is denoted as a feature map.

Grid maps are ideal for general purpose applications because no explicit knowledge of the features in the environment is needed. Moreover, they fit well in

a stochastic framework by allowing the occupancy variables to take values between zero and one, corresponding to the degree of belief in the space being occupied. However, they are only useful if the grid resolution is sufficiently small and this may impose constraints in available memory and computational power. Feature maps on the other hand are best utilized when the environment can be adequately modeled by a finite number of geometric primitives or uniquely identifiable landmarks. It is far easier for a human user to produce such a map and far less computational resources are needed to manipulate and store the feature entries.

Taking into account the above, the feature map approach was chosen since the orchard environment consists of dense tree rows in parallel, which can be mapped as straight lines. The particular map considered in the simulations is best seen in Figure 2.5. Each black line represents a tree row and there are six in total along with a seventh perpendicular row. The tree rows have a length of 80 m and are spaced by 4 m each. The map then, is a table which includes the start and end points of each line.

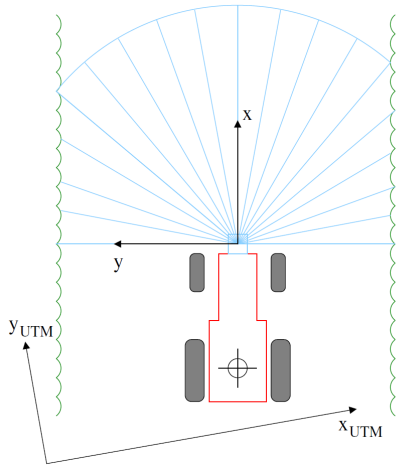

Figure 2.8: Laser Scanner functionality in MATLAB [10].

Having defined the map used for this environment, the formulation of a suitable model for the laser scanner is then considered. Figure 2.8 illustrates the scanner's functionality. It is a device mounted on the front of the vehicle which emits a cone of laser light pulses and receives them when they are reflected from the nearest surface. The time of flight of each ray is measured to determine the distance of that object from the scanner. If no object is detected then the maximal effective distance of the laser scanner is returned.

With this information, a single ray can be represented in polar coordinates as:

$$y_k^i = \begin{bmatrix} r_i \\ \phi_i \end{bmatrix} \tag{2.25}$$

and every ray can be stacked to form the measurement vector:

$$y_k = \begin{bmatrix} r_1 & \phi_1 & r_2 & \phi_2 & \dots & r_n & \phi_n \end{bmatrix}^T \tag{2.26}$$

where r_i and ϕ_i correspond to the returned distance and the angle (measured from the local x axis) of the ith ray. The number of rays can be determined from knowledge of the field of view (size of the cone) and of the resolution of the scanner. For the SICK LMS-200, these are 180 degrees and 0.5 degrees respectively, leading to 361 rays and thus 722 elements for the vector y_k.

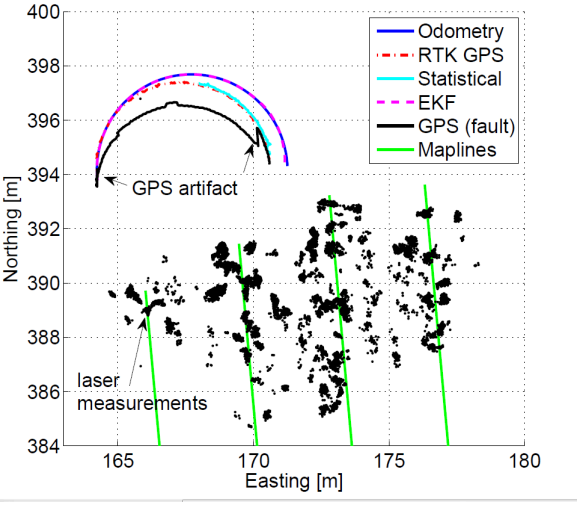

Figure 2.9: Real laser scan measurements from an orchard mission. The green lines represent the map and the black dots represent measured points by the scanner [2].

For visualization purposes, the measurements can be converted from polar to cartesian coordinates and then rotated, by use of (2.5), to the global reference frame. Figure 2.9 shows such a representation of real measurement data, taken during operation in the orchard by [2]. As can be seen, the data points do not lie

exactly on where the map lines have been assigned but are concentrated around them. This is due to the tree foilage which form non-smooth, non-symmetric surfaces from which the laser rays are reflected.

Fortunately, such a phenomenon can be modeled in a stochastic framework. Each ray can be augmented with a two dimensional, normally distributed stochastic vector $e_k^i \sim N(0, R_2^i)$. The same line of reasoning as that used for determining the distribution of the process noise vector w_k is applied.

$$y_k^i = \begin{bmatrix} r_i \\ \phi_i \end{bmatrix} + \mathbf{e}_k^i \tag{2.27}$$

The covariance matrix R_2 is usually chosen to be diagonal if one has no knowledge of the correlation between the ray angle and the returned distance. This matrix is similar to R_1 in the sense that it is effectively a tuning parameter that reflects the engineer's confidence in the measurement. Typically it is identified from real life data. An alternative formulation comes from first converting the rays to the global, cartesian coordinates before adding the vector. In this case R_2 acquires a physical meaning; its diagonal elements are directly related to the area spanned by the tree foilage.

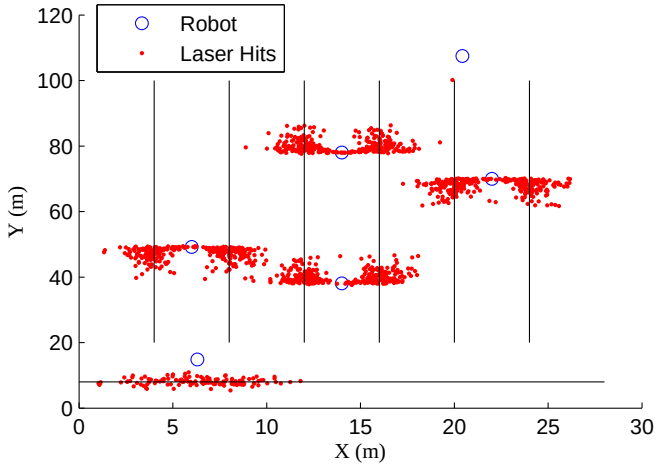

Figure 2.10: Simulation of the HAKO tractor and associated laser scanner measurements.

A simulation of the vehicle driving in the orchard is repeated, this time including the laser scanner function. This function takes as arguments the vehicle's

current position and the map. Since the map is a collection of start and end points, these are transformed into the robot's local reference frame using (2.6). Thereafter, the line parameters corresponding to each pair of start and end points are computed leading to the equation for the ith line:

$$A_i x + B_i y + C_i = 0 \tag{2.28}$$

which can be rewritten in terms of the start and end points as:

$$\frac{(y_{end} - y_{start})x - (x_{end} - x_{start})y + x_{end}y_{start} - y_{end}x_{start}}{\sqrt{(y_{end} - y_{start})^2 + (x_{end} - x_{start})^2}} = 0. \tag{2.29}$$

with the denominator of serving as a normalization of the parameters by the distance of the two defining points. Such a normalization is very useful because (2.29) can then be used directly to compute distances between the robot and each line. Since the field of view and resolution of the laser scanner are known, the angles ϕ_i of each ray can be determined by starting from one end of the field of view and incrementing by the resolution. The points that belong to a particular ray will be constrained as:

$$x_{ray} = cos(\phi_i) \tag{2.30}$$
$$y_{ray} = sin(\phi_i). \tag{2.31}$$

Substituting these in (2.29), yields the distance from the intersection of the ray and the line. This process is carried out for every line on the map and the minimum of these distances and the laser scanner effective range is returned for that ray, leading to the vector y_k.

Figure 2.10 shows snippets of the robot's position during the mission (blue circle) along with the measurements taken in those positions (red dots). A close up view to one of these instances is also provided in Figure 2.11 along with a visual of the rays (red lines). The R_2 matrix was assigned arbitrarily to be:

$$R_2 = \begin{bmatrix} 1 & 0 \\ 0 & 0.01 \end{bmatrix}. \tag{2.32}$$

It can be seen that at least in terms of variance and mean, the simulated measurements are distributed similarly to the real data. More accurate modeling would require access to and evaluation of the statistical properties of the real measurements. Regardless, a Gaussian approximation of the distribution has been shown in [22] and [10] to yield good results in the context of state estimation.

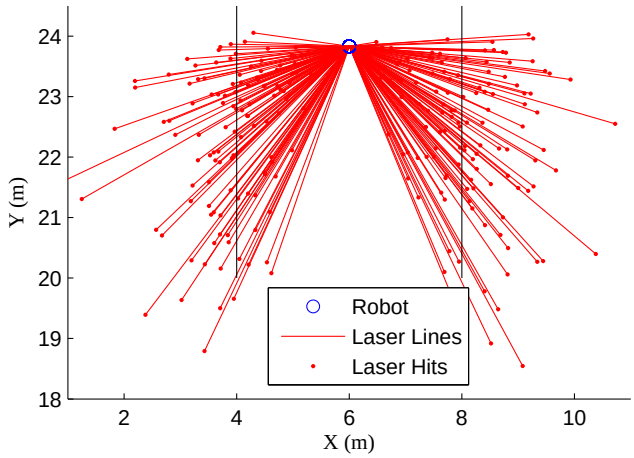

Figure 2.11: Close view of the laser measurements and rays.

Chapter 3

Nonlinear Filtering Methods

This chapter describes the nonlinear state-estimation problem (otherwise referred to as filtering) and presents some of the more prevalent methods for its solution. The Kalman filter has played a central role in state estimation since its conception in 1958, thus its direct nonlinear extension, the Extended Kalman Filter (EKF), is discussed first in Section 3.1. Section 3.2 presents the Unscented Kalman Filter (UKF), a newer and arguably more accurate extension of the Kalman filter relying on statistical linearization. Thereafter, Section 3.3 discusses the Particle Filter (PF) a method resulting from the application of Monte Carlo approximation to filtering. The unification of the later two ideas into one method known as Unscented Particle Filter (UPF) is the subject of Section 3.4.

Filtering is the problem of producing a state estimate $\hat{x}_k$ of a dynamical system, which is frequently modeled using (2.1) and (2.2), given the observation history set $Y_k = \{y_1, y_2, \dots y_k\}$ containing all measurements from the system's initialization until the current time k. If one adopts a Bayesian framework [19],[32] then such an estimate can be derived from the posterior density $p(x_k|Y_k)$, and, the application of Bayes' theorem gives the relationship

$$p(x_k|Y_k) = \frac{p(Y_k|x_k)p(x_k)}{p(Y_k)} \tag{3.1}$$

which then can be manipulated using straightforward probability rules and the Markov property into the form of the recursive Bayesian estimator.

$$p(x_k|Y_k) = \frac{p(y_k|x_k)p(x_k|Y_{k-1})}{p(y_k|Y_k)} \propto p(y_k|x_k)p(x_k|Y_{k-1}) \tag{3.2}$$

Equation (3.2) shows that the desired posterior is proportional to the prior density $p(x_k|Y_{k-1})$ multiplied by the observation likelihood density $p(y_k|x_k)$. The term $p(y_k|Y_k)$ is a normalizing constant and can be found with the application of the theorem of total probability to be

$$p(y_k|Y_k) = \int p(y_k|x_k)p(x_k|Y_{k-1})dx_k \tag{3.3}$$

The prior density can also be found with the application of the same theorem as

$$p(x_k|Y_{k-1}) = \int p(x_k|x_{k-1})p(x_{k-1}|Y_{k-1})dx_{k-1} \tag{3.4}$$

with $p(x_{k-1}|Y_{k-1})$ being the posterior at time $k-1$. Since the estimator is meant to be applied recursively, this density is presumed to have been computed in the previous iteration and is therefore known. The term $p(x_k|x_{k-1})$ is denoted as the state transition prior and follows from (3.5) with the application of (2.1) and the process noise density $p(w_k)$:

$$p(x_k|x_{k-1}) = \int \delta(x_k - f_k(x_{k-1}, u_{k-1}, w_{k-1}))p(w_{k-1})dw_{k-1} \tag{3.5}$$

Similarly, the observation likelihood density is given by use of (2.2) and the measurement noise density $p(e_k)$:

$$p(y_k|x_k) = \int \delta(y_k - h_k(x_k, e_k))p(e_k)de_k \tag{3.6}$$

where δ is the Dirac-delta function. Thus all the components of the Bayesian estimator are either known or can be computed using the system equations. With

the posterior available, any optimal estimate can be computed, the conditional mean (3.7) being the most notable among them and what will be used herein.

$$\hat{x}_k = E[x_k|Y_k] = \int x_k p(x_k|Y_k) dx_k \tag{3.7}$$

The Bayesian estimator is the optimal solution to the filtering problem. However, the integrals in (3.3 - 3.6) can become intractable for nonlinear or non-Gaussian systems. Hence the need for the following approximate methods.

3.1 Extended Kalman Filter

In the case of a system with linear dynamics and normally distributed stochastic variables, the Bayesian estimator reduces to the Kalman filter [18]. This filter however, does not require specifically that the system equations be linear, rather the requirements imposed are the following [32]:

1. It is possible to derive consistent minimum variance state estimates using only the first and second moments of distributions of interest.
2. The estimator 3.2 can be accurately approximated by a linear function.
3. Accurate predictions of the states and the associated measurements can be made (for example by use of (2.1) and (2.2)).

As long as these assumptions hold, the Kalman filter is still the optimal linear estimator. The idea of the EKF then, is to linearize the system, using the Taylor series expansion about the latest state estimate x^+_{k-1} (3.8), where the + superscript denotes that the estimate was made taking into account the measurement at time $k-1$.

$$\begin{aligned} f_{k-1}(x_{k-1}, u_{k-1}, w_{k-1}) = f_{k-1}(\hat{x}^+_{k-1}, u_{k-1}, \hat{w}_{k-1}) + F_{k-1}(x_{k-1} - \hat{x}^+_{k-1}) \\ + G_{k-1}(w_{k-1} - \hat{w}_{k-1}) + O[(x_{k-1} - \hat{x}^+_{k-1})^2] \end{aligned} \tag{3.8}$$

If the second and higher order derivative terms $O[(x_{k-1} - \hat{x}^+_{k-1})^2]$ of this expansion are neglected, then the procedure leads to the two constant partial derivative matrices for the process equation:

$$F_k = \nabla_x f_k(\hat{x}_k^+, u_k, \hat{w}_k)$$
$$G_k = \nabla_w f_k(\hat{x}_k^+, u_k, \hat{w}_k) \tag{3.9}$$

with $\nabla_x = \left(\frac{\partial}{\partial x_1}, \ldots, \frac{\partial}{\partial x_n}\right)$ being the vector differential operator. F_k can be thought of as the matrix representing the dynamics of the state deviation from the estimate, and G_k as being the input matrix relating the process noise to the state. Due to the first assumption, the prior $p(x_k|Y_{k-1})$ is then fully characterized by:

$$\hat{x}_k^- = f_{k-1}(\hat{x}_{k-1}^+, u_{k-1}, \hat{w}_{k-1}) \tag{3.10}$$
$$P_k^- = F_k P_k F_k^T + G_k R_1 G_k^T \tag{3.11}$$

Here the expactation of the nonlinear model has been used to propagate the mean from time $k-1$ to k directly. The superscript $-$ denoting that this estimate does not include information from the latest measurement. The covariance matrix of the state vector P_k is also maintained and propagated using the matrices in (3.9) according to the discrete Lyapunov equation. This step is referred to as the time update of the Kalman filter and is demonstrated visually in Figure 3.1.

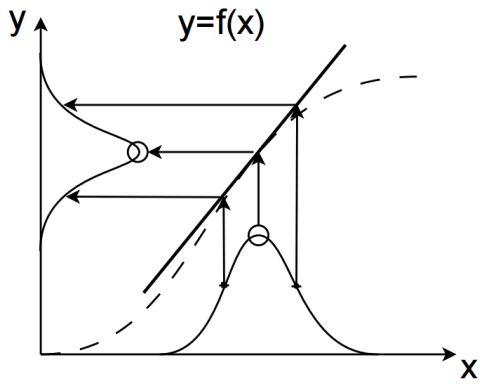

Figure 3.1: Visual demonstration of linearization of a non-linear function and the subsequent propagation of the mean and covariance of the prior density [25].

The next step, commonly referred to as the measurement update, is to include the latest measurement using the observation likelihood $p(y_k|x_k)$. The measurement function (2.2) is therefore linearized as well around x_k^-

$$h_k(x_k, e_k) = h_k(\hat{x}_k^-, \hat{e}_k) + H_k(x_k - \hat{x}_k^-) + D_k(e_k - \hat{e}_k) + O[(x_k - \hat{x}_k^-)^2] \quad (3.12)$$

Neglecting the higher order terms again, leads to an approximation with the two Jacobians:

$$\begin{aligned} H_k &= \nabla_x h_k(x_k^-, \hat{e}_k)\big|_{x_k=\hat{x}_k^-} \\ D_k &= \nabla_e h_k(\hat{x}_k^-, e_k)\big|_{e_k=\hat{e}_k}. \end{aligned} \quad (3.13)$$

H_k is the linearized measurement matrix and D_k is the direct input matrix relating the measurement noise e_k to the observation. With these matrices, the covariance P_y of the measurement vector can be calculated again by use of the discrete Lyapunov equation. Afterwards, one can compute the Kalman gain, defined as the matrix K_k which minimizes the trace of the covariance P_k^+ of the posterior density $p(x_k|Y_k)$ [28].

$$P_y = H_k P_k^- H_k^T + D_k R_2 D_k^T \quad (3.14)$$

$$K_k = P_k^- H_k^T (P_y)^{-1} \quad (3.15)$$

Then the posterior mean x_k^+ is updated first by evaluating the conditional mean of the measurement function and adding it to the prior after scaling it by the Kalman gain. The posterior covariance matrix is also found by subtracting the measurement covariance scaled by the Kalman gain from the prior. The magnitude of the Kalman gain, which is inversely proportional to the magnitude of the measurement covariance matrix, determines the contribution of the measurement to the posterior estimate.

$$\hat{x}_k^+ = \hat{x}_k^- + K_k(y_k - h_k(\hat{x}_k^-, \hat{e})) \quad (3.16)$$

$$P_k^+ = P_k^- - K_k P_y K_k^T \quad (3.17)$$

This completes the state estimation for time k. The described procedure is then applied recursively with the time update step being carried out when the index k increments and the measurement update step whenever a new measurement is acquired.

While the EKF is arguably the standard in nonlinear state estimation [17] it comes with some drawbacks. The linearization presented makes the implicit assumption that the second and higher order derivatives of the expansion are negligible. Moreover, the linearization is only as valid as the estimate, since that is the point where the Taylor expansion is being made. If the estimate deviates significantly from the truth, then this will lead to significant estimation errors, the accumulation of which may cause the estimate to diverge after a few iterations. This is regardless of the accuracy with which the state uncertainty is represented by the matrix P_k; the linearization makes no use of such information.

Modifications of the EKF exist that include higher order terms in the linearization (High order EKF) or which repeat the linearization at x_k+ and repeat the filtering process so as to reduce error (Iterative EKF). Any performance improvements by these versions however are associated with a significant increase in computational cost. Even if these issues were circumvented, the EKF requires the computaton, and thus the existence, of derivative terms for both the process and measurement functions. As a result, it cannot be applied near regions of the state-space where these functions are discontinuous or if the functions are given in a non-explicit form (e.g. look-up tables).

3.2 Unscented Kalman Filter

Motivated by the above limitations, one logical step is to avoid linearization entirely. It was required in the EKF, not because the functions are unknown, but because it is difficult to apply a general nonlinear transformation on a probability density function, hence the need to approximate the transformation. An alternative approach then, is to approximate the density function itself which, as shown in [13], [16], is accomplished by the Unscented Transform (UT), the cornerstone of the UKF.

The UT is a deterministic sampling technique; it involves the selection of a minimal set of points (sigma points) in the state-space, whose ensemble mean and covariance match those of the prior density. Each point is then propagated through the exact nonlinear functions (Figure 3.2) and the first two moments of the resulting set of transformed points can be computed to yield good estimates of the mean and variance of the posterior.

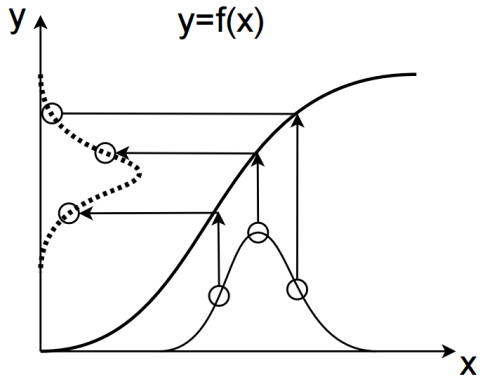

Figure 3.2: Representation of sigma points of a prior density being propagated through a non-linear function and forming the posterior [25].

There are several ways of selecting these sigma points; The original unscented transform [28] exhibits a proportional relationship between the distance of the sigma points and the dimension of the state vector. The scaled unscented transform [14] allows one to counteract this effect and also to better approximate higher moments. Finally, the spherical simplex unscented transform [15] reduces the number of sigma points at the expense of potential numerical instability [17]. Since they share many commonalities only the scaled unscented transform will be presented in the context of the UKF for the general model given by (2.1 - 2.2).

First, an augmented state vector $\hat{x}_k^a$ is formulated, containing the means of the state as well as all non-additive process and measurement noise terms. Their covariances are also collected in the covariance matrix P_k^a.

$$\hat{x}_k^a = \begin{bmatrix} \hat{x}_k & \hat{w}_k & \hat{e}_k \end{bmatrix}^T \tag{3.18}$$

$$P_k^a = \begin{bmatrix} P_k & 0 & 0 \\ 0 & R_1 & 0 \\ 0 & 0 & R_2 \end{bmatrix} \tag{3.19}$$

It is important to note that this augmentation differs from the case when the Kalman filter is used to estimate disturbances or parameters, in the sense that the noise means and covariances will not be estimated; they will simply be used for the correct approximation of the prior. Letting n be the dimension of the

augmented vector, exactly $2n + 1$ sigma points are selected according to the following scheme:

$$\bar{x}_{k-1}^{0} = \hat{x}_{k-1}^{a+} \tag{3.20}$$

$$\bar{x}_{k-1}^{i} = \hat{x}_{k-1}^{a+} + \left(\sqrt{(n+\lambda)P_{k-1}^{a+}}\right)_{i}^{T} \qquad i = 1, \ldots, n \tag{3.21}$$

$$\bar{x}_{k-1}^{n+i} = \hat{x}_{k-1}^{a+} - \left(\sqrt{(n+\lambda)P_{k-1}^{a+}}\right)_{i}^{T} \qquad i = 1, \ldots, n \tag{3.22}$$

The matrix square root in (3.21 - 3.22) can be computed using, for instance, the Cholesky decomposition and $\left(\sqrt{(n+\lambda)P_{k-1}^{a+}}\right)_{i}$ is the ith row of that matrix. The first sigma point is the mean itself and the other $2n$ points are selected as a symmetric pair around it. In addition to these sigma points, the following weights are defined:

$$W_0^m = \frac{\lambda}{n+\lambda} \tag{3.23}$$

$$W_0^c = \frac{\lambda}{n+\lambda} + (1 - \alpha^2 + \beta^2) \tag{3.24}$$

$$W_i^m = W_i^c = \frac{1}{2(n+\lambda)} \tag{3.25}$$

W_i^m being the weights associated with the computation of the mean and W_i^c the weights pertaining to the evaluation of the covariance. The weights can be positive or negative regardless of the unscented transform applied, but in order for the estimate to be unbiased, each set of weights must obey the condition [17]:

$$\sum_{i=0}^{2n} W_i = 1 \tag{3.26}$$

In equations (3.21 - 3.25), $\lambda = \alpha^2(n + \kappa) - n$ is a scaling parameter [34]. The scalar values $\alpha \in (0, 1]$ and $\kappa \leq 0$ influence how spread out the sigma points will be from the mean and are typically chosen to be small (10^{-3} and 0 respectively). The variable β is used to incorporate prior knowledge of the distribution of

x_k. Specifically, it increases the weighing of the zeroth sigma point which was shown in [14] to yield partial information of the fourth-order term of the Taylor series expansion of the covariance. Its optimal value is $\beta = 2$ for Gaussian distributions.

The sigma points are then collected into one $n \times (2n+1)$ size matrix X^a_{k-1} which can be interpreted as the set of sigma points. Here $X^x_{i,k-1}$ denotes the component of the ith sigma point associated with the state vector and similarly $X^w_{i,k-1}$ and $X^e_{i,k-1}$ with the components associated with the process and measurement noise respectively.

$$X^a_{k-1} = \begin{bmatrix} \bar{x}^0_{k-1} & \bar{x}^1_{k-1} & \dots & \bar{x}^{2n}_{k-1} \end{bmatrix} \tag{3.27}$$

$$= \begin{bmatrix} \begin{bmatrix} \bar{X}^x_{0,k-1} \\ \bar{X}^w_{0,k-1} \\ \bar{X}^e_{0,k-1} \end{bmatrix} & \begin{bmatrix} \bar{X}^x_{1,k-1} \\ \bar{X}^w_{1,k-1} \\ \bar{X}^e_{1,k-1} \end{bmatrix} & \dots & \begin{bmatrix} \bar{X}^x_{2n,k-1} \\ \bar{X}^w_{2n,k-1} \\ \bar{X}^e_{2n,k-1} \end{bmatrix} \end{bmatrix} \tag{3.28}$$

These points are now propagated to the next time step k using (2.1). Since the statistics of the noise components are static, only X^x_{k-1} needs to be updated. Note that, for each sigma point, its process noise component is used in the function instead of the estimate $\hat{w}_k$.

$$X^x_{i,k} = f_{k-1}(X^x_{i,k-1}, u_k, X^w_{i,k-1}) \tag{3.29}$$

Having propagated the points, their first and second order moments are computed using the previously defined weights and the definitions of the mean and the variance.

$$\hat{x}^-_k = \sum_{i=0}^{2n} W^m_i X^x_{i,k} \tag{3.30}$$

$$P^-_k = \sum_{i=0}^{2n} W^c_i (X^x_{i,k} - \hat{x}^-_k)(X^x_{i,k} - \hat{x}^-_k)^T \tag{3.31}$$

Thereby, the estimates $\hat{x}^-_k$ and P^-_k are acquired in a manner reminiscent of the EKF. In fact, the procedure so far can be considered as the time update step of the UKF. The measurement step follows thereafter, with the sigma points being

propagated now through the measurement function to acquire the observation set Y_k (note that this set is unrelated to the previously defined observation history set). Then the observation estimate is computed as the weighted mean of the points in the set.

$$Y_{i,k} = h_k(X^x_{i,k}, X^e_{i,k}) \tag{3.32}$$

$$\hat{y}_k = \sum_{i=1}^{2n} W^m_i Y_{i,k} \tag{3.33}$$

The covariance of the observation and its cross-covariance with the state are subsequently computed using their respective definitions and the covariance weights.

$$]P_y = \sum_{i=0}^{2n} W^m_i (Y_{i,k} - \hat{y}_k)(Y_{i,k} - \hat{y}_k)^T \tag{3.34}$$

$$P_{xy} = \sum_{i=0}^{2n} W^c_i (X^x_{i,k} - \hat{x}^-_k)(Y_{i,k} - \hat{y}_k)^T \tag{3.35}$$

With these covariance matrices, the Kalman gain is computed and the measurement step is completed by evaluating the mean and covariance of the posterior density. This step is nearly identical for both the EKF and the UKF. At this point the, augmented state vector $\hat{x}^a_k$ and covariance P^a_k are also updated for use in the next iteration.

$$K_k = P_{xy} P_y^{-1} \tag{3.36}$$

$$\hat{x}^+_k = \hat{x}^-_k + K_k(y_k - \hat{y}_k) \tag{3.37}$$

$$P^+_k = P^-_k - K_k P_y K^T_k \tag{3.38}$$

This concludes the presentation of the UKF. It has been argued in [30] and [34] that for Gaussian inputs, the filter approximates the mean and the covariance up to the third momement terms regardless of the nonlinearities. For non-Gaussian inputs, accuracy up to the second order is guaranteed. Because the UKF does not explicitly truncate the third and higher order terms, the errors in those terms

can also be reduced [13]. The third and fourth moments of the distributions are also approximated to an accuracy determined by the parameters α and β. The UKF has a computational complexity of the same order as the EKF [34]. The fact that it is more accurate and can accommodate discontinuities make it preferable for all applications where one might consider the use of the EKF. However, because it still relies in a linearization method, this filter may also diverge if initialized too far from the true state.

Besides the UKF, alternative derivative free Kalman filters exist such as the Central Difference Kalman Filter (CDKF)[12] and the closely related Divided Difference Filter (DD1) [24], which use Sterling's polynomial interpolation method for the linearization. These have been reported [32] to have comparable performance to the UKF, hence only the later is considered in this report as it is more widespread in the literature [28],[31].

3.3 Particle Filter

While the UKF is an improvement over the EKF, it still bound by the Kalman filtering framework, namely that it requires the estimated density to be well approximated by a Gaussian. As a result there will be significant estimation errors for distinctly non-Gaussian and especially multimodal densities, because the mean may lie outside the areas of high probability (Figure 3.3).

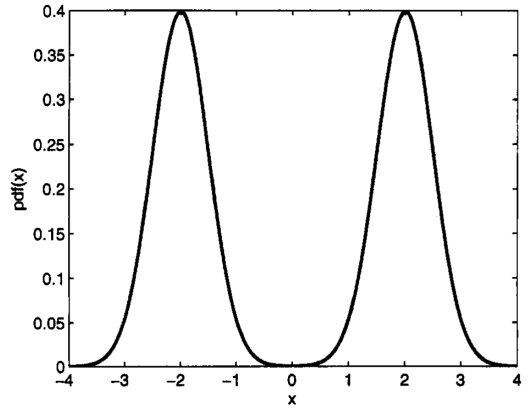

Figure 3.3: A bimodal probability density function. Due to the symmetry, the mean lies at 0 which has the lowest probability [28].

In order to accommodate such densities, an alternative approach based on Sequential Monte Carlo (SMC) methods was developed at first by the control community in the late 70s [9] and subsequently refined for practical applications in the early 90s [8]. SMC methods make no linearity or Gaussianity assumptions but instead stochastically approximate probability densities as clouds of point masses. This eliminates the need for the assumptions imposed by the Kalman filter and theoretically leads to superior estimation performance, provided a sufficient number of point masses (and therefore, computational power) is available.

The main idea of these methods, which are commonly referred to as Particle Filters, is to generate a large number of samples (particles) from some proposed probability density function and, using the available process and measurement models, propagate them through time and evaluate their likelihood. Thereafter, a resampling step is employed where the highest weighted samples multiply at the expense of the least weighted ones. These procedure is repeated until the density formed by the samples converges to the posterior density.

The development of the Particle Filter here follows mainly the work of [6] and [26]. The basis of SMC methods is Monte Carlo integration, which addresses the numerical evaluation of the following multidimensional integral:

$$E[g(x)] = \int g(x)\pi(x)dx \tag{3.39}$$

Examples of such integrals are those previously mentioned in the Bayesian estimator (3.3 - 3.6). The integrand of (3.39) is a factorization so that $\pi(x)$ is a probability density function satisfying the conditions $\pi(x) \geq 0$ and $\int \pi(x)dx = 1$. For the filtering application considered here, this density is the joint posterior density $p(X_k|Y_k)$ with the set $X_k = \{x_1, x_2, \ldots x_k\}$ denoting the state history up to time k. If one now draws $N \gg 1$ samples $\{X_k^i, i = 1, 2, \ldots, N\}$ distributed according to the joint posterior, then the Monte Carlo estimate of the above integral becomes:

$$\bar{E}[g(X_k)] = \frac{1}{N}\sum_{i}^{N} g(X_k^i) \tag{3.40}$$

Because the samples are drawn independently, the resulting estimate will be unbiased and due to the law of large numbers, it will almost surely converge to the integral as the number of samples tend to infinity. Thus the potentially

intractable integral is being mapped to a discrete sum which can be computed. Furthermore, if the variance of $g(X_k)$ is bounded

$$\sigma^2 = \int \big(g(X_k) - E[g(X_k)]\big)^2 p(X_k^i|Y_k) dX_k < \infty \tag{3.41}$$

Then by the central limit theorem, the estimation error will converge in distribution to a normal distribution with the same variance:

$$\lim_{N\to\infty} \sqrt{N}\big(\bar{E}[g(X_k)] - E[g(X_k)]\big) \sim N(0, \sigma^2) \tag{3.42}$$

More importantly, the convergence rate will be of the order $O(1/N^2)$ and will be independent of the dimension of the integral to be estimated. If it were possible to sample from the joint posterior, then the above approach could be applied and the estimation would be complete. Unfortunately this is not the case because the posterior is typically known up to a proportionality constant, however this difficulty can be surmounted using the importance sampling method.

In importance sampling, the samples are generated instead from a known density $q(X_k|Y_k)$ referred to as proposal density. This density needs to be similar to the posterior in the sense that it has the same support:

$$p(X_k|Y_k) > 0 \Rightarrow q(X_k|Y_k) > 0 \qquad \forall X_k \in \mathbb{R}^n \tag{3.43}$$

If this condition is valid, then the integral (3.39) can be rewritten in terms of the proposal density

$$E[g(X_k)] = \int g(X_k) p(X_k|Y_k) dX_k = \int g(X_k) \frac{p(X_k|Y_k)}{q(X_k|Y_k)} q(X_k|Y_k) dX_k \tag{3.44}$$

And, having generated N independent samples distributed according to $q(X_k|Y_k)$, the Monte Carlo estimate can is given by

$$\bar{E}[g(X_k)] = \sum_i^N g(X_k^i) w_k^i \tag{3.45}$$

or more explicitly by

$$\hat{p}(X_k|Y_k) = \sum_{i=1}^{N} w_k^i \delta(X_k - X_k^i) \tag{3.46}$$

with $w_k^i \in [0,1]$ being the importance weights, normalized so as to eliminate the need for the normalizing factor of $p(X_k|Y_k)$:

$$w_k^i = \frac{\bar{w}_k^i}{\sum_{j=1}^{N} \bar{w}_k^j} \tag{3.47}$$

$$\bar{w}_k^i = \frac{p(X_k^i|Y_k)}{q(X_k^i|Y_k)} \tag{3.48}$$

While Monte Carlo integration can be accomplished this importance sampling method, it is unsuitable for recursive estimation. Because of the way the importance weights are defined (3.48), they need to be recomputed across the entire state sequence every time the observation history vector Y_k is updated. This causes to the computational complexity of these weights to grow over time. The solution to this problem is to reformulate importance sampling so that the previously simulated states X_k are left unmodified. That is, only the marginal posterior density $p(x_k|Y_k)$ is of interest at each time step.

A recursive formulation of $p(X_k|Y_k)$ can be derived in terms of $p(X_{k-1}|Y_{k-1}$ which is assumed to be known from the previous time step and the marginals $p(y_k|x_k)$ and $p(x_k|x_{k-1})$. This is accomplished by the application of Bayes' Rule as well as the Markov property:

$$\begin{aligned} p(X_k|Y_k) &= \frac{p(y_k|X_k, Y_{k-1})p(X_k|Y_{k-1})}{p(y_k|Y_{k-1})} \\ &= \frac{p(y_k|X_k, Y_{k-1})p(x_k|X_{k-1}, Y_{k-1})p(X_{k-1}|Y_{k-1})}{p(y_k|Y_{k-1})} \\ &= \frac{p(y_k|x_k)p(x_k|x_{k-1})}{p(y_k|Y_{k-1})} p(X_{k-1}|Y_{k-1}) \qquad (3.49) \\ &\propto p(y_k|x_k)p(x_k|x_{k-1})p(X_{k-1}|Y_{k-1}) \qquad (3.50) \end{aligned}$$

The proposal density can also be factorized by imposing a Markov assumption.

One can therefore sample from this proposal density by augmenting the each of the previous samples available from $q(X_k|Y_k)$ with the new state.

$$q(X_k|Y_k) = q(x_k|X_{k-1}, Y_k)q(X_{k-1}|Y_{k-1}) \tag{3.51}$$

Substituting this expression and 3.50 into (3.48) leads to the desired recursive expression for the importance weights:

$$\begin{aligned} w_k^i &\propto \frac{p(y_k|x_k^i)p(x_k^i|x_{k-1}^i)p(X_{k-1}^i|Y_{k-1})}{q(x_k^i|X_{k-1}^i, Y_k)q(X_{k-1}^i|Y_{k-1})} \\ &= w_{k-1}^i \frac{p(y_k|x_k^i)p(x_k^i|x_{k-1}^i)}{q(x_k^i|x_{k-1}^i, y_k)} \end{aligned} \tag{3.52}$$

The weights w_{k-1}^i are available prior to the iteration, while the likelihood $p(y_k|x_k^i)$ and transition probability $p(x_k^i|x_{k-1}^i)$ can be computed at each time step using the available process and measurement models. Then the estimate of the desired marginal posterior density can be given as

$$\hat{p}(x_k|Y_k) = \sum_{i=1}^{N} w_k^i \delta(x_k - x_k^i) \tag{3.53}$$

This method is referred to as Sequential Importance Sampling (SIS). The samples are initialized to be distributed according to $q(x_0)$, each having a weight equal to $1/N$. At each time step, new samples are drawn from $q(x_k|x_{k-1}, y_k)$ and then their weights are evaluated using (3.52). The weights are normalized and from the resulting approximated posterior, any statistic of interest can be computed to serve as a state estimate, namely the conditional mean.

There is, however, a problem with the SIS method; it has been shown [7] that the variance of the importance weights w_k^i grows stochastically over time. After a few recursions, one particle will have "survived" and have a normalized importance weight tending to 1 while the rest will tend to 0. This degeneracy phenomenon leads to a poor approximation of the posterior and computation power is wasted in updating the negligible samples, making them impractical for applications. A way to mitigate this problem, is to introduce an additional selection (resampling) step in the algorithm.

The purpose of this resampling step is twofold; to eliminate the samples whose importance weight is zero and multiply the number of "surviving" samples proportionally to their importance weight in a manner reminiscent of genetic algorithms. This is accomplished by generating N i.i.d. samples from the discrete approximation of the posterior given by (3.53) and weighing them equally such that $w_k^i = 1/N$.

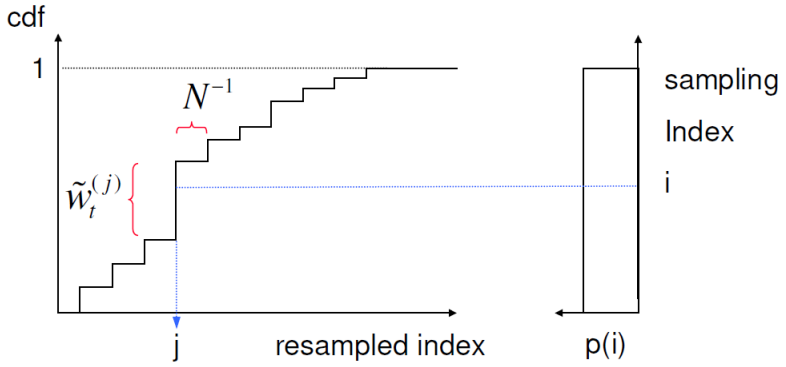

Figure 3.4: Illustration of the general resampling process [33].

This sampling process is further depicted in Figure 3.4. Firstly, a random variable U is drawn from the uniform distribution on the [0,1] interval. Secondly, the weights are summed cumulatively until the accumulated sum exceeds the random variable. When this occurs, the particle corresponding to the last added weight w_k^j is selected as a sample for the next filter iteration. The process is repeated N times for N samples in total. Observe that in this way, samples with a larger weight are more likely to be selected, since a larger weight has a higher probability of expanding the accumulated sum past the index i.

There are several resampling schemes that one might employ, the most common being multinomial, residual, stratified and systematic resampling [11]. Among these, systematic resampling appears to yield the highest estimation quality and has been argued in [26] and [31] to minimize the variance $sigma^2$ of the filter.

In this sampling method, N ordered numbers U^i are generated such that they obey:

$$U^i = \frac{(i-1)+U}{N} \tag{3.54}$$

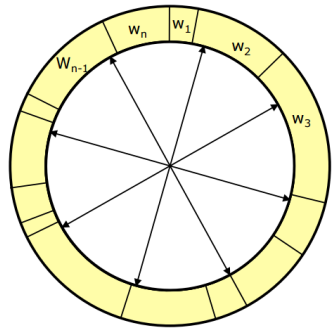

Figure 3.5: Graphical representation of systematic resampling [31].

With U being again a random variable sampled from the uniform distribution U(0,1). Then the weights are again summed cumulatively and whenever this sum exceeds one of these ordered numbers for the first time, the particle associated with the last added weight is chosen as the next sample. The method can be visualized as a roulette whose circumference is occupied by the importance weights proportionally to their magnitude with equally spaced arrows pointing towards them (Figure 3.5). Each arrow corresponds to one number U^i. Drawing from U(0,1) is the equivalent of spinning the roulette once and having the particles which share the same index as the weights pointed to by the arrows, "survive".

In order to determine when the resampling step should be employed, a suitable measure of the degeneracy phenomenon is needed. Such a measure is the effective sample size N_{eff} defined as [26]:

$$N_{eff} = \frac{1}{\sum_{i=1}^{N}(w_k^i)^2} \tag{3.55}$$

So that $0 \leq N_{eff} \leq N$. In the case of the samples being uniformly weighted $N_{eff} = N$ and in the case of extreme degeneracy where only one weight is non-zero, $N_{eff} = 1$. Then it is up to the engineer to assign a threshold below which resampling will be carried out.

The augmentation of the SIS method with a resampling step leads to what is called the Sequential Importance Resampling (SIR) method, otherwise known as the bootstrap filter. The functionality of the filter is depicted in Figure 3.6. Here the filter is initialized with $N = 10$ uniformly weighted particles drawn from

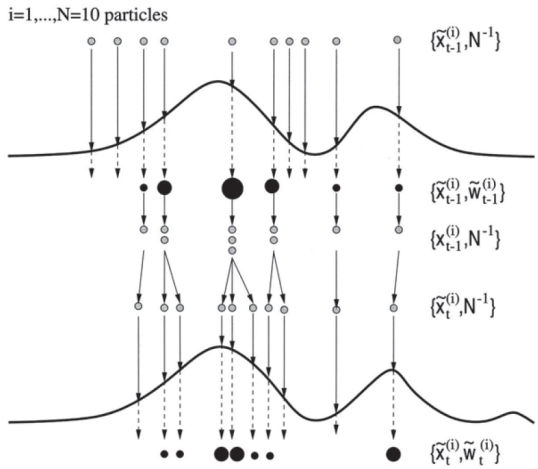

Figure 3.6: Visual demonstration of the particle filter [6].

some initial proposal density $q(x_0)$. The importance weight of each particle is then updated using (3.52), the weights being proportional to the likelihoods of each particle, given the latest available measurement y_k. The particles along with their weights approximate the target density $p(x_k|Y_k)$. If the resampling step is applied, then the highly weighted samples multiply at the expense of the less weighted ones and the resulting set of particles becomes uniformly weighted once again while still approximating the target density. These samples are then propagated to the next time step again through importance sampling, varying the particles and thus the above procedure is repeated.

The only issue left to address is perhaps the most critical one; the selection of the proposal density $q(x_k|x_{k-1}^i, y_k)$. It was shown [5] that the optimal proposal density is the one which minimizes the variance of the importance weights, conditional on x_{k-1}^i and y_k. The same author then showed in [7] that this density is:

$$\begin{aligned} q(x_k|x_{k-1}^i, y_k)_{opt} &= p(x_k|x_{k-1}^i, y_k) \\ &= \frac{p(y_k|x_k, x_{k-1}^i)p(x_k|x_{k-1}^i)}{p(y_k|x_{k-1}^i)} \end{aligned} \tag{3.56}$$

After applying Bayes' rule, it can be substituted into the importance weight

equation (3.52) to yield:

$$w_k^i \propto w_{k-1}^i p(y_k|x_{k-1}^i) \tag{3.57}$$

This result indicates that the importance weights can be computed before the samples are updated to time k. However, in order to use the optimal proposal density, one needs to be able to sample from $p(x_k|x_{k-1}^i, y_k)$ and subsequently evaluate $p(y_k|x_{k-1}^i)$ up to a normalizing constant which is not always possible in the general. This leads to the use of suboptimal proposal densities, the most popular among them arguably being the transition prior $p(x_k|x_{k-1}^i)$

$$q(x_k|x_{k-1}^i, y_k) = p(x_k|x_{k-1}^i) \tag{3.58}$$

Two arguments motivate this choice of density. Firstly, the transition prior typically has a larger support than the target density and is more likely to include it. Secondly, it simplifies the computation of the importance weights. If (3.58) is substituted into (3.52), it will cancel out the nominator of the later, leading to:

$$w_k^i \propto w_{k-1}^i p(y_k|x_k^i) \tag{3.59}$$

Thus only the evaluation of the observation likelihood density is needed to update the weights.

3.4 Unscented Particle Filter

The particle filter imposes few, easily satisfied assumptions, namely that the process and measurement functions are available, that it is possible to sample from the process noise and prior densities and that it is possible to evaluate the likelihood density at the samples. Otherwise it is able to handle any nonlinearities and arbitrary distributions. However, the use of the transition prior instead of the optimal proposal density for importance sampling, can have a significant negative impact on the filter's performance.

Specifically, the prior density may not overlap significantly with the observation likelihood density (e.g. the likelihood density lies at one of the tails of the prior) or the likelihood density may be too narrow. In both cases, it is very likely that the particles will not be located in the area of high observation likelihood

(Figure 3.7). This is because of the discrete nature of the approximation, which is only perfect for infinite particles. In practice however, they are finite and if there are computational considerations, they should be as few as possible.

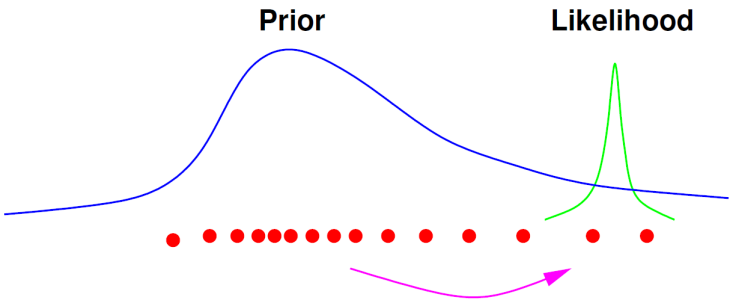

Figure 3.7: Situation where the prior and the observation likelihood densities do not overlap. The importance sampling using the optimal proposal density is the only step where the samples can be moved closer to the likelihood density [33].

Given this situation, the particles will not be weighted appropriately based on the observation likelihood. If the resampling step is not applied, the result will be the aforementioned degeneracy phenomenon. If it is applied, then a different kind of degeneracy occurs; the resampling causes the particles to multiply randomly and after several iterations (again due to their discrete nature), there will be more and more particles occupying the exact same position and subsequent resampling will amplify this phenomenon until all particles are stacked on the same point. This phenomenon is referred to as sample depletion.

If the optimal proposal density were used, then it would be conditioned on the measurement y_k and thus it would move the particles to the area of high observation-likelihood, thus avoiding sample depletion. But as it cannot be used, the task is to find a usable proposal density which incorporates the latest measurement. The main idea then is to approximate the optimal proposal density by a tractable Gaussian density, generated by the Kalman filter so that [32]:

$$q(x_k|x_{k-1}^i, y_k)_{opt} \approx p_N(x_k|x_{k-1}^i, y_k) = N(\hat{x}_{i,k}^+, P_{i,k}^+) \tag{3.60}$$

This approach was originally proposed by [7] where the EKF was used to gen-

erate the approximation. Subsequently, the same idea was used in conjunction with a UKF, resulting in the Unscented Particle Filter [33]. It is essentially a particle filter with a bank of N UKFs. Each sample, having been generated by the initial proposal $q(x_0)$, is propagated through the equations (3.29 - 3.38) using the latest measurement y_k. This means that each particle maintains its own augmented mean and covariance estimates as discussed in the UKF section. Once each particle is updated, it is redrawn from the updated Gaussian distribution (3.60). Then the importance weighting is carried out just as in the SIR filter, but the weights are updated according to:

$$w_k^i = w_{k-1}^i \frac{p(y_k|x_k^i)p(x_k^i|x_{k-1}^i)}{p_N(x_k^i|x_{k-1}^i, y_k)} \tag{3.61}$$

with the transition prior $p(x_k^i|x_{k-1}^i)$ also approximated using the prior mean and covariance of the UKF, if it cannot be evaluated directly:

$$p(x_k^i|x_{k-1}^i) = N(x_{i,k}^-, P_{i,k}^-) \tag{3.62}$$

The resampling step remains largely unaffected, with the caveat that both the particles and their respective means and covariances from their associated UKFs are multiplied or discarded. This completes one iteration of the algorithm. While the UPF has been shown to be more accurate than the standard particle filter [33], the fact that N Unscented Kalman filters need to be run in parallel means that it will impose larger computational and memory requirements. It is argued however [26] that this computational load is significantly offset by the reduction of the total samples required to achieve a certain performance.

Chapter 4

Implementation and Results

This chapter deals with the application of the presented non-linear state-estimation techniques for the localization of the HAKO tractor in its simulation environment. In the first section, the three filters are implemented for the case where the map represents the relevant orchard features correctly. A discussion is given on each filter's performance as well as any specific variations employed in its application. In the second section, these state-estimation techniques are tested when the orchard's tree rows have holes which are not represented by the map. It is shown that this negatively impacts their performance and some techniques for dealing with this problem are given for each filter. Finally, a comparison of the resulting variant filters is given.

4.1 Application with Normal Map

4.1.1 Unscented Kalman Filter

The UKF was implemented first because the Kalman filter has a good track-record in applications thus, from an engineering perspective, it is imperative to try one of its extensions before proceeding to a different solution. Furthermore,

as it has also been applied by [10, 22], it could serve as a validation step in the early development stages of the system and the laser-scanner function.

The filter is implemented as a standalone function whose inputs are the previous augmented state mean and covariance estimates as well as the input vector estimate $\hat{u}_k$ and the laser scanner measurement y_k. The intrinsic parameters of the vehicle and laser-scanner as well as the covariance matrices $R_{1,k}$ and R_2 of the process and measurement noise respectively, can be made available as global variables or passed to the function directly. The function is executed as soon as a new measurement is available.

The implemented filter follows closely the algorithm described in Section 3.2 with one key difference; the augmented state vector does not include the measurement noise estimates:

$$\hat{x}_k^a = \begin{bmatrix} \hat{x}_k & \hat{y}_k & \hat{\theta}_k & \hat{\mathbf{w}}_k \end{bmatrix} \tag{4.1}$$

This is possible because the measurement noise is additive, which allows the modification of the measurement step so that the matrix P_y is computed as in the standard Kalman filter. This change is motivated by the large number of elements (722) of the measurement vector. If the augmentation took place, it would lead to a 727 by 727 size P_k^a matrix which not only would increase the computation effort needed, but also lead to numerical instabilities.

The filter was initialized with the true state as its mean and a covariance matrix given as follows:

$$\hat{x}_0^a = \begin{bmatrix} 6 & 50 & -\pi/2 & 0_{1\times 2} \end{bmatrix}^T \tag{4.2}$$

$$P_0^a = \begin{bmatrix} 0.1 & 0 & 0 & 0_{1\times 2} \\ 0 & 0.1 & 0 & 0_{1\times 2} \\ 0 & 0 & 0.1 & 0_{1\times 2} \\ 0_{2\times 1} & 0_{2\times 1} & 0_{2\times 1} & R_{1,k} \end{bmatrix} \tag{4.3}$$

It is imperative that the filter is initialized close to the true position both to ensure a reasonable linearization, but more importantly because the environment (Figure 2.5) is not sufficiently distinct since the tree rows are identical. Therefore the relative distance information from the laser scanner will not be sufficient to identify the specific row in which the vehicle is driving. This will result in the UKF moving the estimate to the most likely point of the row closest to the initial state estimate. In practice, this restriction is not significant

because either the vehicle begins operation from a known position (its recharging station) or it has a good initial estimate provided by the RTK-GPS and its gyroscope.

The covariance matrix given by (4.3) defines the initial uncertainty of the position and orientation. If this information is not available (as in this case) it can be chosen arbitrarily, bearing in mind that it determines the spread of the sigma points in the first iteration of the UKF. Thus it should not be chosen too large, else some of the sigma points may be moved to a point of high likelihood in an adjacent row and cause the estimate to diverge.

In the time update step, the ith sigma point was propagated through (2.9) with the process noise component $X^{w}_{i,k-1}$ included nonlinearly as:

$$X^{x}_{i,k} = X^{x}_{i,k-1} + \begin{bmatrix} (v_{k-1} + X^{w_1}_{i,k-1})cos\left(\theta_{i,k-1} + \frac{\Delta\theta_{i,k-1}}{2}\right) \\ (v_{k-1} + X^{w_1}_{i,k-1})sin\left(\theta_{i,k-1} + \frac{\Delta\theta_{i,k-1}}{2}\right) \\ \Delta\theta_{i,k-1} \end{bmatrix} \tag{4.4}$$

$$\Delta\theta_{i,k} = \frac{(v_k + X^{w_1}_{i,k}) + tan(\theta_{SA,k} + X^{w_2}_{i,k})}{l} \tag{4.5}$$

Here, w_1 and w_2 denote the first and second elements of the process noise vector respectively.

In the modified measurement step, the sigma points were passed through the laser-scanner function described in Section 2.3 and the covariance P_y was computed by exploiting the additivity of the noise:

$$Y_{i,k} = h_k(X^{x}_{i,k}, \hat{e}_k) \tag{4.6}$$

$$P_y = \sum_{i=0}^{2n} W^{m}_{i}(Y_{i,k} - \hat{y}_k)(Y_{i,k} - \hat{y}_k)^T + R_2 \tag{4.7}$$

The Kalman gain and posterior estimates $\hat{x}^{+}_{k}$ and P^{+}_{k} follow as elaborated in the previous chapter. The phase plot of one full simulation using this filter is given in Figure 4.1. It is observed that the filter is successful in tracking the robot's position as it drives, with the estimate almost overlapping the true trajectory except at the upper right corner where there is a slight deviation. This can also

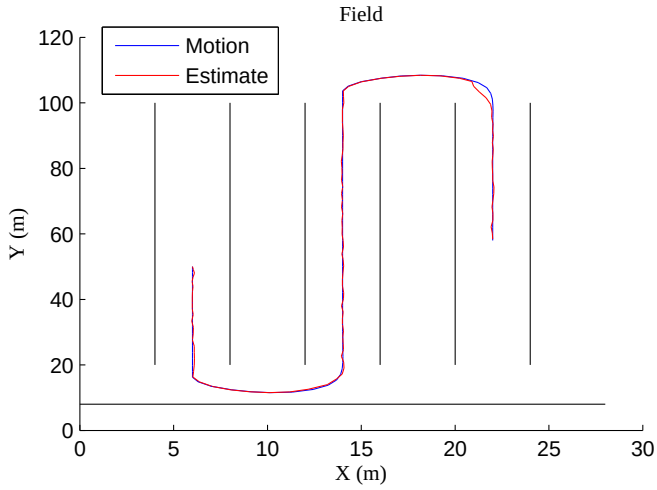

Figure 4.1: Phase plot of UKF estimate of the robot's position.

be observed if each state is plotted individually along with the corresponding estimate as done in Figure 4.2.

Since the estimates are so close to the true states, their root squared errors $e_k^x = \sqrt{(\hat{x}_k - x_k)^2}$ are given as well in Figure 4.3 in order to observe where any deviations occur and their relative magnitude. Here it is seen that, for the states x_k and θ_k, the estimation error is consistently small except near the two time samples 30 and 80-85. Cross-referencing with the state plots in Figure 4.2 reveals that these are the points where the vehicle turns from one tree row to the next.

These errors can be attributed to two causes. First, the system nonlinearities are being excited by the turning (specifically, θ_k changes value, causing x_k and y_k to evolve nonlinearly) which leads to the linearization errors of the UKF becoming more dominant. Second, there is limited information coming in from the laser-scanner during the first turn as only a single tree row perpendicular to the others is being observed (Figure 2.10) and no row can be observed during the second turn. Thus the UKF relies heavily on predictions based on odometry during turning.

It is customary in Kalman filtering to report the state variances estimated by the filter, as they provide a measure of the uncertainty during operation. These are shown in Figure 4.4. It can be observed again that they, like the estimation error, are kept small everywhere except for when the turns take place, with the

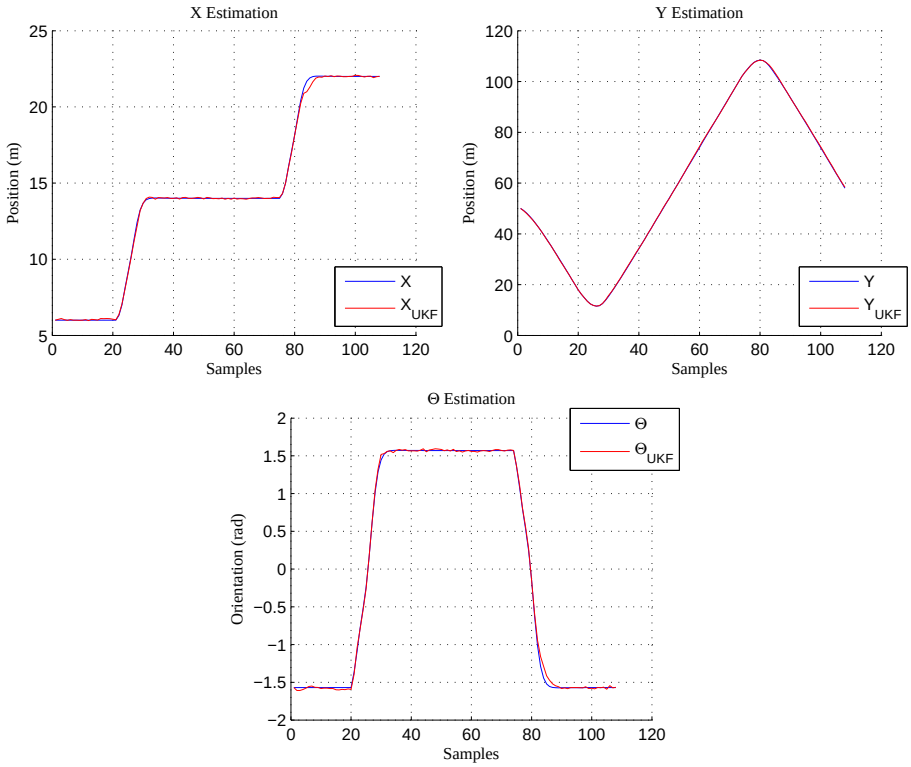

Figure 4.2: Individual plots of UKF estimates versus the true states.

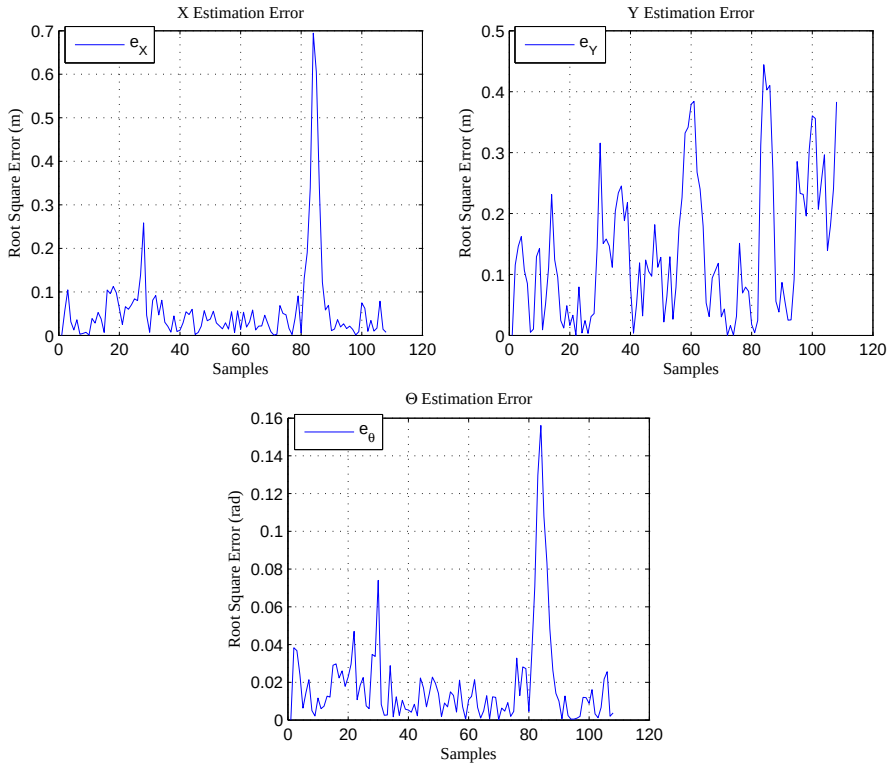

Figure 4.3: UKF Root Square errors for each state.

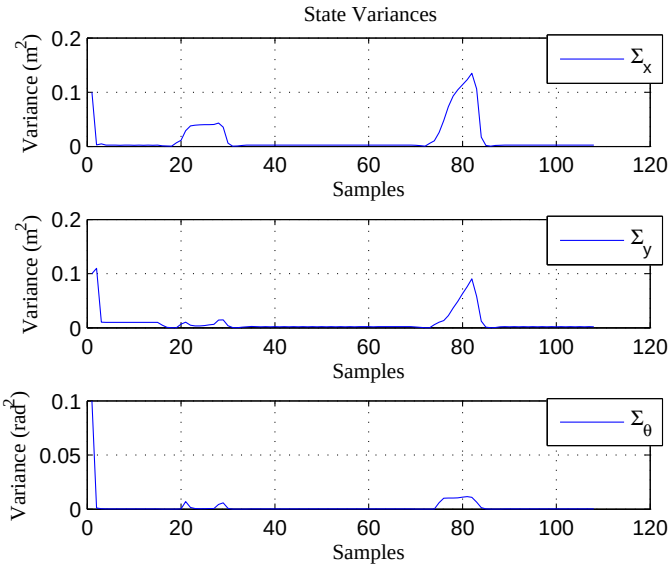

Figure 4.4: Evolution of UKF variance estimates over time.

second turn causing the most significant uncertainty increase.

Lastly, an assessment of the estimation performance and computation time of the filter is needed for comparison with subsequent solutions. For this purpose, the Root Mean Square Error (RMSE) across 100 realizations was chosen because it was frequently employed in the referenced material (e.g. [7, 10, 26, 33]). It is defined by:

$$E_X = \frac{1}{100}\frac{1}{T}\sum_{i=1}^{100}\sum_{k=1}^{N}\sqrt{(\hat{x}_k^i - x_k^i)^2} \tag{4.8}$$

The RMSE values for each state estimate of the UKF as well as their sum, are tabulated on the first row of Table 4.1 along with the average total running time of each simulation.

4.1.2 Particle Filter

The next step was to implement the SIR filter, both to compare it to the UKF solution but also to set up the foundations of the subsequent UPF implementation. The particle filter is also implemented as a single function to be called each time a measurement is taken. Its inputs are again the input estimate and the measurement as well as a structure which contains the particles, their associated weights and the state estimate.

The initialization of the filter involves the generation of $N = 100$ samples. This number of particles was chosen after a simulation trial where, starting with a small number of samples, it was incremented until there was no apparent performance increase. The samples are drawn from a Gaussian distribution centered around the true state with the following covariance matrix:

$$\Sigma_0 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0.5 \end{bmatrix} \tag{4.9}$$

As a result, using the 3σ rule, the particles are distributed within at least 3m of the initial position, spanning the entire width of the row. The maximum deviation of the orientation is 17.19 degrees allowing for a large uncertainty from the gyro measurement. While a uniform distribution would better represent complete uncertainty, the availability of an initial guess (based on prior knowledge or the RTK-GPS and gyro measurements) makes it more sensible to assume a Gaussian distribution around it.

These initial samples need to be drawn from the proposal density, in this case the transition prior $p(x_k|x^i_{k-1})$. This is accomplished simply by propagating them through the kinematics (2.9):

$$x^i_k = x^i_{k-1} + \begin{bmatrix} (v_{k-1} + w^i_{1,k-1})cos\left(\theta^i_{k-1} + \frac{\Delta\theta^i_{k-1}}{2}\right) \\ (v_{k-1} + w^i_{1,k-1})sin\left(\theta^i_{k-1} + \frac{\Delta\theta^i_{k-1}}{2}\right) \\ \Delta\theta^i_{k-1} \end{bmatrix} \tag{4.10}$$

$$\Delta\theta^i_k = \frac{(v_k + w^i_{1,k}) + tan(\theta_{SA,k} + w^i_{2,k})}{l} \tag{4.11}$$

Where $w^i_{1,k}$ and $w^i_{2,k}$ are elements of the process noise vector w^i_k, which is drawn

from the process noise probability density, in this case the Gaussian $N(0, R_{1,k})$.

The importance weights, denoted here as q_k^i, of each sample are then evaluated according to (3.59). The observation likelihood $p(y_k|x_k^i)$ can be evaluated as the multivariate Gaussian distribution of the measurement noise:

$$p(y_k|x_k^i) = p[y_k - h(x_k^i, \hat{e}_k)] \tag{4.12}$$

$$= \frac{1}{\sqrt{2\pi^n|R_2|}} exp\left(\frac{-[y_k - h(x_k^i, \hat{e}_k)]^T R_2^{-1}[y_k - h(x_k^i, \hat{e}_k)]}{2}\right) \tag{4.13}$$

Notice that R_2 needs to be inverted. In general, the inversion of a matrix of this dimension can be problematic, both in terms of numerical stability and computational effort. Therefore it is possible to exploit the assumption of independence imposed on the laser rays, namely that:

$$p(y_k|x_k^i) = \prod_j p(y_k^j|x_k^i) \tag{4.14}$$

With y^j containing the distance and angle returned from the jth laser ray. Thus the problem of evaluating a 722-dimensional, multivariate Gaussian distribution is reduced to evaluating a bivariate one 361 times, then multiplying them together. Once the importance weights are assigned, the conditional mean is computed to serve as the state estimate:

$$\hat{x}_k = \sum_{i=1}^{N} q_k^i x_k^i \tag{4.15}$$

Thereafter, in principle, the N_{eff} is computed to determine if resampling should take place. However, it was found through simulation trials that resampling every time yields the best estimation quality for this application. The phase plot of the SIR filter estimate is given in Figure 4.5. The particle clouds formed at every 5 time samples are also shown on the same figure, since they convey information about the confidence the filter has in the estimate.

It can be observed that the initial particle cloud of the SIR filter quickly converges around the estimate and expands very briefly during turning. The state estimate overlaps the true trajectory even more so than the UKF, implying

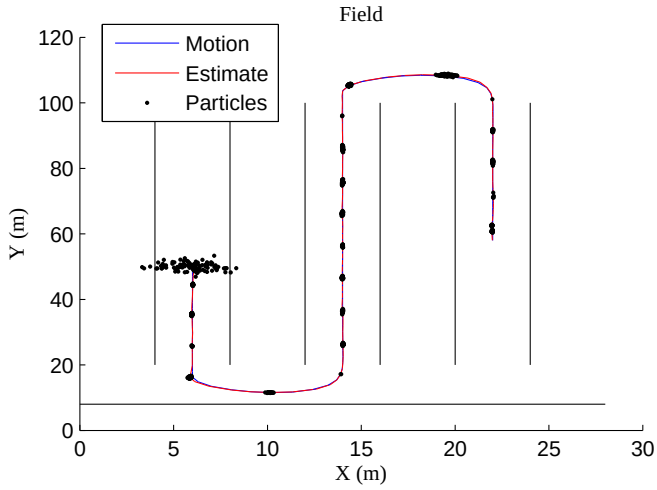

Figure 4.5: Phase plot of SIR filter estimate of the robot's position.

a performance increase. This is also suggested by the individual state plots in Figure 4.5, where it is almost impossible to distinguish the state from the estimate.

The root square errors of each state can again be used to assess the performance and are given in Figure 4.7. Here it is seen that in the states x_k and θ_k, similarly to the UKF, there is an increase in the estimation error when the vehicle turns. However, the overall magnitude of the error is significantly reduced and its spikes, due to lack of information, are less than half those of the ones found in the UKF. Notably, there is a high initial estimation error for the y_k, likely caused by the lack of information regarding the vehicle's position on the y axis as it drives down the row. This observation is supported from the fact that, as the vehicle approaches the turning point (where the horizontal tree row can be observed), the estimation error shrinks and thereafter grows again slightly when that row cannot be observed anymore.

Finally, the RMSE values for every state are calculated across 100 realizations and tabulated in the second row of Table 4.1 along with the average simulation time. It was found that, on average, the SIR filter has a superior performance over the UKF but takes about 10 times as long to execute. However, the computation time is not prohibitive and it is expected that implementation in more efficient languages such as C or C++ will help reduce the computational difference between the two filters, since the SIR filter executes significantly more for loops, which are very inefficient in MATLAB.

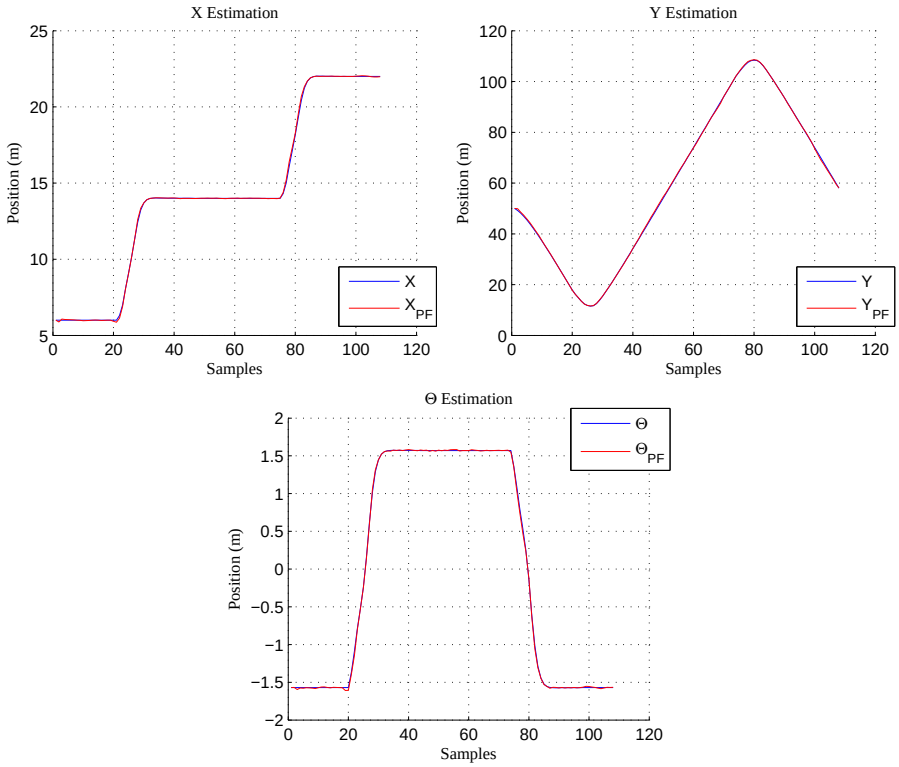

Figure 4.6: Individual plots of SIR filter estimates versus the true states.

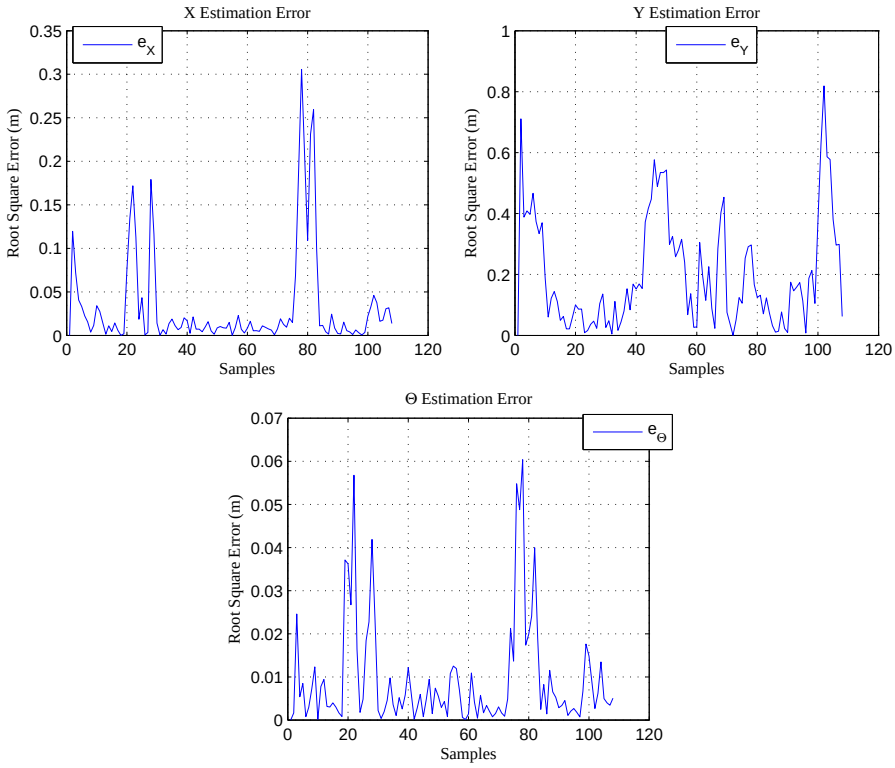

Figure 4.7: SIR filter Root Square errors for each state.

Notice that an alternative approach would have been to draw the initial samples from a uniform or Gaussian distribution, spanning the entire map, in the case where no good initial guess is available. However, such an approach would require a very large number of particles and even then, a distinct map would be needed for only the particles in the correct row to survive. Otherwise, a good estimate could only be obtained by using the highest weighted particle, since the mean would not necessarily lie in an area of high probability. Due to these reasons, as well as the high likelihood of a good initial guess, this approach is not pursued in this project.

4.1.3 Unscented Particle Filter

Having successfully implemented the SIR filter, it was augmented with the UKF based proposal distribution so as to determine if a higher performance/computation time ratio could be achieved over the conventional particle filter. The new filter was implemented as a function similar in structure to that of the SIR one. Its inputs are the usual control estimate u_k and measurement y_k and then an augmented particle structure, containing each particle in augmented state form, the associated augmented covariances and the importance weights.

The filter is initialized identically to the SIR, with the only difference being that only $N = 10$ particles were used. Instead of propagating these samples through the kinematics, the filter makes a call to the UKF function for each particle. The initial covariances used in these UKF calls are the same as those in (4.3). Then each particle is redrawn from its respective Gaussian proposal density $N(x^+_{i,k}, P^+_{i,k})$ to complete the time update step.

The importance weights need to be evaluated according to (3.61). The density $p(y_k|x^i_k)$ is computed identically to how it was in the SIR filter to avoid numerical issues. Since the process noise is injected in a nonlinear way, the density $p(x^i_k|x^i_{k-1})$ was approximated by the distribution $N(x^-_{i,k}, P^-_{i,k})$, since its parameters can also be extracted from the previous UKF calls.

$$p(x^i_k|x^i_{k-1}) = \frac{1}{\sqrt{2\pi^n|P^-_{i,k}|}} exp\left(\frac{-[x^i_k - x^-_{i,k}]^T(P^-_{i,k})^{-1}[x^i_k - x^-_{i,k}]}{2}\right) \tag{4.16}$$

The proposal density term in the denominator of (3.61) is given similarly as:

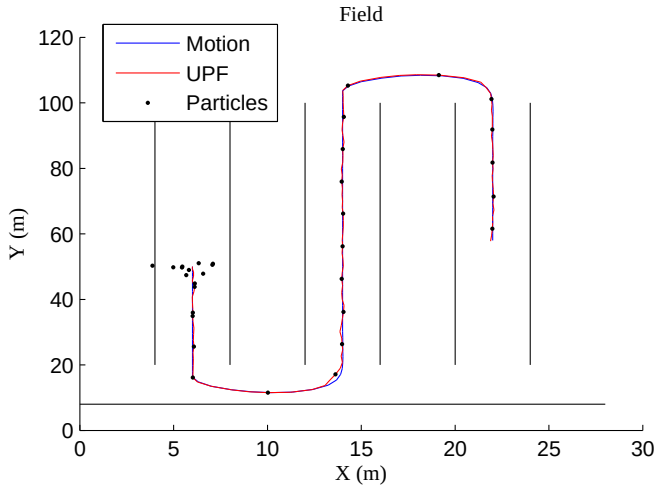

Figure 4.8: Phase plot of UPF estimate of the robot's position.

$$p_N(x_k^i|x_{k-1}^i, y_k) = \frac{1}{\sqrt{2\pi^n|P_{i,k}^+|}} exp\left(\frac{-[x_k^i - x_{i,k}^+]^T(P_{i,k}^+)^{-1}[x_k^i - x_{i,k}^+]}{2}\right) \quad (4.17)$$

Otherwise, the evaluation of the state estimate as well as the resampling step, is identical to the SIR filter. It was also found best to resample at every iteration. The phase plot of the UPF estimate and associated particles is given in Figure 4.8. Similarly to the two previously considered filters, the estimate overlaps the true state. The individual state estimate plots are given in Figure 4.9 and confirm the high estimation quality shown by the phase plot.

More information can be gained by inspecting the root square errors. Overall, the UPF displays the same estimation quality as the SIR filter, with the most notable difference being a significant improvement in the estimation of the y_k state; after the first turn, the estimation error does not grow as large as the error in the SIR and UKF filters.

The RMSE values are again computed for 100 realizations and tabulated on the third row of Table 4.1. It can be seen that indeed the estimates of the x_k and θ_k are near those of the SIR while the estimation quality of y_k is improved by approximately 24%. The total time required to run this filter is averaged at 45.33s a time very close to that of the SIR filter.

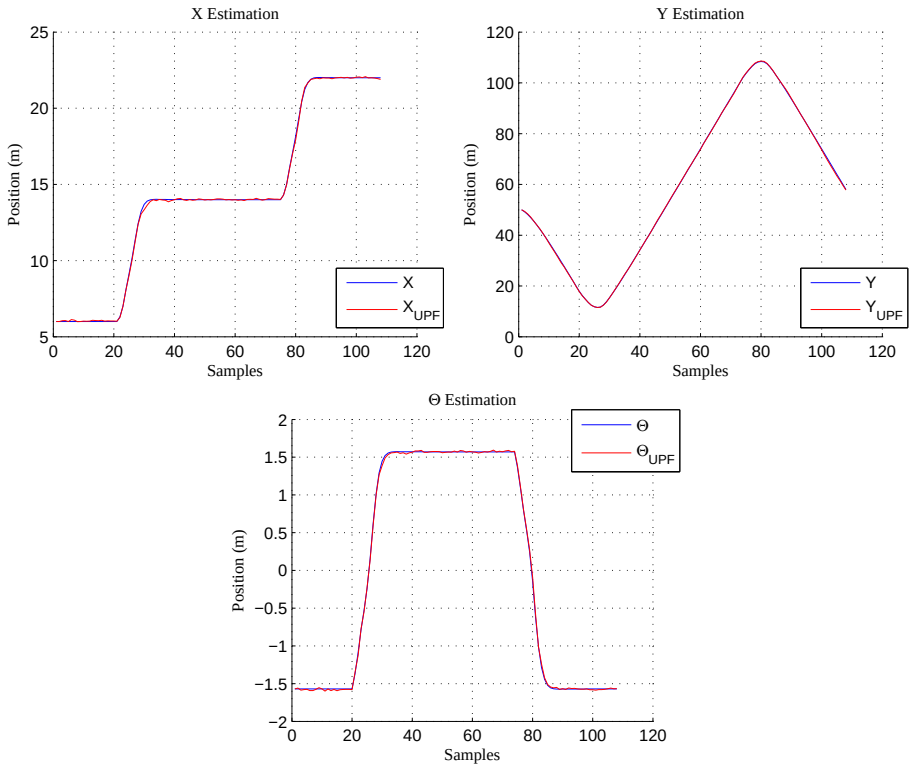

Figure 4.9: Individual UPF estimates versus the true states.

Root Mean Square Errors

Filter	E_X	E_Y	E_Θ	E_{total}	Time (s)
UKF	0.0180	0.0802	0.0056	0.1038	5.0882
PF	0.0123	0.0502	0.0035	0.0660	46.8917
UPF	0.0130	0.0381	0.00058	0.0505	45.3309

Table 4.1: RMSE values for all filters applied with normal map.

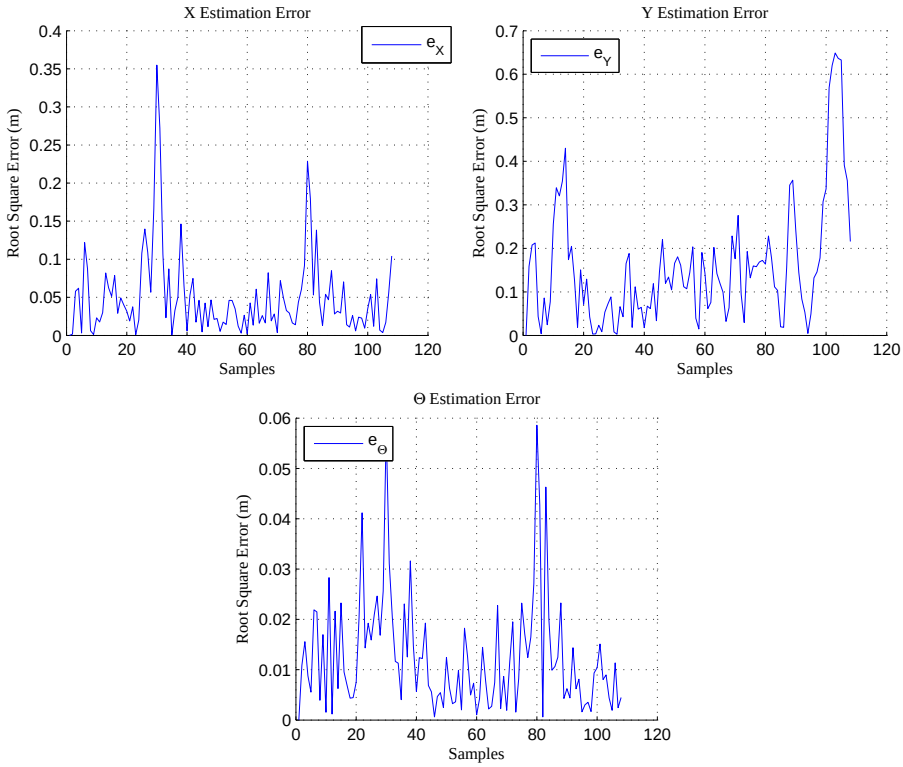

Figure 4.10: UPF Root Square errors for each state.

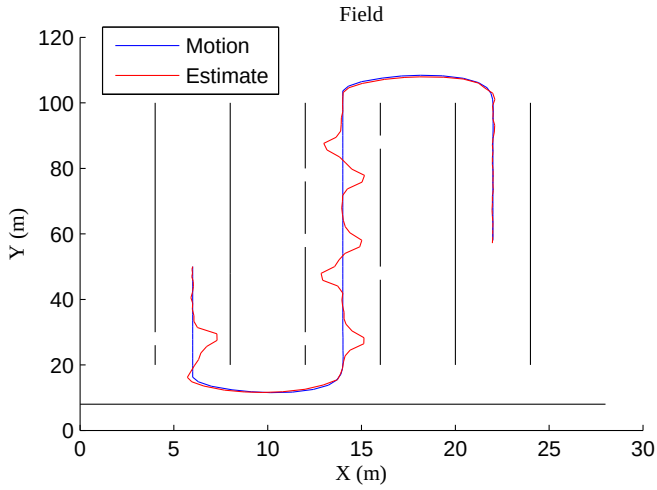

Figure 4.11: Phase plot of UKF estimate using an imperfect map.

4.2 Application with Imperfect Map

In the previous section, it was shown that all three filters succeed in tracking the vehicle's position with minor deviations in estimation quality. A baseline assumption in this assessment was that, even though there was significant measurement uncertainty, the map correctly represents the means of the tree rows in the environment. In practice, it may well be the case that sections of the tree rows are missing either because a diseased tree has been cut off or some trees are being replanted. In such a case, a scenario where the tractor's map has not been updated, either out of neglect or due to the inconvenience of documenting the changes, becomes very realistic. It is therefore imperative to assess the estimation quality of these filters in the presence of such a mismatch.

The UKF algorithm was applied first in a scenario were a number of 4m holes were placed on the vertical tree rows. As can be seen in Figure 4.11, the holes introduce systematic errors in the state estimation. In fact, if the holes were any larger, the estimate would diverge completely. Since the UKF cannot distinguish between correct and incorrect map information, it will move the state estimate such that the majority of the ray scan estimates are in consensus with the measured ones.

A degradation of estimation quality is also experienced with the SIR filter as shown in Figure 4.12. The error occurs at the importance weighting step where,

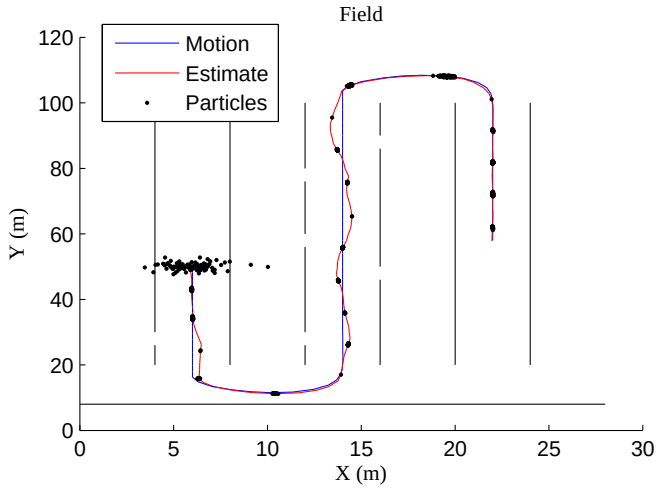

Figure 4.12: Phase plot of SIR filter estimate using an imperfect map.

from the available samples, those nearest to the opposite wall of the hole are weighted highly since these have the greatest consensus with the measurement according to the imperfect map. If the holes or the process noise were larger, then the samples would be allowed to drift even further away and inevitably the estimate would diverge. These observations show that additional steps need to be taken towards outlier rejection.

4.2.1 Unscented Kalman Filter with Outlier Rejection and Feature Extraction

A reasonable approach to using the UKF with an incorrect map, is to reject those measurements that occur from incorrectly mapped regions. Then, while the filter will not be able to use them towards improving its estimate, it will not integrate them incorrectly either. Such an outlier rejection method, that is popular in the robotics community [27], is validation gating using the Mahalanobis distance.

The Mahalanobis distance is a distance measure between two arbitrary points in a vector space and can be interpreted as the multi-dimensional equivalent of measuring how many standard deviations apart these two points are. The two points of interest are the measurement mean $\hat{y}_i$ of the ith laser ray, and the true

measurement y_i. Then the distance is defined as:

$$M = (y_i - \hat{y}_i)^T P_y^i (y_i - \hat{y}_i) \tag{4.18}$$

with P_y^i corresponding to the 2 by 2 covariance matrix block corresponding to the ith laser ray and can be extracted from the measurement covariance matrix P_y evaluated in 4.7. If the measurement is an outlier, then this distance will be significantly larger than for the other measurements, thus a threshold (gate) γ can be assigned so as to reject all measurements above it.

The threshold γ is a tuning parameter and defines how many standard deviations away a measurement can be before it is rejected. It can be chosen manually, or taken from the inverse χ^2 cumulative distribution with degrees of freedom equal to the dimension of the measurement and at the level α which determines the percentage of accepted measurements [1]. A common choice is $\alpha = 0.95$ and since $k = 2$, the resulting threshold is found to be $\gamma = 6$. The measurement which surpasses this threshold can be rejected by setting the corresponding elements of the Kalman gain to 0 before executing the measurement update step.

In addition to outlier rejection, the idea of extracting line features from the measurements was explored, since the tree rows are already modeled as straight lines. The motivation was twofold; reduce the influence of laser rays that do not match the available map and reduce the dimension of the measurement vector so as to improve the accuracy/computation ratio of the UKF.

The chosen method follows closely the one described in [2, 10]. First, each laser ray measurement y_i is converted from polar to cartesian coordinates and then transformed from the robot's local reference frame to the global one, using the latest state estimate $\hat{x}_k$ in conjunction with the homogeneous transform (2.5). Since the line parameters A, B and C (2.28) are available for every tree row on the map, the Euclidean distance of each point to each line can be evaluated. The point is then assigned an index, corresponding to the closest tree row, provided this distance is below a manually assigned threshold. If this is not the case, the point is rejected.

Thereafter, the points assigned to each index i are used to identify a straight line by means of a least squares fit such that the error e_i (4.19) is minimized:

$$\hat{A}_i x + \hat{B}_i y + \hat{C}_i = e_i \tag{4.19}$$

Having extracted the line parameters from the measurements, the aim is to reduce them to polar coordinates for a more compact representation. This is accomplished by computing the distance from the vehicle to each identified line as well as the angle of the line relative to its orientation:

$$f_k^i = \begin{bmatrix} d_k^i \\ \psi_k^i \end{bmatrix} = \begin{bmatrix} \frac{\hat{A}_i \hat{x}_k + \hat{B}_i \hat{y}_k + \hat{C}_i}{\sqrt{\hat{A}_i^2 + \hat{B}_i^2}} \\ arctan\big(-\frac{\hat{A}_i}{\hat{B}_i}\big) - \hat{\theta}_k \end{bmatrix} \tag{4.20}$$

leading to the features f_i, which represent each identified line. These serve as the measurements for filtering. They can be stacked together to form the feature vector:

$$f_k = \begin{bmatrix} f_k^1 & f_k^2 & \dots & f_k^m \end{bmatrix}^T \tag{4.21}$$

The dimension m is practically restricted to $0 \leq m \leq 3$ since the vehicle can only observe at most three tree rows simultaneously. Thus, this feature extraction method will result in a significantly smaller measurement vector but will also introduce further error because it is constructed using the state estimate x_k^-. Furthermore, it became apparent through simulation trials that, scaling down the R_2 matrix during the measurement update step by a factor of 10^2, produced better state estimates.

The UKF was then implemented with the above mentioned outlier rejection method. The first implementation used the original measurement vector while the second made use of feature extraction. The phase plots of both filters are superimposed in Figure 4.13. It is observed that the systematic errors are eliminated due to the validation gating technique. The standard UKF overlaps that of the true state, while the estimate produced by the UKF with line features, deviates noticeably during the second turn.

The individual state estimates for the two filters are given in Figure 4.14. Here it is shown that most of the error during the turn comes from the estimation of the x_k state, while the other states are estimated at a similar accuracy for both filters. This is further supported from the root square error plots in Figure 4.15. It becomes clear that there is a significantly larger error spike during the 75-85 time window for the feature-based UKF. For the other states, the estimation quality is comparable, with the original UKF performing slightly better.

Finally, the estimated variances of each state are reported in Figure 4.16. Several spikes occur at the variance of x_k for both filters which can be attributed

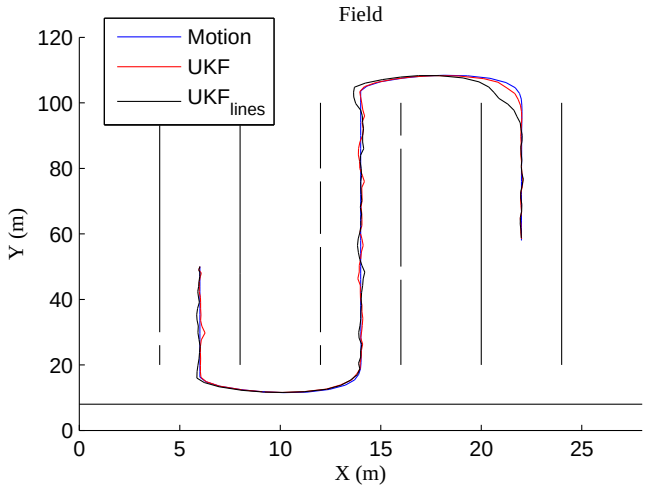

Figure 4.13: Phase plot of the estimates generated by the standard and feature-based UKF.

to equation (3.38). Because the outlier rejection is carried out by setting the relevant elements of the Kalman gain to 0, the covariance estimate P_k^+ is reduced to a lesser extent during the steps where some measurements have been rejected. Otherwise, the large covariance increase, occurring at the second turn, are due to the same reasons as those in the normal map case.

The RMSE values for the standard and feature-based UKF are tabulated in the first and second rows of Table 4.2 respectively. For the standard UKF, it is observed that the RMSEs of each state are higher than its normal map counterpart, which can be attributed to the less information available due to the rejection. The feature-based UKF has twice the total RMSE error but is executed at less than half the time, leading to a superior performance/computation ratio.

4.2.2 Particle Filter with modified Likelihood

For the SIR filter, the validation gating method was implemented as follows. First, as soon as the samples were generated from the proposal density, the empirical measurement mean $\hat{y}_k$ and covariance matrix P_y are computed using the observation hypotheses of every particle. Then, during the importance weighting step, the Mahalanobis distance between each ray measurement and

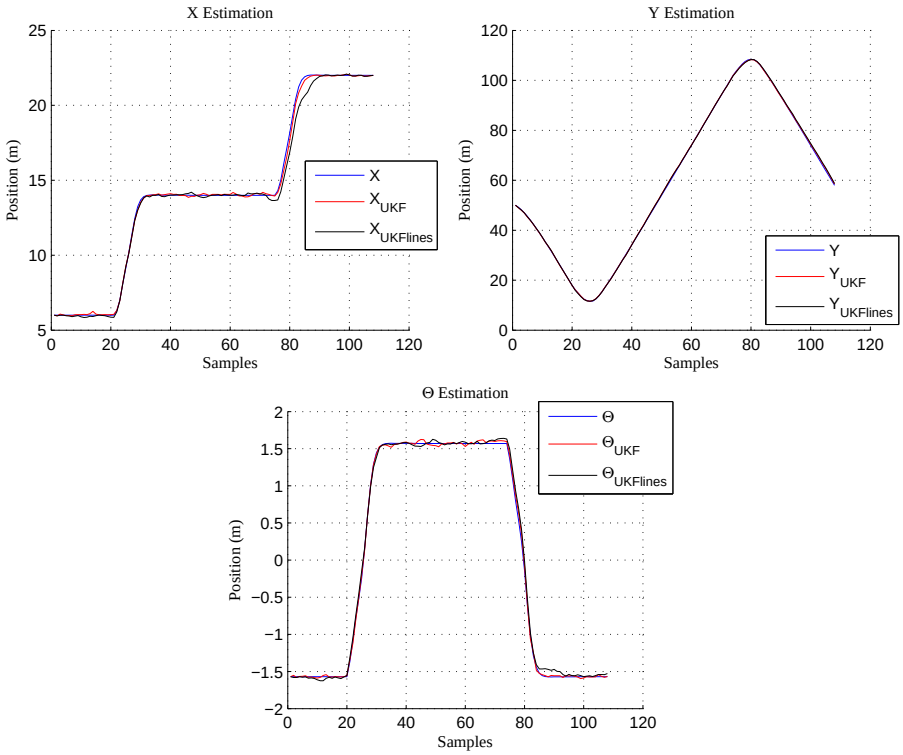

Figure 4.14: Individual state estimates of standard UKF and feature-based UKF.

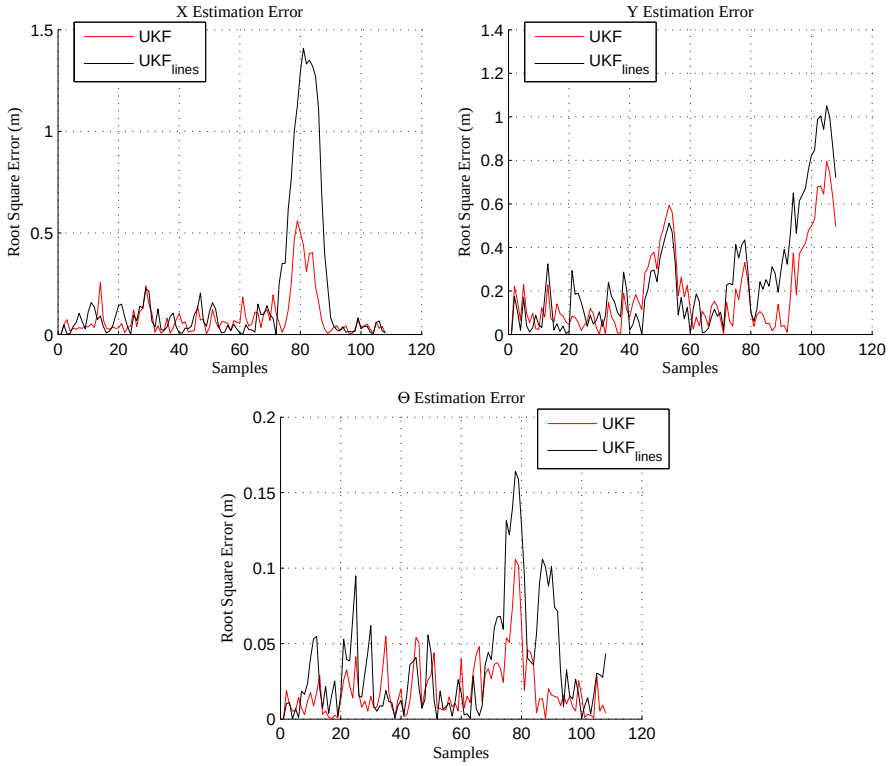

Figure 4.15: Root Square errors of each state estimate generated by the standard and the feature-based UKF.

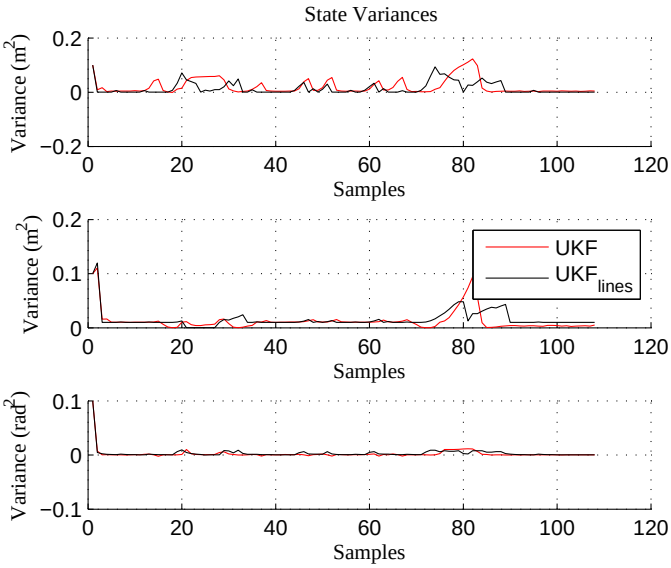

Figure 4.16: Evolution of state variances maintained by the standard and feature-based UKF.

its mean is evaluated. If it is below the threshold, the likelihood $p(y_k^j|x_k^i)$ can be computed as done previously. Otherwise, one has the option of rejecting the particle itself by setting this likelihood to 0 (which due to (4.14) will result in $p(y_k|x_k^i) = 0$). Alternatively, the measurement alone can be rejected by setting $p(y_k^j|x_k^i)$ to 1.

With either of these choices however, the SIR filter did not perform well. When the particles themselves were rejected, the filter suffered from degeneracy, since the majority of particles had negligible weights. When the gated measurements were rejected instead, then the particles were only weighted based on those measurements that lied closer to the mean, which distorted their likelihood and allowed incorrect particles to multiply.

Clearly a different approach is necessary. What is actually desired is to reduce the influence each measurement ray has on the overall likelihood $p(y_k|x_k^i)$ so that a few mismatched measurements will not significantly affect the distribution. This is satisfied intuitively by setting the observation likelihood to to equal the

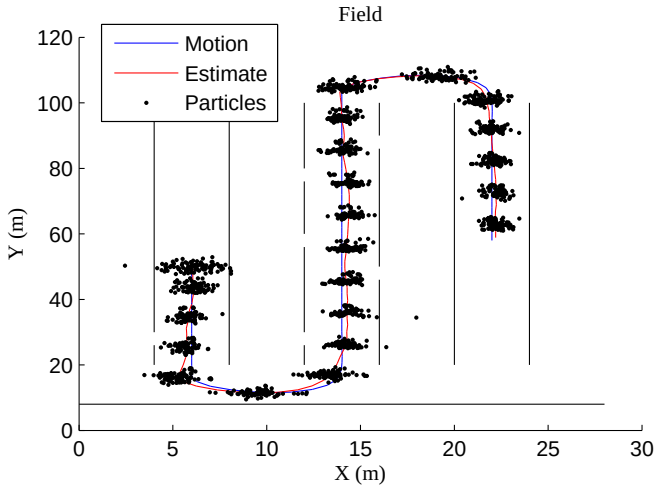

Figure 4.17: Phase plot of the estimate of the robot's position, generated by the modified SIR filter.

average of the ray likelihoods:

$$p(y_k|x_k^i) = \frac{1}{M}\sum_{j=1}^{M} p(y_k^j|x_k^i) \tag{4.22}$$

with this likelihood model, the particles are penalized evenly for incorrect ray hypotheses, regardless of whether the error is due to the map or the particle's position itself. Since the map error does not affect the majority of the rays, it is expected that correctly positioned particles will be weighted more heavily. A simulation is thus carried out with this filter and the phase plot of its estimate is shown in Figure 4.17.

It can be seen that the particle clouds converge in distribution but this distribution has a larger covariance, since the new likelihood model allows more regions of the state-space to have a high probability. The estimate however, appears to match the true state. The individual state plots given by Figure 4.18, show that while there is a good overlap between the states and the estimates, the estimation quality is lower than that achieved by the UKF.

The root square errors are presented in Figure 4.19. The adjustment of the likelihood model has led to a more fluctuating estimation error due to the larger

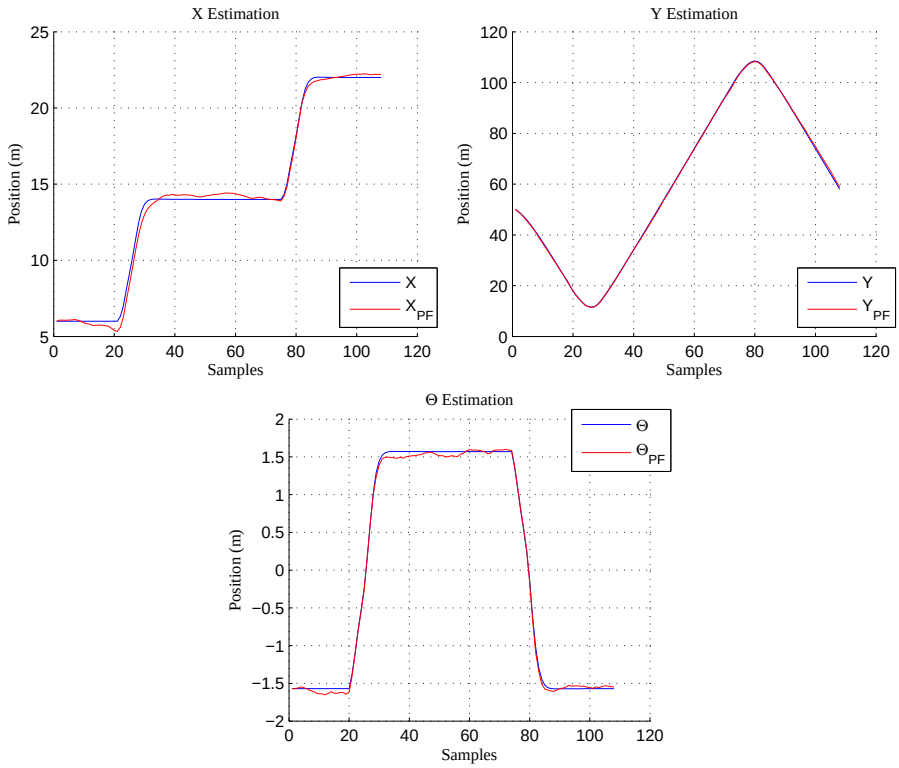

Figure 4.18: Individual estimates generated by the modified SIR filter, versus the true states.

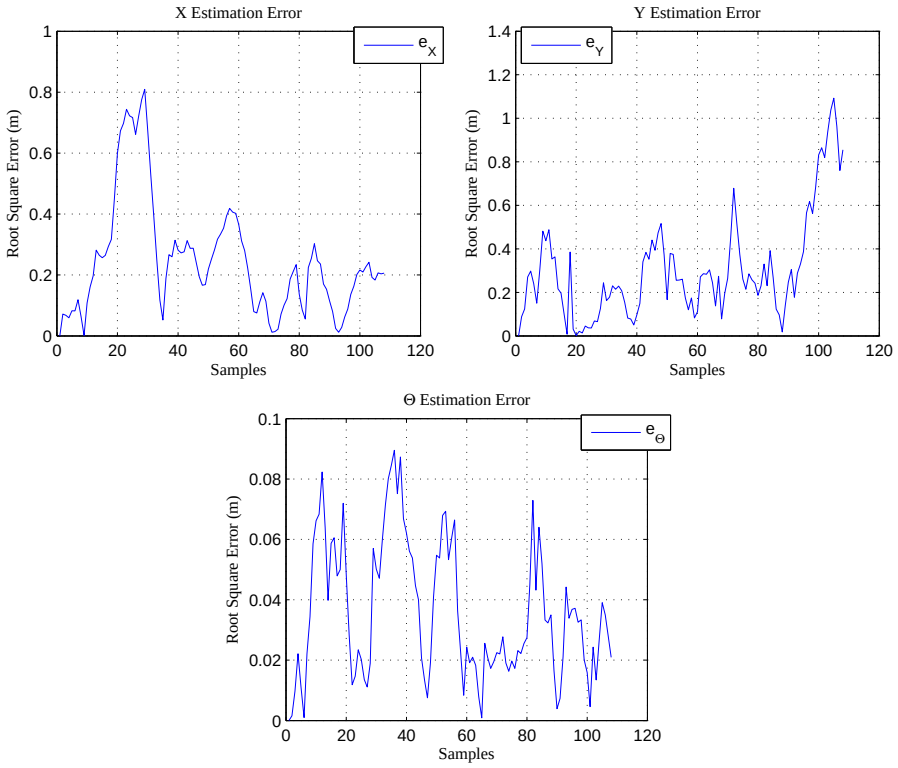

Figure 4.19: Root Square errors for each state estimate of the modified SIR filter.

particle covariance which allows more variation of the particles at each time sample and therefore more variations in the mean.

Lastly, the RMSE errors across 100 realizations are computed for every state and tabulated on the third row of Table 4.2. The RMSE error for every state is the largest so far while the total computation time is about the same as before. Therefore, while the SIR filter can be made to function with an imperfect map, its estimation quality is severely affected by it.

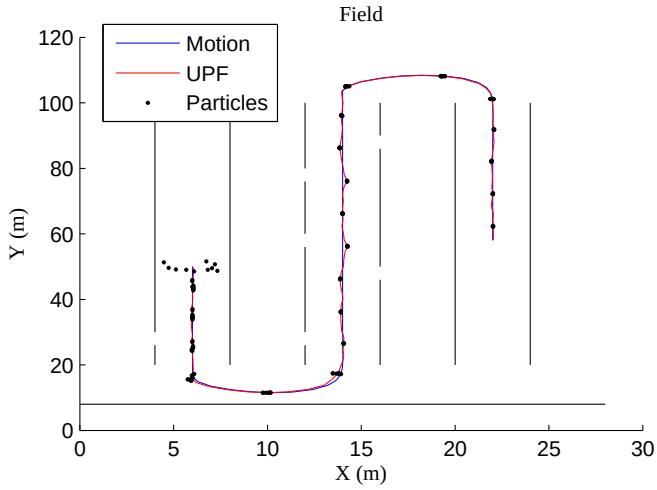

Figure 4.20: Phase plot of the estimate of the robot's position, generated by the standard UPF with outlier rejection.

4.2.3 Unscented Particle Filter with Outlier Rejection and Feature Extraction

The previous subsections showed that both the standard and feature-based UKF outperform the modified SIR filter. The UPF is the last filter to be considered and it was investigated whether it was sufficient to handle the outlier rejection with just the UKF-based proposal densities. The first version of the UKF performs estimation using the normal measurement vector and functions exactly as described in Section 4.1.3. The second makes use of features used during the UKF calls and the original measurements y_k to update the importance weights of each particle.

The phase plots for the two filters are given in Figures 4.20 and 4.21. It is evident that the systematic errors are again eliminated and the estimate matches closely the true state for both approaches.

The individual state plots for both filters are superimposed in Figure 4.22. It is better observed here that the feature-based UPF shows noticeable error at estimating x_k and θ_k at the turning points, similar to the difference between the standard and feature-based UKF.

The root square errors of the state estimates in Figure 4.23 also indicate that

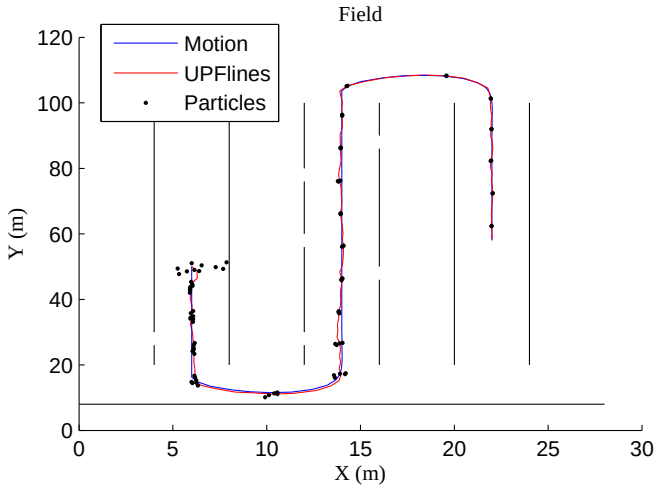

Figure 4.21: Phase plot of the estimate of the robot's position, generated by the feature-based UPF.

Root Mean Square Errors

Filter	E_X	E_Y	E_Θ	E_{total}	Time (s)
UKF	0.0275	0.1079	0.0071	0.1425	5.5006
$UKFlines$	0.1555	0.1377	0.0209	0.3142	1.8100
PF	0.2561	0.2140	0.0194	0.4895	46.4231
UPF	0.0194	0.0435	0.0037	0.0666	45.4114
$UPFlines$	0.0225	0.0535	0.0036	0.0796	10.8549

Table 4.2: RMSE values for all filters applied with an imperfect map.

there is a noticeable initial estimation error for the state y_k until the first turning point where the horizontal tree row is encountered. Comparing to the estimation errors for the UKF filters (Figure 4.15), it is concluded that the estimation errors of the UPF filters are significantly smaller.

The RMSE values across 100 realizations which are tabulated on the last two rows of Table 4.2 show that the UPF filters have the best performance by a considerable margin, approaching the performance of their counterparts that use a normal map (Table 4.1). For the standard UPF, the total computation time is the same as its counterpart, but the feature-based UPF is more than four times faster than its counterpart and only twice as slow as the standard UKF. Thus it showed the largest improvement from the addition of feature extraction.

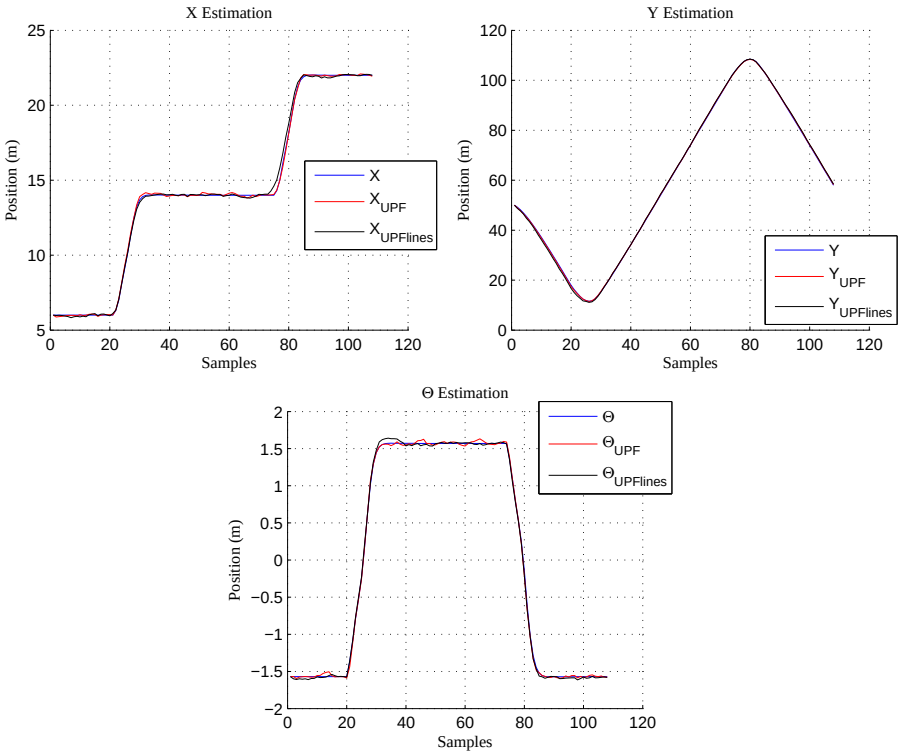

Figure 4.22: Standard and feature-based UPF estimates versus the true states.

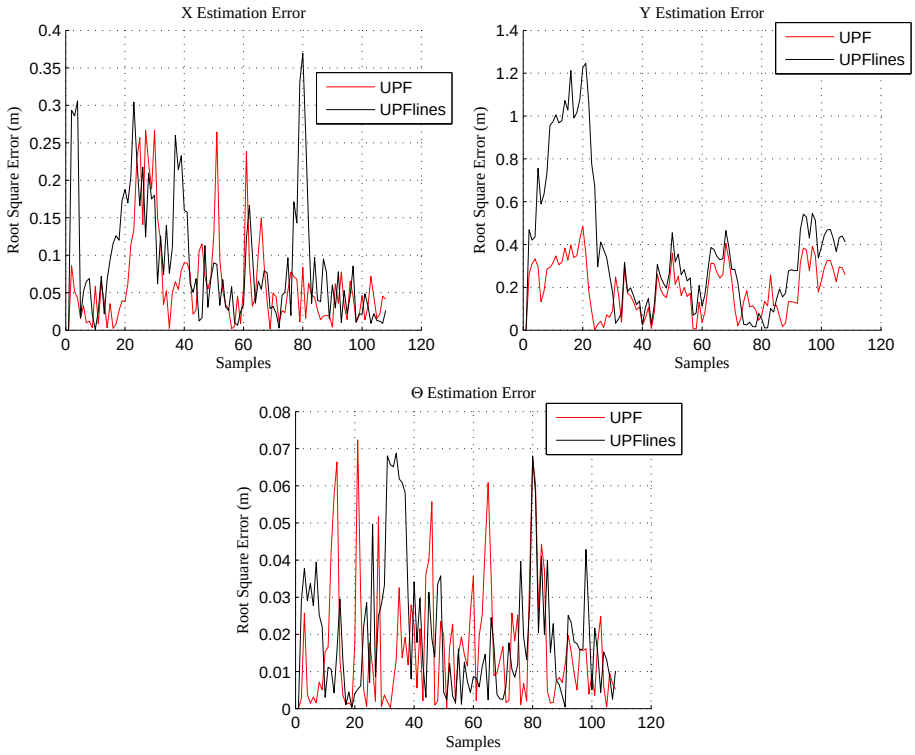

Figure 4.23: Standard and feature-based UPF Root Square errors for each state.

Chapter 5

Conclusions

5.1 Summary and Discussion

In summary, the current thesis investigated the application of state of the art nonlinear state estimation methods for the localization of an autonomous guided vehicle in an outdoor orchard environment, inspired by the works of [2, 10, 22]. The vehicle considered was an automated HAKO tractor, outfitted with rotary encoders for odometry and a laser range scanner for feature detection.

The vehicle was modeled using a simple kinematic model developed by use of its steering geometry. The model was motivated by the success it has had in the robotics community. The uncertainties due to unmodeled dynamics and external disturbances are incorporated as stochastic inputs to this model, whose distributions were modeled arbitrarily as Gaussian which is a standard procedure in the absence of real data. For the simulation of the vehicle to be possible, a Pure Pursuit controller was assigned so that the tractor could follow a trajectory specified by via points. These points along with the tuning parameters of the control were chosen such that the vehicle's path is feasible at a forward velocity not exceeding 2 m/s.

Thereafter, the tree rows present in the orchard were modeled as straight lines in the field, similar to those shown in [10]. A laser scanner function was constructed

that emulated the behavior of the laser range scanner the tractor is equipped with. Lastly, a normally distributed noise vector was added to the measurements so that their spread approximates that of the real laser scans shown in the work of [2].

Having described the system and the environment, the filtering problem was formally defined and the Bayesian estimator was presented as its optimal solution which is however infeasible in the general case of nonlinear systems. The Extended Kalman Filter was then presented as it is the standard of nonlinear estimation and to illustrate its suboptimal nature, since it uses a first-order approximation of the nonlinear functions. The Unscented Kalman Filter, shown to approximate the state distribution instead of the nonlinear functions, was subsequently presented, highlighting both its similarity with the EKF and its improved estimation accuracy.

Particle filtering was presented in depth, discussing its fundamental aspects; importance sampling, resampling, degeneracy and choice of proposal density. It was shown to require weak assumptions and to be able to approximate any probability density, provided a sufficient number of samples were used. Finally, the persisting problem of sample depletion was described and the Unscented Particle Filter was presented as a proposal density choice which helps alleviate the problem and raise the filter's performance/computation ratio.

The implementation of these filters and an assessment of their performance was then carried out, focusing mainly on plots of the state estimates versus the true states and the evaluation of Root Mean Square Errors. The results showed that the UPF had the best performance among the filters but its performance/computation ratio was lower than the UKF.

The evaluation of the same filters in the special case of having an incorrect map showed that their estimation quality deteriorates and that they need modifications so as to be able to reject outliers. Validation gating based on the Mahalanobis distance and line feature extraction were the two methods investigated for outlier rejection and for computational burden reduction. These methods were applied on the UKF and were shown to eliminate the systematic errors caused by the incorrect map. The feature-based UKF was shown to be three times faster than its standard counterpart but had twice its total RMSE.

In the case of the particle filter, both methods led to divergence of the state estimate and a modification of the observation likelihood was proposed to reduce the large impact faulty measurements had on the importance weighting step. The assorted figures showed that while it could adequately estimate the vehicle's trajectory, it exhibited the worst performance of all filters. Lastly, the UPF was implemented as the normal particle filter which made calls to the modified UKF

functions. It was shown that this was sufficient to eliminate the systematic errors and for both variants, the estimation quality was competitive with the filters when applied to the normal map. The feature-based UPF in particular was found to be twice as slow as the standard UKF yet having approximately half its RMSE.

The conclusion supported by these results is that for the most computationally demanding, real-time applications, the UKF has the advantage as it can carry out solid state estimation very quickly. In applications where resources are more plentiful and the maximum performance is expected, the UPF appears to be most promising.

5.2 Future Problems

This section briefly discusses the next steps that could be undertaken to extend this work. The most important step would be to validate the simulation results by implementing the filters on a real system or, at the very least, using real measurements. It is expected that this step will highlight previously unforeseen problems which will lead to further modifications of the current methods. Thereafter, the most promising filter could be selected (e.g. the UPF) and extended for global localization using some adaptive sampling technique to enable an efficient transition between global localization and tracking. Another course of action would be to investigate the extend to which the chosen filter can solve the mapping and obstacle detection problems.

Bibliography

[1] Y. Bar-Shalom and E. Fortmann, T. *Tracking and Data Association*, volume 179 of *Mathematics in Science and Engineering*. Elsevier Science, 1988.

[2] M. Blanke, M. R. Blas, S. Hansen, J. C. Andersen, and F. Caponetti. *Autonomous Robot Supervision using Fault Diagnosis and Semantic Mapping in an Orchard*, chapter 1. iConcept Press Ltd, 2012.

[3] J. Borenstein, R. Everett, H., and L. Feng. *Where am I? Sensors and Methods for Mobile Robot Positioning*. University of Michigan, 1996.

[4] P. Corke. *Robotics, Vision and Control*. Springer, 2013.

[5] A. Doucet. *Monte Carlo Methods for Bayesian Estimation of Hidden Markov Models: Applications to Radiation Signals*. PhD thesis, University Paris-Sud, Orsay, France, 1997.

[6] A. Doucet, N. Freitas, and N. Gordon. *Sequential Monte Carlo Methods in Practice*. Springer, 2001.

[7] A. Doucet, S. Godsill, and C. Andrieu. On sequential monte carlo sampling methods for bayesian filtering. *Statistics and Computing*, 10(3):197 – 208, 2000.

[8] J. Gordon, N., D. J. Salmond, and F. M. Smith, A. Novel approach to nonlinear/non-gaussian bayesian state estimation. *IEEE Proceedings-f Radar and Signal Processing*, 140(2):107 – 113, 1993.

[9] J. M. Hammersley and W. Morton, K. Monte carlo techniques to estimate the conditional expectation in multi-stage non-linear filtering. *International Journal of Control*, 9(5):547 – 559, 1969.

[10] S. Hansen, E. Bayramoglu, C. Andersen, J, O. Ravn, N. Andersen, and N. K. Poulsen. Orchard navigation using derivative free kalman filtering. In *American Control Conference on O'Farrell Street*, San Francisco, CA, USA, June 29 - July 01, 2011.

[11] D. Hol, J., B. Schon, T., and F. Gustafsson. On resampling algorithms for particle filters. In *NSSPW. Nonlinear Statistical Signal Processing Workshop*, 2006.

[12] K. Ito and K. Xiong. Gaussian filters for nonlinear filtering problems. *IEEE Transactions on Automatic Control*, 45, 5:910–927, May 2000.

[13] J. Julier, S., K. Uhlmann, J., and F. Durrant-Whyte, H. A new approach for filtering nonlinear systems. In *The Proceedings of the American Control Conference*, pages 1628 – 1632, Seattle, Washington, 1995.

[14] S. J. Julier. The scaled unscented transformation. In *Proceedings of the American Control Conference*, pages 4555 – 4559, 2002.

[15] S. J. Julier. The spherical simplex unscented transformation. In *Proceedings of the American Control Conference*, pages 2430 – 2434, 2003.

[16] S. J. Julier and J. K. Uhlmann. A new extension of the kalman filter to nonlinear systems. In *The Proceedings of AeroSense: The 11th International Symposium on Aerospace/Defense Sensing, Simulation and Controls*, SPIE, Multi Sensor Fusion, Tracking and Resource Management II, Orlando FL, USA, 1997.

[17] S. J. Julier and J. K. Uhlmann. Unscented filtering and nonlinear estimation. In *Proceedings of the IEEE*, volume 92, no 3, pages 401 – 422, Mar 2004.

[18] R. E. Kalman. A new approach to linear filtering and prediction problems. *Transactions of the ASME–Journal of Basic Engineering*, 82(Series D):35–45, 1960.

[19] R. Karlsson. *Particle Filtering for Positioning and Tracking Applications.* PhD thesis, Department of Electrical Engineering, Linkoping University, Linkoping, Sweden, 2005.

[20] L. Kleeman. Advanced sonar and odometry error modeling for simultaneos localisation and map building. In *Proceedings of the 2003 IEEE International Conference on Intelligent Robots and Systems*, Las Vegas, 2003.

[21] L. M. Path tracking for a miniature robot. Master's thesis, Umea University, Department of Computer Science, Sweden, 2003.

[22] L. V. Mogensen, S. Hansen, C. Andersen, J, O. Ravn, N. A. Andersen, M. Blanke, and N. K. Poulsen. Kalmtool used for laser scanner aided navigation in orchard. In *15th IFAC Symposium on System Identification*, Saint-Malo, France, July 6-8 2009.

[23] Y. Morales and T. Tsubouchi. Gps moving performance on open sky and forested paths. In *Proceedings of the 2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, San Diego, CA, USA, Oct 29 - Nov 2, 2007.

[24] M. Norgaard, N. K. Poulsen, and O. Ravn. New developments in state estimation for nonlinear systems. *Automatica*, 36, 11:1627 – 1638, November 2000.

[25] F. Orderud. *Comparison of Kalman Filter Estimation Approaches for State Space Models with Nonlinear Measurements.* Sem Saelands vei 7-9, NO-7491 Trondheim.

[26] B. Ristic, S. Arulampalam, and N. Gordon. *Beyond the Kalman Filter: Particle Filters for Tracking Applications.* Artech House, 2004.

[27] R. Siegwart, R. Nourbakhsh, I., and D. Scaramuzza. *Introduction to Autonomous Mobile Robots.* MIT Press, 2011.

[28] D. Simon. *Optimal State Estimation: Kalman, H_∞ and Nonlinear Approaches.* Wiley, 2006.

[29] W. Sprong, M., S. Hutchinson, and M. Vidyasagar. *Robot Modeling and Control.* Wiley, 2006.

[30] G. A. Terejanu. *Unscented Kalman Filter Tutorial.* Department of Computer Science and Engineering, University at Buffalo, Buffalo, NY 14260.

[31] S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics.* MIT Press, 2006.

[32] R. van der Merwe. *Sigma-Point Kalman Filters for Probabilistic Inference in Dynamic State-Space Models.* PhD thesis, OGI School of Science & Engineering, Oregon Health & Science University, 2004.

[33] R. van der Merwe, A. Doucet, N. Freitas, and E. Wan. The unscented particle filter. Technical Report CUED/F-INFENG/TR 380, Cambridge University Engineering Department, June 2009.

[34] E. A. Wan and R. van der Merwe. The unscented kalman filter for nonlinear estimation. In *Proceedings of Symposium on Adaptive Systems for Signal Processing Communications and Control*, pages 153 – 158, 2000.