UCSD Winter 2021 ECE276A: Project 3 Visual Inertial Simultaneous Localization and Mapping (SLAM)

Steven Wong
Machine Learning and Data Science
University of California, San Diego
San Diego, CA 92093
scw039@ucsd.edu

March 13, 2021

1 Introduction

In this project, we focus on Visual Inertial Simultaneous Localization and Mapping (SLAM). This is an important problem because it defines a way a robot can understand its current surroundings, allowing the observer to better gauge the robot's physical behaviors. The possibility of mapping a robot's environment while also localizing it is widely used in autonomous driving. Given feature extractions from images and an IMU that measures linear velocity and angular velocity, we are able to perform this algorithm in real time. In our approach, we perform a localization and mapping step separately before combining them into a SLAM problem. All of these steps invoke the extended Kalman Filter (EKF) predict and update step. The format of this paper will follow said road map.

2 Problem Formulation

In this section, we will dive into the problem formulation and the questions we should ask to solve the SLAM problem.
First, we will define our data and our variables. In the localization step, we make the following assumptions:

1. Linear velocity $v_t \in \mathbb{R}^3$ and angular velocity $\omega_t \in \mathbb{R}^3$ given through an IMU
2. The data association $\Delta_t : 1, ..., M \longrightarrow 1, ..., N_t$ is known.

The statement in (2) means that we have some underlying known function that maps each observation to its corresponding landmark. For example, if we see features x, y, z in a particular frame, we have the function Δ_t that maps those x, y, z observations to their corresponding landmark number. We can combine (1) into a single general velocity vector $u = [v_t^T, \omega_t^T] \in \mathbb{R}^6$.
To perform visual mapping, we make the following assumptions:

1. We are given the observations z_t as in localization.
2. The data association Δ_t is again known
3. The landmarks m_i are static so we do not consider a motion model and thus we do not need a prediction step.

Putting these together, we are able to solve the SLAM problem:

Problem: Maximize the data likelihood conditioned on the parameters such that

$$\boxed{\max_{x_{0:T},m} \log(p(z_{0:T}, u_{0:T-1}|x_{0:T}, m))}$$

Where z are the observations, u are the control inputs (IMU general velocity data), $x \in \mathbb{R}^{4x4xT}$ are the IMU poses, and $m \in \mathbb{R}^{3xM}$ is the current map.

3 Technical Approach

In this section, we will go into the technical approach of each of the three main sections mentioned above: localization, mapping, and finally SLAM.

3.1 Localization

To do localization only, we deploy the Extended Kalman Filter, in which we need to update the mean of the IMU pose and the covariance of the IMU pose each time step. We define tao as the difference between time step $t+1$ and t. The initialized covariance and the initialized pose is simply the identity matrix. In addition, we calculate the M, the camera calibration matrix, as follows:

$$M = \begin{bmatrix} fs_u & 0 & c_u & 0 \\ 0 & fs_v & c_v & 0 \\ fs_u & 0 & c_u & -fs_u b \\ 0 & fs_v & c_v & 0 \end{bmatrix} \tag{1}$$

From here, we solve for H. It is important to note that H spans the indices of i. This means that there will be an H for each landmark. We will stack these into a singular H matrix as follows:

$$H_{t+1} = \begin{bmatrix} H_{t+1,1} \\ ... \\ H_{t+1,N_{t+1}} \end{bmatrix} \tag{2}$$

The question remains to solve for H, which is the Jacobian of the observation model with respect to T_{t+1} evaluated at the mean pose $\mu_{t+1|t}$, and K, the Kalman Gain. We aim to solve for H_i and K:

$$\boxed{H_{t+1,i} = -M\frac{d\pi}{dq}({}_O T_I \mu^{-1}_{t+1|t} \underline{\mathrm{m}}_j)\, {}_O T_I (\mu^{-1}_{t+1|t} \underline{\mathrm{m}}_j)\odot} \tag{3}$$

$$\boxed{K_{t+1} = \Sigma_{t+1|t} H^T_{t+1|t} (H_{t+1|t} \Sigma_{t+1|t} H^T_{t+1|t} + I \otimes V)^{-1}} \tag{4}$$

Where M is the camera calibration matrix, T_I is the IMU to optical transformation, T is the IMU pose from the last time step, $\underline{\mathrm{m}}_j$ is the homogeneous world coordinates of observation j, and v is noise. i represents the landmark we are currently updating from 1 to N_{t+1}. The feature observations is a 4xN_t matrix of the following form:

$$\begin{bmatrix} u^1_l ... u^{N_t}_l \\ v^1_l ... v^{N_t}_l \\ u^1_r ... u^{N_t}_r \\ v^1_r ... v^{N_t}_r \end{bmatrix}$$

Where u_l, v_l are the pixel coordinates of the features in the left camera, and u_r, v_r are the pixel coordinates of the features in the right camera.
The goal is to use the above measurements to predict the pose of the IMU $T_t \in SE(3)$ over time. The problem then becomes to define the EKF prediction step and EKF update step for localization only. Let the prior distribution of the pose be a normal distribution with mean $\mu_{t|t}$ and covariance $\Sigma_{t|t}$ where $\mu_{t|t} \in SE(3)$ and $\Sigma_{t|t} \in \mathbb{R}^{6x6}$. We need a motion model to predict the next mean in the next time step. The nominal kinematics of $\mu_{t|t}$ and pertubation kinematics of $\delta\mu_{t|t}$ with time discretization is the following:

$$\mu_{t+1|t} = \mu_{t|t}(\tau * \hat{u})$$
$$\delta\mu_{t+1|t} = \exp(-\tau * \overset{\curlywedge}{u})\delta\mu_{t|t} + w_t$$

We denote $\hat{u}$, $\overset{\curlywedge}{u}$, and exp as the hatmap, adjoint, and the matrix exponential of the inputs. The EKF prediction step can then be condenced into two very straightforward equations:

$$\boxed{\mu_{t+1|t} = \mu_{t|t}(\tau * (\hat{u}))} \tag{5}$$

$$\boxed{\Sigma_{t+1|t} = \exp(-\tau * \overset{\curlywedge}{u_t})\Sigma_{t|t}\exp(-\tau * \overset{\curlywedge}{u_t})^T + W} \tag{6}$$

Using the following observation model, we can create the EKF update step for localization only:

$$z_{t+1,i} = M\pi(T_I T^{-1} m_j) + v_{t+1,i} \tag{7}$$

Using the observation model, we can them perform the EFK update by solving the following equations:

$$\boxed{\mu_{t+1|t+1} = \mu_{t+1|t}\exp((K_{t+1}(z_{t+1} - \tilde{z}_{t+1}))^\curlywedge)} \tag{8}$$

$$\boxed{\Sigma_{t+1|t+1} = (I - K_{t+1}H_{t+1})\Sigma_{t+1|t}} \tag{9}$$

The next problem is to deal with frames in which there are no features. This means there is nothing to update, so we simply skip this iteration and inherit the predicted IMU pose as that time step's updated pose. This concludes the technical approach of the localization problem. The formulas and edge cases have been taken care of up to this point. Next, we will dive into the technical approach to visual mapping.

3.2 Visual Mapping

To perform Visual Mapping, we once again deploy the extended Kalman Filter. Since we can skip the motion model (based off the assumption that all landmarks are static), we only need to use the same observation model as in the previous section. This allows us to generate the EKF Update equations for mapping:

$$\boxed{\mu_{t+1} = \mu_t + K_{t+1}(z_{t+1} - \tilde{z}_{t+1})} \tag{10}$$

$$\boxed{\Sigma_{t+1} = (I - K_{t+1}H_{t+1})\Sigma_t} \tag{11}$$

The question then remains to define H, which is the Jacobian of the observation model evaluated at μ_t, the mean position of the landmark, and K, the Kalman gain. H can will be defined as follows:

$$\boxed{H_{t+1,i,j} = M\frac{d\pi}{dq}({}_O T_I T^{-1}_{t+1|t}\underline{\mu}_{t,j})\,{}_O T_I T^{-1}_{t+1|t} P^T} \tag{12}$$

$$\boxed{K_{t+1} = \Sigma_t H_{t+1}^T (H_{t+1} \Sigma_t H_{t+1}^T + I \otimes V)^{-1}} \tag{13}$$

Where $P = [I, 0] \in \mathbb{R}^{3x4}$ is a projection matrix to convert from homogeneous to regular coordinates. Notice now we have i, j. The H matrix only has value when the data association function of j is equivalent to i. Simply put, if we observe landmark i, there will be value only in the columns in which the data association function maps i to j.
The final problem is to account for when we wish to update, and when we wish to initialize the landmark coordinates. For example, if we see landmarks a, b, c in a frame t, and in frame $t+1$ we see landmarks b, c, d, then we must send landmarks b, c through the update step, and intialize d. This is done by using a simple set different and set union function. This concludes the technical approach of the visual mapping problem. The formulas and edge cases have been taken care of up to this point. Next, we will dive into the technical approach to SLAM. With these equations defined, we are able to map the environment around the IMU.

3.3 SLAM

In slam we would like to do a predict and update step per time iteration simultaneously. Only this time, since we are doing SLAM, we do not actually know the IMU pose or the map landmark points. We are to use their means of their respective distributions as calculated in the last two sections. The equations very similar, with some key differences. The first, we combine the covariance of the IMU pose and landmark points. Combining the map and pose update steps into one captures the correlation between the landmarks positions and the robot pose, which hopefully improves the estimation quality.
We now will combine certain quantities together to show relationships, and then use the same EKF predict and update equations shown in the above section to execute SLAM. The following are the quantities we want put together:

1. We need to combine the Jacobian Matrices together.

$$H = [H_p, H_m] \in \mathbb{R}^{3M+6} \tag{14}$$

2. We need to combine the Covariance matrix together and initialize it as follows.

$$\Sigma = \begin{bmatrix} \Sigma_p & 0 \\ 0 & \Sigma_m \end{bmatrix} \in \mathbb{R}^{(3M+6)x(3M+6)} \tag{15}$$

3. We will then come up with a new perturbation term, which we call denote as $\delta\mu$

$$\delta\mu = K_{t+1}(z_{t+1} - \tilde{z}_{t+1}) \in \mathbb{R}^{3M+6} \tag{16}$$

The first six entries of this $\delta\mu$ will represent the pose perturbation, in which we utilize equation (8) to update. The next $3M$ will denote the perturbation of the landmark positions, in which we utilize equation (10) to update.

4. We will have to reorganize the covariance matrix in order to predict and update it. Let

$$\Sigma = \begin{bmatrix} \Sigma_p & \Sigma_{pm} \\ \Sigma_{mp} & \Sigma_m \end{bmatrix} \in \mathbb{R}^{(3M+6)x(3M+6)}$$

Using equation (6), to predict the next Σ, we define $A = \exp(-\tau * \hat{\dot{u}})$ and thus:

$$\begin{bmatrix} A & 0 \\ 0 & I \end{bmatrix} \begin{bmatrix} \Sigma_p & \Sigma_{pm} \\ \Sigma_{mp} & \Sigma_m \end{bmatrix} \begin{bmatrix} A^T & 0 \\ 0 & I \end{bmatrix} + \begin{bmatrix} W & 0 \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} A\Sigma_p A^T + W & A\Sigma_{pm} \\ \Sigma_{mp} A^T & \Sigma_m \end{bmatrix} \tag{17}$$

Once we have predicted the new Σ, we update it using same formulation in equation (11). The final problem, like in visual mapping and localization, is to keep track of which landmarks have been seen and which have not. SLAM accounts for both edge cases. We first predict the joint covariance and the IMU pose. If we see landmarks in need of updating, we will update the mean and covariance by the formulation above. If we see a new landmark, we will initialize the pose and map locations. If no features are seen, we will simply inherit the predicted IMU pose as that time step's updated pose. One can see that this is the combination of both edge cases mentioned in sections 3.1 and 3.2. This concludes the technical approach section. We will now dive into the results.

4 Results

In the results section, we will have four main points of interest: Dead reckoning trajectory and mapping, Localization only, Mapping only, and finally SLAM. Note that in these sections, we down sample the 13,289 features by a factor of 10 for computational efficiency.

4.1 Dead Reckoning Trajectory and Mapping

In dead reckoning trajectory, we do not account for seen and unseen landmarks. We simply just use the motion model described in equation (5) until the last time iteration. Likewise, in dead reckoning mapping, we simply plot the world coordinates of the observations.

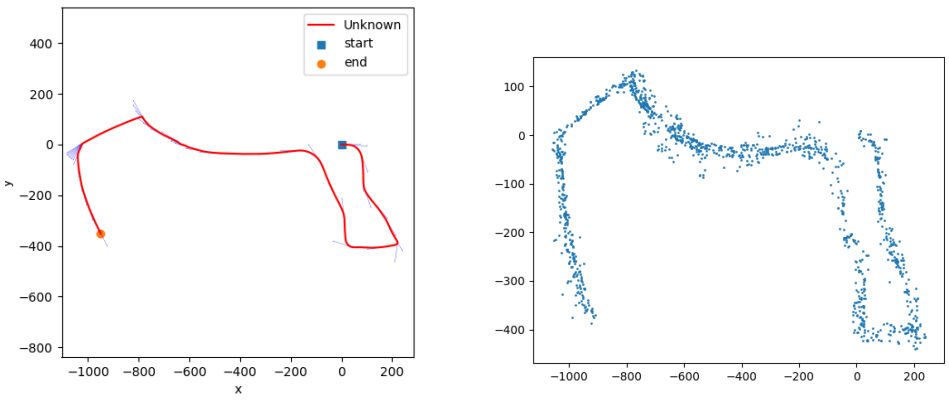

Figure 1: Dead Reckoning: Trajectory and Map

We can see here that the features line up fairly well with the trajectory, showing landmarks on both sides of the road.

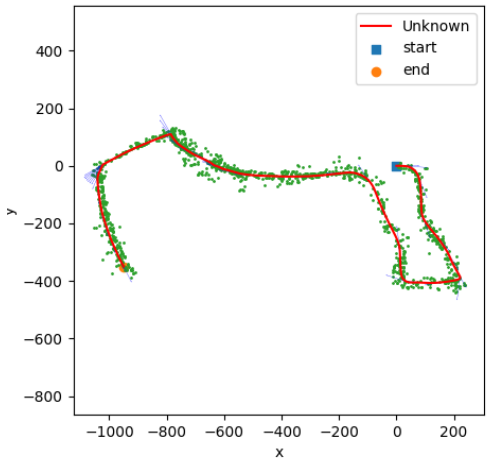

Figure 2: Dead Reckoning Both Trajectory and Map

4.2 Localization only

In localization only, we deploy the strategy described in the technical approach. Here we use a V noise of 7 pixels and a W of $5 * 10^{-5}$.

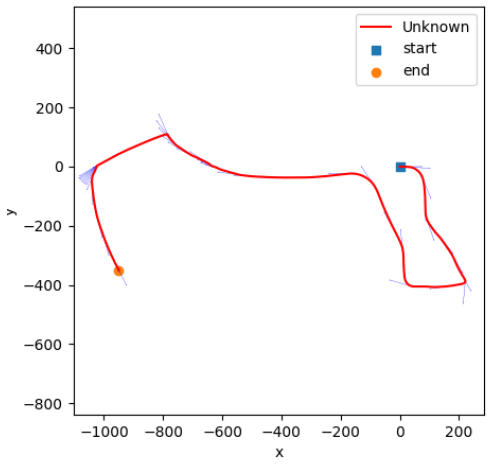

Figure 3: EKF Localization Only

The trajectory looks essentially identical to the dead reckoning, as in both cases we use the same equation (5) to update our IMU pose.

4.3 Visual Mapping only

In visual mapping only, we account for the cases in which landmarks are re-seen by the next frame, and update those while only initializing the landmarks we haven't seen. Here we use a V noise of 7 pixels.

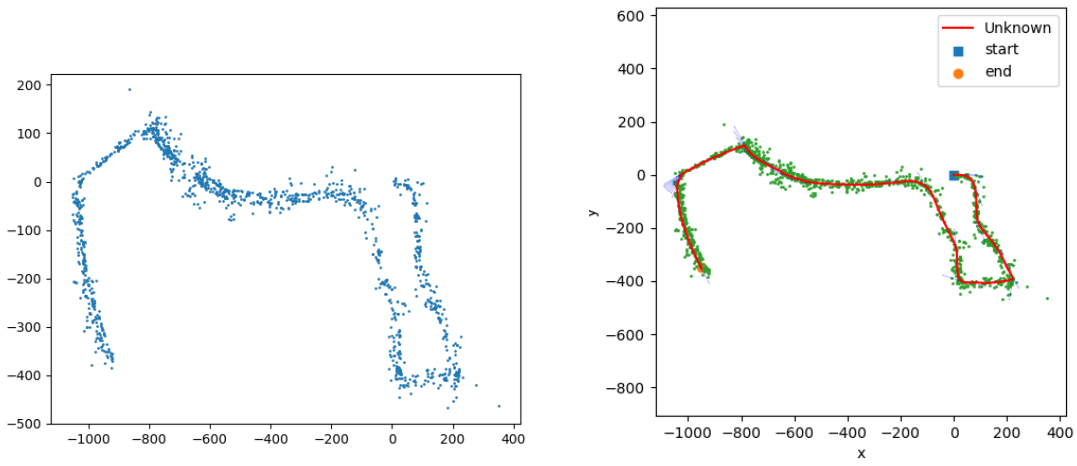

Figure 4: EKF Mapping Only

4.4 SLAM

In SLAM, we notice that the final results is extremely sensitive to noise input, particularly with W. Below we use a W of $5 * 10^{-4}$, and is our best achieved results. We will dive in closer to this result, before going into the comparative cases where the variability of W is changed.

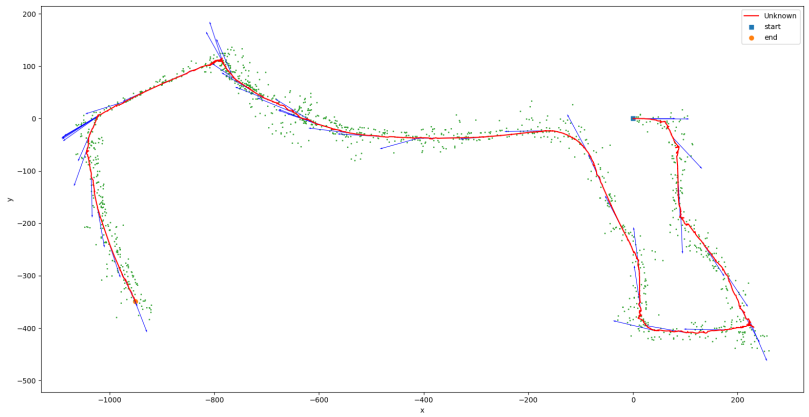

Figure 5: SLAM

The image is enlarged to show noise detail. Notice how the trajectory (red line) is more noisy than that of the dead reckoning path. Overall, the map points are reasonable enough to the left and right of the trajectory. For a close up, observe Figure 6.

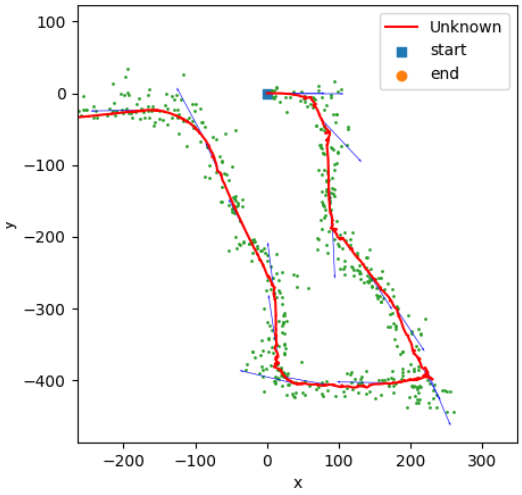

Figure 6: SLAM zoomed

We can see the full SLAM algorithm build in real time by observing Figure 7.

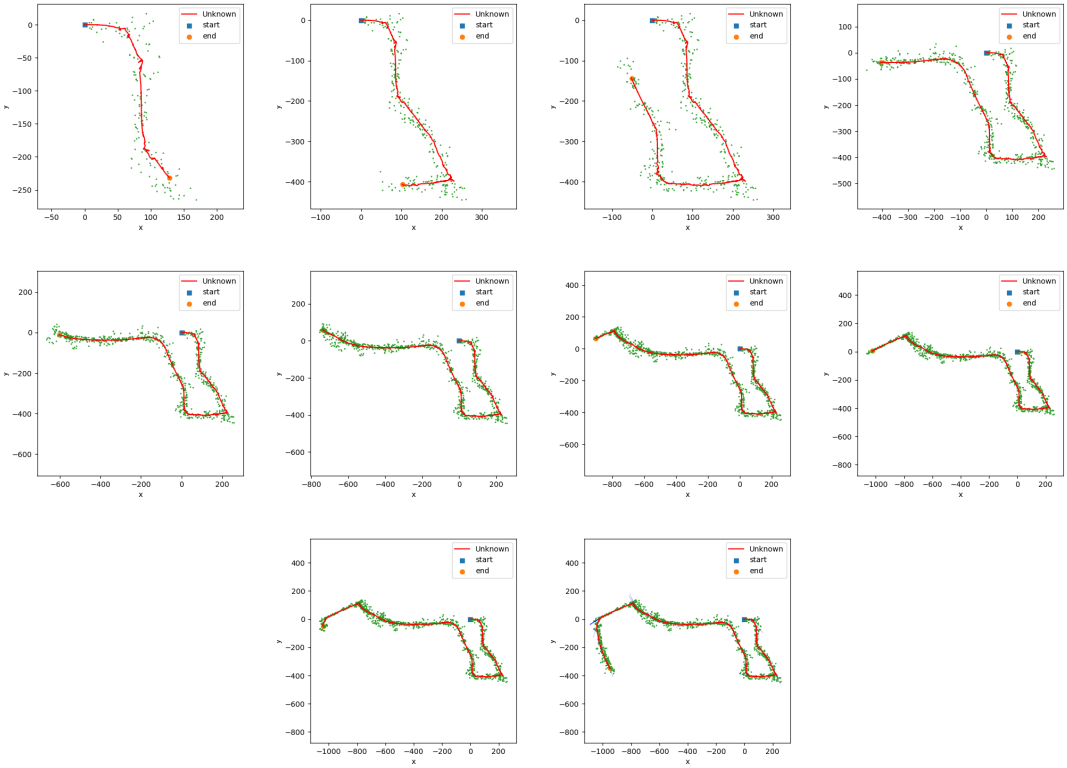

Figure 7: SLAM built in real time

We next show the results of when W is changing to show the SLAM algorithm's sensitivity to the motion noise. In specific, we choose two versions of W, each spaced off by a factor of 10, shown in Figure 8 and 9. These results are down sampled by 20 for computational efficiency. This is to show trajectory noise.

4.5 Conclusion/Discussion

It seems like our SLAM implementation was successful. In specific, we are able to both localize and map in real time. The issue comes with hyper parameter tuning. For further improvements, we would like to tune the hyper parameters more, that is, the noise variability in W and V. We believe that our method is sensitive to W because W directly influences our $\delta\mu$ calculation, which for the first six entries will affect the way we guess our next IMU pose. If the pose is far off, then it makes sense that the mapping would be far off as well, as the mapping in SLAM is directly related to the pose of the IMU. This is why we believe that W is the main contributing factor to the SLAM result. V is the observation noise, and through some experiments we found that the increase of V improves the results. This is because it created less noise in the map landmark locations, which can be inferred

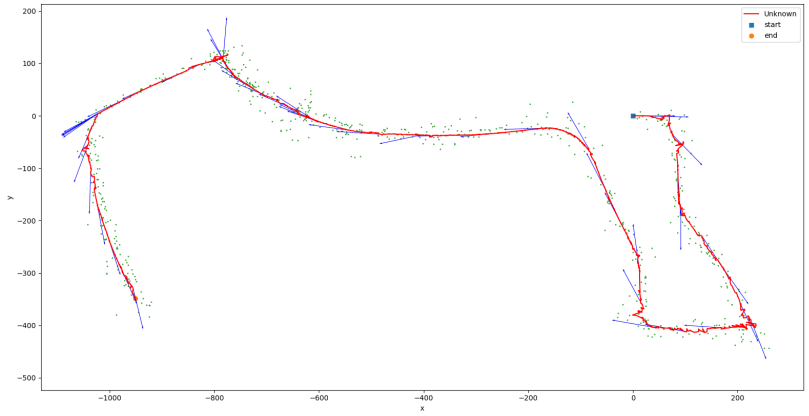

Figure 8: SLAM Noise ($W = 5 * 10^{-3}$)

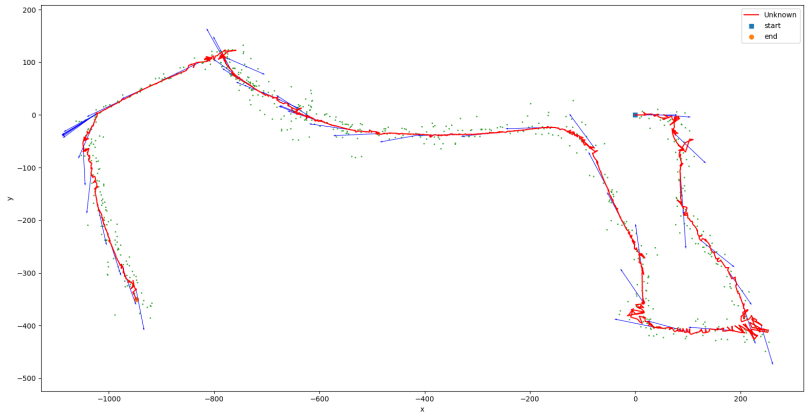

Figure 9: SLAM Noise ($W = 5 * 10^{-2}$)

in equation (11). Additional improvements include a dynamic array update for the Kalman Gain, H, and the covariances. Theses matrices in particular are extremely sparse, thus computing the inverse and doing the dot product is computationally expensive. By condensing these matrices into only the non-zero values of the indices we need to update, we can significantly reduce computation time. One can see the head start on this method in the get_patch_idx() and Kalman_gain_patch() functions in functions.py. This concludes the Visual Inertial SLAM algorithm summary for a moving vehicle.