3D SLAM Using Kalman Filter

Sanjay John

***Abstract*— The hallmark of understanding an environment through sensor data lies in SLAM - Simultaneous Localization and Mapping. In this project, we implement a 3D SLAM algorithm utilizing an Extended Kalman Filter and various sensors in a car driving through a scene to create a 2-dimensional map of the environment observed from the top down. Stereo camera data allows for mapping of general structure of landmarks in the scene, while IMU data allows for localization of the car. We present results of this algorithm on three datasets, one of which is a test set released a few days prior to submission.**

I. INTRODUCTION

Understanding an environment through remote sensors has infinitely many applications, such as self-driving cars, unmanned machinery for mining, bomb-defusing, space exploration etc. The state-of-the-art in representing sensor data in a user-friendly manner lies in SLAM - Simultaneous Localization and Mapping. The gist of the algorithm is straightforward - given that you start at (0,0), start by mapping your environment. Then, predict your next position using your data, correlate this position with your current map, update your position based on this, and repeat.

In this project, we will use a vehicle equipped with an IMU, and stereo cameras to map out the environment that the robot is exploring. The stereo camera will be our observational input, while the IMU will serve as control inputs into the motion model. It becomes necessary to model the control inputs, observations, and states as probabilities. To therefore bring in sensor data into this probability space and effectively compute probable trajectories and maps, it becomes necessary to implement a form of a Bayesian filter. In this project, we will use an Extended Kalman Filter.

II. PROBLEM FORMULATION

SLAM ultimately becomes a chicken-and-egg problem, namely that:

- Mapping: given the robot state trajectory $x_{0:T}$, build a map m of the environment.
- Localization: given a map m of the environment, localize the robot and estimate its trajectory $x_{0:T}$.

This ends up becoming a parameter estimation problem for the parameters $x_{0:T}$ and m. Since we are given a dataset of observations $z_{0:T}$ and control inputs $u_{0:T}$, we can use MLE, MAP and Bayesian filtering to find the appropriate posterior likelihood.

SLAM exploits the Markov decomposition of the joint probability density function such that:

$$p(x_{0:T}, m, z_{0:T}, u_{0:T-1}) = p_{0|-1}(x_0, m) \prod_{t=0}^{T} p_h(z_t|x_t, m) \prod_{t=1}^{T} p_f(x_t|x_{t-1}, u_{t-1}) \quad (1)$$

where p_h refers to the observation model, and p_f refers to the motion model. Therefore it becomes necessary to arrive at an evaluation of the observation model and motion model for every time step. It also becomes necessary to define an approach for propagating this density over time. We will do this using the Bayesian filter.

To implement the Bayesian filter, we will use the Extended Kalman Filter. We assume

- The prior probability is a Gaussian
- The motion model is affected by Gaussian noise
- The observation model is affected by Gaussian noise
- The process noise w_t and measurement noise v_t are independent of each other, of the state x_t, and across time
- The posterior pdf is forced to be Gaussian via approximation

A. Motion Model

The motion model in the Extended Kalman Filter is calculated as

$$x_{t+1} = f(x_t, u_t, w_t) \quad (2)$$

where $x_t \sim \mathcal{N}(\mu_{t|t}, \Sigma_{t|t})$, $w_t \sim \mathcal{N}(0, W)$. The EKF uses a first-order Taylor series approximation to calculate $f(x_t, u_t, w_t)$.

$$f(x_t, u_t, w_t) \approx f(x_t, u_t, 0) + F*(x_t - \mu_{t|t}) + Q*(w_t - 0) \quad (3)$$

where $F = \frac{df(x_t, u_t, 0)}{dx}$ and $Q = \frac{df(x_t, u_t, 0)}{dw}$. Since w_t and x_t are independent and the Gaussian distribution is stable, we know that the distribution of x_{t+1} is Gaussian: $\mathcal{N}(\mu_{t+1|t}, \Sigma_{t+1|t})$ We just need to compute its mean and covariance:

$$\mu_{t+1|t} = f(\mu_{t|t}, u_t, 0) \quad (4)$$

$$\Sigma_{t+1|t} = F_t \Sigma_{t|t} F_t^T + Q_t W Q_t^T \quad (5)$$

B. Observation Model

The observation model is similar:

$$z_{t+1} = h(x_{t+1}, v_{t+1}) \quad (6)$$

where $v_t \sim \mathcal{N}(0, V)$. first-order Taylor series approximation gives us

$$h(x_{t+1}, v_{t+1}) \approx h(\mu_{t+1|t}, 0) + H*(x_{t+1} - \mu_{t+1|t}) + R*(v_{t+1} - 0) \tag{7}$$

where $H = \frac{dh(\mu_{t+1},0)}{dx}$ and $R = \frac{dh(\mu_{t+1},0)}{dv}$ The conditional Gaussian distribution is then

$$\mu_{t+1|t+1} := \mu_{t+1|t} + K_{t+1|t}(z_{t+1} - m_{t+1|t}) \tag{8}$$

$$\Sigma_{t+1|t+1} := \Sigma_{t+1|t} K_{t+1|t} S_{t+1|t} K_{t+1|t}^T \tag{9}$$

$$K_{t+1|t} := C_{t+1|t} S_{t+1|t}^{-1} \tag{10}$$

$$m_{t+1|t} \approx h(\mu_{t+1|t}, 0) \tag{11}$$

$$S_{t+1|t} \approx H_{t+1} \Sigma_{t+1|t} H_{t+1}^T + R_{t+1} V R_{t+1}^T \tag{12}$$

$$C_{t+1|t} \approx \Sigma_{t+1|t} H_{t+1}^T \tag{13}$$

C. Stereo Camera Theory

Stereo cameras employ the same principle as our own vision system does in order to gain an implicit evaluation of depth of the scene. If the calibration matrices giving the distances per pixel are known, as well as the baseline distance between the cameras, actual depth can be calculated from the disparity leading us to a rich texture combined depth sensor that can simply be implemented by two cameras. Given that $[u_L, v_L, u_R, v_R]^T$ corresponds to the world $[x, y, z]$ coordinates of a point m, we can say that

$$\begin{bmatrix} u_L \\ u_R \\ v_L \\ v_R \end{bmatrix} = \begin{bmatrix} fs_u & 0 & c_u & 0 \\ 0 & fs_v & c_v & 0 \\ fs_u & 0 & c_u & -fs_u b \\ 0 & fs_v & c_v & 0 \end{bmatrix} \frac{1}{z} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \tag{14}$$

The last two rows of the stereo camera matrix are identical due to the stereo setup. We can then drop the v_R and simply calculate our needed depth z through

$$d = u_L - u_R = \frac{1}{z} fs_u b \tag{15}$$

$$z = \frac{fs_u b}{u_L - u_R} \tag{16}$$

III. TECHNICAL APPROACH

A. Dataset

The data consists of 2 parts.

- IMU: Accelerometer and Gyroscope data from an IMU attached to the car in the form of linear and angular velocities. We will only need to use the pitch and yaw signals as this is 2D from the top down.
- Camera: Stereo camera data given pixel values of a unique set of features. The disparity between pixel values will give the corresponding depth and therefore is used in the optical frame to world coordinate transformation.

B. Localization

To perform the localization, we use our IMU to understand where our position in space is. We can consider our motion model to be a discretized version of nominal kinematics and zero-mean perturbation kinematics; we employ the exponential map to change our perturbations from the $\mathfrak{se}(3)$ to the nominal pose $\in$ SE(3) such that

$$\mu_{t+1|t} = exp(-\tau \hat{u}_t \mu_{t|t}) \tag{17}$$

where $\hat{u}$ corresponds to the hat map (skew symmetric matrix) of the velocity vector $[v_t, w_t]^T$. We mirror the equations employing nominal twist ζ and small perturbation due to noise w to come up with the covariance:

$$\Sigma_{t+1|t} = exp(-\tau \hat{\hat{u}}_t) \Sigma_{t|t} exp(-\tau \hat{\hat{u}}_t)^T + \mathrm{W}\tau^2 \tag{18}$$

where the double hat signifies the twist on the velocity vector such that $\hat{\hat{u}} = \begin{bmatrix} \hat{\omega} & \hat{v} \\ 0 & \hat{\omega} \end{bmatrix}$.

W was taken as a covariance matrix calculated as $50I$.

C. Mapping

Mapping involves transforming the information from our depth sensors into world coordinates, then transforming it into map coordinates. In this case, our depth sensor is our stereo camera set, with selective features indexed uniquely.

We model each landmark as being part of a larger matrix of means and covariances. The values of each mean are initialized by first checking if the landmark is visited before - this is done by checking if the pixel value prior is not equal to -1 and the current mean is (0,0,0) as first initialized. When a new landmark is found, we convert the feature pixels $[u_L, v_L, u_R, v_R]^T$ to world coordinates. We take the left camera's pixels as the basis, and use the difference between the pixel location distance to get the disparity which allows us to get the projection:

$$(X_w, Y_w, Z_w) = H_{cw}^{-1} R_{oc}^{-1} Z_0 K^{-1} (u_L, v_L) \tag{19}$$

where (u_L, v_L) refers to the pixels of the left camera, K refers to the camera matrix, Z_0 refers to the optical axis and therefore the depth (calculated from disparity by $(fs_u * b)/(u_L - u_R)$ where b is the baseline distance between the cameras, and fs_u comes from the camera matrix), R_{oc} refers to the transform that flips the axes from camera to the optical frame, and H_{cw} refers to the world to body to camera frame transform.

When encountering a landmark that has already been seen, we update our corresponding mean and covariance using the Extended Kalman Filter approach. The new predicted observation of pixels given the landmark mean is calculated as the inverse of the previous step:

$$\hat{z}_{t,i} = M\pi(T_{OI}\mu_{t,j})$$

where M refers to the stereo camera calibration matrix in equation 14. We then take the Jacobian of this by taking the

derivative of the projection function, done by simply applying a derivative map such that for a vector $q = [q_1, q_2, q_3, q_4]$ we obtain $d\pi/dq = \begin{bmatrix} 1 & 0 & -q_1/q_3 & 0 \\ 0 & 1 & -q_2/q_3 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & -q_4/q_3 & 1 \end{bmatrix}$. The steps follow:

$$H_t = M\frac{d\pi}{dq}(T_{OI}\mu_{t,j})T_{OI}D \tag{20}$$

$$K_t = \Sigma_t H_t^T(H_t\Sigma_t H_t^T + V)^{-1} \tag{21}$$

$$\mu_{t+1} = \mu_t + DK_t(z_t - \hat{z}_t) \tag{22}$$

$$\Sigma_{t+1} = (I - K_t H_t)\Sigma_t \tag{23}$$

where M is the stereo camera calibration matrix (made by stacking the left camera matrix twice), K_t is the Kalman gain for the time step, D is the dilation matrix, V is the process noise (taken as $1000I$ as the covariances were of similar range), and z refers to the new observed feature. The means of each landmark correspond to the world coordinates, and we plot the X_w, Y_w along with the trajectory world coordinates.

IV. Results

A. Training Set

Figures 1 through 2 show the first dataset (identified as #27) with different values for noise. The video of this dataset indicates that the car drove around a busy street block, returning nearly to the same position as the start. We find that varying w_t did not produce much change if at all, but varying v_t was immediately observable; quite expectedly, we observe landmarks placed very far away which indicates that the observation noise was not taken into consideration during the update.

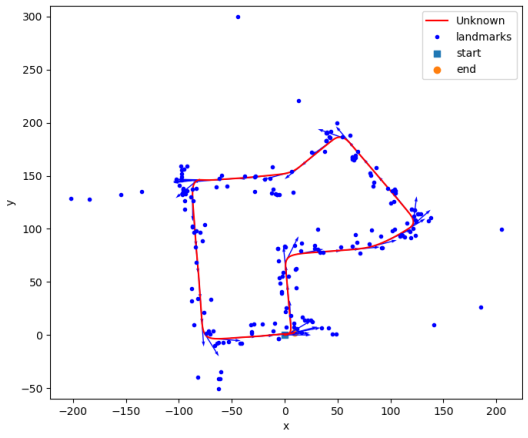

Fig. 1. SLAM Map of dataset #27, w = 500, v = 1000, red - trajectory, blue - landmark locations

Figures 3 and 4 show the second dataset (identified as #42). For the rest of the results, we adopted w = 50, v = 1000. We also show halfway time steps; the red line going to origin was due to the initialization of the IMU mean. The video of this dataset shows the car getting onto a freeway and driving straight down for a while.

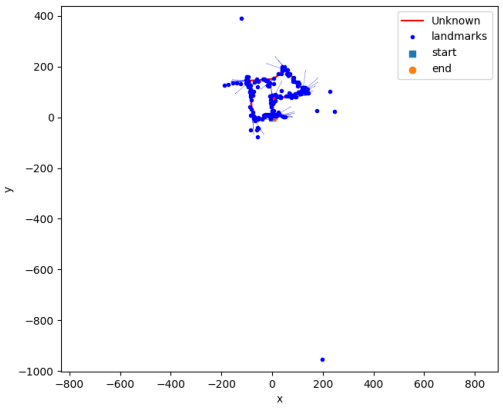

Fig. 2. SLAM Map of dataset #27, w = 50, v = 0.000001 red - trajectory, blue - landmark locations

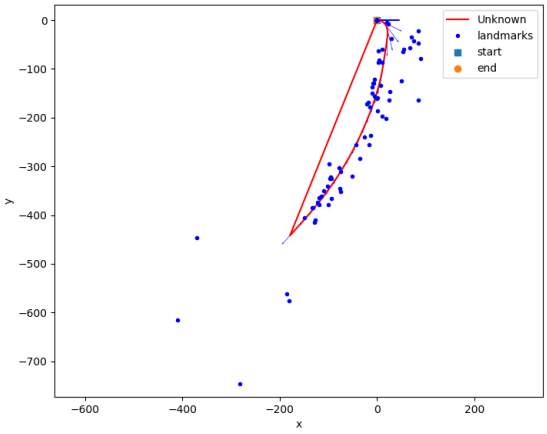

Fig. 3. SLAM Map of dataset #42, half way though, red shows trajectory, blue shows landmark locations

B. Test Results

From Figure 5 onwards, we can see the progression of the test set's map. The video indicates that the test set was a fairly straight road with the car driving down, taking a U-turn, driving up, taking a U-turn again, and ending at the end of the road before the first U-turn.

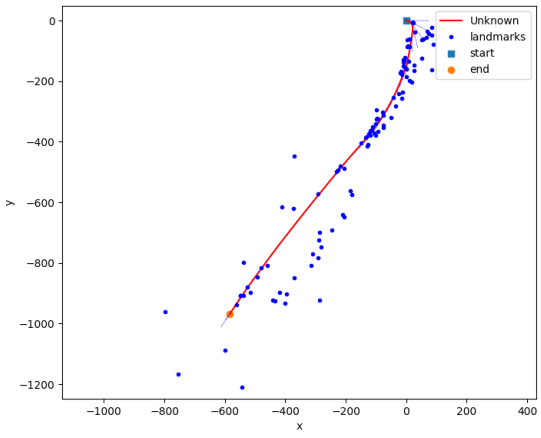

Fig. 4. SLAM Map of dataset #42, last time step, red - trajectory, blue - landmarks

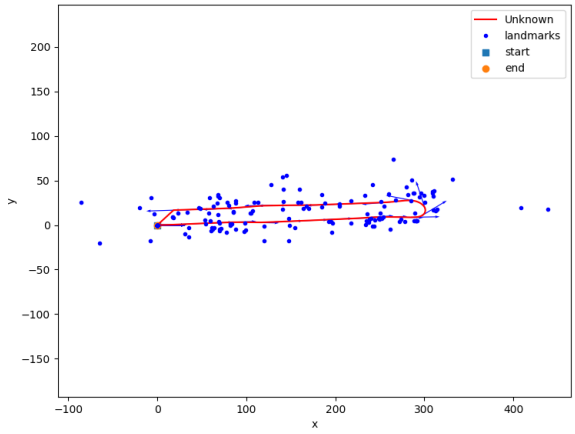

Fig. 5. Test set, halfway through, red - trajectory, blue - landmarks

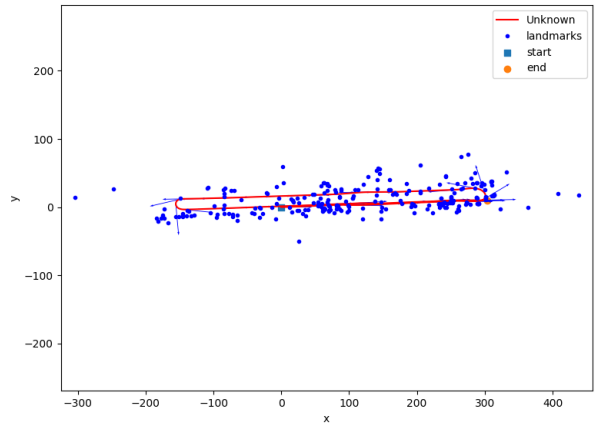

Fig. 6. Test set, last time step, red - trajectory, blue - landmarks

V. CONCLUSIONS

We can see that the stereo cameras gives a good projection of depth of particular landmarks and therefore allows rough understanding of the scene. We can also deduce that our IMU data benefits from the Kalman's prediction step, allowing us to predict trajectory without much drift, showing the car return back to positions visited before. We can see that while the localization and mapping work, the real task ahead becomes updating the trajectory to match these landmark observations, thereby improving the landmark observations as well and providing a feedback loop with which to truly implement the benefits of the Kalman Filter. In the freeway scene, we can see that the points start becoming more and more spread out as time goes on, indicating that the process and measurement noise is perhaps carrying the measurements further away and therefore in dire need of updates to the localization based on these observations.

ACKNOWLEDGMENT

Thanks to Professor Nikolay Atanasov and TAs Tianyu Wang and Ibrahim Akbar.

REFERENCES

[1] Barfoot, T. "State Estimation For Robotics." Cambridge, United Kingdom: Cambridge University Press, 2017