CSE 276A Introduction to Robotics - HW3

Wilson Liao
Electrical and Computer Engineering
PID: A59016079

1 Algorithm

For the Simultaneous Localization and Mapping (SLAM) problem, we adopt Kalman Filter as the backend of our SLAM algorithm. We aim to estimate the robot trajectory and the map of the environment, where it contains the position of the apriltag landmarks without knowing the number of landmarks in advance.

1.1 State vector and observation vector

The state vector x contains the pose of the robot and the observed landmarks in the *world frame*. x can be written as follows:

$$x = [x_r \quad y_r \quad \theta_r \quad x_1 \quad y_1 \quad \theta_1 \quad ...]^T \tag{1}$$

where the size of x is (3n+3, 1), n is the number of total seen landmarks.

The observation vector z contains the pose of current detected landmarks in the *body (robot) frame*. We utilize apriltag detection to get the tag observations in camera frame, and convert them to body frame for simpleness in further computation. z can be written as follows:

$$z = [x'_1 \quad y'_1 \quad \theta'_1 \quad x'_2 \quad y'_2 \quad \theta'_2 \quad ...]^T \tag{2}$$

where the size of z is (3m, 1), m is the number of current detected landmarks and these landmarks should be seen before. For new detected landmarks, we will initialize them (see 1.3.2) to the state vector without updating it; we will update it the next time the robot detect it again.

1.2 Kalman filter matrices

1.2.1 Prediction step

For prediction step in Kalman filter, we need to model the transition of the states using system and motion model. The following is how we model this step and define the system matrix *F* and control matrix *G*:

$$\mu_{t|t-1} = F\mu_{t-1|t-1} + Gu_{t-1} \tag{3}$$

$$\Sigma_{t|t-1} = F\Sigma_{t-1|t-1}F^T + Q \tag{4}$$

where

$$F = I_{(3n+3)\times(3n+3)} \tag{5}$$

$$G = \begin{bmatrix} \Delta t & 0 & 0 \\ 0 & \Delta t & 0 \\ 0 & 0 & \Delta t \\ 0 & 0 & 0 \\ & \cdots & \end{bmatrix}_{(3n+3)\times 3} \qquad u_t = \begin{bmatrix} v_x \\ v_y \\ \omega \end{bmatrix}_{3\times 1} \tag{6}$$

where n is the number of total seen landmarks, Δt is the time (short term, around 0.2 sec) that the current motion u_t (generalized velocity) command last. Thus, in prediction step, the robot state will be updated as:

$$\begin{bmatrix} x_r \\ y_r \\ \theta_r \end{bmatrix}_t = \begin{bmatrix} x_r \\ y_r \\ \theta_r \end{bmatrix}_{t-1} + \begin{bmatrix} v_x \Delta t \\ v_y \Delta t \\ \omega \Delta t \end{bmatrix} \tag{7}$$

while the landmarks mean (not the covariance, covariance will be affect by the system noise *Q*) remain the same.

1.2.2 Update step

For update step in Kalman filter, we need to convert the state space into observation space using observation model. The following is how we model this step and define the measurement matrix *H*:

$$\mu_{t|t} = \mu_{t|t-1} + K_t(z_t - H\mu_{t|t-1}) \tag{8}$$

$$K_t = \Sigma_{t|t-1} H^T (H \Sigma_{t|t-1} H^T + R)^{-1} \tag{9}$$

$$\Sigma_{t|t} = (I - K_t H)\Sigma_{t|t-1} \tag{10}$$

where

$$H = \begin{bmatrix} cos\theta_r & sin\theta_r & 0 & 0 & 0 & 0 & \\ -sin\theta_r & cos\theta_r & 0 & 0 & 0 & 0 & \\ 0 & 0 & 1 & 0 & 0 & 0 & ... \\ 0 & 0 & 0 & cos\theta_r & sin\theta_r & 0 & \\ 0 & 0 & 0 & -sin\theta_r & cos\theta_r & 0 & \\ 0 & 0 & 0 & 0 & 0 & 1 & \\ & ... & & & & & \end{bmatrix}_{3m \times 3m} \begin{bmatrix} -1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & \\ 0 & -1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & \\ 0 & 0 & -1 & 0 & 0 & 1 & 0 & 0 & 0 & ... \\ -1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & \\ 0 & -1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & \\ 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 1 & \\ & & & ... & & & & & & \end{bmatrix}_{3m \times (3n+3)} \tag{11}$$

m is the number of current landmarks (seen before) detected, n is the length of the state. The update step is basically getting relative pose of landmarks to the robot, and convert it from world frame to body frame, which is the predicted observation ($\hat{z}_t$) that need to be compared with real observation (z_t).

$$\hat{z}_t = Hx = \begin{bmatrix} cos\theta_r & sin\theta_r & 0 \\ -sin\theta_r & cos\theta_r & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} -1 & 0 & 0 & 1 & 0 & 0 \\ 0 & -1 & 0 & 0 & 1 & 0 \\ 0 & 0 & -1 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_r \\ y_r \\ \theta_r \\ x_1 \\ y_1 \\ \theta_1 \end{bmatrix} = {}_{\{body\}}R_{\{world\}} \begin{bmatrix} x_1 - x_r \\ y_1 - y_r \\ \theta_1 - \theta_r \end{bmatrix} \tag{12}$$

Note that we only process the detected and seen landmarks in update step; thus, for those detected before but not detected this time, we will not convert them from state space to observation space through *H*, that is, the columns of where the those landmarks are will all be zero in the *H* matrix.

1.3 Initialization

1.3.1 Robot state

At the beginning of the slam algorithm, the state vector is initialized as $x = [0, 0, 0]^T$, as we view the point, where we place our robot, as the origin of the world frame, and the orientation is identity (yaw equals zero).

The covariance is initialized as $\Sigma = diag([10^{-5}, 10^{-5}, 10^{-5}])$ (something small and close to zero) since we are certain about where the robot is in the very beginning.

1.3.2 New landmarks

As mentioned in (1.2.2), we only process the seen landmarks in the update step. For detected landmarks that is new to our Kalman filter, that is, not tracked in current state vector, we initialize

their state (in world frame) based on the robot current pose, and then augment them to our state vector as follows:

$$\begin{bmatrix} x_1 \\ y_1 \\ \theta_1 \end{bmatrix}_{\{world\}} = \begin{bmatrix} cos\theta_r & -sin\theta_r & 0 \\ sin\theta_r & cos\theta_r & 0 \\ 0 & 0 & 1 \end{bmatrix}_{\{world|body\}} \begin{bmatrix} x_1\prime \\ y_1\prime \\ \theta_1\prime \end{bmatrix}_{\{body\}} + \begin{bmatrix} x_r \\ y_r \\ \theta_r \end{bmatrix}_{\{world\}} \tag{13}$$

where the state with $\prime$ is the observation from apriltag detection, we need to convert it to world frame and put into our state vector.

The covariance of the new detected landmarks will be initialized with a high value since we are uncertain with their position for the first time, in our case we put $I_{3\times3}$, and augment it to our state covariance; the covariance of this landmark will shrink quickly the next time we detect it again.

1.4 Noise

The system noise Q depends on how accurate our motion model is. In our case, the motor is not stable, and the calibration is not accurate, we determine our system noise Q relatively larger than the observation noise R:

$$Q = \begin{bmatrix} Q_r & 0 & 0 & \\ 0 & Q_1 & 0 & \\ 0 & 0 & Q_2 & ... \\ & ... & & \end{bmatrix}_{(3n+3)\times(3n+3)}, \quad R = \begin{bmatrix} R_1 & 0 & 0 & \\ 0 & R_2 & 0 & \\ 0 & 0 & R_3 & ... \\ & ... & & \end{bmatrix}_{3m\times3m} \tag{14}$$

where Q_r is the system noise of the robot, in our case $diag([5e^{-3}, 5e^{-3}, 1.5e^{-2}])$, we give larger noise for θ since we think angular estimation is much more noisy; Q_{idx} is the system noise of the landmark, in our case $diag([1e^{-5}, 1e^{-5}, 1e^{-5}])$ since they are static. R_{idx} is the observation noise of the landmarks, in our case $diag([5e^{-4}, 1e^{-3}, 1.5e^{-3}])$, we give larger noise on landmark y and θ since apriltag measurement of x (front distance to tag) is more accurate than y (lateral distance to tag), and angular measurement is noisy.

2 Result

2.1 Visualization

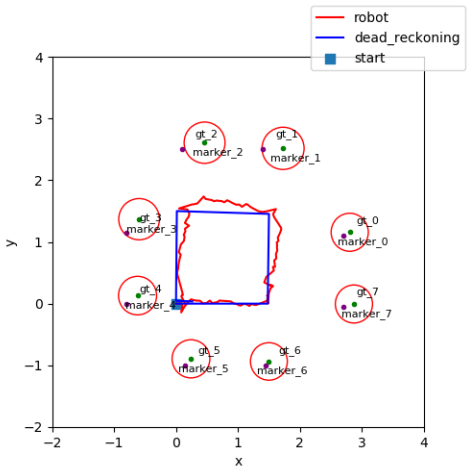

Figure 1: SLAM Result with one round square path

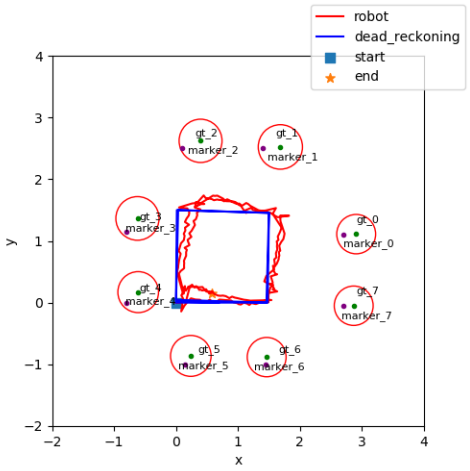

Figure 2: SLAM Result with two rounds square path

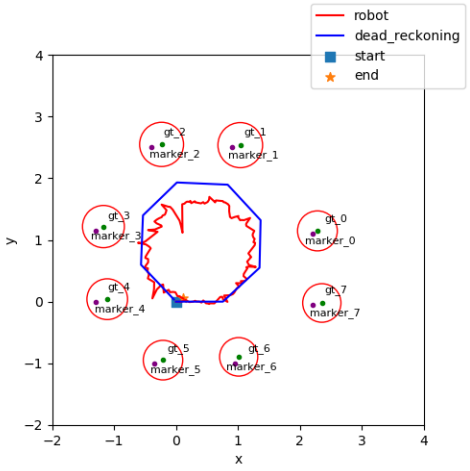

Figure 3: SLAM Result with one round 8-point octagon path

We shift the starting point of the 8-point octagon path with 50 cm in +x direction to avoid collision and better see all the landmarks better.

The green points and the purple points in the figures above represent the estimated position and the actual (ground truth) position of the landmarks respectively.

2.1.1 Quantitative result

	Square_1	Square_2	Octagon_1
Error (m)	0.217	0.213	0.147

Table 1: Average error of landmarks between the measured map and the ground truth

where index below the path shape is the rounds/times for that route.

2.1.2 Discussion

We can see from the table that in terms of driving times through the route, driving multiple times improves the estimation, which is pretty straight forward, since we get more observations to update, and multiple path means we can perform loop closure and the robot is able to know where it should be when it sees the seen landmarks.

In terms of shape of the path, we can see that driving with octagon (8-point) path yields a lower error of landmark estimation, since with this path, the robot is able to see the landmark with better viewing angle, and thus provides better observation; while driving in square path, some of the landmarks is

actually on the side of the camera view, which severely affects the detection quality due to limited viewing angle and the distortion of the camera.

Nevertheless, from the figures/result above, we can still see that both the estimation of the trajectory and the map is not perfect. There are several factors that can be modified to improve the performance:

- System noise, observation noise: These parameter should be tuned since we do not know how noisy is our current system.
- Calibration for motion command: Although Kalman filter will handle the localization of the robot, but if the motion model sucks, it's still hard to provide a good estimation.
- Better backend for SLAM: Kalman filter assumes that both of our motion and observation model are linear, if there's non-linearity in our system, Kalman filter can not capture it well. We can use non-linear filters like EKF or UKF or even sampling-based filter like particle filter.

In summary, appropriate route shape and multiple drive times through the environment improves the quality of the map.

3 Video Demo

Square path video link

Octagon path video link