Visual-Inertial SLAM : An academic project

Orish Jindal
Department of Electrical Computer Engineering
University of California, San Diego
ojindal@ucsd.edu

I. *Abstract*— Autonomous vehicles are equipped with sensors that are used to collect information about the motion and surroundings of the vehicle. With the help of the information collected, our goal is to build a virtual map of the surroundings while simultaneously locating the vehicle in that map. A popular technique which is used for this task is called SLAM: Simultaneous Localization and Mapping. This paper will focus on one of the approaches to solve the SLAM problem for a moving car and the advantages as well as disadvantages of the given implementation.

Index Terms— SLAM, Kalman Filter.

II. IMPORTANCE OF SLAM

Development of mobile robots and autonomous vehicles is becoming more and more popular as to replace the humans for the tasks that are monotonous or dangerous. Primary function that these machines need is to navigate in an unknown environment without the help of human. A solution to this problem falls into the family of problems called SLAM. There are various methods in this family such as particle filter, extended Kalman filter, covariance intersection, and GraphSLAM, that are popular for specific type of problems of navigation and creation of a map for any unknown or changing space where a map is either not available or needs to be updated in real-time.

III. PROBLEM FORMULATION

SLAM is basically a parameter estimation problem for $\mathbf{x_{0:T}}$ (position at time T) and **m (**map of surroundings) given $\mathbf{u_{0:T-1}}$ (control input at time T-1) and observations $\mathbf{z_{0:T}}$ (observations at time T).

There are two parameters that needs to be estimated at each interval: Position of the robot with respect to surroundings and Map of the surroundings in which the robot is moving. They both need the existence of each other to estimate them, so SLAM becomes a chicken - egg problem where it is difficult to keep track of both the parameters simultaneously.

Technically, we need the robot pose to build the map with the help of mapping sensors like stereo camera and again need the map information to estimate the robot pose at any time.

The equation of estimating the position of the robot with highest probability at some time T given the information of all the sensors before that time is as follows:

$$p(\mathbf{x}_{0:T}, \mathbf{m}, \mathbf{z}_{0:T}, \mathbf{u}_{0:T-1}) = \underbrace{p_0(\mathbf{x}_0, \mathbf{m})}_{\text{prior}} \prod_{t=0}^{T} \underbrace{p_h(\mathbf{z}_t \mid \mathbf{x}_t, \mathbf{m})}_{\text{observation model}} \prod_{t=1}^{T} \underbrace{p_f(\mathbf{x}_t \mid \mathbf{x}_{t-1}, \mathbf{u}_{t-1})}_{\text{motion model}} \prod_{t=0}^{T-1} \underbrace{p(\mathbf{u}_t \mid \mathbf{x}_t)}_{\text{control policy}}$$

where the joint pdf is formed using Bayes rule, and Markov assumption properties given the deterministic observation and motion models throughout. For solving this problem, We need to estimate two equations at each time

First: Motion Model

Robot's estimated position for a discrete time interval dataset at each time as a function of its previous position and control input subject to the motion noise is written as follows:

$$\mathbf{x}_{t+1} = f(\mathbf{x}_t, \mathbf{u}_t, \mathbf{w}_t)$$

Second: Observation Model

The observations as a function of robot's position at time t, as a function of its position, map of the environment subject to the measurement noise is as follows:

$$\mathbf{z}_t = h(\mathbf{x}_t, \mathbf{m}_t, \mathbf{v}_t)$$

Specific to the problem discussed in this report:

The project discussed in this paper is a three-step problem formulation written below:

A. IMU Localization

The task is to predict the trajectory of the car using the values of the velocity and angular velocity in all the 3 dimensions from the dataset of the IMU sensor.

B. *Landmark Mapping*

The task here is to estimate the coordinates of all the observations in the world frame using the 2D coordinates given in left and right camera of the stereo camera of each feature using the trajectory information calculated in IMU localization.

C. *Visual-inertial SLAM*

The task here is to simultaneously localize the car using IMU data and map the landmarks using the data of camera sensor.

IV. TECHNICAL APPROACH: EXTENDED KALMAN FILTER (EKF)

The goal of the project that is discussed in this paper is to use a visual-inertial simultaneous localization and mapping (SLAM) using an extended Kalman filter (EKF) in Python with given the data of the synchronized measurements from an inertial measurement unit (IMU) and a stereo camera as well as the intrinsic camera calibration and the extrinsic calibration between the two sensors. Where EKF is an extension to the typical Kalman filter approach applied to the models that does not satisfies the conditions of the typical one.

Dataset:

We are given datasets of the readings from two sensors IMU and Stereo Camera. We are given two datasets in '.npz' format files. Each of them corresponds to the different motion and visuals of a car. In these files, a master synced data of both the sensors is given for each time stamp. The data given by sensors is as follows:

IMU:

3D velocity and 3D angular velocity at each timestamp.

Stereo Camera:

2D position of all the pixels at all timestamps in both: left and right camera along with the distance between two cameras.

Also, intrinsic camera calibration and the extrinsic calibration between the two sensors is included in the files as well.

This complete problem is divided into three steps:

Step 1 – IMU Localization via EKF Prediction.

In this step, the readings of the IMU sensors are assumed true and the help of the camera sensor (observation model) for correction at each step, was not taken. That way, coordinates of the car can be calculated at each timestamp by forming rotation matrices using the data and multiplying it with the homogenized coordinates of the car at each time step in an iterative manner. In this project, two plots were then formed to have a rough idea about how our later plots should look like. One of the two plots showed only the trajectory and the other showed the direction of velocities at some timestamps as we move along the trajectory. This can also be done using quaternions, but our project is limited to the method of rotation matrices only for the sake of simplicity as there is no possibility of a gimble lock in a car that had not met with an accident.

The intermediate step involves the calculation of hat map and curly hat map of the 6D vector- control input u_t which is calculated as follows:

$$\text{where } \mathbf{u}_t := \begin{bmatrix} \mathbf{v}_t \\ \boldsymbol{\omega}_t \end{bmatrix} \in \mathbb{R}^6 \quad \hat{\mathbf{u}}_t := \begin{bmatrix} \hat{\boldsymbol{\omega}}_t & \mathbf{v}_t \\ \mathbf{0}^\top & 0 \end{bmatrix} \in \mathbb{R}^{4\times 4} \quad \hat{\hat{\mathbf{u}}}_t := \begin{bmatrix} \hat{\boldsymbol{\omega}}_t & \hat{\mathbf{v}}_t \\ 0 & \hat{\boldsymbol{\omega}}_t \end{bmatrix} \in \mathbb{R}^{6\times 6} \tag{27}$$

v_t and w_t represents the linear and angular velocities respectively.

These equations were further used to calculate the mean and variance of the prediction step at each discretized timestamp with added gaussian noise which in this first step was assumed to be zero for dead reckoning. The formular are as follows:

Motion Model: nominal kinematics of $\boldsymbol{\mu}_{t|t}$ and perturbation kinematics of $\delta\boldsymbol{\mu}_{t|t}$ with time discretization τ_t:

$$\boldsymbol{\mu}_{t+1|t} = \boldsymbol{\mu}_{t|t} \exp(\tau_t \hat{\mathbf{u}}_t)$$
$$\delta\boldsymbol{\mu}_{t+1|t} = \exp\left(-\tau_t \hat{\hat{\mathbf{u}}}_t\right) \delta\boldsymbol{\mu}_{t|t} + \mathbf{w}_t$$

EKF Prediction Step with $\mathbf{w}_t \sim \mathcal{N}(0, W)$:

$$\boldsymbol{\mu}_{t+1|t} = \boldsymbol{\mu}_{t|t} \exp(\tau_t \hat{\mathbf{u}}_t)$$
$$\Sigma_{t+1|t} = \mathbb{E}[\delta\boldsymbol{\mu}_{t+1|t}\delta\boldsymbol{\mu}_{t+1|t}^\top] = \exp\left(-\tau\hat{\hat{\mathbf{u}}}_t\right)\Sigma_{t|t}\exp\left(-\tau\hat{\hat{\mathbf{u}}}_t\right)^\top + W$$

Step 2 – Landmark Mapping via EKF Update.

In this step, the readings of the stereo camera are also considered to plot an estimated map of each feature as landmarks in the world frame.

In this, I calculated the mean and covariance of the landmark positions using the trajectory information calculated in the previous step. Three equations need to run simultaneously which are written below:

$$K_{t+1} = \Sigma_t H_{t+1}^\top \left(H_{t+1}\Sigma_t H_{t+1}^\top + I \otimes V\right)^{-1}$$
$$\boldsymbol{\mu}_{t+1} = \boldsymbol{\mu}_t + K_{t+1}(\mathbf{z}_{t+1} - \tilde{\mathbf{z}}_{t+1})$$
$$\Sigma_{t+1} = (I - K_{t+1}H_{t+1})\Sigma_t$$

The first one represents Kalman gain, the second one represents the predicted mean of observations and the third one represents the co-variance matrix for observation noise. The intermediate steps involve the calculation of two values represented below:

Predicted observations based on $\boldsymbol{\mu}_t$ and known correspondences Δ_{t+1}:

$$\tilde{\mathbf{z}}_{t+1,i} := K_s \pi\left(o T_I T_{t+1}^{-1} \underline{\boldsymbol{\mu}}_{t,j}\right) \in \mathbb{R}^4 \qquad \text{for } i = 1, \ldots, N_{t+1}$$

Jacobian of $\tilde{\mathbf{z}}_{t+1,i}$ with respect to $\mathbf{m}_j$ evaluated at $\boldsymbol{\mu}_{t,j}$:

$$H_{t+1,i,j} = \begin{cases} K_s \frac{d\pi}{d\mathbf{q}}\left(o T_I T_{t+1}^{-1} \underline{\boldsymbol{\mu}}_{t,j}\right) o T_I T_{t+1}^{-1} P^\top & \text{if } \Delta_t(j) = i, \\ \mathbf{0}, & \text{otherwise} \end{cases}$$

M matrix M was formed to convert the pixel feature coordinates in 2D for both the cameras into the 3D coordinates in the optical frame which were later to camera frame then IMU frame (which

is assumed to body frame) and finally to the world frame. The conversion of 2D to 3D is as follows:

$$\begin{bmatrix} u_L \\ v_L \\ u_R \\ v_R \end{bmatrix} = \underbrace{\begin{bmatrix} fs_u & 0 & c_u & 0 \\ 0 & fs_v & c_v & 0 \\ fs_u & 0 & c_u & -fs_u b \\ 0 & fs_v & c_v & 0 \end{bmatrix}}_{M} \frac{1}{z} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \qquad \begin{bmatrix} x \\ y \\ z \end{bmatrix} = {}_oR_r R^\top (\mathbf{m} - \mathbf{p})$$

Also, some mappings were used which are as follows:

Projection function and its derivative:

$$\pi(\mathbf{q}) := \frac{1}{q_3}\mathbf{q} \in \mathbb{R}^4 \qquad \frac{d\pi}{d\mathbf{q}}(\mathbf{q}) = \frac{1}{q_3}\begin{bmatrix} 1 & 0 & -\frac{q_1}{q_3} & 0 \\ 0 & 1 & -\frac{q_2}{q_3} & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & -\frac{q_4}{q_3} & 1 \end{bmatrix} \in \mathbb{R}^{4\times 4}$$

Where q is a 4D vector.

Step 3 – Visual-inertial SLAM.

In this step, I incorporated an update step immediately after the prediction step in order to enhance our estimation about the motion and simultaneously the map of the surroundings.

we again need to run three equations in loop at each discretized time immediately after the prediction steps.

$$K_{t+1} = \Sigma_{t+1|t} H_{t+1}^\top \left(H_{t+1} \Sigma_{t+1|t} H_{t+1}^\top + I \otimes V \right)^{-1}$$
$$\mu_{t+1|t+1} = \mu_{t+1|t} \exp\left((K_{t+1}(\mathbf{z}_{t+1} - \tilde{\mathbf{z}}_{t+1}))^\wedge \right)$$
$$\Sigma_{t+1|t+1} = (I - K_{t+1} H_{t+1}) \Sigma_{t+1|t}$$

The intermediate steps involve the calculation of two values represented below:

Predicted observation based on $\mu_{t+1|t}$ and known correspondences Δ_t:

$$\tilde{\mathbf{z}}_{t+1,i} := K_s \pi \left({}_oT_I \mu_{t+1|t}^{-1} \underline{\mathbf{m}}_j \right) \qquad \text{for } i = 1, \ldots, N_{t+1}$$

Jacobian of $\tilde{\mathbf{z}}_{t+1,i}$ with respect to T_{t+1} evaluated at $\mu_{t+1|t}$:

$$H_{t+1,i} = -K_s \frac{d\pi}{d\mathbf{q}} \left({}_oT_I \mu_{t+1|t}^{-1} \underline{\mathbf{m}}_j \right) {}_oT_I \left(\mu_{t+1|t}^{-1} \underline{\mathbf{m}}_j \right)^\odot \in \mathbb{R}^{4\times 6}$$

Also, the special operator used in the code are as follows:

where for homogeneous coordinates $\underline{\mathbf{s}} \in \mathbb{R}^4$ and $\hat{\boldsymbol{\xi}} \in \mathfrak{se}(3)$:

$$\hat{\boldsymbol{\xi}} \underline{\mathbf{s}} = \underline{\mathbf{s}}^\odot \boldsymbol{\xi} \qquad \begin{bmatrix} \mathbf{s} \\ 1 \end{bmatrix}^\odot := \begin{bmatrix} I & -\hat{\mathbf{s}} \\ 0 & 0 \end{bmatrix} \in \mathbb{R}^{4\times 6}$$

V. Results

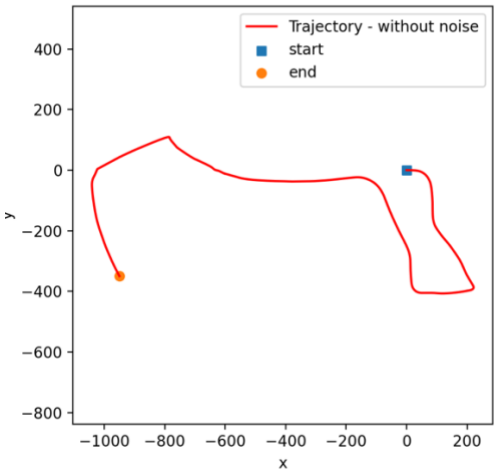
I have formulated a function that gives the features with reduced number of features equal the count that is passed as an argument to the function. The function basically ranks the features based on the number of times they were visible to the camera sensor of the car at each timestamp. Then it returns the desired number of top features. This step is used to reduce the time taken for calculations by slightly compromising on the results.

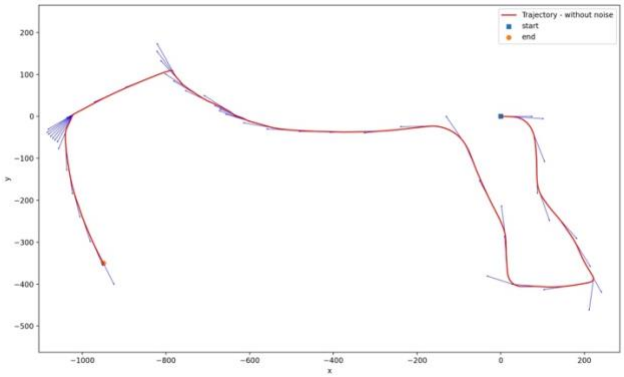
Here are the images shown that were plotted for all the three parts of the problem with both the datasets.

For **landmark mapping, the images shown are with the zero noise in motion model** to plot the dead reckoning of the map whereas in the **SLAM step, noise is added to both motion and observation model** and updated all the values by their comparison. So, in the later one, the trajectories are not as smooth as the trajectories without noise which makes proper sense in the real world.

For dataset '10.npz':

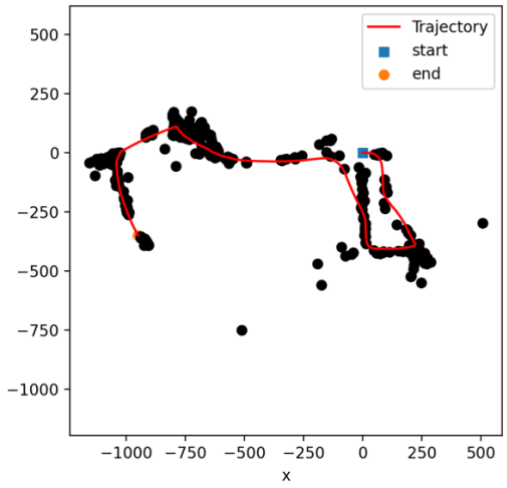
Localization step: showing two images, one of just the predicted trajectory assuming no noise and the other of the velocity directions as we move along the trajectory respectively.



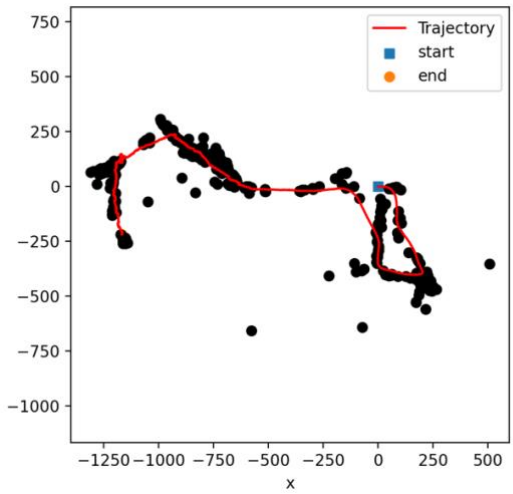


On 500 features:

Landmark Mapping

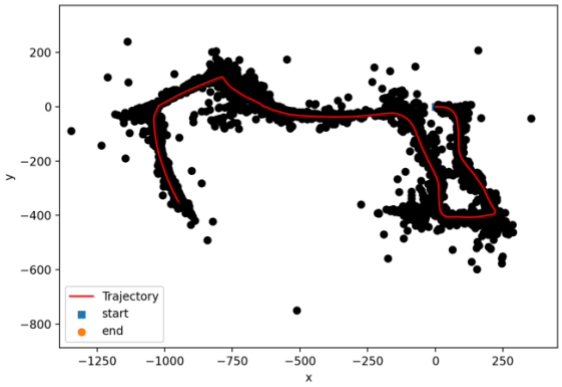


SLAM

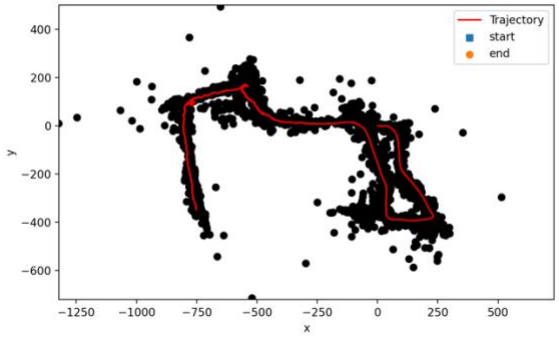


On 5000 features:

Landmark Mapping

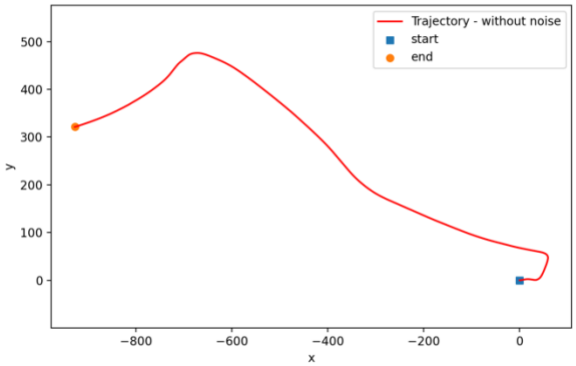


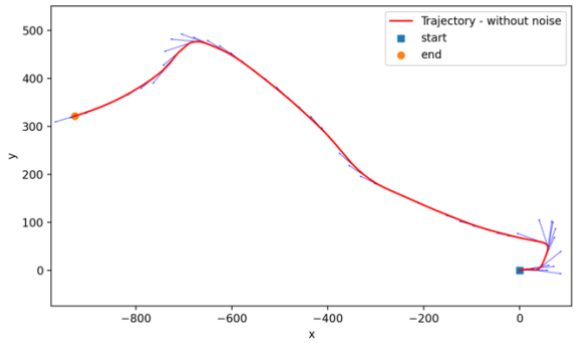
SLAM



For dataset '03.npz':

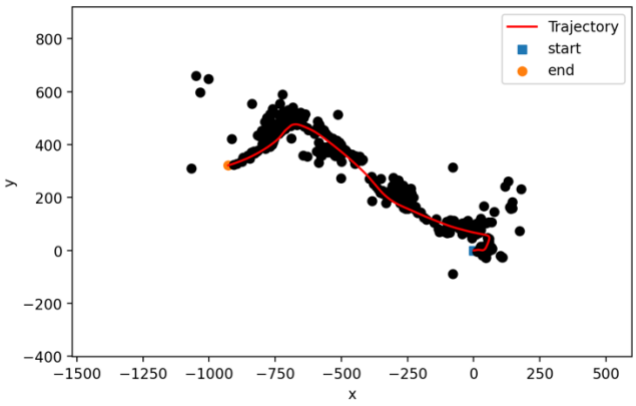
Localization step: showing two images, one of just the predicted trajectory assuming no noise and the other of the velocity directions as we move along the trajectory respectively.



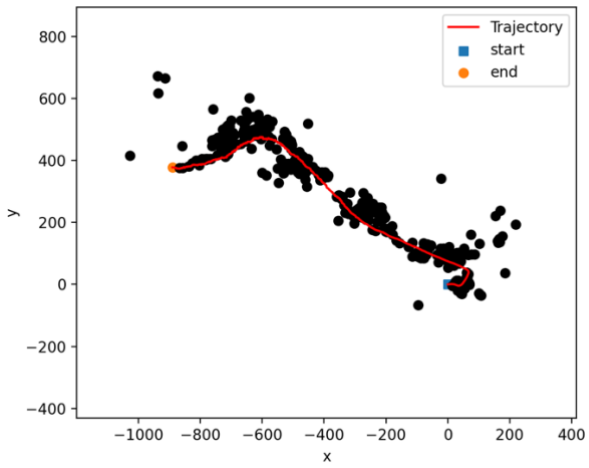


On 500 features:

Landmark Mapping

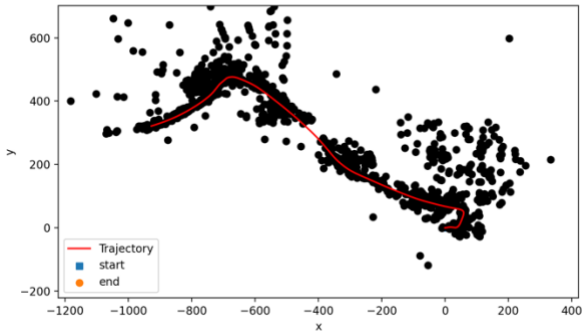


SLAM

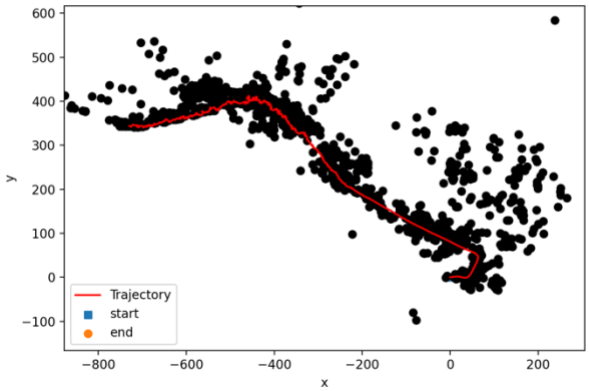


On 2000 features:

Landmark Mapping



SLAM



VI. Advantages and Disadvantages of EKF

This method has its own set of advantages and limitations as compared to other methods.

Advantages:

- It removes the impractical assumptions of the typical Kalman Filter i.e., motion model and observation model linearity.
- Fairly simple to implement as it works in the gaussian families of probability which have a very nice property of closure.
- Light in computation.

Limitations:

- Not compatible with higher dimensional systems (more that 3 dimensions).
- If the initial estimate of the state is wrong, or if the process is modeled incorrectly, the filter may quickly diverge.