NYU - RESEARCH TRACK EXPLORATION (RTE) PROGRAM

IMPLEMENTATION OF SLAM ALGORITHM FOR UNDERWATER ROBOTICS

July 25, 2016

Mayank Mittal
Department of Electrical Engineering
I.I.T. Kanpur

Mentor: Prof. Farshad Khorrami
Tandon School of Engineering
New York University

Contents

I Introduction

With the recent advancements in Autonomous Underwater Vehicles (AUVs), their applications has expanded to a variety of tasks such data collection in marine environments, detecting underwater mines, conducting oceanic surveys as well as mapping seafloors.

However, for any mobile robot to be completely autonomous it is important for the robot to know it's position in the given environment.This is often referred to as localization of the robot. Unfortunately as the case normally is, for a localizing a robot it needs to know the map of the environment, which might always be readily available. This leads to so called *'chicken-or-egg'* problem. The solution to is lies in the Simultaneous Localization and Mapping (SLAM) technique. Several algorithms are present to implement SLAM, however, what makes it difficult to use SLAM for an AUV is the rapid attenuation of GPS and radio frequency signals underwater resulting in a difficulty to localizing the pose of the vehicle accurately.

The aim of the project was to explore the possibility of SLAM implementation for underwater robotics. Considering the massiveness of the problem, the project was limited to implementing Kalman Filter SLAM in a simple virtual environment, while studying data fusion of Inertial Navigation System for an AUV to improve its odometry estimates.

II SLAM Problem

Simultaneous Localization and Mapping (SLAM) is the basis of most navigation problems. It is considered a fundamental problem for truly autonomous robots. Two major models exist for solving the SLAM problem: Full SLAM and Online SLAM. The latter is easier to solve than the former by taking the complete state assumption (also called Markov assumption) which postulates that the past and future data are independent if current state x_t is known.

SLAM consists of several parts such as landmark extraction, data association, state estimation, state update and landmark update. The front-end problem comprises of landmark extraction and data association, while the other three are included in the back-end problem [9].

Throughout the period of the project the following three paradigms for solving a SLAM problem were studied with an emphasis on the back-end implementation of them:

1. Kalman Filter 2. Particle Filter 3. Graph-Based

Common to each of the above paradigms is the notion of motion and observation models.

(a) **Motion Model:** It specifies the posterior probability that action u carries the robot from pose x_t to x_{t+1} i.e. $p(x_t|u_t, x_{t-1})$. It can be odometry based or velocity based model.

(b) **Observation Model:** It specifies the posterior probability of the observations given a particular state i.e $p(z_t|x_t)$. It can be thought of as a noisy projection of the state and the model depends upon the type of sensors being used.

Although the study involved a detailed mathematical review of the various algorithms [10] available for the three paradigms mentioned above, only a comparative analysis of each of the algorithms is given in Table 1.

Table 1: Various State Estimators used in SLAM Algorithms

SLAM Method	**Description**
Kalman Filter (KF)	State distribution assumed to be Gaussian characterized by mean μ and covariance Σ. Optimal solutions for linear model and Gaussian distributions
Extended Kalman Filter (EKF)	Extension of Kalman filter to solve for non-linear systems by carrying out local linearization. Large uncertainty leads to increased approximation error. Prediction operation is fast, while correction step is slow. Cost per step dominated by number of landmarks $\mathcal{O}(n^2)$
Unscented Kalman Filter (UKF)	Provides better approximation than EKF for non-linear models. Instead of mapping (μ, Σ) through the non-linear functions, multiple âĂŸsigmaâĂŹ points are chosen, which are transformed and then re-parametrized as Gaussian from assigned weights.
Sparse Extended Information Filter (SEIF)	Uses canonical representation of belief $\mathbf{\Omega} = \mathbf{\Sigma}^{-1}$, $\xi = \mathbf{\Sigma}^{-1}\mu$. Concept of active and passive landmarks ensures sparsification of matrix.Requires linear memory complexity. Prediction step can be slow, however correction step is fast.
Particle Filter (PF)	Uses random samples rather than parametric models to represent distributions.Each discrete sample is assigned a weight. More samples results in a better estimate at the cost of higher computation requirement.
Least Squares (LS)	Through least square optimization it obtains maximum *a posteriori* state estimate. Has the advantage as past states are stored which can be used for full trajectory optimization later.
Graph-based	Uses graph to represent the problem where node represents pose of robot during mapping while edge represents spatial constraints between two nodes obtained from observations of environment or odometry data.

III Kalman Filter (KF)

Kalman filter is one of the most useful estimation tools available to obtain an *optimal* estimate of the desired quantities from data obtained in a noisy environment. It follows a recursive data processing algorithm, which estimates the current value of variables of interest by processing the following:

- knowledge of the system and measurement device dynamics
- statistical description of system noises, errors and uncertainty in dynamic models
- available information about the variable's initial conditions

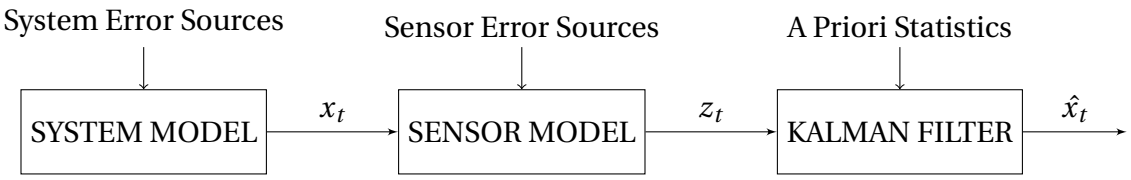


Figure 1: A Model of the estimator

One reason for the gain of popularity of Kalman Filtering is its easy application to linear systems. It is considered one of the best linear unbiased filter under the assumption of linearity of the system along with modeling the system noise as white Gaussian in nature.

Consider a discrete time-varying linear system, for which the state space equations are as follows [1]:

$$x_t = \boldsymbol{A_t} x_{t-1} + \boldsymbol{B_t} u_t + \epsilon_t$$
$$z_t = \boldsymbol{C_t} x_{t-1} + \delta_t$$

where ϵ_t and δ_t are random processes which are independent and normally distributed with covariance R_t and Q_t respectively.

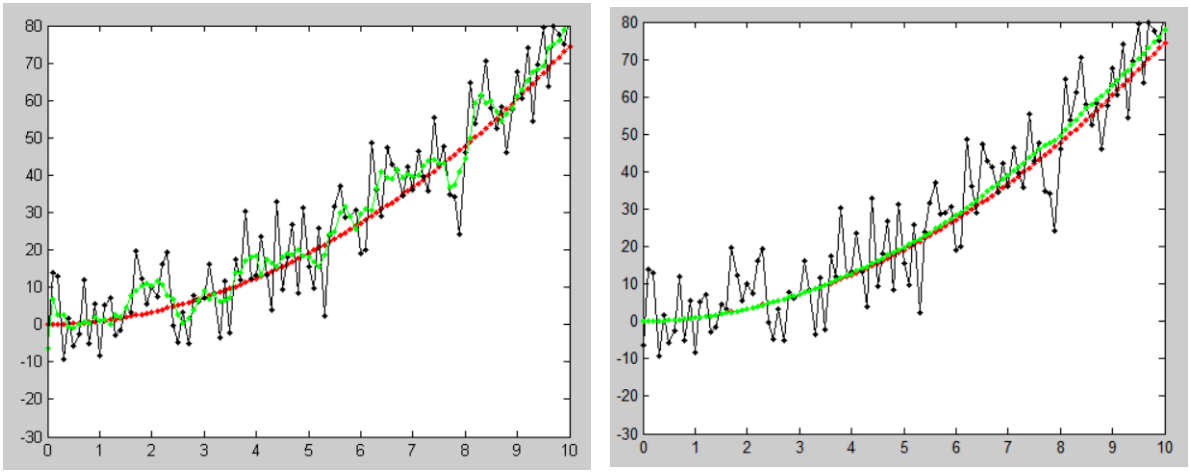
For the above system the Kalman filter equations can be summarized as follows [10]:

Algorithm 1 Kalman Filter Algorithm

procedure $KF\_Algo(\mu_{t-1}, \Sigma_{t-1}, u_t, z_t)$
  Stage one: prediction
    $\bar{\mu}_t = \boldsymbol{A_t}\mu_{t-1} + \boldsymbol{B_t}u_t$
    $\bar{\boldsymbol{\Sigma}}_{\boldsymbol{t}} = \boldsymbol{A_t}\boldsymbol{\Sigma_{t-1}}\boldsymbol{A_t^T} + \boldsymbol{R_t}$
  Stage two: correction
    $\boldsymbol{K_t} = \bar{\boldsymbol{\Sigma}}_{\boldsymbol{t}}\boldsymbol{C_t^T}(\boldsymbol{C_t}\bar{\boldsymbol{\Sigma}}_{\boldsymbol{t}}\boldsymbol{C_t^T} + \boldsymbol{Q_t})^{-1}$
    $\mu_t = \bar{\mu}_t + \boldsymbol{K_t}(z_t - \boldsymbol{C_t}\bar{\mu}_t)$
    $\boldsymbol{\Sigma_t} = (\boldsymbol{I} - \boldsymbol{K_t}\boldsymbol{C_t})\bar{\boldsymbol{\Sigma}}_{\boldsymbol{t}}$
return μ_t, $\boldsymbol{\Sigma_t}$

Kalman filter finds itself applicable in a variety of fields such as in tracking objects, economics, navigation as well as in computer vision (stabilizing dept measurements, feature tracking, fusing data from different sensors etc.)

Example: An example of implementation of Kalman Filter for a 1-D tracking problem was ran to develop a better understanding of the estimator. The problem and the code used can be found in [11].



(a) Without Kalman Filter **(b)** With Kalman Filter

Figure 2: Kalman Filter in 1-Dimension

The black line plot shows the visual data of the position of the quail observed by the observer. The red line plot is true position of the quail in a global reference frame. The green line plot is odometry estimate made by the observer.

IV Application to an AUV

In mobile robotics, position estimate of the robot is called dead reckoning. It is achieved by integrating odometry measurements of the robot. Kalman filter acts as a sophisticated tool to fuse the inputs to the robot (commands) with the measurements from its odometry sensors. Consider a mobile robot which state is defined by $\boldsymbol{x} = [x_r, v_r]$ where x_r and v_r are the robot position and velocity respectively; u denotes the real-valued force applied to the robot.

For an autonomous underwater vehicle (AUV), in a global frame $x_r = [x, y, z, \theta]$ and $v_r = \dot{x_r} = [u, v, r, \Theta]$.The dynamic model, as given by Wang 2006, comprises of decoupled motions, quadratic terms and considering no tether effects, i.e.$x_{t+1} = f(x_t, u_{t=1})$.

IV.1 Sensor Suite

The most common sensor technology used in almost all AUV these days are:

- **Inertial/ dead reckoning**: Compass, DVL, IMU, Pressure Sensor
- **Optical**: Cameras
- **Acoustic**: imaging SONAR, ranging SONAR

Table 2: Dead-Reckoning Sensors

Sensor	Description
Inertial Measurement Unit (IMU)	Comprises of Gyroscope which measures angular rates and accelerometer which measures the force required to accelerate a proof mass. The two (along with magnetometer) helps in estimating the vehicle's orientation, and velocity, however are plagued by drift and biasing errors.
Compass	It provides a globally bounded heading reference by measuring the magnetic field vector. It is subject to bias in the presence of objects with strong magnetic signature
Pressure Sensor	Used to measure underwater depth. It can attain high accuracy as pressure gradient is steeper underwater.
Doppler Velocity Log (DVL)	Uses acoustic measurements (doppler effect) to capture bottom tracking in order to determining the AUV's velocity vector (heave, surge and sway) relative to the seafloor. DVL consists of typically 4 beams.

IV.2 Model for Inertial System

A basic kinematic model for dead-reckoning pose estimation [5] can be given by following equations:

$$\dot{x} = v cos\psi + w sin\psi$$
$$\dot{y} = v sin\psi + w cos\psi$$
$$\dot{\psi} = 0$$

where the heading ψ and velocities (v, w) can be obtained using a compass and DVL respectively. However, the above model doesn't account for water current.

A common problem with inertial system is that they drift over time. An inertial system aims to minimize this error by considering drift as part of the state space as discussed in [4]. Sensor data fusion using Unscented Kalman Filter for underwater navigation has been described in detail in [3]. Further improvements to INS navigation can be done by making appropriate modifications in the kinematic model shown above.

V Robot Simulation for EKF SLAM

Robot simulator helps in implementing algorithms without the requirement of a physical robot, thus saving time and cost. For the purpose of this project Virtual Robot Experimentation Platform (V-REP 3.3.1) was used. V-REP, by Coppelia Robotics, allows fast robot prototyping and provides a variety of models for dynamics. Moreover, it supports bindings with a number of programming languages such as Python, C++, MATLAB through it's remote API calls.

Figure 3: V-REP Logo

While the tutorials are provided by Coppelia Robotics [7], they are insufficient for a beginner to get a grip over the software. Instead there is a tutorial series [**?**] available which builds on V-REP effectively. More understanding of writing scripts can be done by reading those available for various simulation examples available with the software.

With the additional plugin for Open Motion Planning Libraries (OMPL), it is easier to implement path planning algorithm as required for this project.

V.1 Setup

- **Robot used:** Pioneer 3DX
- **Motion model:** Velocity-based differential drive
- **Sensor:** Range bearing sensor
- Uses a go_to_goal controller with an Incremental A* Algorithm for path planning
- Only a static environment is present i.e. moving objects haven't been accounted
- Detects walls as landmark features

V.2 SLAM Process

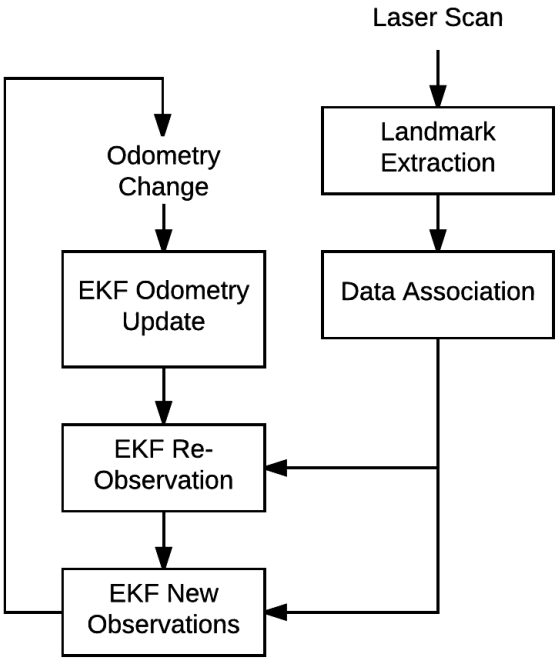


Figure 4: Flow process for EKF SLAM [6]

V.3 Result of Simulation

The code and V-REP environment used for carrying out this simulation can be found in [8].

The red dot in Figure 6 shows the pose of the robot in the 2-D map, while the green dot is the goal to which the robot has been set to reach. The blue ones on the other hand are the nodes of the path solved using the incremental A* algorithm [2].

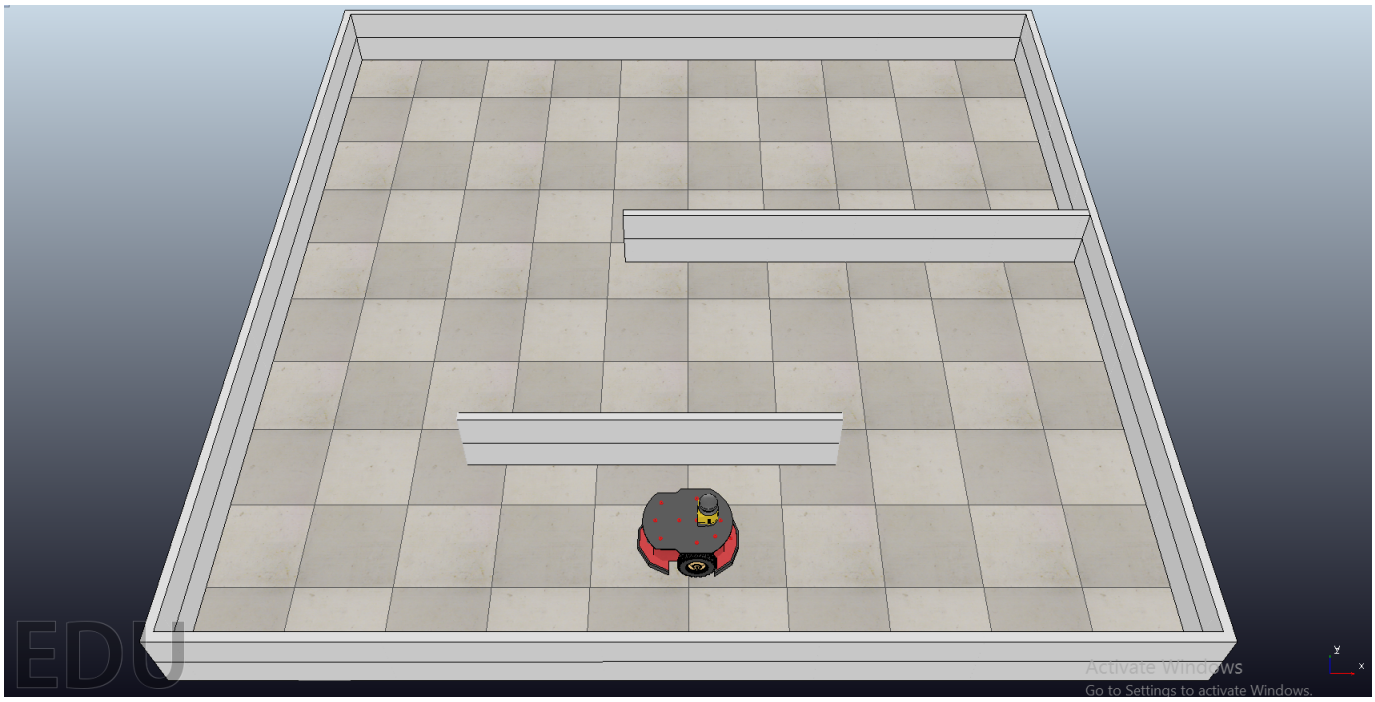


Figure 5: Screen-shot of V-REP Environment Used

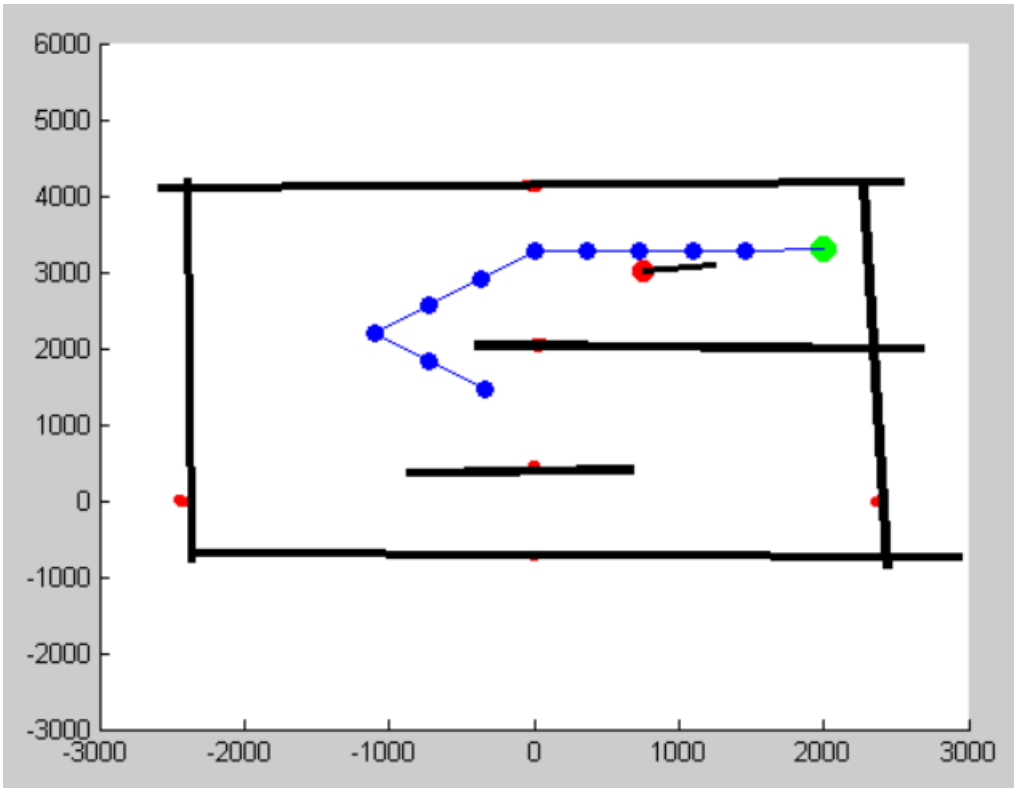


Figure 6: Map created using EKF SLAM on MATLAB

VI Final Remarks

Implementing the SLAM algorithm is a challenge in itself. While writing the scripts for the EKF SLAM implementation, solving the data association problem seemed tough considering the short period of the project. However, while searching a way out a number of methods were discovered such as Iterative Close Point (ICP) and Iterative Maximum Likelihood (IML). Although these techniques were studied, implementing them showed the difference between practical and theoretical knowledge.

Doing the above for an AUV requires a lot of deeper study into the dynamics of AUV, and the SLAM algorithms. While implementation EKF SLAM performed rather slowly, as expected. With the current popular FastSLAM technique, the computation complexity of the problem can be reduced.

However, the major takeaway from this project, which is using Kalman Filter as an estimator for pose of a mobile robot, would supplement the project *Varun* (I.I.T. Kanpur's first AUV underdevelopment). Although it is not currently equipped with a Doppler Velocity Log, the progress made through this project would be beneficial for the future years to come, and would allow experimental tests to be run on it as well.

Bibliography

[1] Mohinder S. Grewal and Angus P. Andrews. *Kalman filtering : theory and practice using MATLAB*. Wiley, New York, Chicester, Weinheim, 2001. La p. de t. porte en plus : A Wiley-Interscience publication.

[2] Sven Koenig and Maxim Likhachev. Incremental a*. In *NIPS*, pages 1539–1546, 2001.

[3] P Krishnamurthy and Farshad Khorrami. A self-aligning underwater navigation system based on fusion of multiple sensors including dvl and imu. In *Control Conference (ASCC), 2013 9th Asian*, pages 1–6. IEEE, 2013.

[4] Y. Zhao P. Miller, J. Farrell and V. Djapic. âĂIJautonomous underwater vehicle navigation,âĂİ. *Oceanic Engineering, IEEE Journal of*, 35(3):663–678, 2010.

[5] Liam Paull, Sajad Saeedi, Mae Seto, and Howard Li. Auv navigation and localization: A review. *IEEE Journal of Oceanic Engineering*, 39(1):131–149, 2014.

[6] SÃÿren Riisgaard and Morten Rufus Blas. Slam for dummies a tutorial approach to simultaneous localization and mapping by the âĂŸdummiesâĂŹ.

[7] Coppelia Robotics. Virtual robot experimental platform user manual.

[8] Bijo Sebastian. The slam diary.

[9] Cyrill Stachniss. Robot mapping.

[10] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic robotics*. Intelligent robotics and autonomous agents. the MIT Press, Cambridge (Mass.) (London), 2005.

[11] Student Dave's Tutorials. Kalman filter with matlab.