Tutorial on KF SLAM

Author: MINA HENEIN

Code Structure

What is SLAM?

"SLAM is concerned with the problem of building a map of an unknown environment by a mobile robot while at the same time navigating the environment using the map."

Søren Riisgaard and Morten Rufus Blas.

"A Tutorial Approach to Simultaneous Localization and Mapping", 2005

Calculates the robot new pose given previous pose and motion

Input:

- absolute coordinate of robot

→ [x1,y1,theta1]

- motion command with respect to robot frame

→ [dx, dy, dtheta]

Output:

- absolute coordinate of robot next pose

← [x2,y2,theta2]

```
def Relative2AbsolutePose (robot_abs, u):

    x1 = robot_abs[0][0]
    y1 = robot_abs[1][0]
    theta1 = robot_abs[2][0]
    dx = u[0][0]
    dy = u[1][0]
    dtheta = u[2][0]

    #R is the transition matrix of robot frame
    R = [[np.cos(theta1), -np.sin(theta1), 0],
         [np.sin(theta1), np.cos(theta1), 0],
         [0, 0, 1]]

    #Calculate Jacobian H1 with respect to X1
    H1 = [[1, 0, -dx*np.sin(theta1)-dy*np.cos(theta1)],
          [0, 1,  dx*np.cos(theta1)-dy*np.sin(theta1)],
          [0, 0, 1]]

    #Calculate Jacobian H2 with respect to u
    H2 = [[np.cos(theta1), -np.sin(theta1), 0],
          [np.sin(theta1), np.cos(theta1), 0],
          [0, 0, 1]]

    next_robot_abs = np.dot(R,u) + robot_abs

    return next_robot_abs, H1, H2
```

Calculates Landmark's absolute coordinate

Input:

- robot's absolute coordinate

→[x, y, theta]

- landmark's measurement with respect to robot frame

→[x, y]

Output:

- landmark's absolute coordinate

←[xl, yl]

```python
def Relative2AbsoluteXY(robot_abs,landmark_meas_xy):

    x1 = robot_abs[0][0]
    y1 = robot_abs[1][0]
    theta1 = robot_abs[2][0]
    x2 = landmark_meas_xy[0]
    y2 = landmark_meas_xy[1]

    landmark_meas = [[x2],
                     [y2],
                     [1]]

    #R is the transition matrix to robot frame
    R = [[np.cos(theta1), -np.sin(theta1), 0],
         [np.sin(theta1), np.cos(theta1), 0],
         [0, 0, 1]]

    #Calculate Jacobian H1 with respect to X1
    H1 = [[1, 0, -x2*np.sin(theta1)-y2*np.cos(theta1)],
          [0, 1,  x2*np.cos(theta1)-y2*np.sin(theta1)]]

    #Calculate Jacobian H2 with respect to X2
    H2 = [[np.cos(theta1), -np.sin(theta1)],
          [np.sin(theta1),  np.cos(theta1)]]

    landmark_abs = np.array(np.dot(R,landmark_meas))
                        + np.array(robot_abs)

    return [landmark_abs[0][0],landmark_abs[1][0]], H1, H2
```

Kalman Filter

Time Update ("Predict")

(1) Project the state ahead

$$\hat{x}_k^- = A\hat{x}_{k-1} + Bu_k$$

(2) Project the error covariance ahead

$$P_k^- = AP_{k-1}A^T + Q$$

Measurement Update ("Correct")

(1) Compute the Kalman gain

$$K_k = P_k^- H^T (HP_k^- H^T + R)^{-1}$$

(2) Update estimate with measurement z_k

$$\hat{x}_k = \hat{x}_k^- + K_k(z_k - H\hat{x}_k^-)$$

(3) Update the error covariance

$$P_k = (I - K_k H)P_k^-$$

Initial estimates for $\hat{x}_{k-1}$ and P_{k-1}

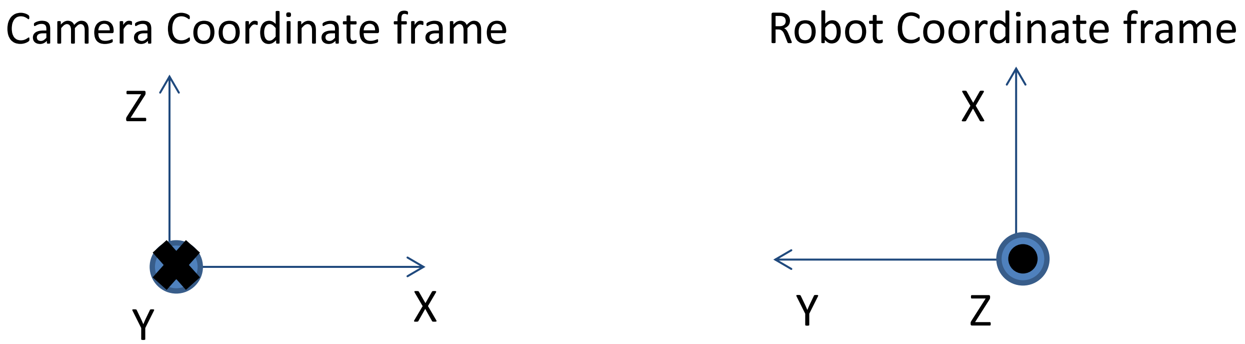
Algorithm Skeleton

- 1- Subscribe to the /odom & /cylinderTopic topics (and/or to any other topic you think might be useful).

- 2- Define callback functions to call whenever you have a new message received on the subscribed topics.

- Def CallbackOdometryMotion:

- Def CallbackLandmarkMeasurement:



- 3- Write down the Kalman Filter equations in matricial form.

- Def KF_predict (motion, motionCovariance):
- a- [nextRobotAbsPose, F, W] = move robot given robot current pose and u.
- b- Set priorStateMean(0:3, 0) to nextRobotAbs.
- c- Set robot new pose to priorStateMean(0:3, 0).

- d- Set robot prior covariance to $\overline{\Sigma}_t$
- e- Set priorStateCovariance(ii, jj) for ii, jj = 0,1,2 to

$$\overline{\Sigma} = F\,\Sigma_r\,F^T + W\,\Sigma_{motion}\,W^T$$

- f- Return priorStateMean and priorStateCovariance

- Def KF_update (measurement, measurementCovariance):
- a- [landmarkAbs, G1, G2] = robot inverse sense given robot current pose and measurement.
- b- Determine if it’s a new landmark or a measurement of an already existing landmark.

- i- if new → augment the stateMean by the new landmarkAbs and augment $\bar{\Sigma}_t$ to $[\Sigma_r, \Sigma_{xl}; \Sigma_{xl}^T, \Sigma_{\boldsymbol{l}}]$

  where $\Sigma_l = G_1 \Sigma_r {G_1}^T + G_2 \Sigma_{meas.} {G_2}^T$

  and $\Sigma_{xl} = G_1 [\, \Sigma_r \; \Sigma_{rm.}]$

  and $\Sigma_{rm} = \Sigma_{t-1}(1: dimR, \quad dimR + 1: end)$

- ii- if old → stateMean and stateCovariance don't change (These will change later at update phase)

- c- $[z_{exp.}, H_r, H_l]$ = Absolute2RelativeXY given robot current absolute pose and landmark absolute position
- d- Build C matrix = $[H_r, 0, \ldots, 0, H_l]$

- e- Compute Kalman gain :

$$K_t = \bar{\Sigma}_t C_t^T (C_t \bar{\Sigma}_t C_t^T + Q_t)^{-1}$$

- f- Update posteriorStateMean:

$$\mu_t = \bar{\mu}_t + K_t (z_t - C_t \bar{\mu}_t)$$

- g- Set robot new pose to posteriorStateMean(0:3,0)
- h- Update posteriorStateCovariance: $\Sigma_t = (I - K_t C_t)\bar{\Sigma}_t$
- i- Set robot new covariance to Σ_t (ii, jj) for ii, jj = 0,1,2
- j- Return posteriorStateMean, posteriorStateCovariance

Some Key points to consider:

- The state mean in KF is the robot pose augmented with seen landmark positions (in world coordinates)

$$\mu = [r; l_1; l_2 ;\ l_3 ; \dots ;\ l_{n,}]$$

- The state covariance is a square matrix with the form

$$\boldsymbol{\Sigma} = [\Sigma_{RR}, \Sigma_{RM} ;$$
$$\Sigma_{\boldsymbol{MR}}, \Sigma_{MM}]$$

- Motion only affects robot's mean and covariance (μ_r, Σ_r) (in an absolute representation of the map)
- Measurement affects the whole state through kalman update

- When a landmark is seen for the first time, the state mean is augmented with the new landmark absolute position estimate (output of the function Relative2AbsoluteXY(robotpose, measurement))

$$\mu = [r;\ l_1;\ l_2\ ;\ l_3\ ]$$

- When a landmark is seen for the first time, the state covariance is augmented as follows: $[\Sigma_r, \Sigma_{xl}; \Sigma_{xl}^T , \Sigma_{\boldsymbol{l}}]$
- where

$$\Sigma_l = G_1 \Sigma_r \, {G_1}^T + G_2 \Sigma_{meas.} \, {G_2}^T$$

$$\Sigma_{xl} = G_1 [\, \Sigma_r \, \Sigma_{rm.}]$$

$$\Sigma_{rm} = \Sigma_{t-1}(1 : dimR, dimR + 1 : end)$$

- When a landmark is re-seen, the state mean and covariance only change at the update stage through the kalman gain.

- To perform an update, you need to compute a correction, so you need a measurement estimate z_{exp}. output of the function Absolute2RelativeXY(robot pose, landmarkAbs).

- To update, you also need to compute Kalman gain, so you need C matrix = [$H_{\boldsymbol{r}}$, 0, …, 0, $H_{\boldsymbol{l}}$] where $H_{\boldsymbol{r}}$ and $H_{\boldsymbol{l}}$ are output of the Absolute2RelativeXY(robotpose, landmarkAbs).

REFERENCES

- Søren Riisgaard and Morten Rufus Blas. “A Tutorial Approach to Simultaneous Localization and Mapping”, 2005
- Michael Kaser & Frank Dellaert. iSAM: Incremental Smoothing and Mapping,2008. url:http://people.csail.mit.edu/kaess/isam/doc/Tutorial.html
- Graph SLAM maps, SLAM map building. url:http://www.mrpt.org/Graph-SLAM_maps
- Kalman Filter Basics. url:http://biorobotics.ri.cmu.edu/papers/sbp_papers/integrated3/kleeman_kalman_basics.pdf