SLAM (Simultaneous Localization and Mapping) Using Extended Kalman Filter and Stereo-Odometry Data

Harmeet Gill
Department of Electrical Computer Engineering
University of California, San Diego
hsgill@eng.ucsd.edu

Abstract- This paper describes the theory and experimental results behind using an extended kalman filter with camera stereo data and odometry data to simultaneously localize and map the movement and surroundings of a robot (or car in the case of the data). The paper will discuss the success of the results as well as some parameters that were tested given the experimental setup and model.

I. Introduction

SLAM is an important and useful logical system for allowing devices with sensors to understand their surroundings better over time. This has obvious bearing in fields such as the automotive industry for driving automation, as well as other use cases where the surroundings might not be so dynamic (uses such as training a manufacturing robot to find objects or be more careful when a new object appears in the surrounding). In our case, SLAM relies on probabilistically reasoning where the robot state is based on maintaining and updating a pdf of robot states over time that's informed by measurements. Extended Kalman filter SLAM is a way of tracking non-linear equation driven robot states and over time by having a gaussian distribution that represents the robot state be updated by data that also has gaussian characteristics. The central limit theorem allows many data types and uncertainties in measurement in experiment to become gaussian, which is why Kalman filters are widely used. With the use of extended Kalman filter, measurement error in the robot's pose or scan reading can be handled and improve end results of the SLAM logic.

For the method described here, the problem was broken down into three areas. How do we use lie algebra and SE3 space to track robot pose? How can we use an EKF model to update out landmarks over time? And third, how can we use an EKF model to update robot pose informed by landmark positions over time?

The transformations were largely handled by matrix transformations in SE3 space from camera coordinates to IMU coordinates. The features observed in the camera were tracked and updated over time but doing inverse transformations from camera to IMU. These then would be used in subsequent time steps to inform the correction of the pose, as will be described in the problem formulation.

The fundamental assumptions of a Kalman filter are that update and prediction equations are linear and that our noise injection in the system follows a gaussian, however with the conversion from camera to world coordinates our update steps become non-linear, thus an extended Kalman filter is necessary to use Taylor expansion about a point to linearize the calculation. As will be discussed, the extended Kalman filter allowed us to use the Kalman filter in our non-linear case, with the test emphasis being around testing a tweaking noise injection for the motion and observation model to best match reality.

II. Problem Formulation

A. SE3 Kinematics and Prediction

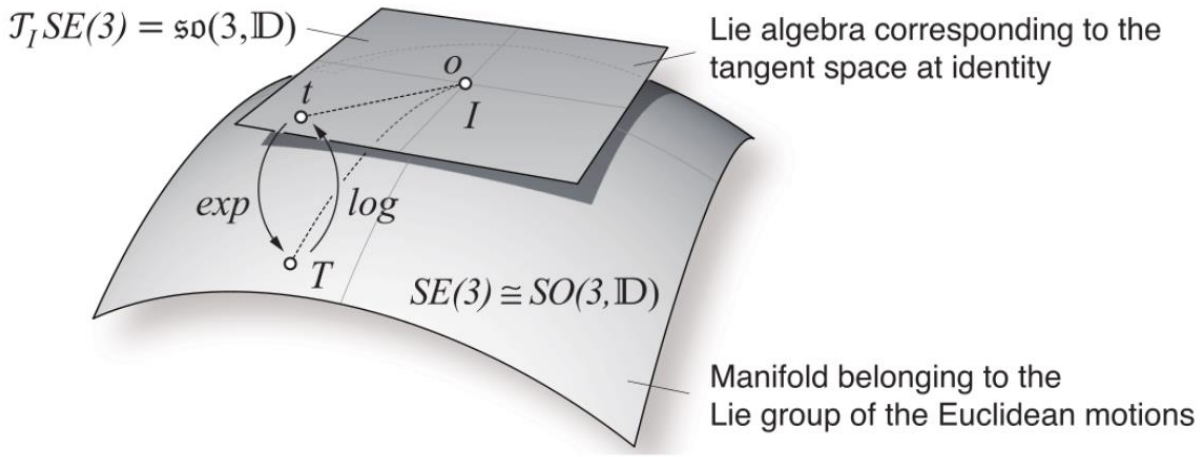

Figure 1 Visualization of SE3 Space and Lie Algebra

The prediction step that is initially implemented is informed purely from rotation and linear velocity given from the IMU. Because we also have the time stamps for each measurement, we can easily turn the velocities into absolute position and rotation changes by multiplying any instance of rotation and linear velocities by the delta time between it and the last measurement. With this, we have the change in linear and rotational positions over time, that can be used to understand robot trajectory with some simple transformation equations. To simplify the math, however, pose information can be translated into SE3 space using lie groups, which hold the property of multiplying previous poses rather than adding for new trajectory.

$$\mu_{t+1|t} = \exp(-\tau\hat{u}_t)\mu_{t|t} \quad \epsilon \text{ 4x4 (1)}$$

$$\delta\mu_{t+1|t} = \exp(-\tau\hat{u}_t)\,\mu_{t|t} + w_t \ (2)$$

$$\Sigma_{t+1|t} = \mathrm{E}\left[\delta\mu_{t+1|t}\delta\mu_{t+1|t}^{T}\right] = \exp(-\tau\tilde{u}_t)\Sigma_{t|t}\exp(-\tau\tilde{u}_t)^T + W \ \epsilon \text{ 6x6} \quad (3)$$

$$\hat{\mu}_t = \begin{bmatrix}\hat{\omega}_t & v_t \\ 0^T & 0\end{bmatrix}, \tilde{u}_t = \begin{bmatrix}\hat{\omega}_t & v_t \\ 0^T & \hat{\omega}_t\end{bmatrix} \ (4)$$

Here, $\mu_{t+1|t}$ represents a new pose being informed by the old pose, $\mu_{t|t}$, and a new measurement from the INS ($\hat{u}_t$). Similarly, the variance of the predicted pose follows being updated from the old variance with a noise W injected in, which is one of the necessary attributes of a Kalman filter. This indicated that the prediction of new robot pose is becoming less accurate over time in the absence of correcting the pose with other features (such as the landmarks in our case as seen by the camera). The $\delta\mu_{t+1|t}$ term is generated by considering a perturbation model, where new information about the robot pose comes in as a separation between deterministic values, and a delta value that acts as the perturbation introduced by noise. This model can be reduced into the curl hat argument shown above.

B. Landmark Updating

Assuming the predicted trajectory in part A is true, the next step would be to introduce and update landmarks in the trajectory map which can be used later to improve our pose information. To start, we must initialize a landmark using the following.

$$M = \begin{bmatrix} fs_u & 0 & c_u & 0 \\ 0 & fs_v & c_v & 0 \\ fs_u & 0 & c_u & -fs_u b \\ 0 & fs_v & c_v & 0 \end{bmatrix}, z_c = \begin{bmatrix} u_l \\ v_l \\ u_r \\ v_r \end{bmatrix}, m = \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}, d = u_l - u_r = \frac{fs_u b}{z_w} \ (5)$$

$$\begin{bmatrix} u_l \\ v_l \\ d \end{bmatrix} = M\frac{1}{z}m, x_w = \frac{(u_l - c_u)b}{d}, y_w = \frac{(v_l - c_v)b}{d}, z_w = \frac{fs_u b}{d} \text{ (6)}$$

The equations above are derived from a stereo camera model, with both the intrinsic and extrinsic parameters given. z_c represents the correspondence camera coordinates from the left and right frame. The world coordinates are subscripted with w. Equation 6 is simplified by solving equations 5 for world coordinates. It's worth noting that for our case both fsu and fsb were the same value, so equations 6 are further simplified accordingly. Solving the equation out, we can find closed form solution to a tracked features cartesian position in the left camera frame (x, y and z). With this, the robots pose as well as a transformation matrix can be used to go from camera to world frame. We assume the data is associated in . z_c and that landmarks are static for our case.

$$\begin{bmatrix} x_{world} \\ y_{world} \\ z_{world} \\ 1 \end{bmatrix} = U_{imu}^{world} T_{cam}^{IMU} \begin{bmatrix} x_{cam} \\ y_{cam} \\ z_{cam} \\ 1 \end{bmatrix} \text{ (7)}$$

With this, we can initialize our landmark map with an acquisition of data. To then update landmark positions based on an EKF model and given pose data, the inverse of mapping world to features must be done to compare new observations against what we already know about our landmarks.

$$z_t = M\pi\left(T_{imu}^{optical} U m_j\right) + v_t, \epsilon\ 4\text{x}N_t \quad v_t = N(0, I \otimes V) \text{ (8)}$$

$$\pi(q) = \frac{1}{q_3}q, \frac{d\pi}{dq} = \frac{1}{q_3}\begin{bmatrix} 1 & 0 & -\frac{q_1}{q_3} & 0 \\ 0 & 1 & -\frac{q_2}{q_3} & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & -\frac{q_4}{q_3} & 1 \end{bmatrix}, P^t = [\text{I } 0] \text{ (9)}$$

$$H_t = M\frac{d\pi}{dq}\left(T_{imu}^{optical} U_t m\right) T_{imu}^{optical} U_t P^t \text{(10)}$$

$$K_{t,i} = \Sigma_t H_t^T (H_t \Sigma_t H_t^T + I \otimes V)^{-1}\ \epsilon\ 3\text{x}4 \text{ (11)}$$

$$\mu_{t+1,i} = \mu_t + K_t\left(z_t - M\pi\left(T_{imu}^{optical} U m\right)\right) \epsilon\ 3\text{x}1 \text{ (12)}$$

$$\Sigma_{t+1,i} = (I - \text{K}_t H_t^T)\Sigma_t\ \epsilon\ 3\text{x}3 \text{ (13)}$$

In the model above, μ_t is a 3xM matrix representing landmark positions in the world coordinates. The coordinates for the next time step are corrected by the difference between new measured features and expected features, $z_t - M\pi\left(T_{imu}^{optical} U m\right)$, which is scaled by the Kalman gain which allows us to control our confidence in measurement. The difference between measured and predicted observations is only of a size relative to how many features were measured at time t ($4\text{x}N_t$). With this, new landmark positions can be instantiated, and old ones corrected to get accurate landmark readings with a given trajectory.

C. Pose Updating

The final step is to propagate new information regarding features to correct our robot pose. This completes the loop for our SLAM problem and allows data to better map the area and localize the robot based on the map. The equations for pose updating is as follows.

$$\tilde{z}_t = M\pi\left(T_{imu}^{optical} U m_j\right) \epsilon\ 4N_t\text{x}1 \text{ (14)}$$

$$H_t = M\frac{d\pi}{dq}\left(T_{imu}^{optical} U_t m\right) T_{imu}^{optical} (U_t m)^*, [s\ 1]^* = \begin{bmatrix} I & \hat{S}_t \\ 0 & 0 \end{bmatrix} \text{ (15)}$$

$$K_{t+1|t} = \Sigma_{t+1|t} H_{t+1|t}^T (H_{t+1} \Sigma_{t+1} H_{t+1}^T + I \otimes V)^{-1} \epsilon\ 6x4N_t \quad (16)$$

$$\mu_{t+1|t+1} = \exp(K_{t+1|t}(z_{t+1} - \tilde{z}_{t+1}))\mu_{t+1|t} \epsilon\ 4x4 \quad (17)$$

$$\Sigma_{t+1|t+1} = (I - K_{t+1|t} H_{t+1|t}^T)\Sigma_{t+1|t}\ \epsilon\ 6x6 \quad (18)$$

Here, $\mu_{t+1|t+1}$ represents updated robot pose with the difference between measured and expected features informing the correction according to a Kalman gain scale term. Our variance similarly scales using Kalman gain and the previous variance measurements. Both these sets of equations and the ones describing landmark updating are derived from Extended Kalman filter assumptions. Because the camera to world transformation is non-linear, estimations of the function based on Taylor expanding the projection function using perturbation logic around the point of consideration is necessary for the Kalman functions to apply and be used.

III. Technical Approach

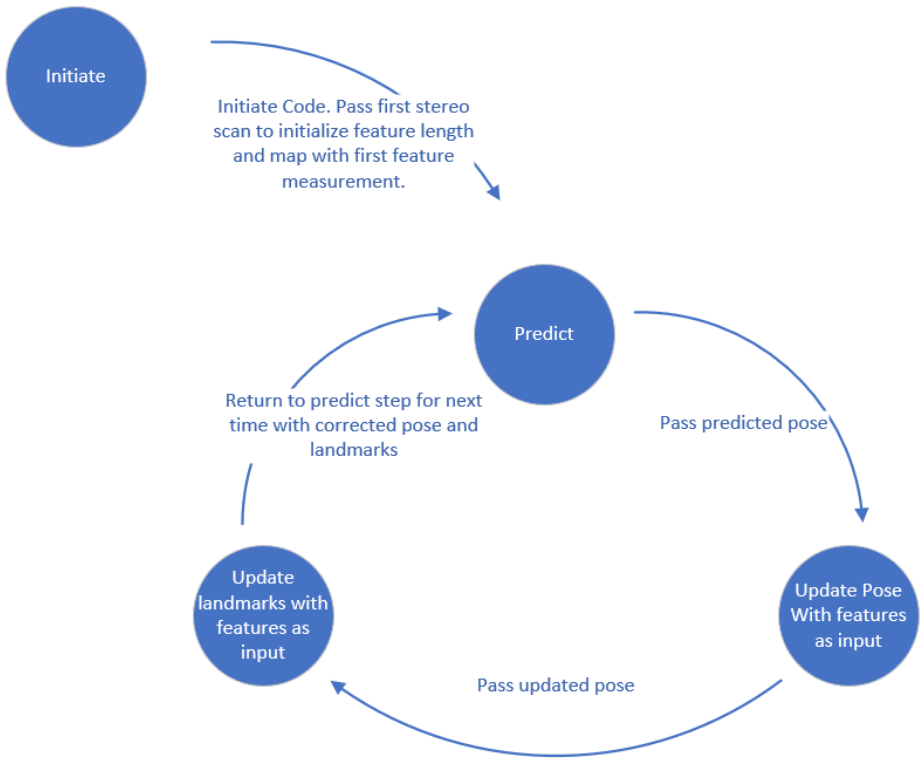

Figure 2 Visualization of Robot State

The first step in running the Kalman filter was initializing the landmark array by taking as input the first feature. This would allow us to define the length of the array necessary to account for all landmark's and give a baseline of landmarks to work with when we get to the update landmark step. Additionally, the first IMU data can't be used since there's no delta time between it and some scan from a time before. The pose was initialized as being centered at the middle of the map, with zero rotations. The robot class handles the robot state at any given timestamp, so a for loop over all time is required to continually update the parameters until the robot has closed its positional loop. The robot class tracks both SE3 and se3 pose and uses the se3 space with the grid center coordinates

to properly position the trajectory and landmarks on the map. At any given time stamp the robot get passed both stereo data and IMU data.

Using equations 5, 6 and 7, the feature map is initialized in the world frame with the first stereo measurement. In this equation, disparity between the camera y coordinates between the left and right camera were thrown out. Because the robot doesn't have a lot of variation in z, including vertical disparity would only add noise to the system. At time 0, for any landmarks that don't have associations, the landmark map for those features are kept as a zero vector.

Using equations 1 and 3, the robot then first predicts its new pose given IMU data starting at time step 1. Input data informs both the new inverse pose and covariance matrix with noise variable W injected into the new covariance matrix. The choice of W (and V) were based purely on what did not give an error in the pose update step, which was discovered to be extremely dependent on noise variables.

With a new predicted pose calculated, the pose prediction is then passed to a landmark updating function. In the landmark update, features observed are compared against a prediction of where the features might be based on a projection from world to camera coordinates of tracked landmarks as described in equation 8. With that, and using equations 13 and 13, each individual landmark positions and their covariance matrices are updated (at least the ones that are present with the observed features during the specific time step). Any landmarks that were newly observed (meaning was set as zero vectors in the robots landmark list), are instantiated using the same methods described to instantiate the landmarks at the zero-time step (equations 6).

Between pose prediction and landmark updating, the pose update step was implemented using equations 16-18. Like with the landmark update step, the stereo observations were matched with predicted observations from the existing list of landmarks projected into the camera coordinates. With this, a Kalman gain term was created that considers all measured landmarks at time t and used to calculate new pose mean and covariance. One difference between the theoretical and implemented pose update model was that a scaling constant was added to the Kalman gain equation to control unstable feedback of pose mean and covariance updates over time. With no scaling constant, the Kalman gain equation experienced big matrix values that perpetuated over time until the code threw singularity issues because of the inversion part of the gain equation. The use of scaling added another degree of freedom but allowed the pose updates to become stable and give decent results in the case of all datasets.

$$K_{t+1|t} = \Sigma_{t+1|t} H_{t+1|t}^T (H_{t+1} \Sigma_{t+1} H_{t+1}^T + I \otimes V)^{-1} * \mathrm{A} \in 6\mathrm{x}4N_t \quad (17)$$

The pose update ultimately gave better results for the case of dataset 34. The pose update also allowed the loop in dataset 27 to close better (however not completely). A scaling constant of 0.001 was found after trial and error to give the best results.

IV. Results

A. Prediction and Landmark Updates (Part A and B)

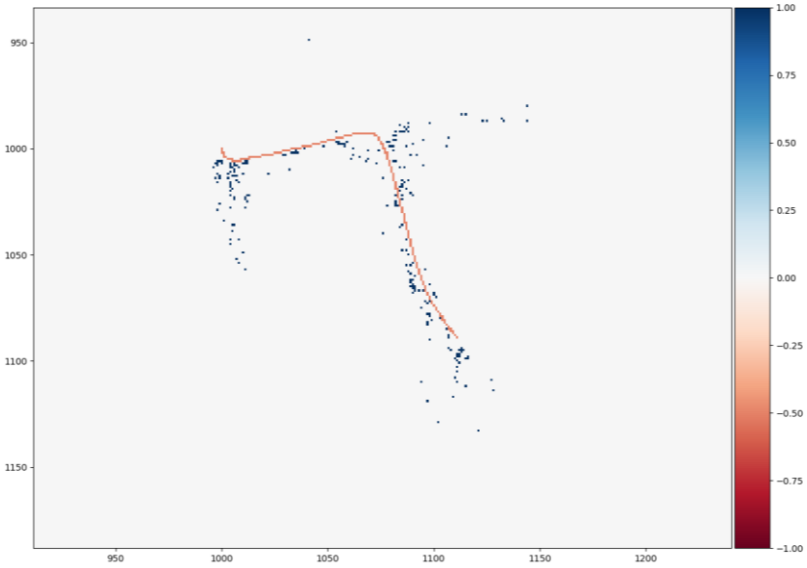

Figure 3 Data 27 1/4 Finished

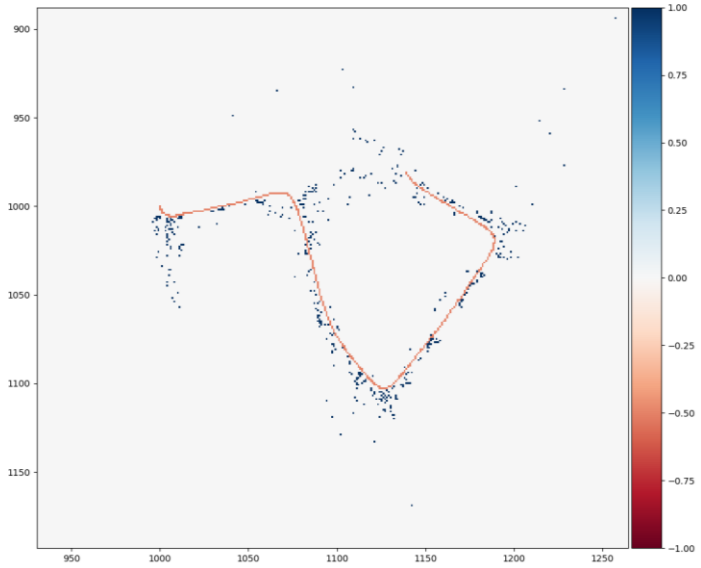

Figure 4 Data 27 2/4 Finished

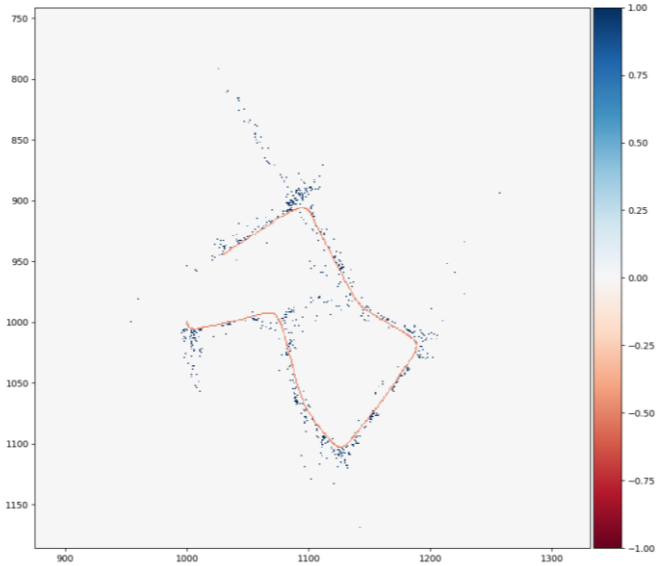

Figure 6 Data 27 3/4 Finished

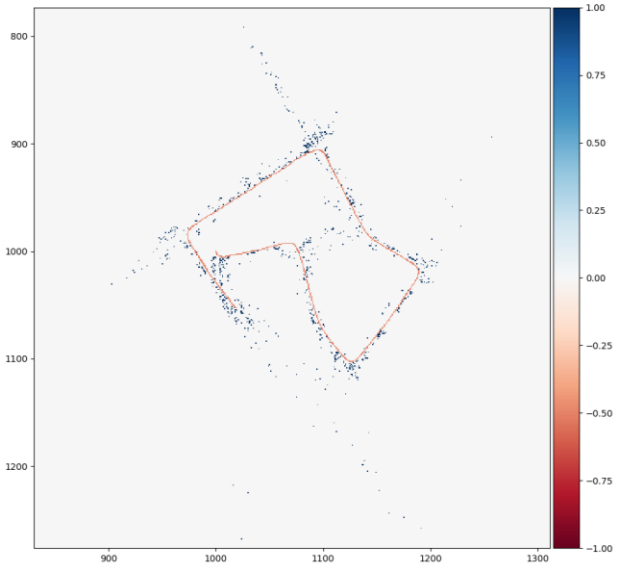

Figure 5 Data 27 Finished

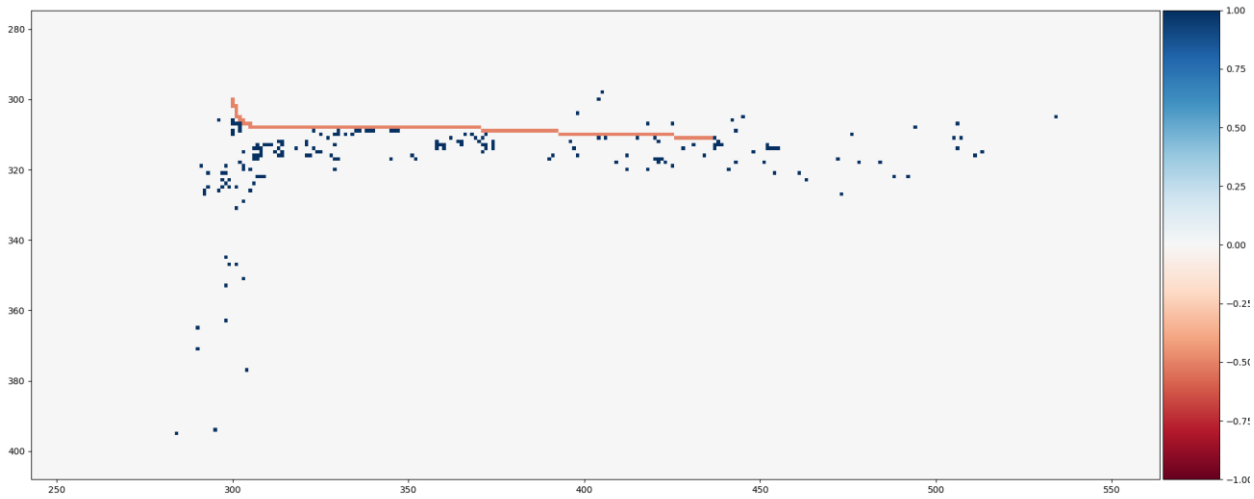

Figure 8 Data 22 2/4 Finished

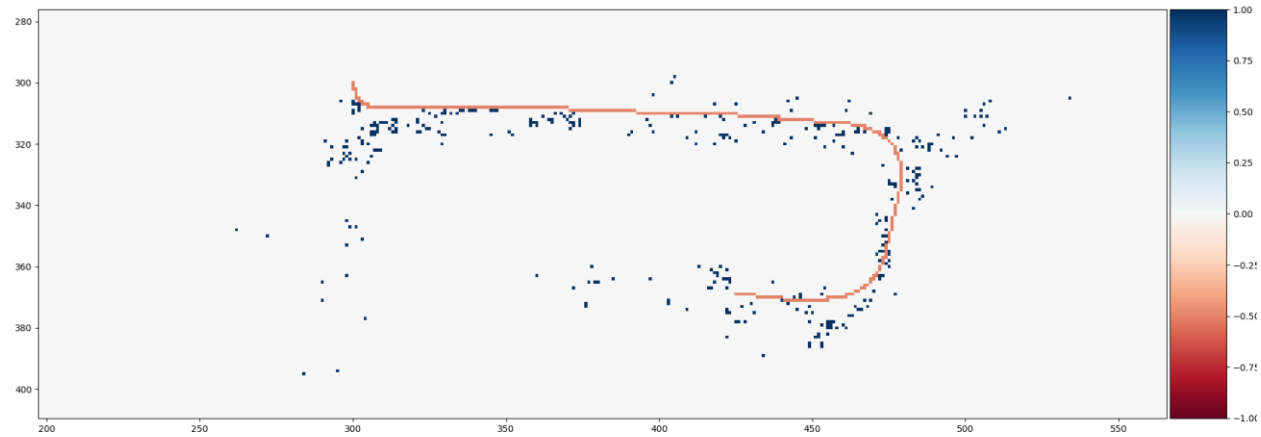

Figure 7 Data 22 1/4 Finished

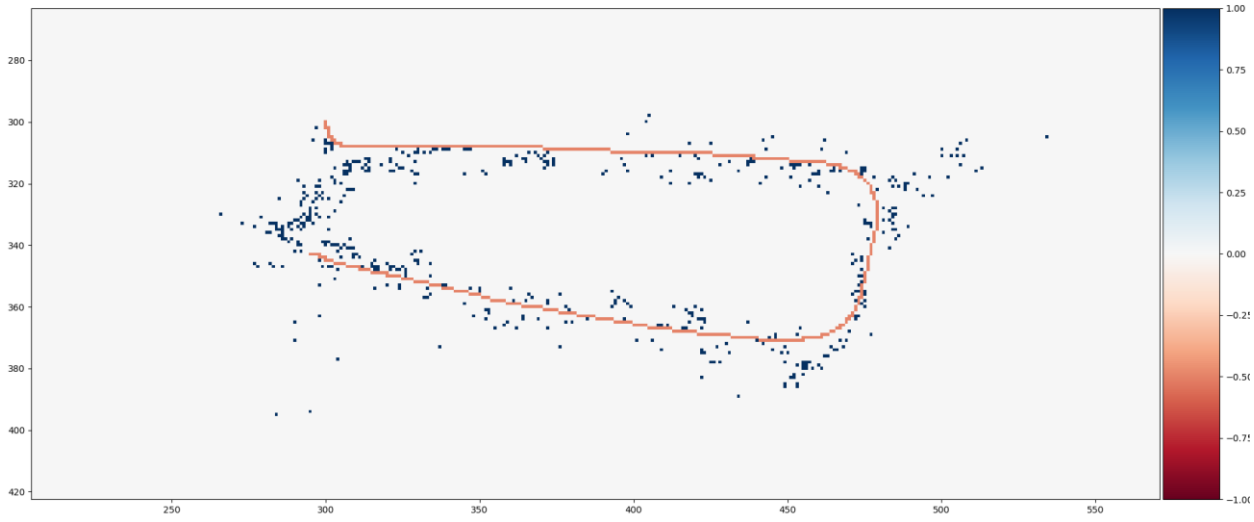

Figure 9 Data 22 3/4 Finished

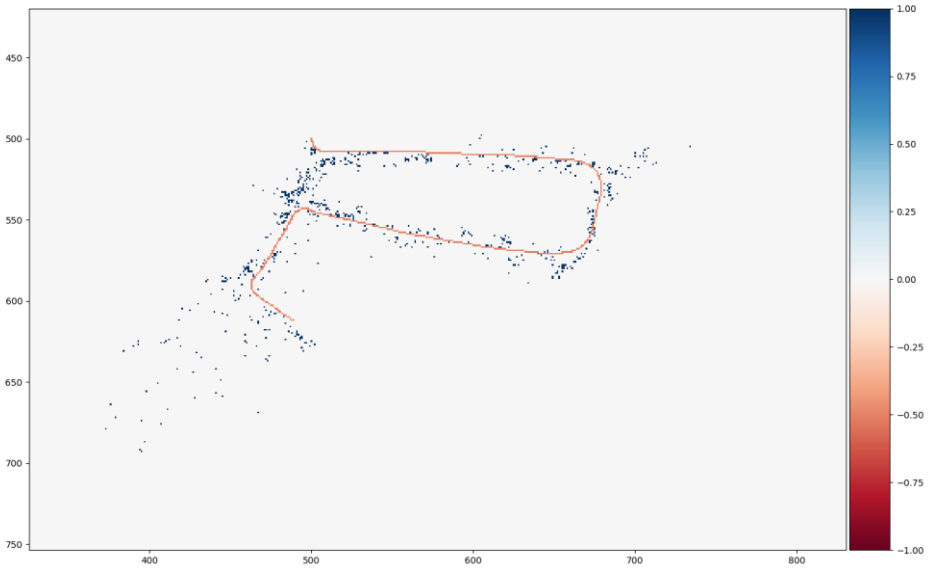

Figure 11 Data 22 Finished

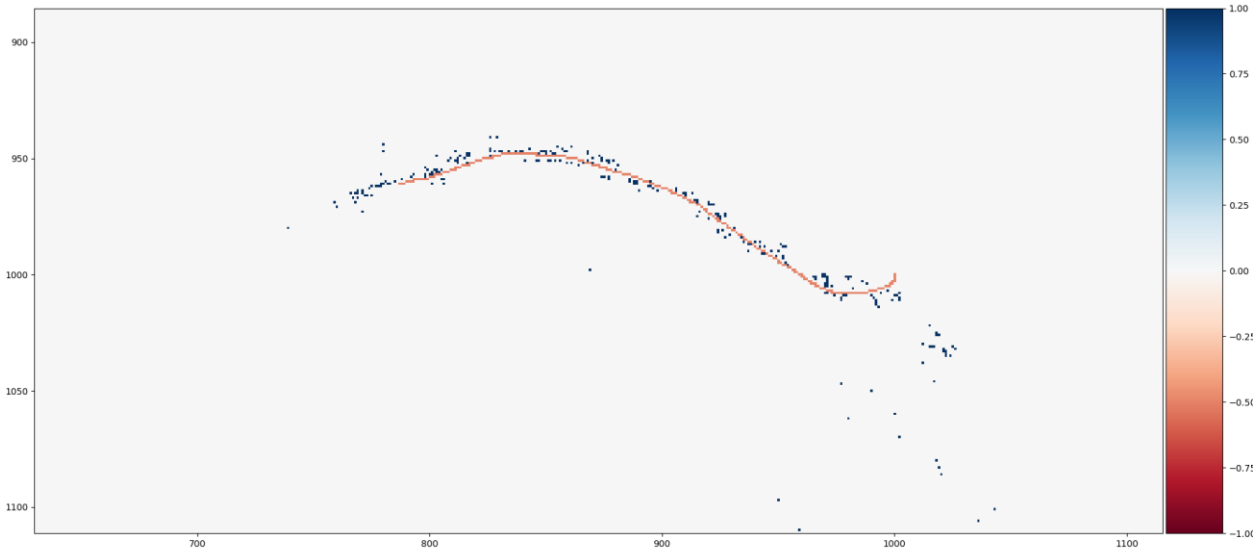

Figure 10 Data 34 1/4 Finished

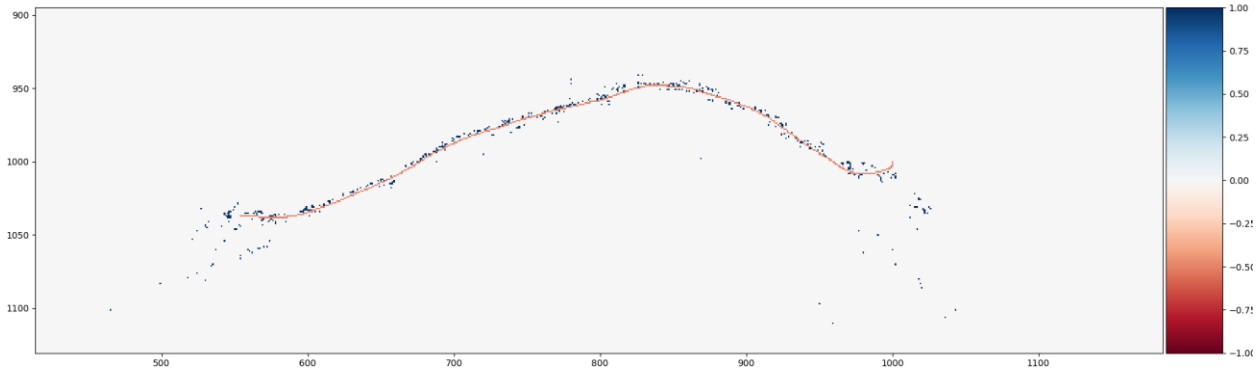

Figure 13 Data 34 2/4 Finished

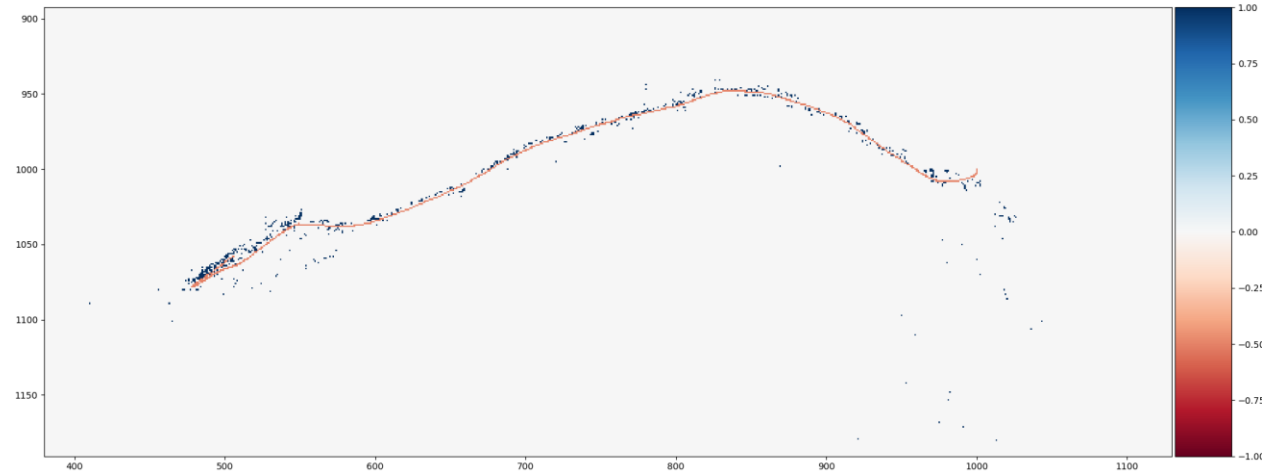

Figure 12 Data 34 3/4 Finished

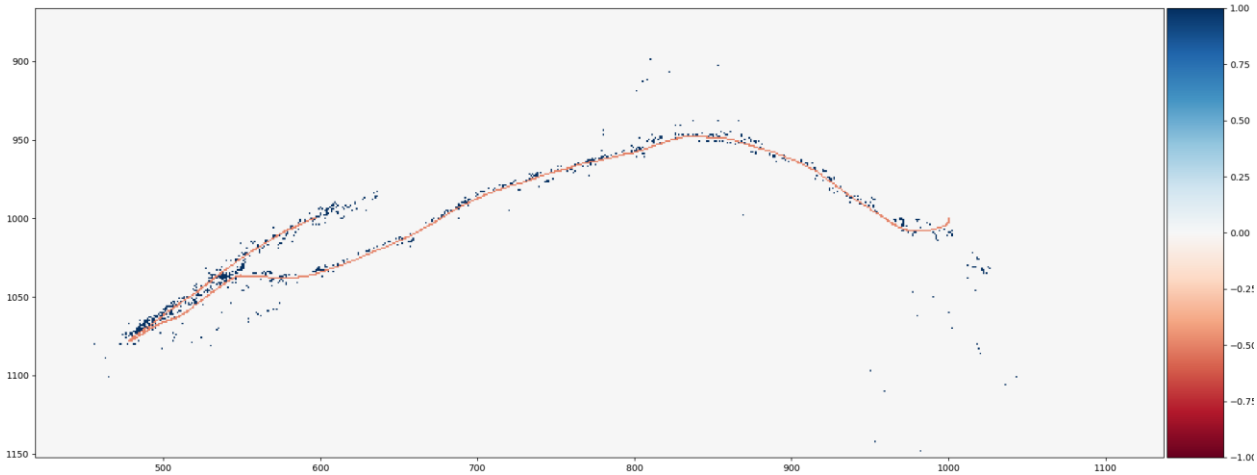

Figure 14 Data 34 Finished

B. Prediction, Landmark, and Pose Updates (Part A and B)

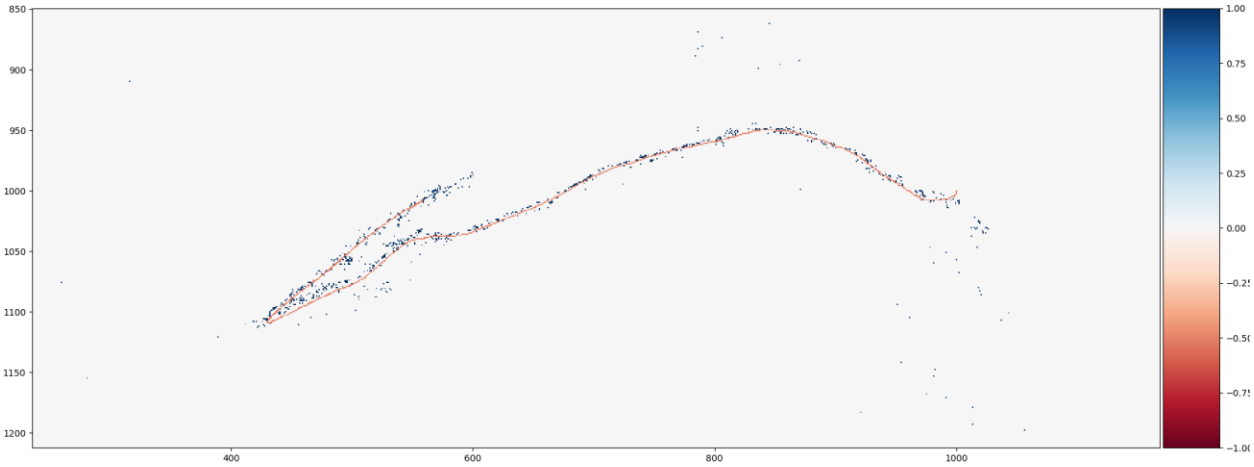

Figure 15 Dataset 34 Complete

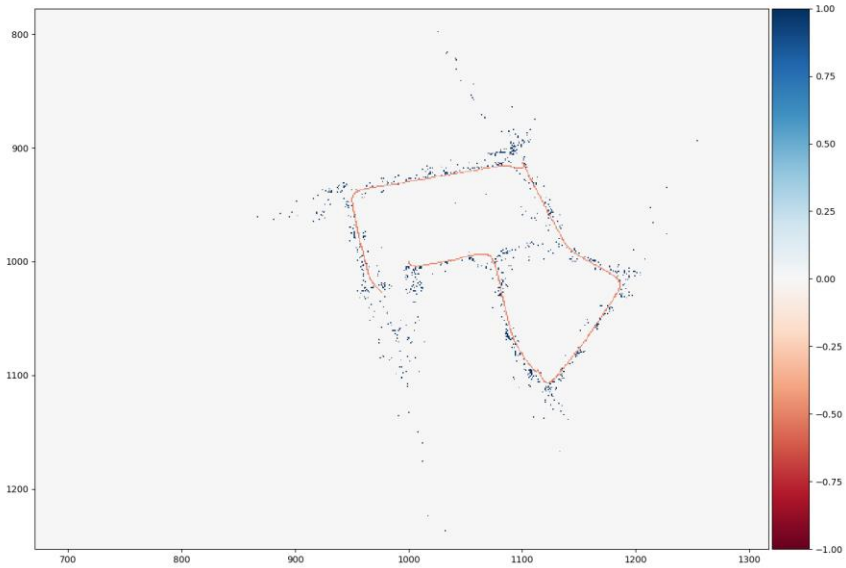

Figure 16 Dataset 27 Complete

Figure 17 Dataset 22 Complete

Overall the results for data set 22 and 24 are fairly good. The landmarks appear in the right orientation and 90 degrees turns are handled well in the pose trajectory data. Dataset 24 comes close to closing its loop but seems to overshoot in the case of pure landmark update. For the case of pose and landmark update, dataset 24 came closer to completing its loop and be true to the actual car trajectory. Dataset 34 does well in tracking the initial ride down the road but fails to execute the sharp right turn to the correct rotation for the landmark update only. In reality, the sharp turn wasn't as sharp as the grid map trajectory might suggest, which indicated some overshooting from the IMU rotational velocity data. This was corrected when pose update was introduced, as can be seen in figure 15 where the sharp right turn isn't as sharp.

While the pose update step could have drastically helped in the dataset 27, ultimately the noise injection values were too volatile and so a scaling constant had to be introduced to stabilize the update.