Chapter 1

Introduction

1.1 Context

Global Navigation Satellite Systems (GNSS), based on trilateration, are used for positioning and navigation of the vehicle. They are accessible sources all over the world used for finding the position of the user by estimating the distance between the receiver and the satellites. At least 4 satellites are needed for a PVT solution. Once the position of each satellite is known, we could get an estimation of the user position. However, there still exist errors as the signal propagates such as ionospheric and tropospheric errors, clock errors, multipath, interferences, etc. In harsh environments (the presence of large buildings for instance), these systems are not efficient. These systems can be ineffective especially in urban environments because the signals sent by satellite are reflected, diffracted, and refracted leading to an inaccurate position estimation, adding a large bias to the actual position of the user (which is not desirable). Also, the signals from the satellites are lost, for example, when driving through a tunnel, and then the receiver cannot estimate its position and therefore, track the vehicle trajectory. For simple calculations, we can use least squares estimation techniques which are not very accurate. In reality, when we implement a PVT solution for the estimation of vehicle/user trajectory, our problem is non-linear. Thus, it becomes important to consider these characteristics to correctly model the system accurately.

1.2 Objective

The idea of this project is to implement different estimators. The project uses 3 estimation techniques namely, Least Square Estimation (LSE), Extended Kalman Filter (EKF), and Particle Filter (PF), along with comparison of their performance. A detailed study of PF is performed by studying its various types. Next, the concept of Map-Matching is implemented, where the estimated position of the user obtained after the implementation of the estimator (PF) is merged with the map information. These maps can be used to provide a priori information on the receiver trajectory and act as assistance data. The project implements map matching with PF as it can handle non-standard priors and likelihood. Two methods of map matching are proposed: the first method uses the distance between the map and particles to update the observation equation and the second method draws a proposal distribution from intersecting points of a circle with the center at the estimated position, at the previous epoch. The project aims to develop a positioning algorithm that merges the observations of a GNSS receiver and a map based on a particle filter in order to improve the positioning of a vehicle on the road, especially in an urban environment. It is developed assuming a 2D space (x, y position of the satellite). Later, this would be used to develop MAGNITUDE, the TELECOM/SIGNAV library for GNSS data processing.

Chapter 2

State of Art about GNSS

Global Navigation Satellite System (GNSS) refers to a constellation of satellites that transmit positioning and timing data from space to GNSS receivers via signals. The concept of GNSS positioning is the idea of computation of the user position, velocity, and time using this available data. It uses the measurements of time-of-arrival from multiple satellite signals to find estimates of distances to the satellites. For this, first, the satellite position is computed, before the clock correction. To that extent, several parameters (such as semimajor axis, corrected mean motion, delta t) are extracted from the data [1]. Then, rotation arguments and corrected orbital parameters are computed in order to obtain the satellite position in the Earth-Centred Earth Fixed (ECEF) frame at the time of transmission. To have satellite ECEF coordinates in reception time, the Sagnac effect is taken into account. Finally, the satellite positions in ECEF are converted in the Latitude-Longitude-Altitude coordinate system (WGS-84). This estimated satellite position is an input to the estimator, as well as the estimated satellite clock bias. The output is a rough estimate of the user position. This rough estimate allows computing tropospheric and ionospheric errors. Finally, we use this estimator to obtain a fine and correct estimate of the user position by taking into account all possible errors. In the section for estimation techniques, we find the use of an estimator. We can implement simple estimators like the Least Square Estimator (LSE), the Extended Kalman Filter, or the Particle Filter. As discussed, LSE techniques have very low accuracy. This accuracy degrades even more in an urban environment, due to multipath. To solve this problem and to ensure that we can have accurate positioning, Bayesian filters like Kalman filters(KF), Extended Kalman filters(EKF), Particle filters(PF), etc may be used. This project aims to compare various estimators and therefore, draws a performance comparison. Methods for improving the position estimation after the implementation of estimators (PF) will also be discussed.

In the rest of this work, we will assume a 2D positioning problem where all the various terms, except the receiver clock bias and noise, are perfectly corrected. In this context, the equation of the pseudorange of the i-th satellite is expressed as [1]:

$$\tilde{\rho}^i = \sqrt{(x - x_{sat})^2 + (y - y_{sat})^2} + b_u + n^i, \tag{2.1}$$

where

- (x, y) denotes the position of user in the 2D frame
- (x^i, y^i) denotes the position of the i-th satellite expressed in the same 2D frame
- b_u is the satellite clock bias
- n^i represents the measurement noise.

Chapter 3

Problem Formulation

In this chapter, we will present the formulation of our problem. This includes the different estimation techniques used for positioning. The concept and formulae of some useful estimation techniques will be discussed that were implemented on MATLAB. This chapter also introduces the concept of map-matching and the approaches taken in view of this project.

3.1 Estimation Techniques

In this section, we will present the estimators and their formulations. Classical estimators like least square estimators will be presented along with Bayesian estimators like Kalman filters, Extended Kalman Filters, and Particle filters.

3.1.1 Least Square Estimation

Ordinary Least Squares
The first estimation technique for positioning is the least square estimation (LSE) or simply the ordinary least squares. By definition, the LSE minimizes the squared norm of estimation residuals, i.e., the difference between the real observations and their predictions for a given value of the state vector. In the case of a linear model we thus have:

$$\min_{\boldsymbol{x}} \|\boldsymbol{y} - \boldsymbol{H}\boldsymbol{x}\|_2^2 \tag{3.1}$$

leading

$$\hat{\boldsymbol{x}}_{LS} = (\boldsymbol{H}^T\boldsymbol{H})^{-1}\boldsymbol{H}^T\boldsymbol{y}. \tag{3.2}$$

This method involves building a measurement matrix from the partial derivatives of the geometric distances between the satellite and the user with respect to each coordinate axis. The difference between the previous value of the state vector and its new value is obtained by the formation of its pseudoinverse matrix and factoring it into the measurement equation with all the corrections of the pseudorange. Therefore, their new value needs to be added to the previous one as an iteration process, which is interrupted when the said difference is low enough or when the number of iterations for the same epoch is high enough. Since the geometric range and the user clock bias depends on the current estimation of the user position, the newly obtained values are reported back to the initial phase of the process if it has not been stopped by the aforementioned conditions, and the process is then repeated.

Mathematically, let us consider the state vector to be estimated as $\boldsymbol{x}$, which gathers the position and the clock bias. Then the measurement vector is modeled as $\boldsymbol{y} = \boldsymbol{h}(\boldsymbol{x})$, where $\boldsymbol{y}$ represents

the GNSS pseudoranges or measurements. The noise vector is assumed Gaussian with zero mean and with covariance matrix R, i.e., $\boldsymbol{n} \sim \mathcal{N}(0, \boldsymbol{R})$. The resulting measurement equation is: [9]

$$\boldsymbol{y} = \boldsymbol{h}(\boldsymbol{x}) + \boldsymbol{n} \tag{3.3}$$

But for the GNSS problem, the above measurement equation is not linear, therefore we can not use ordinary least squares and therefore, need to resort to the non-linear least squares (explained below).

Non-linear Least Squares

When the measurement model is not linear, ordinary least squares can not be used instead we use non-linear least squares, where the positioning solution is based on the linearization of the measurement model around a supposed state vector, say, $\hat{\boldsymbol{x}}_0$:

$$\boldsymbol{h}(\boldsymbol{x}) = \boldsymbol{h}(\hat{\boldsymbol{x}}_0) + \frac{\partial \boldsymbol{h}}{\partial \boldsymbol{x}} \boldsymbol{h}(\hat{\boldsymbol{x}}_0).(\boldsymbol{x} - \boldsymbol{x}_0), \tag{3.4}$$

where $\frac{\partial \boldsymbol{h}}{\partial \boldsymbol{x}} \boldsymbol{h}(\hat{\boldsymbol{x}}_0) = \boldsymbol{H}$, called as the Jacobian matrix of $\boldsymbol{h}(\boldsymbol{x})$. Thus the measurement model can be re-written as:

$$\boldsymbol{y} = \boldsymbol{h}(\hat{\boldsymbol{x}}_0) + \boldsymbol{H}.(\boldsymbol{x} - \boldsymbol{x}_0) + \boldsymbol{n} \tag{3.5}$$

OR

$$\Delta \boldsymbol{y} = \boldsymbol{H} \Delta \boldsymbol{x} + \boldsymbol{n} \tag{3.6}$$

Hence, this reformulated model is linear between:

(a) The transformed measurement, $\Delta \boldsymbol{y} = \boldsymbol{y} - \boldsymbol{h}\hat{\boldsymbol{x}}_0$, and

(b) The transformed state, $\Delta \boldsymbol{x} = \boldsymbol{x} - \hat{\boldsymbol{x}}_0$

Therefore, the Least Square Estimator formula can be then re-used as:

$$\Delta \hat{\boldsymbol{x}}_{LS} = (\boldsymbol{H}^T \boldsymbol{H})^{-1} \boldsymbol{H}^T \Delta \boldsymbol{y} \tag{3.7}$$

This implies that the final estimate is:

$$\hat{\boldsymbol{x}} = \hat{\boldsymbol{x}}_0 + \Delta \hat{\boldsymbol{x}}_{LS} \tag{3.8}$$

Usually, this method is applied iteratively in order to refine the linearization point, $\hat{\boldsymbol{x}}_0$, and this iterative least square method has been implemented in the code script as well.

3.1.2 Kalman Filter (KF)

In Kalman Filter, we are in a Bayesian framework, and hence we are looking for the posterior distribution, $p(\boldsymbol{x}_k|\boldsymbol{y}_{1:k})$ and the state vector. Each epoch is represented by $'k'$. The following is the state space model with first order Markov assumption for the state model, which is true for KF and also for EKF and PF (described in later sub-sections):

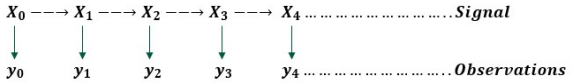


Figure 3.1: System model (adapted from [5])

In this study, the velocity and clock drift is added to the state that makes our state vector as: $\boldsymbol{x} = \begin{bmatrix} \boldsymbol{x}_k & \dot{\boldsymbol{x}}_k & \boldsymbol{y}_k & \dot{\boldsymbol{y}}_k & \boldsymbol{b}_k & \dot{\boldsymbol{b}}_k \end{bmatrix}$ where,

- $\boldsymbol{x}_k$ represents the x-axis position at time epoch k
- $\dot{\boldsymbol{x}}_k$ represents the velocity for the x-axis position at time epoch k
- $\boldsymbol{y}_k$ represents the y-axis position at time epoch k
- $\dot{\boldsymbol{y}}_k$ represents the velocity for the y-axis position at time epoch k
- $\boldsymbol{b}_k$ represents the clock drift at time instant k
- $\dot{\boldsymbol{b}}_k$ represents the clock drift velocity at time instant k

We know that the Kalman Filter is a recursive estimator using extensively the Gaussian assumption. Here, in comparison to LSE, we have a state transition model along with the measurement model. In the case of linear models, these are expressed as: [9]

(a) State Transition Model:

$$\boldsymbol{x}_{k+1} = \boldsymbol{F}_k \boldsymbol{x}_k + \boldsymbol{G}_k \boldsymbol{u}_k + \boldsymbol{v}_k \tag{3.9}$$

where,

- $\boldsymbol{F}_k$ is the assumed state transition matrix, providing the vehicle dynamics equations
- $\boldsymbol{u}_k$ is the command vector (known) that signifies the pilot/driver's actions
- $\boldsymbol{G}_k$ is the command matrix (known) providing a link between the pilot/driver's actions and the state
- $\boldsymbol{v}_k$ is the state noise vector or process noise, also called the modelization error

(b) Measurement Model:

$$\boldsymbol{y}_k = \boldsymbol{H}_k \boldsymbol{x}_k + \boldsymbol{n}_k \tag{3.10}$$

where,

- $\boldsymbol{x}_k$ is the state vector defined previously
- $\boldsymbol{y}_k$ is the measurement vector gathering all the pseudoranges at time k
- $\boldsymbol{n}_k$ is the measurement noise vector at time instant k that affects the measurement

Regarding the state model, in our study, we will use a first-order random walk model, defined by:

$$\boldsymbol{F}_k = \begin{bmatrix} 1 & \Delta t & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & \Delta t & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & \Delta t & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.11}$$

Assumptions: The KF assumes that in equations 3.9 and 3.10, in addition to the linear aspect of these equations,the corresponding noises v_k and n_k are Gaussian. Therefore, both v_k and w_k are independent and Gaussian with distributions $\mathcal{N}(0, \boldsymbol{Q}_k)$ and $\mathcal{N}(0, \boldsymbol{R}_k)$ respectively. As Kalman Filter is a recursive process, an initial state must be provided that is Gaussian, namely $\boldsymbol{x}_0 \sim \mathcal{N}(\hat{\boldsymbol{x}}_0, \boldsymbol{\Sigma}_0)$. It can be shown that, due to the recursive equations, the system state vector $\boldsymbol{x}_k$ is also random and Gaussian, and is therefore described by two parameters, i.e., it's mean ($\hat{\boldsymbol{x}}_k = \boldsymbol{E}[\boldsymbol{x}_k]$) and its covariance matrix ($\boldsymbol{\Sigma}_k = Cov[\boldsymbol{x}_k]$).

Principle: The principle of the Kalman Filter can be explained in the following two steps:

- **State Prediction:** The first step in KF is the prediction phase where the internal state vector and the state covariance matrix are estimated based on only the state transition model. In other words, a prediction of the state distribution is obtained using the state transition model, and without the use of a new observation. As this distribution is Gaussian, it is fully determined by its mean and covariance matrix, expressed as

  (a) State prediction:

$$\hat{\boldsymbol{x}}_{k|k-1} = \boldsymbol{F}_{k-1}\hat{\boldsymbol{x}}_{k-1|k-1} + \boldsymbol{G}_{k-1}\boldsymbol{u}_{k-1} \tag{3.12}$$

  (b) State Covariance prediction:

$$\boldsymbol{\Sigma}_{k|k-1} = F_{k-1}\hat{\boldsymbol{\Sigma}}_{k-1|k-1}\boldsymbol{F}_{k-1}^T + \boldsymbol{Q}_{k-1} \tag{3.13}$$

  After the prediction step, a prior state is obtained given by: $\hat{\boldsymbol{x}}_{k|k-1} = \boldsymbol{E}[\boldsymbol{x}_k|\boldsymbol{y}_0, \ldots, \boldsymbol{y}_{k-1}] = \boldsymbol{E}[\boldsymbol{x}_k|\boldsymbol{y}_0^{k-1}]$ with covariance matrix, $\boldsymbol{\Sigma}_{k|k-1} = \boldsymbol{Cov}[\boldsymbol{x}_k|\boldsymbol{y}_0^{k-1}]$.

- **Measurement Update:** The second step is the update step of the apriori estimated state vector and the covariance matrix at the current epoch k. Now, the predicted state distribution is used as the prior distribution with the measurement in order to obtain the Bayesian estimator of the state vector. In the linear Gaussian model, it can be shown that this distribution is also Gaussian, whose mean is denoted as $\hat{x}_{k|k}$ and whose covariance matrix as $\Sigma_{k|k}$, which are expressed as

  (a) State Update:

$$\hat{\boldsymbol{x}}_{k|k} = \hat{\boldsymbol{x}}_{k|k-1} + \boldsymbol{K}_k(\boldsymbol{y}_k - \boldsymbol{H}_k\hat{\boldsymbol{x}}_{k|k-1}) \tag{3.14}$$

  $\boldsymbol{K}_k$ denotes the Kalman gain at epoch k, as seen in equation 3.14, can be interpreted as the ratio between the estimated apriori covariance matrix $\boldsymbol{\Sigma}_{k|k-1}$ and the sum of the estimated apriori covariance matrix and the observation covariance matrix $\boldsymbol{R}_k$ and is expressed as

$$\boldsymbol{K}_k = \boldsymbol{\Sigma}_{k|k-1}\boldsymbol{H}_k^T(\boldsymbol{H}_k\boldsymbol{\Sigma}_{k|k-1}\boldsymbol{H}_k^T + \boldsymbol{R}_k)^{-1} \tag{3.15}$$

  (b) State Covariance Update:

$$\boldsymbol{\Sigma}_{k|k} = (\boldsymbol{I} - \boldsymbol{K}_k\boldsymbol{H}_k)\hat{\boldsymbol{\Sigma}}_{k|k-1} \tag{3.16}$$

After taking into account the k^{th} measurement $\boldsymbol{y}_k$, we obtain the posterior state given by: $\hat{\boldsymbol{x}}_{k|k} = \boldsymbol{E}[\boldsymbol{x}_k|\boldsymbol{y}_0, \ldots, \boldsymbol{y}_k] = \boldsymbol{E}[\boldsymbol{x}_k|\boldsymbol{y}_0^k]$ with covariance matrix, $\boldsymbol{\Sigma}_{k|k} = \boldsymbol{Cov}[\boldsymbol{x}_k|\boldsymbol{y}_0^k]$.

Kalman filter works with linear and Gaussian systems but our model is non-linear. We, therefore, resort to Extended Kalman filters that can deal with non-linear and Gaussian systems.

3.1.3 Extended Kalman Filter (EKF)

For nonlinear models, the Extended Kalman Filter (EKF) can be used. Kalman filters also assume Gaussian noise. The model equations are linearized around the previous estimate to be able to apply the Kalman Filter algorithm. This implies that now we have: $\boldsymbol{x}_{k+1} = \boldsymbol{f}(\boldsymbol{x}_k) + \boldsymbol{G}_k\boldsymbol{u}_k + \boldsymbol{v}_k$ and $\boldsymbol{y}_k = \boldsymbol{h}(\boldsymbol{x}_k) + \boldsymbol{n}_k$, where $\boldsymbol{f}(\boldsymbol{x})$ and $\boldsymbol{h}(\boldsymbol{x})$ are non-linear functions. Therefore, now, we need to linearize the model. For the state transition model, the linearization point is the last estimated state, $\hat{\boldsymbol{x}}_{k|k}$, given by:

$$\boldsymbol{f}(\boldsymbol{x}_k) \approx \boldsymbol{f}(\hat{\boldsymbol{x}}_{k|k}) + \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}_k}(\hat{\boldsymbol{x}}_{k|k})(\boldsymbol{x}_k - \hat{\boldsymbol{x}}_{k|k}). \tag{3.17}$$

Similarly, for the measurement model, the linearization point is the last predicted state, $\hat{\boldsymbol{x}}_{k|k-1}$, given by

$$\boldsymbol{h}(\boldsymbol{x}_k) \approx \boldsymbol{h}(\hat{\boldsymbol{x}}_{k|k-1}) + \frac{\partial \boldsymbol{h}}{\partial \boldsymbol{x}_k}(\hat{\boldsymbol{x}}_{k|k-1})(\boldsymbol{x}_k - \hat{\boldsymbol{x}}_{k|k-1}). \tag{3.18}$$

Linearization: For linearizing the equations, first order Taylor's series expansion has been used and the system of equations becomes:

$$\boldsymbol{x}_{k+1} = \boldsymbol{f}(x_k) + \boldsymbol{G}_k\boldsymbol{u}_k + \boldsymbol{v}_k = \boldsymbol{f}(\hat{\boldsymbol{x}}_{k|k}) + \boldsymbol{F}_k(\boldsymbol{x}_k - \hat{\boldsymbol{x}}_{k|k}) + \boldsymbol{G}_k\boldsymbol{u}_k + \boldsymbol{v}_k \tag{3.19}$$

$$\boldsymbol{y}_k = \boldsymbol{h}(\boldsymbol{x}_k) + \boldsymbol{n}_k = \boldsymbol{h}(\hat{\boldsymbol{x}}_{k|k-1}) + \boldsymbol{H}_k(\boldsymbol{x}_k - \hat{\boldsymbol{x}}_{k|k-1}) + \boldsymbol{n}_k \tag{3.20}$$

where $\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}_k}(\hat{\boldsymbol{x}}_{k|k}) = \boldsymbol{F}_k$ and $\frac{\partial \boldsymbol{h}}{\partial \boldsymbol{x}_k}(\hat{\boldsymbol{x}}_{k|k-1}) = \boldsymbol{H}_k$. Denoting the known quantities (the models and the Jacobians that are known analytically) as:

$$\tilde{\boldsymbol{u}}_k = \boldsymbol{G}_k\boldsymbol{u}_k + f(\hat{\boldsymbol{x}}_{k|k}) - \boldsymbol{F}_k(\hat{\boldsymbol{x}}_{k|k}) \tag{3.21}$$

$$\tilde{\boldsymbol{y}}_k = \boldsymbol{h}(\hat{\boldsymbol{x}}_{k|k-1}) - \boldsymbol{H}_k\hat{\boldsymbol{x}}_{k|k-1} \tag{3.22}$$

Thus, we have:

$$\boldsymbol{x}_{k+1} = \boldsymbol{F}_k\boldsymbol{x}_k + \tilde{\boldsymbol{u}}_k + \boldsymbol{v}_k \tag{3.23}$$

$$\boldsymbol{y}_k - \tilde{\boldsymbol{y}}_k = \boldsymbol{H}_k\boldsymbol{x}_k + \boldsymbol{n}_k \tag{3.24}$$

Thus, this new system is now linear and the conventional KF algorithm can be used. Hence, the EKF algorithm can be written as:

- **State Prediction:** $\hat{\boldsymbol{x}}_{k|k-1} = \boldsymbol{f}(\hat{\boldsymbol{x}}_{k-1|k-1}) + \tilde{\boldsymbol{u}}_{k-1}$
- **State Covariance Prediction:** $\boldsymbol{\Sigma}_{k|k-1} = \boldsymbol{F}_{k-1}\boldsymbol{\Sigma}_{k-1|k-1}\boldsymbol{F}_{k-1}^T + \boldsymbol{Q}_{k-1}$
- **Kalman Gain Computation:** $\boldsymbol{K}_k = \boldsymbol{\Sigma}_{k|k-1}\boldsymbol{H}_k^T(\boldsymbol{H}_k\boldsymbol{\Sigma}_{k|k-1}\boldsymbol{H}_k^T + \boldsymbol{R}_k)^{-1}$
- **State Update:** $\hat{\boldsymbol{x}}_{k|k} = \hat{\boldsymbol{x}}_{k|k-1} + \boldsymbol{K}_k(\boldsymbol{y}_k - \boldsymbol{h}(\hat{\boldsymbol{x}}_{k|k-1}))$
- **State Covariance Update:** $\boldsymbol{\Sigma}k|k = (\boldsymbol{I} - \boldsymbol{K}_k\boldsymbol{H}_k)\boldsymbol{\Sigma}_{k|k-1}$

3.1.4 Particle Filter (PF)

Particle filters [6] are Bayesian filters suited for nonlinear and non-Gaussian models. They involve the implementation of Sequential Importance Sampling (SIS) which is a Monte-Carlo (MC) method. SIS is a technique to implement a recursive Bayesian filter by MC simulations. Thus, the idea behind the particle filter is to discretize the initial distribution with particles with associated weights, and to propagate them through the prior and the likelihood, resulting in a discrete version of the a posteriori distribution. Then the maximum a posteriori (MAP) estimator can be used to select the best path that fits the observations. In other words, the required posterior density function is represented by a set of random samples with associated weights and computes estimates based on these weights. As the number of samples becomes very large, the Monte-Carlo characterization becomes equivalent to the usual functional description of the posterior PDF, and the SIS filter approaches the optimal Bayesian estimate.

Algorithm: The estimation of the posterior density of the state variable is the main aim of the PF algorithm. In other words, the idea behind PF is to find out the sequential estimation of the values of the hidden states, $\boldsymbol{X}_k$, when the values of the observation process at any time k are given, i.e., $\boldsymbol{y}_0, \boldsymbol{y}_1, \boldsymbol{y}_2, ..., \boldsymbol{y}_k$. The same has been expressed in figure 3.1.

As we know that a posterior density, $p(\boldsymbol{x}_k|\boldsymbol{y}_0, \boldsymbol{y}_1, \boldsymbol{y}_2, ..., \boldsymbol{y}_k)$ is followed by all the Bayesian estimates of $\boldsymbol{X}_k$. An approximation of these conditional probabilities is provided by the PF technique. In contrast to it, the full posterior, $p(\boldsymbol{x}_0, \boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_k|\boldsymbol{y}_0, \boldsymbol{y}_1, \boldsymbol{y}_2, ..., \boldsymbol{y}_k)$ is processed by the Markov Chain Monte-Carlo (MCMC) method. Thus, the model used for the particle filter is given as:

$$\boldsymbol{X}_k|\boldsymbol{X}_{k-1} = \boldsymbol{x}_k \sim p(\boldsymbol{x}_k|\boldsymbol{x}_{k-1}) \tag{3.25}$$

$$\boldsymbol{Y}_k|\boldsymbol{X}_k = \boldsymbol{y}_k \sim p(\boldsymbol{y}_k|\boldsymbol{x}_k) \tag{3.26}$$

Now comes the step for the approximation of the probability density of the PF, $p(\boldsymbol{x}_k|\boldsymbol{y}_0, \boldsymbol{y}_1, \boldsymbol{y}_2, ..., \boldsymbol{y}_k)$ by a weighted set of the samples of the total number of particles, referred to as N_p. Thus, a particle is represented as a possible value for the state vector along with the corresponding probability, which is the value of the posterior PDF at this point, i.e., a particle is represented as:

$$(\boldsymbol{x}_k^{(i)}, \boldsymbol{w}_k^{(i)}), \quad \forall i = 1, 2, ..., N_p \tag{3.27}$$

The principle of importance sampling (IS) relies on the Monte Carlo integration. Assume that we have N independent samples $x_1, \ldots, x_N$ drawn from a distribution with PDF $p(x)$. Then, the Monte-Carlo approximations states that for any function f, we have:

$$\int \boldsymbol{f}(\boldsymbol{x})p(\boldsymbol{x})dx = \boldsymbol{E}_{\boldsymbol{x}\sim p(\boldsymbol{x})}[\boldsymbol{f}(\boldsymbol{x})] \approx \frac{1}{N}\sum_{i=1}^{N} \boldsymbol{f}(\boldsymbol{x}_i). \tag{3.28}$$

This last expression is equivalent to considering a discretization of the pdf $p(x)$ such as:

$$p(\boldsymbol{x}) \approx \frac{1}{N}\sum_{i=1}^{N} \delta(\boldsymbol{x} - \boldsymbol{x}_i) \tag{3.29}$$

where $\delta(\boldsymbol{x})$ denotes the Dirac distribution. However in some situations, drawing samples from $p(\boldsymbol{x})$ might be impossible for practical reasons. The idea of IS is to draw samples $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N$ from a distribution $q(\boldsymbol{x})$ which contains the support of the target distribution $p(\boldsymbol{x})$, referred to as the proposal distribution, from which we are able to draw samples. Noting that

$$\int f(\boldsymbol{x})p(\boldsymbol{x})dx = \int \boldsymbol{f}(\boldsymbol{x})\frac{p(\boldsymbol{x})}{q(\boldsymbol{x})}q(\boldsymbol{x})dx = \boldsymbol{E}_{\boldsymbol{x}\sim q(\boldsymbol{x})}\left[\frac{p(\boldsymbol{x})}{q(\boldsymbol{x})}\boldsymbol{f}(\boldsymbol{x})\right] \tag{3.30}$$

and denoting

$$\boldsymbol{w}_i = \frac{p(\boldsymbol{x}_i)}{q(\boldsymbol{x}_i)} \tag{3.31}$$

the so-called importance weights, we have:

$$\int \boldsymbol{f}(\boldsymbol{x})p(\boldsymbol{x})dx = \boldsymbol{E}_{\boldsymbol{x}\sim p(\boldsymbol{x})}[\boldsymbol{f}(\boldsymbol{x})] \approx \frac{1}{N}\sum_{i=1}^{N} \boldsymbol{w}_i\boldsymbol{f}(\boldsymbol{x}_i). \tag{3.32}$$

This last expression is equivalent to considering a discretization of the PDF $p(\boldsymbol{x})$ such as:

$$p(\boldsymbol{x}) \approx \frac{1}{N}\sum_{i=1}^{N} \boldsymbol{w}_i\delta(\boldsymbol{x} - \boldsymbol{x}_i). \tag{3.33}$$

To improve this approximation, we rather use:

$$p(\boldsymbol{x}) \approx \frac{1}{N}\sum_{i=1}^{N} \overline{\boldsymbol{w}}_i\delta(\boldsymbol{x} - \boldsymbol{x}_i) \tag{3.34}$$

where

$$\overline{\boldsymbol{w}}_i = \frac{\boldsymbol{w}_i}{\sum_{i=1}^{N} \boldsymbol{w}_i} \tag{3.35}$$

to ensure that the approximate pdf has a unit integral. In the case of PF, the posterior distribution is therefore approximated as ...

One key element of PF is the proposal distribution: $\Pi(\boldsymbol{x}_k|\boldsymbol{x}_{0:k-1}, \boldsymbol{y}_{0:k})$. The target distribution is the optimal proposal distribution which would be the optimal choice, where the target distribution is given as :

$$\Pi(\boldsymbol{x}_k|\boldsymbol{x}_{0:k-1}, \boldsymbol{y}_{0:k}) = p(\boldsymbol{x}_k|\boldsymbol{x}_{k-1}, \boldsymbol{y}_k) = \frac{p(\boldsymbol{y}_k|\boldsymbol{x}_k)}{\int p(\boldsymbol{y}_k|\boldsymbol{x}_k)p(\boldsymbol{y}_k|\boldsymbol{x}_{k-1}d\boldsymbol{x}_k)}.p(\boldsymbol{y}_k|\boldsymbol{x}_{k-1}d\boldsymbol{x}_k) \tag{3.36}$$

When $N_p \to \infty$, the proposal becomes: [5]

$$\Pi(\boldsymbol{x}_k|\boldsymbol{x}_{0:k-1}, \boldsymbol{y}_{0:k}) \approx \frac{p(\boldsymbol{y}_k|\boldsymbol{x}_k)}{\int p(\boldsymbol{y}_k|\boldsymbol{x}_k)\hat{p}(d\boldsymbol{x}_k|\boldsymbol{x}_{k-1})}.\hat{p}(d\boldsymbol{x}_k) = \sum_{i=1}^{N_p} \frac{p(\boldsymbol{y}_k|\boldsymbol{x}_k^i(\boldsymbol{x}_{k-1}))}{\sum_{j=1}^{N_p} p(\boldsymbol{y}_k|\boldsymbol{x}_k^j(\boldsymbol{x}_{k-1})}.\delta_{\boldsymbol{x}_k^i(\boldsymbol{x}_{k-1})}.d\boldsymbol{x}_k \tag{3.37}$$

Therefore, when we approximate the result, we have:

$$\hat{p}(d\boldsymbol{x}_k|\boldsymbol{x}_{k-1}) = \frac{1}{N_p}\sum_{i=1}^{N_p} \delta_{\boldsymbol{x}_k^i}(\boldsymbol{x}_{k-1}).d\boldsymbol{x}_k \approx_{N_p\uparrow\infty} p(\boldsymbol{x}_k|\boldsymbol{x}_{k-1})d\boldsymbol{x}_k \tag{3.38}$$

where we use the notation $\delta_{\boldsymbol{x}_k^i}(\boldsymbol{x}_{k-1}) = \delta(\boldsymbol{x}_{k-1} - \boldsymbol{x}_k^i)$.These are associated with N_p random samples, $\boldsymbol{x}_k^i(\boldsymbol{x}_{k-1}), i = 1, 2, 3, ..., N_p$ with a conditional distribution of the random state, $\boldsymbol{x}_k$, given by $\boldsymbol{x}_k = \boldsymbol{x}_{k-1}$.

When the **importance density** is chosen to be the **transition prior probability**, it becomes easier to draw samples or particles and compute the corresponding importance weight.

$$\Pi(\boldsymbol{x}_k|\boldsymbol{x}_{0:k-1}, \boldsymbol{y}_{0:k}) = p(\boldsymbol{x}_k|\boldsymbol{x}_{k-1}) \tag{3.39}$$

where $\Pi(\boldsymbol{x}_k|\boldsymbol{x}_{0:k-1}, \boldsymbol{y}_{0:k})$ is the proposal density and $p(\boldsymbol{x}_k|\boldsymbol{x}_k - 1)$ is the prior density. With the SIS particle filters, there exists a problem of degeneracy. It is seen that after a few iterations, the weight will concentrate on a few particles only and most particles will have negligible weight. Degeneracy occurs when the weight of one particle is close to one and the weights of all other particles are close to or nearly zero. The filtering distribution in this case is represented by a single particle, which results in a very poor approximation. To solve this problem, the idea is to perform resampling because we want to eliminate particles that have small weights and concentrate on the particles with large weights, and hence, what we implement is the SIR (Sequential Importance Resampling) algorithm particle filter. The algorithm for a SIR PF is given by: [5], [6]

- **Initialization:** The particles are randomly generated from the initial prior Gaussian distribution, $p(\boldsymbol{x}_k|\boldsymbol{x}_{k-1}, \boldsymbol{y}_k)$
- For $i = 1, 2, ..., N_p$, first the samples are drawn from the proposal: $\boldsymbol{x}_k^i \sim \Pi(\boldsymbol{x}_k|\boldsymbol{x}_{0:k-1}, \boldsymbol{y}_{0:k})$
- For $i = 1, 2, ..., N_p$, the importance weights are updated as the particle filters are also an approximation, therefore the non-linear filter equation requires recursive implementation, which is the update step: $\hat{w}_k^i = \hat{w}_{k-1}^i.\frac{p(\boldsymbol{y}_k|\boldsymbol{x}_k^i)p(\boldsymbol{x}_k^i|\boldsymbol{x}_{k-1}^i)}{\Pi(\boldsymbol{x}_k^i|\boldsymbol{x}_{0:k-1}^i, \boldsymbol{y}_{0:k})}$.

  Since the proposal distribution is chosen as the prior distribution, the above equation is reduced to:

  $$w_k^i = w_{k-1}^i p(\boldsymbol{y}_k|\boldsymbol{x}_k^i) \tag{3.40}$$

- The normalized importance weights are expressed as: $w_k^i = \frac{\hat{w}_k^i}{\sum_{j=1}^{N_p} \hat{w}_k(k)}$
- Now the computation of the estimate of the effective number of particles is performed as: $\hat{N}_{eff} = \frac{1}{\sum_{i=1}^{N_p}(w_k^i)^2}$

■ If $\hat{N}_{eff} < N_{threshold}$, then resampling is performed, given by:

(a) Draw N_p particles from the current particle set with probabilities proportional to their weights. Then replace the current particle set with this one.

(b) For $i = 1, 2, ..., N_p$; set $w_k^i = \frac{1}{N_p}$

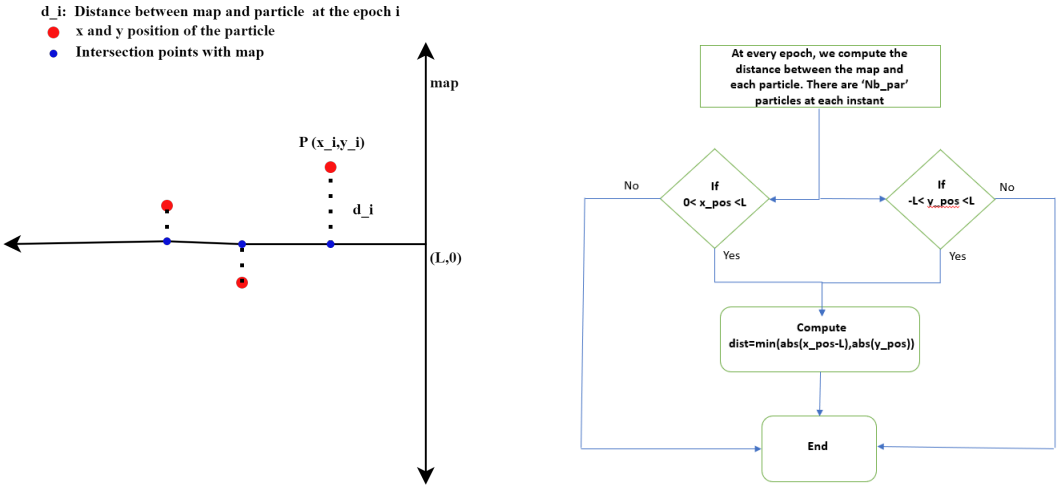
3.2 Map matching

This section introduces the concept of map-matching and the methods used for its implementation with particle filters for GNSS positioning.
Map matching determines a vehicle's position on a digital road database. In this project, the map-matching technique is used along with GNSS observations in order to have a more accurate estimation of the vehicle's position. The map database provides a priori information for the system and acts like 'assistance' data. The advantages of implementing map matching involve cost-effective and reliable solution that provides confidence, and more accuracy solves data efficiency problems, and can overcome bad GNSS observations due to harsh environments.

The maps are defined by the line segments with nodes at the crossing. This technique uses non-standard priors. For example, at a crossing, the prior distribution is a mix of distributions containing all the possible directions that the vehicle can take. Particle filters can only be used in this case because classical filters cannot handle such non-standard priors and likelihood. In this project, two map-matching techniques were implemented with particle filters and a T-shaped map has been defined with a particular length.

3.2.1 Method 1: Map information is used in the observations

The first method is a simple process where we find the distance between the map and the generated particles at that epoch. These distances are combined inside the observation equation along with PVT equations. This technique is used where the observation equations are updated (or in other words, the likelihood PDF is changed) with no change in the proposal distribution. The following figure 3.2 shows the representation of the method used and the algorithm to compute the distance:



(a) Distance between map and estimated position

(b) Flowchart of implementation

Figure 3.2: Method 1: Observations with the distance between map and particles

To account for the map, we are able at any time to measure the distance from a state vector $\boldsymbol{x}$ to the map, that we denote $d(\boldsymbol{x})$. If we assume that the relation between a position (x, y) and the closest point to the map (x_m, y_m) is as

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} x_m \\ y_m \end{bmatrix} + \boldsymbol{e}_m \tag{3.41}$$

where $\boldsymbol{e}_m$ accounts for the uncertainty of the map and is as $\boldsymbol{e}_m \sim \mathcal{N}(0, \sigma_m^2 I_2)$, then

$$d(\boldsymbol{x}) \sim \mathcal{R}(\sigma_m^2) \tag{3.42}$$

where $\mathcal{R}(\sigma_m^2)$ denotes the Rayleigh distribution with parameter σ_m^2 whose support is $\mathbb{R}^+$ and whose PDF is

$$p(d) = \frac{d}{\sigma_m^2} \exp\left(-\frac{1}{2\sigma_m^2} d^2\right) \tag{3.43}$$

. We therefore augment the observations $\boldsymbol{y}_k$ as $\boldsymbol{z}_k = \begin{bmatrix} \boldsymbol{y}_k^T & d(\boldsymbol{x}_k) \end{bmatrix}^T$. Assuming the observations $\boldsymbol{y}_k$ and $d(\boldsymbol{x}_k)$ are independent, we have

$$p(\boldsymbol{z}_k|\boldsymbol{x}_k) = p(\boldsymbol{y}_k|\boldsymbol{x}_k) p(d(\boldsymbol{x}_k)). \tag{3.44}$$

Hence, the update (3.40) is as

$$\begin{aligned} w_k^i &= w_{k-1}^i p(\boldsymbol{y}_k|\boldsymbol{x}_k^i) p(d(\boldsymbol{x}_k^i)) \\ &\propto w_{k-1}^i \exp\left(-\frac{1}{2}\left[\boldsymbol{y}_k - h(\boldsymbol{x}_k^{(i)})\right]^T \boldsymbol{R}_k^{-1} \left[\boldsymbol{y}_k - h(\boldsymbol{x}_k^{(i)})\right]\right) d(\boldsymbol{x}_k^i) \exp\left(-\frac{d(\boldsymbol{x}_k^i)^2}{2\sigma_m^2}\right). \end{aligned} \tag{3.45}$$

3.2.2 Method 2: Proposal drawn from possible position of the vehicle on the map

The second method involves defining the proposal distribution of the particle filter with the possible directions or positions that a vehicle can go from an estimated position, at an epoch. To implement this, we used the method of the intersection of circles. At each epoch, a circle is drawn with the center as the position estimated by PF at the previous epoch. The radius of the circle(r_i) is defined as $r_i = \sqrt{v_x^2 + v_y^2}\delta t$; where v_x and v_y are the estimated x and y velocity at the previous epoch. The intersection points of the circle with the map are computed. We also compute the number of intersection points at every epoch. A uniform proposal distribution is drawn from the intersection points. The points are assumed to have equal probabilities. For the computation of the intersection points, we consider a T-shaped map. The map can be seen to be defined by two straight-line segments. The horizontal line segment lies on the x-axis with x∈[0,L] and y=0. The vertical line segment is parallel to the y-axis with y∈[-L,L] and x=L, where L is the length of the map. The equations for the computation of points of intersection(x_{int}, y_{int}) are:

Horizontal line-segment
y=0 and 0<=x<=L
Let the position estimate at the previous epoch be x_i,y_i. We first check if the condition: $r_i^2 > y_i^2$ is true. Then the points of intersection are

$$x_{int} = x_i \pm \sqrt{r_i^2 - y_i^2} \tag{3.46}$$

and $y_{int} = 0$.

Vertical line-segment
x=L and -L<=y<=L

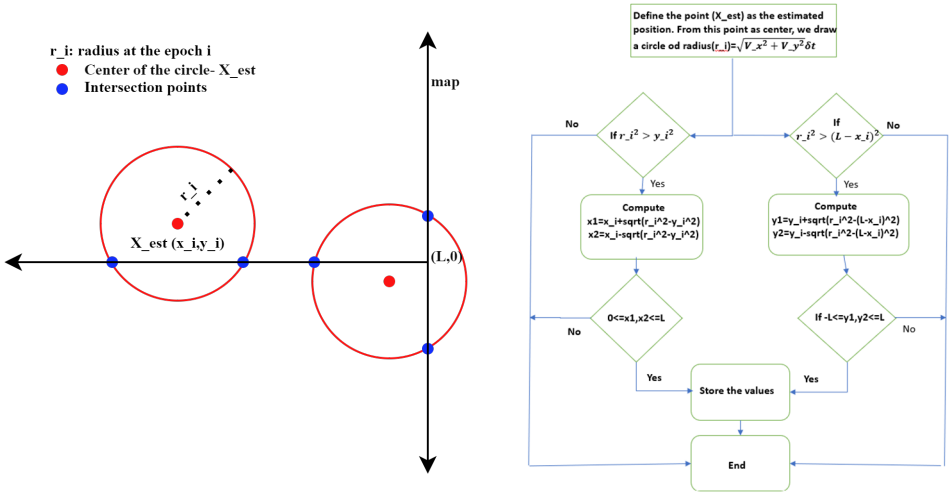
Let the position estimate at the previous epoch be x_i,y_i. We first check if the condition: $r_i^2 > (L - x_i)^2$ is true. Then the points of intersection are

$$y_{int} = x_i \pm \sqrt{r_i^2 - (L - x_i)^2} \tag{3.47}$$

and $x_{int} = L$.

Once we have this information, certain equations for the particle filter are changed to include map matching like the proposal and weight update equations, which are implemented in our algorithm.

The following figure 3.3 shows the circles whose intersections are to be found at each epoch and the algorithm to find them:



(a) Intersection of circles drawn from estimated position with map

(b) Flowchart of implementation

Figure 3.3: Method 2: New proposal density

The equations that need to be updated are:

- The proposal density:

$$\Pi(x_k|x_{0:k-1}, y_{0:k}) = \frac{1}{N_{int}} \sum_{j=1}^{N_{int}} \delta(x_k^i - X_{int}^i) \tag{3.48}$$

  where, N_{int} is the number of intersections, x_k is the estimate from PF and X_{int} is the points of intersection of the map with the circles.

- For i=1,2,...,N_p; samples are drawn from this proposal:

$$x_k^i \sim \Pi(x_k|x_{0:k-1}, y_{0:k}) = \frac{1}{N_{int}} \sum_{j=1}^{N_{int}} \delta(x_k^i - X_{int}^i) \tag{3.49}$$

- For i=1,2,...,N_p; the importance weights are updated:

$$\hat{w}_k^i = \hat{w}_{k-1}^i \frac{p(y_k|x_k^i)p(x_k^i|x_{k-1}^i)}{\Pi(x_k|x_{0:k-1}, y_{0:k})} = \hat{w}_{k-1}^i \frac{p(y_k|x_k^i)p(x_k^i|x_{k-1}^i)}{\frac{1}{N_{int}} \sum_{j=1}^{N_{int}} \delta(x_k^i - X_{int}^i)} \tag{3.50}$$

Since for each intersecting point, $\sum_{j=1}^{N_{int}} \delta(x_k^i - X_{int}^i) = 1$; we have reduced equations for weight update:

$$\hat{w}_k^i = \hat{w}_{k-1}^i \frac{p(y_k|x_k^i)p(x_k^i|x_{k-1}^i)}{\frac{1}{N_{int}}} \tag{3.51}$$

The prior density, normalization of weights, and effective number of particle computations for resampling remain the same as in the SIR particle filter. The resampling of particles is done from the proposal density.