# 2-D Simultaneous Localization and Mapping with Gaussian Processes

Steve Thomas

September 8, 2016

**Abstract**

In this project an attempt has been made to simulate a two dimensional robot moving in different path scenarios. The localization is implemented based on two different algorithms: Map-based localization and Simultaneous Localization and Mapping (SLAM). The robot makes use of a laser sensor to map the obstacles it crosses along its path. The position of the obstacle is fed to a Gaussian Process Kernel which then produces a two dimensional map of obstacles it observes.

## 1 Introduction

Simultaneous Localization and Mapping or SLAM is a process which enables autonomous navigation. When a robot does not have access to GPS coordinates, then an accurate map of the area it navigates is not available and it is on unknown territory. In such a situation the robot has to estimate its position by creating a map of it's surroundings. The robot tries to map an unknown environment while figuring out its present location. Robot uses landmarks to determine its location using a range of measuring devices like sensors, lasers and sonars.

Certain criteria is required for SLAM robots to function efficiently. The main criteria being that the landmark should be stationary. It is unable to determine it's own location if a nearby landmark is continuously moving. Also it requires that the landmarks should be plentiful and should be easily distinguishable from each other. Being able to view the landmark from many different angles is another important requirement.

Once a robot has sensed a landmark it is able to determine its own location by extracting the sensory input and identifying different landmarks. As the robot interacts continuously with the environment, the mapping device and the location, it not only maps the area but also determines it's own position simultaneously.

In this project we focus on the two-dimensional version of the SLAM problem. As discussed earlier, there are two main approaches to Localization, Map based and SLAM. The difference between the two are that in map based localization, the position of the landmarks are known. In SLAM, the location of the landmarks are unknown. Hence to tackle this, in the formulation process of SLAM, we augment the state vector to include the pose (Position and Orientation) of the landmarks that need to be estimated along with the pose of the robot. Additionally we map our surroundings using a Gaussian Process kernel. We make use of Gaussian Process regression for obstacle inference.

## 2 Literature Review

In this project, all of the literature pertaining to Simultaneous localization and mapping(SLAM) and Map based localization has been taken from the class notes of CSCI 5552, taught by professor Stergios Roumeliotis. Understanding the working and implementation of Gaussian Processes has been greatly referenced from the seminal works of Carl Rasmussen and Chris Williams, "Gaussian Processes for Machine Learning", MIT. Additionally, tutorials on Gaussian Processes by Mark Ebden, Oxford university, were also referenced in this project.

## 3 Problem Formulation

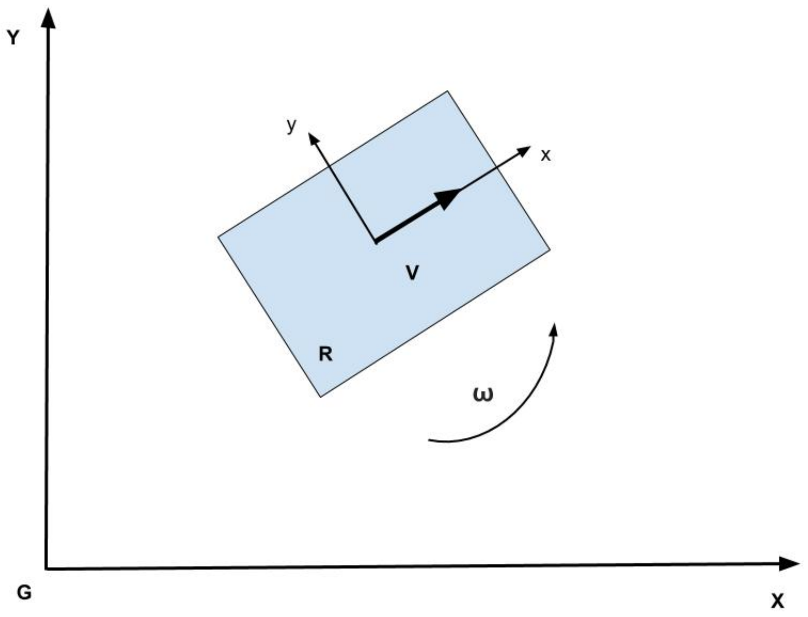

Figure 1: Robot pose

### 3.1 Robot Dynamics

Assume that the robot is moving on a planar surface, with velocity v and rotational velocity $\omega$ in the Robot frame of reference

$$^{R}\dot{x}(t) = v(t) \tag{3.1}$$
$$^{R}\dot{y}(t) = 0 \tag{3.2}$$
$$^{R}\dot{\phi}(t) = \omega(t) \tag{3.3}$$

These same equations expressed with respect to a global frame of reference are

$$^{G}\dot{x}(t) = v(t)cos(\phi)(t) \tag{3.4}$$
$$^{G}\dot{y}(t) = v(t)sin(\phi)(t) \tag{3.5}$$
$$^{G}\dot{\phi}(t) = \omega(t) \tag{3.6}$$

On discretizing the equations, the propagation equations are given as

$$\hat{x}_{k+1} = \hat{x}_k + v_k \delta t cos\hat{\phi}_k \tag{3.7}$$
$$\hat{y}_{k+1} = \hat{y}_k + v_k \delta t sin\hat{\phi}_k \tag{3.8}$$
$$\hat{\phi}_{k+1} = \hat{\phi}_k + \omega_k \delta t \tag{3.9}$$

Here the $\delta t$ is the duration of time step. Here as the measurements are taken in by the robot encoders, we can assume there will be some noise in the pose measurements. Hence we rewrite the propagation equations as

$$\hat{x}_{k+1} = \hat{x}_k + v_{m_k} \delta t cos\hat{\phi}_k \tag{3.10}$$
$$\hat{y}_{k+1} = \hat{y}_k + v_{m_k} \delta t sin\hat{\phi}_k \tag{3.11}$$
$$\hat{\phi}_{k+1} = \hat{\phi}_k + \omega_{m_k} \delta t \tag{3.12}$$

where $v_{m_k}$ and $\omega_{m_k}$ are the velocity and rotational velocity corrupted by zero mean white Gaussian noise

$$v_{m_k} = v_k + \omega_{v_k} \tag{3.13}$$
$$\omega_{m_k} = \omega_k + \omega_{\omega_k} \tag{3.14}$$

The covariances for the white Gaussian noises are given as

$$\mathbf{E}(\omega_{v_k}^2) = \sigma_v^2 \tag{3.15}$$
$$\mathbf{E}(\omega_{\omega_k}^2) = \sigma_\omega^2 \tag{3.16}$$
$$\mathbf{E}(\omega_{v_k}\omega_{\omega_k}) = \sigma_{v\omega} \tag{3.17}$$

## 3.2 Kalman Filter Equations

The Kalman filter is basically an estimator which works in a recursive fashion. Only the estimated state from the previous time step and the current measurement are needed to compute the estimate of the current state. The Kalman filter can be divided in two different stages, the propagation and update stage. The propagation stage takes the state estimate of the previous timestep and creates the estimate of the current time step. This creates an predicted state estimate known as *a priori* estimate. It is *a priori* because the estimate does not include any observation information about the current timestep in it. The update stage takes care of this. The update stage of the algorithm includes the observation information of the timestep to the state estimate. This improved estimate is known as *a posteriori* estimate. The discrete time Kalman filter Equations are given as:

*PROPAGATION*

$$\textit{A priori state estimate} \;\rightarrow\; \hat{\mathbf{x}}_{k+1|k} = \mathbf{\Phi}_k \hat{\mathbf{x}}_{k|k} + \mathbf{G}_k \mathbf{w}_k \tag{3.18}$$

$$\textit{A priori estimate covariance} \;\rightarrow\; \mathbf{P}_{k+1|k} = \mathbf{\Phi}_k \mathbf{P}_{k|k} \mathbf{\Phi}_k^T + \mathbf{G}_k \mathbf{Q}_k \mathbf{G}_k^T \tag{3.19}$$

*UPDATE*

$$\textit{A posterioir measurement estimate} \;\rightarrow\; \hat{\mathbf{z}}_{k+1|k} = \mathbf{H}_{k+1} \hat{\mathbf{x}}_{k+1|k} \tag{3.20}$$

$$\textit{Measurement residual} \;\rightarrow\; \mathbf{r}_{k+1|k} = \mathbf{z}_{k+1} - \hat{\mathbf{z}}_{k+1|k} \tag{3.21}$$

$$\textit{Residual covariance} \;\rightarrow\; \mathbf{S}_{k+1|k} = \mathbf{H}_{k+1} \mathbf{P}_{k+1|k} \mathbf{H}_{k+1}^T + \mathbf{R}_{k+1} \tag{3.22}$$

$$\textit{Optimal Kalman Gain} \;\rightarrow\; \mathbf{K}_{k+1|k} = \mathbf{P}_{k+1|k} \mathbf{H}_{k+1}^T \mathbf{S}_{k+1|k}^{-1} \tag{3.23}$$

$$\textit{A posteriori state estimate} \;\rightarrow\; \hat{\mathbf{x}}_{k+1|k+1} = \hat{\mathbf{x}}_{k+1|k} + \mathbf{K}_{k+1|k} \mathbf{r}_{k+1|k} \tag{3.24}$$

$$\textit{A posterioir estimate covariance} \;\rightarrow\; \mathbf{P}_{k+1|k+1} = \mathbf{P}_{k+1|k} - \mathbf{P}_{k+1|k} \mathbf{H}_{k+1}^T \mathbf{S}_{k+1|k}^{-1} \mathbf{H}_{k+1} \mathbf{P}_{k+1|k} \tag{3.25}$$

The two stages occur alternately at each timestep. If there is an instant where observation information is unavailable, then the update stage is skipped and *a posteriori* estimate is equated to the *a priori* estimate.

## 3.3 Propagation Step

By integrating noisy encoder signals, the robot will diverge from their real values and the robot will eventually be lost. Hence it becomes essential to derive the estimate error propagation. To achieve this we subtract from the equations for the actual pose with the corresponding pose estimates.

$$\tilde{x}_{k+1} = x_{k+1} - \hat{x}_{k+1} \tag{3.26}$$

$$\tilde{y}_{k+1} = y_{k+1} - \hat{y}_{k+1} \tag{3.27}$$

$$\tilde{\phi}_{k+1} = \phi_{k+1} - \hat{\phi}_{k+1} \tag{3.28}$$

Expressing these equations in matrix form:

$$\begin{bmatrix}\tilde{x}_{k+1}\\ \tilde{y}_{k+1}\\ \tilde{\phi}_{k+1}\end{bmatrix} = \begin{bmatrix}1 & 0 & -v_{mk}\delta t sin\hat{\phi}_k\\ 0 & 1 & v_{mk}\delta t cos\hat{\phi}_k\\ 0 & 0 & 1\end{bmatrix}\begin{bmatrix}\tilde{x}_k\\ \tilde{y}_k\\ \tilde{\phi}_k\end{bmatrix} + \begin{bmatrix}-\delta t cos\hat{\phi}_k & 0\\ -\delta t sin\hat{\phi}_k & 0\\ 0 & -\delta t\end{bmatrix}\begin{bmatrix}\omega_{v_k}\\ \omega_{\omega_k}\end{bmatrix} \tag{3.29}$$
$$\Leftrightarrow \tilde{\mathbf{x}}_{k+1} = \mathbf{\Phi}_k\tilde{\mathbf{x}}_k + \mathbf{G}_k\mathbf{w}_k$$

Here $\Phi_k$ is the Jacobian matrix with respect to estimated state $\mathbf{x}_k$ and $G_k$ is the Jacobian matrix with respect to $\mathbf{w}_k$. Having found the $\Phi_k$ and $G_k$ matrices, the covariance propagation matrix can be computed as:

$$\begin{aligned}\mathbf{P}_{k+1|k} &= \mathbf{E}(\tilde{\mathbf{x}}_{k+1}\tilde{\mathbf{x}}_{k+1}^T)\\ &= \mathbf{E}[(\mathbf{\Phi}_k\tilde{\mathbf{x}}_k + \mathbf{G}_k\mathbf{w}_k)(\mathbf{\Phi}_k\tilde{\mathbf{x}}_k + \mathbf{G}_k\mathbf{w}_k^T)]\\ &= \mathbf{\Phi}_k\mathbf{E}(\tilde{\mathbf{x}}_k\tilde{\mathbf{x}}_k^T)\mathbf{\Phi}_k^T + \mathbf{G}_k\mathbf{E}(\mathbf{w}_k\mathbf{w}_k^T)\mathbf{G}_k^T\\ &= \mathbf{\Phi}_k\mathbf{P}_{k|k}\mathbf{\Phi}_k^T + \mathbf{G}_k\mathbf{Q}_k\mathbf{G}_k^T\end{aligned} \tag{3.30}$$

Here as $\mathbf{Q}_k$ may vary with time and hence we make use of the $k$ subscript to denote this and $\mathbf{P}_{k|k}$ is just the covariance of the estimate from the previous time step.

## 3.4 Map Based Localization

If the robot only relies on a prior estimates, the robot will soon drift away from the real values and it will soon be lost. Hence it is essential in estimation to have some form of information or data to be inputed to estimate its future states. In map based localization, we assume that the Landmark position to the global frames is already known. So each landmark has an established pose. What follows is that the robot carries different types of sensors that measure different quantities like measuring relative position, orientation, distance and bearing. Based on these measurements, the Kalman filter provides correct pose estimates of the robot.

### The Relative Position Measurements

Given a known landmark, the robot measures the relative position of the landmark with respect to its local frame. The relative position measurements towards the landmarks and global frame of reference can be defined as:

$${}^{G}\mathbf{p}_L = \mathbf{p} + \mathbf{C}(\phi)^{R}\mathbf{p}_L \tag{3.31}$$

$\mathbf{p}$ is the position of the robot in the global frame of reference and $\mathbf{C}(\phi)$ is a 2 by 2 rotation matrix. The measurement in this case is the position of the landmark with respect to the robot. Therefore rearranging eqn 2.31 in will provide us with our desired measurement.

$$\mathbf{z} =^{R} \mathbf{p}_L =^{R}_{G} \mathbf{C}(\phi_R)(^{G}\mathbf{p}_L -^{G} \mathbf{p}_R) + \mathbf{n} \tag{3.32}$$

Here $\mathbf{n}$ is some zero mean white Gaussian noise. Using the current robot pose estimate and the knowledge (from the map)of the landmark position expressed in global coordinates, the estimated measurement can be written as

$$\hat{\mathbf{z}} =^R \hat{\mathbf{p}}_L =^R_G \mathbf{C}(\hat{\phi}_R)(\mathbf{p}_L - \hat{\mathbf{p}}_R) \tag{3.33}$$

By subtracting eqns 2.32 and 2.33, the error measurement $\tilde{\mathbf{z}}$ can be computed as:

$$\begin{aligned} \tilde{\mathbf{z}} &= \mathbf{z} - \hat{\mathbf{z}} \\ &=^R_G \mathbf{C}(\phi_R)(^G\mathbf{p}_L -^G \mathbf{p}_R) + \mathbf{n} -^R_G \mathbf{C}(\hat{\phi}_R)(\mathbf{p}_L - \hat{\mathbf{p}}_R) \end{aligned} \tag{3.34}$$

Taking the Jacobian of $\tilde{\mathbf{z}}$ with respect to $\hat{\mathbf{p}}$ and $\hat{\phi}$, we get our information matrix $\mathbf{H}$ for relative position. Hence we write $\tilde{\mathbf{z}}$ as:

$$\begin{gathered} \begin{bmatrix} -^R_G\mathbf{C}(\hat{\phi}) & -^R_G \mathbf{C}(\hat{\phi})\mathbf{J}(\mathbf{p}_L - \hat{\mathbf{p}}) \end{bmatrix} \begin{bmatrix} \tilde{\mathbf{p}} \\ \tilde{\phi} \end{bmatrix} + \mathbf{n} \\ \tilde{\mathbf{z}} = \mathbf{H}\tilde{\mathbf{x}} + \mathbf{n} \end{gathered} \tag{3.35}$$

where

$$\mathbf{J} = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \tag{3.36}$$

### The Relative Orientation measurements

The relative orientation that the robot measures with respect to its own frame

$$z =^R \phi_L + n_\phi = \phi_l - \phi_R + n_\phi \tag{3.37}$$

where $n_\phi$ is the measurement noise, assuming zero-mean white Gaussian processes with known variance $\sigma^2$. The expected value $\hat{z}$ of this measurement is given as:

$$\hat{z} =^R \hat{\phi}_L = \phi_L - \hat{\phi} \tag{3.38}$$

Therefore the measurement error equation and the corresponding information matrix are given as:

$$\begin{aligned} \tilde{z} &= z - \hat{z} \\ &= \phi_L - \phi + n_\phi - \phi_L + \hat{\phi} = \hat{\phi} - \phi + n_\phi = -\tilde{\phi} + n_\phi \\ &= \begin{bmatrix} 0 & -1 \end{bmatrix} \begin{bmatrix} \tilde{\mathbf{p}} \\ \tilde{\phi} \end{bmatrix} + n_\phi \\ &= \mathbf{H}\tilde{\mathbf{x}} + n_\phi \end{aligned} \tag{3.39}$$

### Distance and Bearing Measurements

Another way, a robot can measure the distance and bearing of a landmark given its pose. The relative distance to a landmark can be given by:

$$z_{k+1} = \sqrt{(x_{L_{k+1}} - x_{R_{k+1}})^2 + (y_{L_{k+1}} - y_{R_{k+1}})^2} + n_{k+1} \tag{3.40}$$

It measures the bearing angles to the landmarks as :

$$z_{k+1} = atan2((y_{L_{k+1}} - y_{R_{k+1}}), (x_{L_{k+1}} - x_{R_{k+1}})) - \phi_{R_{k+1}} + n_{k+1} \tag{3.41}$$

Since these equations are non-linear, we have to linearize them to obtain the measurement Jacobain matrix for the Kalman filter. The information $\mathbf{H}$ matrix is computed as

$$\tilde{z} \simeq \begin{bmatrix} \frac{-\mathbf{p}_L^T - \hat{\mathbf{p}}^T}{||\mathbf{p}_L - \hat{\mathbf{p}}||} & 0 \\ \frac{-(\mathbf{p}_L - \hat{\mathbf{p}})^T \mathbf{J}^T}{||\mathbf{p}_L - \hat{\mathbf{p}}||^2} & -1 \end{bmatrix} \begin{bmatrix} \tilde{\mathbf{p}} \\ \tilde{\phi} \end{bmatrix} = \mathbf{H}\tilde{\mathbf{x}} + n \tag{3.42}$$

### SLAM Propagate

In map based localization, we know the number of landmarks and their position in the global frame of reference. Hence the state variable estimates get modified at each time step based on these known landmarks. However in SLAM the whole scenario changes. In SLAM, the robot does not know the position of the Landmarks beforehand. Hence at each timestep, when the robot encounters a new landmark, it will determine if its a new landmark or if its already been discovered before. In order to remember old landmarks, the robot has to add every landmark pose in its state variables.This way it ‘remembers’ every landmark it has seen and will accordingly differentiate between old and new landmarks.

The SLAM state vector can be defined as:

$$\mathbf{X} = \begin{bmatrix} x_R \\ x_{L_1} \\ \vdots \\ x_{L_N} \end{bmatrix} \tag{3.43}$$

Here $x_R$ is the robot pose at a timestep and $x_{L_N}$ is the pose of the Nth landmark discovered. The robot state propagation is as under:

$$\tilde{\mathbf{x}}_{R_{k+1}} = \Phi_{R_k} \tilde{\mathbf{x}}_{R_k} + \mathbf{G}_{R_k} \mathbf{w}_{R_k} \tag{3.44}$$

$$\Phi_{R_k} = \begin{bmatrix} 1 & 0 & -v_{mk} \delta t sin\hat{\phi}_k \\ 0 & 1 & v_{mk} \delta t cos\hat{\phi}_k \\ 0 & 0 & 1 \end{bmatrix} \tag{3.45}$$

$$\mathbf{G}_{R_k} = \begin{bmatrix} -\delta t cos\hat{\phi}_k & 0 \\ -\delta t sin\hat{\phi}_k & 0 \\ 0 & -\delta t \end{bmatrix} \tag{3.46}$$

$$\mathbf{X}_{L_{i,k+1}} = \mathbf{X}_{L_{i,k}} \tag{3.47}$$

$$\hat{X}_{L_{i,k+1}} = \hat{X}_{L_{i,k}} \tag{3.48}$$

$$\tilde{\mathbf{X}}_{L_{i,k+1}} = \tilde{\mathbf{X}}_{L_{i,k}} = \mathbf{I}_3\tilde{\mathbf{X}}_{L_{i,k}} + \mathbf{0}_{3\times 2} w_{R_k} \tag{3.49}$$

The state error propagation is given by:

$$\begin{aligned} \tilde{\mathbf{x}}_{k+1} &= \Phi_k\tilde{\mathbf{x}}_k + \mathbf{G}_k\mathbf{w}_k \\ &= \begin{bmatrix} \Phi_{R_k} & 0 & \dots & 0 \\ 0 & I & \dots & 0 \\ 0 & 0 & \dots & I \end{bmatrix} \begin{bmatrix} \tilde{\mathbf{X}}_{R_k} \\ \tilde{\mathbf{X}}_{L_{1,k}} \\ \vdots \\ \tilde{\mathbf{X}}_{L_{N,k}} \end{bmatrix} + \begin{bmatrix} G_{R_k} \\ 0 \\ \vdots \\ 0 \end{bmatrix} \mathbf{w_{R_k}} \end{aligned} \tag{3.50}$$

$$\begin{aligned} \mathbf{P}_{k+1} &= \Phi_k\mathbf{P}_{k|k}{\Phi_k}^T + \mathbf{G}_k\mathbf{Q}_{R_k}{\mathbf{G}_k}^T \\ &= \begin{bmatrix} \Phi_{R_k} & 0 & \cdots & 0 \\ 0 & I & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & I \end{bmatrix} \begin{bmatrix} \mathbf{P}_{RR} & \mathbf{P}_{RL_1} & \cdots & \mathbf{P}_{L_1L_N} \\ \mathbf{P}_{L_1R} & \mathbf{P}_{L_1L_1} & \cdots & \mathbf{P}_{L_1L_N} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{P}_{L_NR} & \mathbf{P}_{L_NL_1} & \cdots & \mathbf{P}_{L_NL_N} \end{bmatrix} \begin{bmatrix} \Phi_{R_k}^T & 0 & \cdots & 0 \\ 0 & I & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & I \end{bmatrix} + \begin{bmatrix} \mathbf{G_{R_k}} \\ 0 \\ \vdots \\ 0 \end{bmatrix} \mathbf{Q_{R_k}} \begin{bmatrix} {\mathbf{G_{R_k}}}^T & 0 & \cdots & 0 \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{P}_{RR_{k|k}} & \Phi_{R_k}\mathbf{P}_{RL_{1k|k}} & \dots & \Phi_{R_k}\mathbf{P}_{RL_{Nk|k}} \\ \mathbf{P}_{L_1R_{k|k}}{\Phi_{R_k}}^T & \mathbf{P}_{L_1L_{1k|k}} & \dots & \mathbf{P}_{L_1L_{Nk|k}} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{P}_{L_NR_{k|k}}{\Phi_{R_k}}^T & \mathbf{P}_{L_NL_{1k|k}} & \dots & \mathbf{P}_{L_NL_{Nk|k}} \end{bmatrix} \end{aligned} \tag{3.51}$$

After repeating the process m times, one can derive the expression for the covariance propagation, after m steps if there is no update during those m steps

$$\mathbf{P}_{k+m+1|k} = \begin{bmatrix} \mathbf{P}_{RR_{k+m+1|k}} & \mathbb{F}_{\mathbb{R}}(k+m,k)\mathbf{P}_{RL_{1k|k}} & \dots & \mathbb{F}_{\mathbb{R}}(k+m,k)\mathbf{P}_{RL_{Nk|k}} \\ \mathbf{P}_{L_1R_{k|k}}\mathbb{F}_{\mathbb{R}}(k+m,k)^T & \mathbf{P}_{L_1L_{1k|k}} & \dots & \mathbf{P}_{L_1L_{Nk|k}} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{P}_{L_NR_{k|k}}\mathbb{F}_{\mathbb{R}}(kk+m,k)^T & \mathbf{P}_{L_NL_{1k|k}} & \dots & \mathbf{P}_{L_NL_{Nk|k}} \end{bmatrix} \tag{3.52}$$

where

$$\mathbb{F}_{\mathbb{R}}(k+m,k) = \Phi_{R_{k+m}}...\Phi_{R_{k+1}}\Phi_{R_k} \tag{3.53}$$

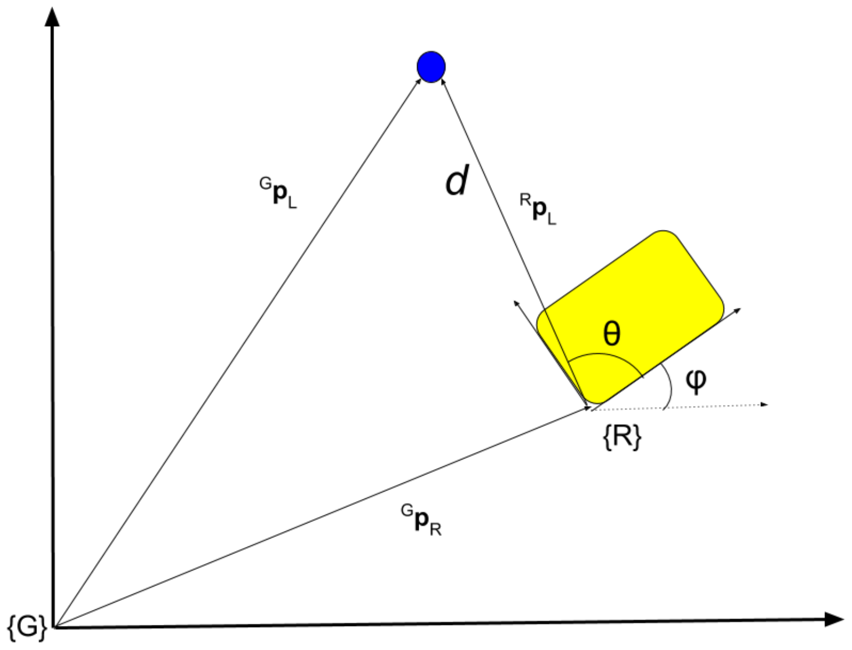

Figure 2: Relative distance and bearing to a landmark

## 3.5 SLAM Update

### The Relative Position Measurement

When calculating the real and estimated measurements we apply the same approach as we did in map based localization. The only main difference is that now we find the jacobian matrix with the estimated state variables of the robot and the estimated position of the landmark. This is done with respect to the global frame which lies within the state variable vector.

$$\begin{aligned} \mathbf{z} &= {}^{R}\mathbf{p}_{L_i} = {}^{R}_{G}\mathbf{C}(\phi_R)({}^{G}\mathbf{p}_{L_i} - {}^{G}\mathbf{p}_R) + \mathbf{n} \\ \hat{\mathbf{z}} &= {}^{R}\hat{\mathbf{p}}_{L_i} = {}^{R}_{G}\mathbf{C}(\hat{\phi}_R)(\hat{\mathbf{p}}_{L_i} - \hat{\mathbf{p}}_R) \end{aligned} \tag{3.54}$$

By subtracting the actual measurement and its estimate, the error measurement $\tilde{\mathbf{z}}$ can be computed as:

$$\begin{aligned} \tilde{\mathbf{z}} &= \mathbf{z} - \hat{\mathbf{z}} \\ &= {}^{R}_{G}\mathbf{C}(\phi_R)({}^{G}\mathbf{p}_{L_i} - {}^{G}\mathbf{p}_R) + \mathbf{n} - {}^{R}_{G}\mathbf{C}(\hat{\phi}_R)(\hat{\mathbf{p}}_{L_i} - \hat{\mathbf{p}}_R) \end{aligned} \tag{3.55}$$

Taking the Jacobian of $\tilde{\mathbf{z}}$ with respect to $\hat{\mathbf{p}}$, $\hat{\phi}$ and $\hat{\mathbf{p}}_{\mathbf{L_i}}$ we get our information matrix $\mathbf{H}$ for relative position. Hence we write $\tilde{\mathbf{z}}$ as:

$$\begin{aligned}
&\left[\begin{array}{cc|c} -{}^R_G\mathbf{C}(\hat{\phi}) & -{}^R_G\mathbf{C}(\hat{\phi})\mathbf{J}(\mathbf{p}_L-\hat{\mathbf{p}}) & {}^R_G\mathbf{C}(\hat{\phi}) \end{array}\right]\left[\begin{array}{c}\tilde{\mathbf{p}}\\ \tilde{\phi}\\ \hline \tilde{\mathbf{p}}_\mathbf{l}\end{array}\right]+\mathbf{n}\\
&=\begin{bmatrix}\mathbf{H}_R\ \mathbf{H}_{L_i}\end{bmatrix}\begin{bmatrix}\tilde{\mathbf{x}}_R\\ \tilde{\mathbf{x}}_{L_i}\end{bmatrix}+\mathbf{n}
\end{aligned}\tag{3.56}$$

or

$$\begin{aligned}
\tilde{\mathbf{z}}&=\begin{bmatrix}\mathbf{H}_R\ 0\dots\ 0\ \mathbf{H}_{L_i}\ 0\dots\ 0\end{bmatrix}\begin{bmatrix}\tilde{\mathbf{x}}_R\\ \tilde{\mathbf{x}}_{L_1}\\ \vdots\\ \tilde{\mathbf{x}}_{L_{i-1}}\\ \tilde{\mathbf{x}}_{L_i}\\ \tilde{\mathbf{x}}_{L_{i+1}}\\ \vdots\\ \tilde{\mathbf{x}}_{L_N}\end{bmatrix}+\mathbf{n}\\
&=\mathbf{H}\tilde{\mathbf{x}}+\mathbf{n}
\end{aligned}\tag{3.57}$$

## The Relative Orientation Measurement

If the relative orientation measurement of a landmark is recorded by the robot, this can be expressed as

$$z_{k+1}={}^R\phi_{L_{i,k+1}}+n_{k+1}=\phi_{L_{i,k+1}}-\phi_{R_{k+1}}+n_{n+1}\tag{3.58}$$

which is similar to eqn 2.37. Similar to the case of Map based localization, the measurement error equation can be derived as:

$$\begin{aligned}
\tilde{z}=z-\hat{z}&=\phi_{L_i}-\phi_R-(\hat{\phi}_{L_i}-\hat{\phi}_R)+n\\
&=\hat{\phi}_{L_i}-\hat{\phi}_R+\tilde{\phi}_{L_i}-\tilde{\phi}_R-(\hat{\phi}_{L_i}-\hat{\phi}_R)+n\\
&\simeq\tilde{\phi}_{L_i}-\tilde{\phi}_R+n\\
&=\left[\begin{array}{cc|cc}0 & -1 & 0 & 1\end{array}\right]\left[\begin{array}{c}\tilde{\mathbf{p}}_R\\ \tilde{\phi}_R\\ \hline \tilde{\mathbf{p}}_{L_i}\\ \tilde{\phi}_{L_i}\end{array}\right]+n\\
&=\begin{bmatrix}\mathbf{H}_R & \mathbf{H}_{L_i}\end{bmatrix}\begin{bmatrix}\tilde{\mathbf{x}}_R\\ \tilde{\mathbf{x}}_{L_i}\end{bmatrix}+n
\end{aligned}\tag{3.59}$$

## Distance and Bearing Measurement

In this section we consider distance and bearing measurements that are transformed into a relative position measurements of each landmark. Then the same derivations for the measurement Jacobian matrix as in the as in the case of map based localization can be used to update the states.

## 3.6 Exploration Phase of SLAM

The SLAM Algorithm works in a very orderly fashion. Initially, we assume that the robot pose and updated covariance matrix is zero. If landmark $L_i$ is now detected, then the measurement matrix is:

$$\mathbf{H} = \begin{bmatrix}\mathbf{H}_R \ \mathbf{H}_{L_1} \ \mathbf{H}_{L_2} \ \dots \ \mathbf{H}_{L_i}\end{bmatrix} \tag{3.60}$$

with

$$\mathbf{H}_R = \begin{bmatrix}-\mathbf{C}^T(\hat{\phi}) & -\mathbf{C}^T(\hat{\phi})\mathbf{J}(\hat{p}_{L_i} - \hat{p})\end{bmatrix} \tag{3.61}$$

$$\mathbf{H}_{L_i} = \mathbf{C}^T(\hat{\phi}) \tag{3.62}$$

Given our covariance $\mathbf{P}$ and measurement $\mathbf{H}$ matrices, the covariance of the residuals $\mathbf{S}$ can be computed as

$$\begin{aligned}\mathbf{S} &= \mathbf{HPH}^T + \mathbf{R} \\ &= \mathbf{H}_R\mathbf{P}_{RR}\mathbf{H}_R^T + \mathbf{H}_{L_i}\mathbf{P}_{L_iL_i}\mathbf{H}_{L_i}^T + \mathbf{R}\end{aligned} \tag{3.63}$$

Next we compute the Kalman Gain

$$\begin{gathered}\mathbf{K} = \mathbf{PH}^T\mathbf{S}^{-1} \\ \begin{bmatrix}\mathbf{K}_R \\ \mathbf{K}_{L_1} \\ \vdots \\ \mathbf{K}_{L_i}\end{bmatrix} = \begin{bmatrix}\mathbf{P}_{RR}\mathbf{H}_R^T\mathbf{S}^{-1} \\ \mathbf{P}_{L_iR}\mathbf{H}_R^T\mathbf{S}^{-1} \\ \vdots \\ \mathbf{P}_{L_iL_i}\mathbf{H}_{L_i}^T\mathbf{S}^{-1}\end{bmatrix}\end{gathered} \tag{3.64}$$

Since $\mathbf{P}_{RR}$, $\mathbf{P}_{L_\mathbf{i}R}$, ..., $\mathbf{P}_{L_\mathbf{i-1}R}$ are bounded, it is

$$\mathbf{K}_R = \mathbf{K}_{L_i} = \ldots = \mathbf{K}_{L_{i-1}} = 0$$

The remaining component $\mathbf{K}_{L_\mathbf{i}}$ is computed as follows:

$$\begin{aligned}\mathbf{K}_{L_\mathbf{i}} &= \mathbf{P}_{L_\mathbf{i}L_\mathbf{i}}\mathbf{H}_{L_i}^T\mathbf{S}^{-1} \\ &= \mathbf{H}_{L_i}^T\mathbf{H}_{L_\mathbf{i}}\mathbf{P}_{L_\mathbf{i}L_\mathbf{i}}\mathbf{H}_{L_i}^T(\mathbf{H}_R\mathbf{P}_{RR}\mathbf{H}_R^T + \mathbf{H}_{L_\mathbf{i}}\mathbf{P}_{L_\mathbf{i}L_\mathbf{i}}\mathbf{H}_{L_i}^T + \mathbf{R})^{-1} \\ &= \mathbf{H}_{L_i}^T[(\mathbf{H}_{L_\mathbf{i}}\mathbf{P}_{L_\mathbf{i}L_\mathbf{i}}\mathbf{H}_{L_i}^T)^{-1}(\mathbf{H}_R\mathbf{P}_{RR}\mathbf{H}_R^T + \mathbf{R}) + \mathbf{I}]^{-1} \\ &= \mathbf{H}_{L_i}^T\end{aligned} \tag{3.65}$$

Therefore the final $\mathbf{K}$ matrix will look like

$$\mathbf{K} = \begin{bmatrix}0 \\ 0 \\ \vdots \\ \mathbf{H}_{L_i}^T\end{bmatrix} \tag{3.66}$$

Hence the state update is given as

$$\hat{\mathbf{x}}_{k+1|k+1} = \hat{\mathbf{x}}_{k+1|k} + \mathbf{K}\mathbf{r} \tag{3.67}$$

After updating the state vector, the next thing to update in the update stage is the covariance matrix

$$\mathbf{P}_{k+1|k+1} = \mathbf{P}_{k+1|k} - \mathbf{K}\mathbf{S}\mathbf{K}^T \tag{3.68}$$

substituting from equation 2.64, $\mathbf{K}\mathbf{S}\mathbf{K}^T$ is given as

$$\mathbf{K}\mathbf{S}\mathbf{K}^T = \begin{bmatrix} \mathbf{P}_{RR}\mathbf{H}_R^T \\ \vdots \\ \mathbf{P}_{L_{i-1}R}\mathbf{H}_R^T \\ \mathbf{P}_{L_iL_i}\mathbf{H}_{L_i}^T \end{bmatrix} \mathbf{S}^{-1} \begin{bmatrix} \mathbf{H}_R\mathbf{P}_{RR} & \dots & \mathbf{H}_R\mathbf{P}_{RL_{i-1}} & \mathbf{H}_{L_i}\mathbf{P}_{L_iL_i} \end{bmatrix} \tag{3.69}$$

Since $\mathbf{P}_{RR}$, $\mathbf{P}_{RL_\mathbf{j}} = \mathbf{P}_{L_\mathbf{j}R}{}^T$ are bounded, all the upper-left submatrices in the expression will be zero. Also expression $\mathbf{P}_{L_iL_i}\mathbf{H}_{L_i}^T\mathbf{S}^{-1}$ can be rewritten as:

$$\begin{aligned} \mathbf{S}^{-1}\mathbf{H}_{L_i}\mathbf{P}_{L_iL_i} &= (\mathbf{H}_R\mathbf{P}_{RR}\mathbf{H}_R^T + \mathbf{H}_{L_\mathbf{i}}\mathbf{P}_{L_\mathbf{i}L_\mathbf{i}}\mathbf{H}_{L_i}^T + \mathbf{R})^{-1}\mathbf{H}_{L_\mathbf{i}}\mathbf{P}_{L_\mathbf{i}L_\mathbf{i}}\mathbf{H}_{L_i}^T\mathbf{H}_{L_\mathbf{i}} \\ &= [(\mathbf{H}_R\mathbf{P}_{RR}\mathbf{H}_R^T)(\mathbf{H}_{L_\mathbf{i}}\mathbf{P}_{L_\mathbf{i}L_\mathbf{i}}\mathbf{H}_{L_i}^T)^{-1} + \mathbf{I}]^{-1}\mathbf{H}_{L_\mathbf{i}} \\ \mathbf{H}_{L_\mathbf{i}} &\Leftrightarrow \mathbf{H}_{L_i}^T = \mathbf{P}_{L_iL_i}\mathbf{H}_{L_i}^T\mathbf{S}^{-1} \end{aligned} \tag{3.70}$$

Therefore $\mathbf{K}\mathbf{S}\mathbf{K}^T$ is defined as

$$\mathbf{K}\mathbf{S}\mathbf{K}^T = \begin{bmatrix} 0 & \dots & 0 & \mathbf{P}_{RR}\mathbf{H}_R^T\mathbf{H}_{L_i} \\ \vdots & \ddots & \vdots & \dots \\ 0 & \dots & 0 & \mathbf{P}_{L_{\mathbf{i-1}}R}\mathbf{H}_R^T\mathbf{H}_{L_i} \\ \mathbf{H}_{L_i}^T\mathbf{P}_R\mathbf{P}_{RR} & \dots & \mathbf{H}_{L_i}^T\mathbf{H}_R\mathbf{P}_{RL_{i-1}} & \mathbf{P}_{L_iL_i}\mathbf{H}_{L_i}^T\mathbf{S}^{-1}\mathbf{H}_{L_i}\mathbf{P}_{L_iL_i} \end{bmatrix} \tag{3.71}$$

Taking this expression we subtract this from $\mathbf{P}_{k+1|k}$. The only submatrix left undetermined is $\mathbf{P}_{k+1|k+1}$

$$\mathbf{P}_{L_iL_{i,k+1|k+1}} = \mathbf{P}_{L_iL_{i,k+1|k}} - \mathbf{P}_{L_iL_{i,k+1|k}}\mathbf{H}_{L_i}^T\mathbf{S}^{-1}\mathbf{H}_{L_i}\mathbf{P}_{L_iL_{i,k+1|k}}$$

inverting the above expression

$$\begin{aligned} \mathbf{P}^{-1}_{L_iL_{i,k+1|k+1}} &= \mathbf{P}^{-1}_{L_iL_{i,k+1|k}} + \mathbf{H}_{L_i}^T(\mathbf{H}_R\mathbf{P}_{RR}\mathbf{H}_R + \mathbf{R})^{-1}\mathbf{H}_{L_i} \\ &= \mathbf{0} + [\mathbf{H}_{L_i}^T(\mathbf{H}_R\mathbf{P}_{RR}\mathbf{H}_R + \mathbf{R})\mathbf{H}_{L_i}]^{-1} \Rightarrow \\ \mathbf{P}_{L_iL_{i,k+1|k+1}} &= \mathbf{H}_{L_i}^T(\mathbf{H}_R\mathbf{P}_{RR}\mathbf{H}_R + \mathbf{R})\mathbf{H}_{L_i} \end{aligned} \tag{3.72}$$

Hence the updated a posterior covariance matrix after initializing the *i*-th landmark becomes

$$\mathbf{P}_{k+1|k+1} = \begin{bmatrix} \mathbf{P}_{RR} & \dots & \mathbf{P}_{RL_{i-1}} & -\mathbf{P}_{RR}\mathbf{H}_R^T\mathbf{H}_{L_i} \\ \vdots & \ddots & \vdots & \vdots \\ \mathbf{P}_{L_{i-1}R} & \dots & \mathbf{P}_{L_{i-1}L_{i-1}} & -\mathbf{P}_{L_{\mathbf{i-1}}R}\mathbf{H}_R^T\mathbf{H}_{L_i} \\ \mathbf{H}_{L_i}^T\mathbf{P}_R\mathbf{P}_{RR} & \dots & \mathbf{H}_{L_i}^T\mathbf{H}_R\mathbf{P}_{RL_{i-1}} & \mathbf{H}_{L_i}^T(\mathbf{H}_R\mathbf{P}_{RR}\mathbf{H}_R + \mathbf{R})\mathbf{H}_{L_i} \end{bmatrix} \tag{3.73}$$

## 3.7 Mahalanobis Distance Test

The Mahalanobis test is computed to test whether the measurement is a good or a bad measurement. When a new landmark is discovered, the Mahalanobis test is used to tell if it is a new landmark or a previous landmark. We first compute the Mahalanobis distance between the current measurement and all the previous observed landmarks. From all these we look to find the minimum distance. If this Mahalanobis distance is smaller than a small threshold $\xi_1$, we consider the current measurement is a measurement of the previous landmark discovered, and use it for updating the previous landmark. If this Mahalanobis distance is larger than a large threshold $\xi_2$, then we consider its a measurement of a new landmark and use it for initialization. If the Mahalanobis distance is between the two thresholds, it is not easy to determine if the current measurement is a measurement of a previous landmark or a new landmark and so that measurement is discarded.

$$d_m = \mathbf{r}^T\mathbf{S}^{-1}\mathbf{r} < \gamma^2 \tag{3.74}$$

## 3.8 Gaussian Process Regression

Gaussian processes are used in prediction problems and have a similar approach to least squares. Given a set of noisy measurements, we are asked to find the best estimate for a new value associated to the given measurements. In the given figure below, the black dots are the noisy measurements and the orange dot is the data we want to estimate. Looking at the figure, one can assume the data to be linear and may use least-squares method to fit a straight line. However, in Gaussian Process Regression, we do not make such assumptions. Without knowing the underlying function of the data, we can still predict or interpolate points solely based on the covariances between each point.

Depending on the type of problem we have to choose our covariance function. This is an essential part of the process. The reliability of the regression depends upon the choice of the covariance function. Consider a case where our data set is $\mathbf{x} = (x_1,....,x_n)$, and we need to estimate $x_*$. With the help of the covariance function, we find the covariance of each data point with each other and put it in a covariance matrix $K$. If the two data points are highly correlated, then the covariance between them will be of a high value. If two points are not so highly correlated, their covariance will be less.

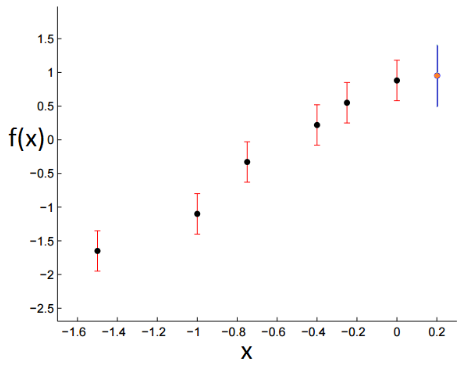

Figure 3: Given data points and interested estimate

From the figure above, we have $x_*$ and we want to find $f(x_*)$. In order to do this, we additionally find the covariance of our $x_*$ with all the data points within our data set. These covariances are also added to the covariance matrix. Hence the final covariance matrix is given as

$$\begin{bmatrix} \mathbf{x} \\ x_* \end{bmatrix} \sim \mathcal{N}(0, \begin{bmatrix} K & K_*^T \\ K_* & K_{**} \end{bmatrix}) \tag{3.75}$$

where $K_{**}$ is the variance of $x_*$. Also we assume that the mean of the Gaussian Process is zero everywhere. The conditional probability that we are interested in is $p(x_*|\mathbf{x})$. Having computed the $K$ matrix, we compute the mean and the variance of $x_*$ as

Best estimate for $x_*$ is the mean of this distribution:

$$\bar{x}_* = K_* K^{-1}\mathbf{x} \tag{3.76}$$

and the uncertainty of our estimate in its variance:

$$var(x_*) = K_{**} - K_* K^{-1} K_*^T \tag{3.77}$$

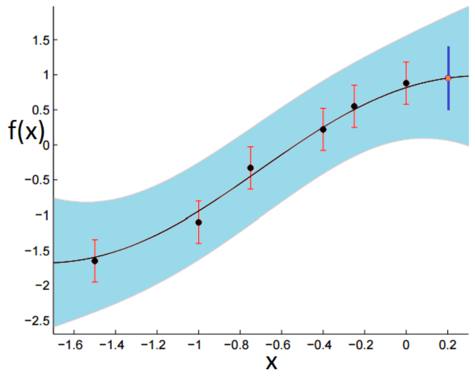

Figure 4: The solid line indicates an estimation for value of $x_*$ given $\mathbf{x}$. Pointwise 95 percent confidence intervals are shaded.

## 4 Numerical Experiments and Results

All simulations of SLAM has been implemented using Ipython notebooks. Two simulations have been done with two different path trajectories. In the first simulation the robot is made to go on a circular path. Landmarks are placed at selected points of the environment for the robot to detect and initialize. In the second simulation, path planning is implemented in the robot dynamics. The planning here is that if the robot encounters an obstacle within certain distance, it will turn 90 degrees on its axis counter clockwise.

In both simulations, boundaries have been added as obstacles in the environment to restrict the robot movement within a particular area. The robot in these simulations has been loaded with two sensors. The first sensor used is to detect landmarks within a selected radius around the robot. The second sensor used is a laser scanner to detect obstacles on it's path. The Gaussian Process kernel, used to map all obstacles in it's surrounding, does this by collecting training data from the laser scanner sensor. In this setup, the only obstacles will be the boundaries within which the robot moves. Also to make the simulations more realistic, zero mean gaussian white noise have been added to the odometry and measurement values.

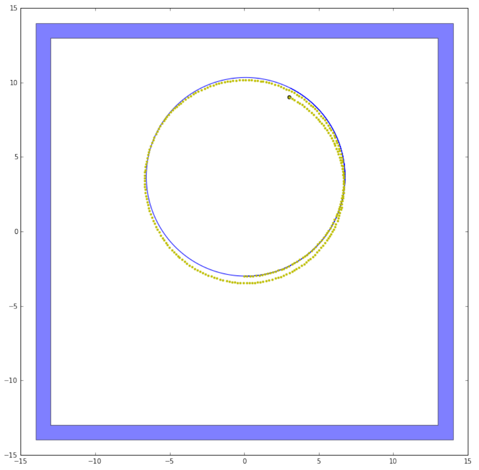

Figure 5: The robot moving in a circular path with no SLAM implemented.

As seen above, when the robot is made to move independently without any knowledge of its surroundings. Hence the robot simply propagates at each timestep and does not follow a circular path due to the odometry noice within its encoders. However after we implement SLAM, the robot is able to localize itself due to the presence of landmarks within its surroundings. This in turn will result in the robot moving in a circle in a more orderly fashion.

As seen above, the robot initially propagates itself on a circular path. However this time, it has the SLAM algorithm functioning. Hence it takes note of all the landmarks within its surroundings as it moves along its path. The position of the landmarks is given by the blue '+' symbols. Whenever the robot detects a landmark within its vicinity, to show it has been detected, we mark that landmark with a red '*' in simulation. For obstacle detection, the laser sensor leaves a red dot on obstacle which it has detected in simulation. This is why we see red dots on boundary walls within the simulation. When the robot completes a round, the robot revisits the same landmarks which has stored in its state vector. On encountering the landmarks again, the robot due to localization

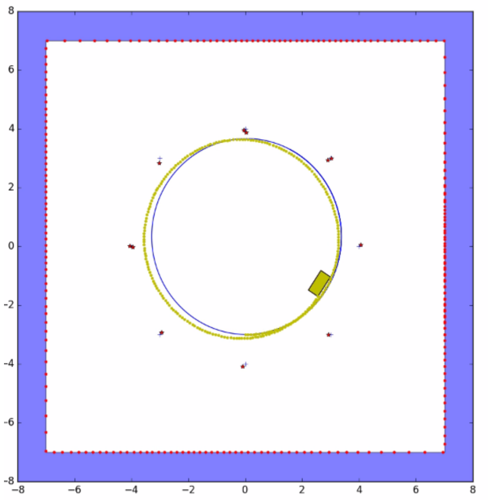

Figure 6: The robot moving in a circular path with SLAM implemented.

is able to tell where it is and will hence align itself on the original path that it once set.

The Mahalanobis thresholds chosen for the simulations were in the range of 10-90. However the thresholds needed to be tuned multiple times in order to gain the right values. On not choosing the right thresholds, the robot might fail to recognize old landmarks and may have to initialize them again. Another observation seen in both simulations. The more landmarks that are initialized, the better is the path trajectory followed by the robot. This would make sense as the presence of more landmarks would mean more points for localization for the robot.

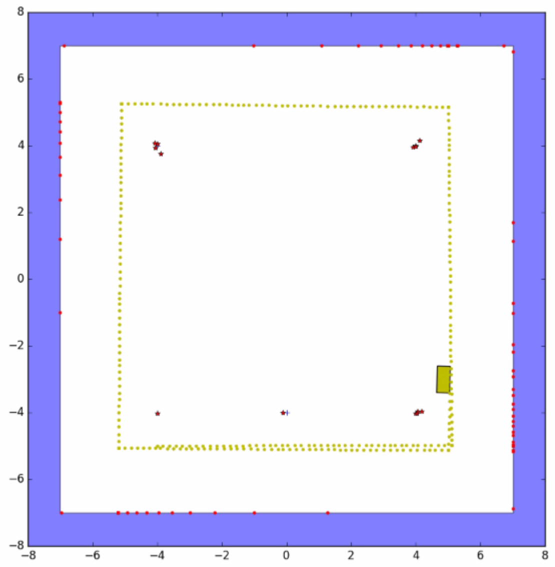

Figure 7: The robot moving in a rectangular path with path planning.

In figure 8, we see the two dimensional map of the obstacles created by the Gaussian Process kernel. In both simulations we see the same map, since the only obstacle in the environment is the surrounding boundary. The gaussian kernel is able to create this map based on the all the

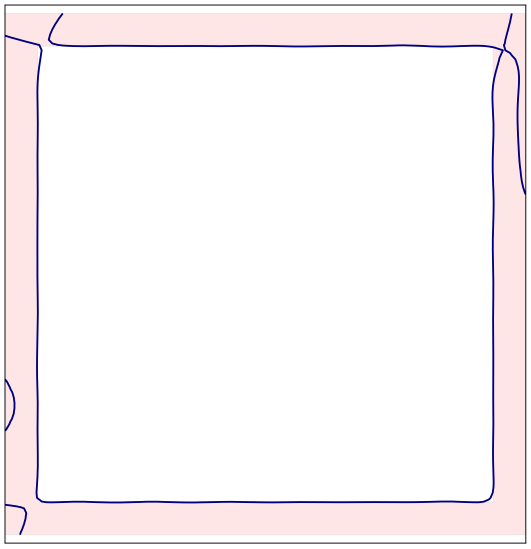

Figure 8: Map of obstacles with Gaussian Process kernel

points the laser scanner detected as the robot moved along its path. The gaussian kernel takes these detected points and using regression is able to estimate the shape of the obstacle.

## 5 Conclusions and Future work

In this project, a simple two-dimensional SLAM was implemented using point landmarks for different path trajectories. Additionally, using Gaussian processes, a two-dimensional map of all the surrounding obstacles was created. Based on the simulation results, we can conclude to say that SLAM was implemented successfully. Future works for this project would include using other methods for landmark detection like stochastic cloning and Delayed landmark initialization.

## References

[1] S. Roumeliotis, Class Notes, 2016

[2] Ebden, M, *"Gaussian Processes for Regression: A Quick Introduction"*, http://www.robots.ox.ac.uk/ mebden/reports/GPtutorial.pdf

[3] Rasmussen, C. and C. Williams (2006), *"Gaussian Processes for Machine Learning"*, MIT Press

[4] Sivia, D. and J. Skilling (2006), *"Data Analysis: A Bayesian Tutorial"* (second ed.), Oxford Science Publications