ECE276A Project 3: Visual Inertial SLAM

Mustafa Shaikh
Department of Electrical and Computer Engineering
University of California, San Diego
La Jolla, U.S.A

I. Introduction

As the scope of applications for autonomous vehicles increases, it is crucial that they are able to operate in new environments and actively create a map of their surroundings. The problem of *simultaneous localization and mapping*, herein referred to as SLAM, is therefore an important problem to solve in order to safely and successfully deploy autonomous vehicles to city streets. SLAM refers to the problem of determining a vehicle's location in its environment, while at the same time creating a map of the environment and using the latest belief of the map state to re-calibrate our belief of the vehicle's pose [1]. Since there is error in the motion of the vehicle [1], as well as noise in the cameras and sensors used to create the map, a two stage iterative approach to the problem naturally follows. In this project we present the results of an Extended Kalman Filter (herein referred to as EKF) approach to the SLAM problem, in which we generate a map of landmarks seen by a vehicle using a stereo camera on board. We are provided with feature matching data obtained from the stereo camera on board the vehicle. Comparing with actual video of the vehicle driving through city streets, we are able to confirm that the trajectory and landmark map obtained from the EKF are quite accurate.

II. Problem Formulation

Consider a vehicle moving driving through city streets, i.e. moving in the xy-plane. The state of the vehicle at time t can be described by its pose x_t, a combination of its position - an (x,y) pair - and orientation, given by an angle θ in the world frame. The current estimate of the environment that the vehicle drives in can be represented by a map of landmarks, denoted by m_t. The motion of the vehicle can be thought of as caused by a set of control inputs u_t - in this case, linear velocity v_t and angular velocity ω_t - applied to the vehicle at every time step, through its engine and drivetrain system. The motion of the vehicle is governed by a discrete-time probabilistic kinematic model, herein referred to as the *motion model* [2]:

$$x_{t+1} = f(x_t, u_t, w_t) \sim p_f(\circ|x_t, u_t),\ w_t = \text{motion noise} \quad (1)$$

The Markov assumption allows us to express the predicted state of the robot in the next time step solely as a function of its current state and current control input, rather than conditioning on the entire history of its states. The presence of error in the motion of the vehicle naturally leads to the probabilistic model seen above.

At each time t, the vehicle captures information about its environment using an on-board stereo camera. These measurements z_t can be modelled by a probabilistic *observation model*, given by [2]:

$$z_t = h(x_t, v_t) \sim p_h(\circ|x_t),\ v_t = \text{observation noise} \quad (2)$$

We now formulate the SLAM problem. SLAM can be thought of as a single optimization problem with two components; localization, and mapping. The aim of the mapping problem is to generate a map m of landmarks in the environment given the vehicle's state trajectory $\mathbf{x_{0:T}}$ and features obtained from stereo camera images $\mathbf{z_{0:T}}$ by minimizing the following cost function:

$$\min_{\mathbf{m}} \sum_{t=0}^{T} \|\mathbf{z_t} - h(x_t, m)\|_2^2, \quad (3)$$

where $h(x_t, m)$ is the observation model described above. The localization problem takes as an input the landmark map m of the environment, the features obtained from camera images, and control inputs in order to estimate the vehicle's state x. In its most general form, this is achieved by minimizing the following cost function:

$$\min_{\mathbf{x_{0:T}}} \sum_{t=0}^{T} \|\mathbf{z_t} - h(x_t, m)\|_2^2 + \sum_{t=0}^{T-1} \|\mathbf{x_{t+1}} - f(x_t, u_t)\|_2^2, \quad (4)$$

where $f(x_t, u_t)$ is the motion model described above. Therefore the complete SLAM problem consists of minimizing the above cost function with respect to both the vehicle state x, and the map m. Alternatively, the probabilistic interpretation of the above problem is to compute the joint posterior distribution of the vehicle's state and the map conditioned on the control inputs and sensor measurements: $p(x_t, m|u_t, z_t)$ [1].

III. Technical Approach

We approach the problem in two parts. The first consists of estimating the vehicle's trajectory and separately estimating the landmark map assuming that the estimated trajectory is the ground truth. The second part is the SLAM implementation, in which we simultaneously estimate both the trajectory and landmark map at each time step.

A. Extended Kalman Filter

We use an Extended Kalman Filter (EKF) approach to solve the SLAM problem. EKF is an extension of the Kalman filter in which a first-order Taylor series approximation to a

possibly non-linear motion and observation model is made at the vehicle's pose and noise means. Recall that the Kalman filter requires the motion and observation model to be linear, and for the prior over both the vehicle's state and the landmark map locations to be Gaussian: $p_{t|t}(x_t) \sim \mathcal{N}(\mu_{t|t}, \Sigma_{t|t})$ and $p_{t|t}(m_t) \sim \mathcal{N}(\mu_{t|t,map}, \Sigma_{t|t,map})$. The EKF maintains the assumption on the prior and adopts a linear assumption of the model through the linearization process described above. As with the Kalman filter, EKF allows us to track only the mean and covariance of the vehicle's pose and the landmark map as these two parameters fully describe the Gaussian distributions over the pose and landmark map. The mean and covariance of the robot pose is denoted by $\mu_{t|t,robot} \in SE(3)$ and $\Sigma_{t|t,robot} \in \mathrm{R}^{6x6}$, while the mean and covariance of the landmark positions are denoted by $\mu_{t|t,landmark} \in \mathrm{R}^{3M}$ and $\Sigma_{t|t,landmark} \in \mathrm{R}^{3Mx3M}$, where M is the number of landmarks.

The main aim of using the EKF is that the covariance of the landmark positions and the robot pose decreases with time, as we observe more features and have more time to correct and update our estimate of the true vehicle pose and landmark positions.

1) Prediction step for localization only: In the prediction step, the pose of the vehicle at the next time step is predicted using the motion model. In this context, the motion of the vehicle is modelled using SE(3) pose kinematics equations. In continuous time, the pose $T(t)$ with noise $w(t)$ is given by:

$$\dot{T} = T(\hat{u} + \hat{w}) \tag{5}$$

where $\hat{u}$ is the hat map of the twist vector $\hat{u} = \begin{bmatrix} v(t) & \omega(t) \end{bmatrix} \in \mathrm{R}^6$. We consider a Gaussian distribution over T and express it as a pose μ with a small perturbation using the exponential map. Using time discretization and some manipulation, we arrive at the discrete time motion model update step for the vehicle pose mean:

$$\mu_{t+1|t,robot} = \mu_{t|t,robot} \exp(\tau_t \hat{u_t}) \tag{6}$$

where $\hat{u_t} = [\hat{v_t}, \hat{\omega_t}]$ is the hatmap of the linear and angular velocity input obtained from IMU readings. The predicted covariance of the robot pose is calculated using the following equation:

$$\Sigma_{t+1|t,robot} = \exp(-\tau \mu_{t,\check{robot}}) \Sigma_{t|t,robot} \exp(-\tau \mu_{t,\check{robot}})^T + W \tag{7}$$

where the curly hat denotes the adjoint hat map: $\check{\mu_t} = \begin{bmatrix} \hat{\omega_t} & \hat{v_t} \\ 0 & \hat{\omega_t} \end{bmatrix} \in \mathrm{R}^6$ Note that since the landmark map is static i.e. we assume the landmarks are arranged in an unknown but deterministic and fixed set of locations, there is no prediction step for the landmark map.

2) Update step for mapping only: In the update step for mapping only, we assume that the trajectory obtained by repeatedly applying the motion model to the vehicle pose mean is the ground truth. In other terms, we use the dead reckoning trajectory to project features obtained from the stereo camera back to the world frame in order to create the landmark map.

The aim in this step is to estimate the coordinates m $\in \mathrm{R}^{3M}$ of the landmarks given a set of observations $\mathrm{z}_t \in \mathrm{R}^{4N_t}$. Note that at any given time step t, only a subset N_t of the landmarks will be observed, hence the observation vector is in $\mathbb{R}^{4N_t}$, since each landmark has homogeneous coordinates with an (x, y, z) triple. We assume that the data association which matches each observation to each landmark is already known. The observation model for the stereo camera is as follows:

$$z_{t,i} = h(T_t, m_j) + v_{t,i} = K_s \pi(T_{imu,opt} T_t^{-1} m_j) + v_{t,i}, \tag{8}$$

where K_s is the intrinsics matrix of the stereo camera. The update step equations for the landmark map are as follows [2]:

$$\begin{aligned} \tilde{z_{t+1,i}} &= K_s \pi(T_{imu,opt} T_t^{-1} \mu_{t,j,landmark}) \\ H_{t+1,i,j} &= K_s \frac{d\pi}{dq}(T_{imu,opt} T_{t+1}^{-1} \mu_{t,j,landmark}) T_{imu,opt} T_{t+1}^{-1} P^T \\ K_{t+1} &= \Sigma_{t,landmark} H_{t+1}^T (H_{t+1} \Sigma_{t,landmark} H_{t+1}^T + I \bigotimes V)^{-1} \\ \mu_{t+1,landmark} &= \mu_{t,landmark} + K_{t+1}(z_{t+1} - \tilde{z_{t+1}}) \\ \Sigma_{t+1,landmark} &= (I - K_{t+1} H_{t+1}) \Sigma_{t,landmark} \end{aligned} \tag{9}$$

where $\tilde{z_{t+1}}$ is the predicted observation at time t+1, K is the Kalman gain, and H is the Jacobian of the predicted observation with respect to each landmark (also known as the innovation). These equations are applied at each time step to obtain updated estimates of the mean and covariance of the landmarks.

The Kalman gain serves as a weight that reflects our confidence in the current observation. If the gain is high, then the estimate for the landmark locations changes significantly depending on the difference between the observation and predicted observation. This means we are confident in the observation and therefore are willing to significantly change our beliefs about the landmark positions.

3) Landmark map initialization: Prior to updating the landmarks, we first initialize their positions by inferring their location from images captured by the stereo camera in which those landmarks are observed. The first time a landmark is seen, its coordinates in the pixel frame are projected to the world frame and used as the initial estimates. These are updated only when the same landmark is observed a second time. In order to project the pixel frame coordinates to world, we first solve the stereo camera model equations, which convert from pixel to optical frame, after which we apply the inverse of the current pose of the vehicle to project to world. Since the intrinsics matrix is rank deficient, it is not invertible and so the solutions for the optical coordinates must be obtained by

solving a system of linear equations. The system was solved analytically, leading to the following equations [3]:

$$\begin{aligned} d &= u_L - u_R \\ z &= -f_{su} b/d \\ x &= (u_L - c_u) * z/f_{su} \\ y &= (v_L - c_v) * z/f_{sv} \end{aligned} \tag{10}$$

The (x, y, z) triple obtained above is then projected back to world coordinates for the initial estimate of the landmark positions.

4) Visual Inertial SLAM: In the SLAM implementation, the key difference to the previous approach is that the vehicle's pose parameters and landmark map parameters are updated simultaneously. In other words, rather than assume the dead reckoning trajectory is true, we use the latest observation from the stereo camera to both update our belief of the landmark positions but also to re-calibrate our belief of the vehicle's true pose. We use an expanded covariance matrix that now contains cross-covariances between the robot pose and landmark positions, whereas earlier the covariance matrix was separate for both. The joint covariance matrix is now of the form $\begin{bmatrix}\Sigma_{LL}\Sigma_{LR}\\ \Sigma_{RL}\Sigma_{RR}\end{bmatrix} \in \mathbb{R}^{(3M+6)\text{x}(3M+6)}$. The landmark and robot pose means remain separate as there are no direct links between the means, and also the update equations for both will take different forms, as will be explained later.

Prediction step: There is also a change to the prediction step. Since we now work with a joint covariance matrix, the covariance prediction step form changes. Rather than applying the prediction equations above to just the robot covariance, we now apply the linearized motion model Jacobian to the off-diagonal entries as follows, while the bottom right block has the same form as in the localization only prediction step: $\begin{bmatrix}\Sigma_{LL} & \Sigma_{LR}F^T \\ F\Sigma_{RL} & F\Sigma_{RR}F^T + W\end{bmatrix}$ This is to capture cross-correlations between the robot pose and landmarks during the prediction step.

Update step: In the update, step, the process is similar to mapping only, with the exception that the covariance, H and Kalman gain matrices are now expanded matrices that include both landmark and vehicle pose sub-matrices. Furthermore, the mean update equations are separate, since the poses of the vehicles are in SE(3) and so updates must remain in the manifold, hence the use of the exponential map below. The landmark mean update remains the same as before. The equations for the update step are as follows:

$$\begin{aligned} \tilde{z_{t+1,i}} &= K_s \pi(T_{imu,opt}\mu^{-1}_{t+1|t,robot}\mu_{t,j,landmark}) \\ H_{t+1,i,j} &= K_s \frac{d\pi}{dq}(T_{imu,opt}\mu^{-1}_{t+1|t,robot} \\ &\mu_{t,j,landmark})T_{imu,opt}(\mu^{-1}_{t+1|t,robot}\mu_{t,j,landmark})^{\circ} \in \mathbb{R}^{(4N_t)\text{x}(3M+6)} \\ K_{t+1} &= \Sigma_{t+1|t,joint}H^T_{t+1}(H_{t+1}\Sigma_{t+1|t,joint}H^T_{t+1} + I \bigotimes V)^{-1} \\ &\in \mathbb{R}^{(3M+6)\text{x}(4N_t)} \\ \mu_{t+1,landmark} &= \mu_{t,landmark} + K_{t+1}(z_{t+1} - \tilde{z_{t+1}}) \\ \mu_{t+1|t+1,robot} &= \mu_{t+1|t,robot}\exp((K_{t+1}(z_{t+1} - \tilde{z_{t+1}})^{)} \\ \Sigma_{t+1|t+1,joint} &= (I - K_{t+1}H_{t+1})\Sigma_{t+1|t+1,joint} \end{aligned} \tag{11}$$

where $\begin{bmatrix}s\\1\end{bmatrix}^{\circ} = \begin{bmatrix}I & \hat{-s}\\ 0 & 0\end{bmatrix}$ and H is now an expanded matrix of the form: $\begin{bmatrix}H_{t+1,landmark} & H_{t+1,robot}\end{bmatrix}$

5) Data Pre-processing: Minimal data processing was required in this project. The sign of the IMU y,z coordinates had to be flipped as the IMU is oriented upside down on the vehicle; this corresponds to a rotation around the x-axis of π rads. Substituting in to the appropriate rotation matrix yields simple sign changes.

The main data processing in this project was for the features from the stereo camera. First, we downsample the features, since the full dataset consists of over 13,000 observations at each time step. Noting that the covariance matrix of the landmarks is of size $\mathbb{R}^{3Mx3M}$, this would lead to a covariance matrix that would be too large to compute with in reasonable time and memory. Therefore, we downsample by a factor of as much as 20x during the implementation. This means we take every 20th observation from each time step. Using a stride of 20 leaves us with a more manageable number of landmarks for computation purposes.

Additionally, we filter out portions of the data stream that correspond to missing observations. At each time step, only a subset of the landmarks are observed and so the remaining vectors are populated with dummy values to distinguish them from the observed features.

6) Noise: We formulated two methods of adding noise to the covariance prediction step. The first method involves constructing a noise matrix W that is added directly in the covariance prediction step for the vehicle pose i.e. in $\exp(-\tau u_t)\Sigma_{t|t}\exp(-\tau u_t)^T + W$.

The second method was to add noise directly to the control input, with variance being set to a percentage of the control input at time t. For example, if the linear velocity $v_{t,x} = 7$, then the variance of the noise in the x-direction could be set to 1% of 7. This method operates under the assumption that the source of the noise comes from the control inputs and motors that propel the vehicle. Mathematically the

linear velocity noise can be represented as $\epsilon_{t_v} \sim \mathcal{N}(0, \sigma_v)$, where $\sigma_v = noisefactor * (v_t)$. The noise factor variable is set experimentally. The noise for angular velocity is similarly defined with respect to angular velocity at each time step.

7) Outliers: Outliers are an important consideration in SLAM problems. They can affect the mean, covariance and cross covariance of landmarks and vehicle pose and therefore affect the calculations in the update step, potentially leading to erroneous trajectories and map updates. Our method for handling outliers is described next.
1. During initialization, the distance between the world coordinates of the landmark and the current vehicle position are compared to a threshold given by $threshold = \|p_{vehicle} - p_{landmark,init}\|_2$.
2. If the distance is greater than the threshold, the landmark is discarded unless it is seen again and is within the distance threshold.
3. In case a point is initialized within the threshold but gets pushed further out than the threshold during map update steps, we identify those landmarks, and arbitrarily set their new location to be at the threshold distance. Other methods for outlier handling could yield better results and can be tested in future work.

B. Technical Details

We use a MacBook Pro with 1.4 GHz Quad-Core Intel Core i5, 8 GB 2133 MHz LPDDR3, Intel Iris Plus Graphics 645 1536 MB. We make use of the well known numpy library for data manipulation.

IV. Results

We first present the results for the vehicle trajectory and the landmark map estimation obtained from dead reckoning. We then present results from the SLAM implementation.

1) Dead Reckoning: We first show the results from dead reckoning for both datasets, which is a successive application of the motion model to the mean of the vehicle pose without introducing noise into the control input. We then plot the trajectory to ensure the motion model is implemented correctly. The figures below shows that the vehicle motion and landmark map predicted by the motion model matches fairly well with the actual path the vehicle takes, when checked against the video provided. However, the final right turn before the car parks is missed in the dead reckoning trajectory. We will see later that it is captured in the SLAM implementation with outlier removal. For dataset 3, we see that there is a small error in the trajectory towards the end before the final left turn. This error is corrected later with the SLAM implementation and outlier removal. Furthermore, we see the distribution of landmarks is fairly spread out in dead reckoning, and this too is corrected with the SLAM implementation; the landmark positions are clustered closer to the road.

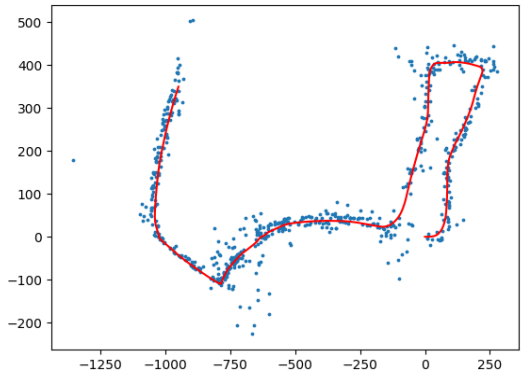

Fig. 1. Dataset 10 dead reckoning trajectory and landmark map

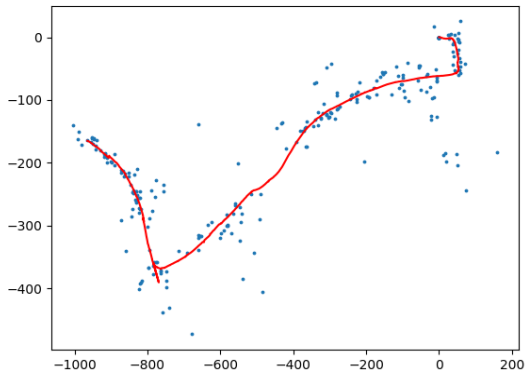

Fig. 2. Dataset 3 dead reckoning trajectory and landmark map

2) SLAM: We present the best trajectory and landmark map estimate obtained for each dataset provided. Various considerations are then discussed. The best results were obtained with stride = 20, 0.5% of the control input as the variance for the noise for both linear and angular velocity, and outlier removal. We see that for dataset 10, this estimate correctly captures the final right turn prior to parking that was not seen in the dead reckoning trajectory estimate.

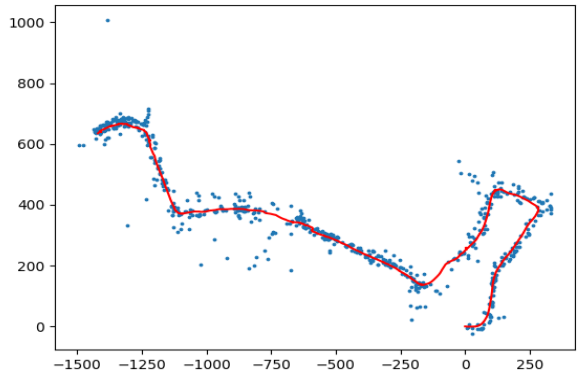

Fig. 3. Dataset 10 best SLAM trajectory and map estimate, stride = 20, noise = 0.01%, outliers removed, outlier threshold distance = 150. Note the final right turn before parking is captured here (plot flipped) but was not captured by dead reckoning.

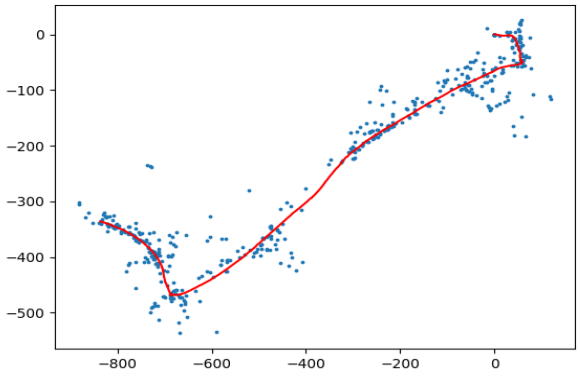

Fig. 4. Dataset 3 best SLAM trajectory and map estimate, stride = 10, noise = 2%, outliers removed, outlier threshold distance = 120. Note that the small error in the dead reckoning trajectory seen towards the end of the path has been corrected by the filter

3) Noise: As discussed earlier, we experimented with two different methods of adding noise, as well as with different levels of noise, to investigate the effects on trajectory estimation. The method of adding noise directly to the control inputs provided much better results than when applying additive noise in the hatmap form, and we also found that low noise performed better than higher noise. The figure below shows that the estimate of the trajectory for the additive noise method is far off the true trajectory and implies that this method of adding noise does not represent the true noise dynamics of the system. From the plots below we see the estimated trajectory has changed quite significantly when increasing the noise even from 2% of the control input at time t to 5% of the control input, for a given stride. Both the trajectory and how smooth it is have changed when increasing the noise. This suggests that the quality of the sensor can be very important in accurate estimation of state.

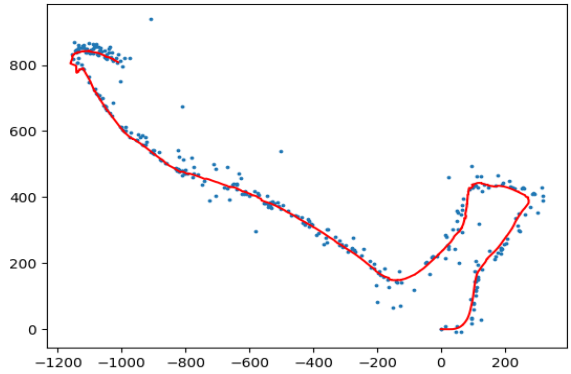

Fig. 5. Dataset 10 SLAM trajectory and map estimate, noise = 2%, stride = 40

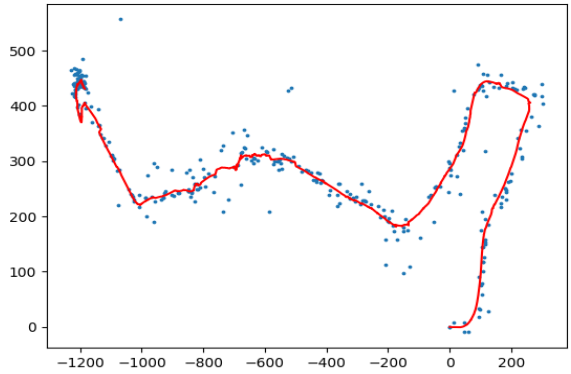

Fig. 6. Dataset 10 SLAM trajectory and map estimate, noise = 5%, stride = 40

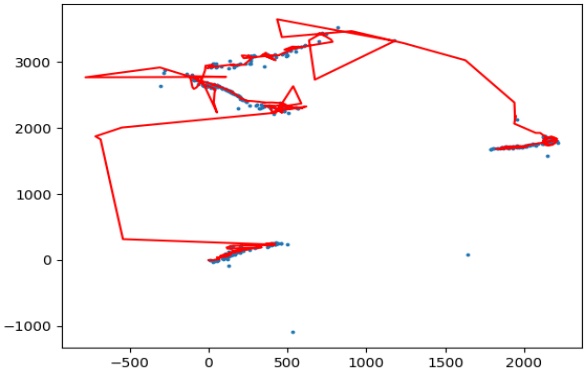

Fig. 7. Dataset 10 SLAM trajectory and map estimate, additive noise method

4) Outliers: We next experiment with outlier removal. From the plots above, we see that removing even a handful of outliers can have a significant impact on the trajectory and map estimates. With outlier removal, the trajectory is smoother and the filter captures the final turn made before parking, even though the direction of this turn shown is incorrect. Note these figures are to illustrate the effect of outlier removal only and do not represent the most accurate trajectory; these were generated with high downsampling for computational efficiency.

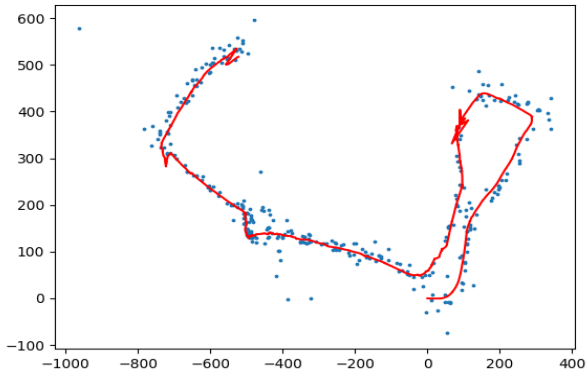

Fig. 8. Dataset 10 SLAM trajectory and map estimate, no outlier removal, stride = 40, distance threshold = 120, noise = 2%

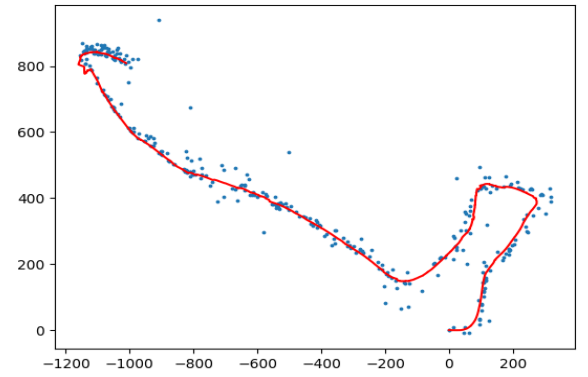

Fig. 9. Dataset 10 SLAM trajectory and map estimate, outliers removed, stride = 40, distance threshold = 120, noise = 2%

5) Further observations: One interesting point noted during this project was that errors in implementation were uncovered when calculating the Kalman gain. The matrix inversion led to singular matrices if the H matrix was not calculated properly; for example, if the individual blocks in the H matrix overlapped, the extra column of zeros would lead to the matrix product $H\Sigma H$ not being positive definite and hence not always invertible.

Additionally, we found that the trajectories were sensitive to stride length, noise and outlier removal threshold. In fact, the direction of the last turn changed depending on the values chosen for the above variables. Below we see an example of this; the final right turn (shown as left in the plot due to flipped axes), is shown as a left turn instead (right turn in the plot). Further investigation would have to be made to understand why a small change in noise, outlier threshold or stride would cause such a significant change in the trajectory.

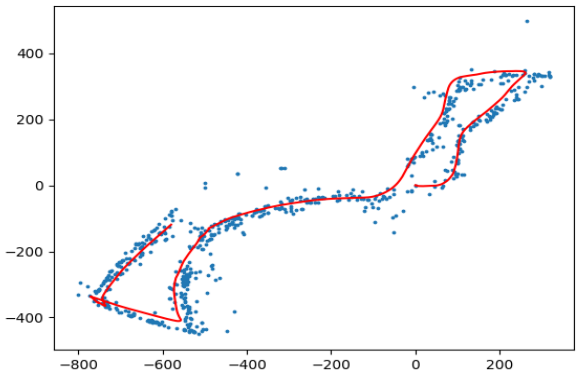

Fig. 10. Dataset 10 SLAM trajectory and map estimate, outliers removed, stride = 20, distance threshold = 100, noise = 2%. Note the final turn is in the opposite direction to the best estimate with a small change to the distance threshold and noise percentage.

REFERENCES

[1] M. Montemerlo, S. Thrun, D. Koller, B. Wegbreit, "FastSLAM: A Factored Solution to the Simultaneous Localization and Mapping Problem"
[2] "https://natanaso.github.io/ece276a/ref/ECE276A_7_BayesFilter.pdf"
[3] "https://natanaso.github.io/ece276a/ref/ECE276A_5_LocalizationOdometry.pdf"