Sistemas Autónomos / Autonomous Systems

Lisbon, Portugal
Projecto [L14] Drone Localization using the Extended Kalman Filter Method
João Ferreira (78101)
Pedro Mendes (81046)
Miguel Malaca (81702)
João Rosa (84089)

Introduction

An autonomous robot should be able of self-localizing in order to find it's own pose in the environment. For that, it is required to compare the environment scan with a map. In order to achieve this goal, the Extended Kalman Filter Algorithm is implemented using the observations os a Depth Camera and an Inertial Measurement Unit (IMU). Thus, it is necessary to evaluate the accuracy and the performance of the algorithm implemented.

Main Objectives

1. Estimate in real time the position and orientation of a drone using the Extended Kalman Filter Method and the available sensors (depth camera and IMU)
2. Evaluate estimation accuracy for absolute localization

Extended Kalman Filter (EKF)

The goal of the EKF algorithm is the estimation of the current state of a time-variant system considering noisy observations of this state. It is divided into two steps:

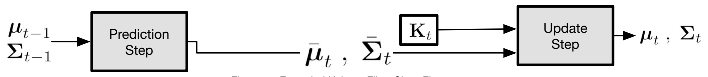

Figure 1 - Extended Kalman Filter Chart Flow

Prediction Step:

This step predicts the current state, through:

$$\bar{\boldsymbol{\mu}}_t = \mathbf{A}_t \boldsymbol{\mu}_{t-1} + \mathbf{B}_t \mathbf{u}_t$$
$$\bar{\boldsymbol{\Sigma}}_t = \mathbf{A}_t \boldsymbol{\Sigma}_{t-1} \mathbf{A}_t^T + \mathbf{R}_t$$

Update Step:

The state is updated according to a new measurement and it is compared with what the robot should be observing if the predicted state is correct.

$$\mathbf{K}_t = \bar{\boldsymbol{\Sigma}}_t \mathbf{H}_t^T (\mathbf{H}_t \bar{\boldsymbol{\Sigma}}_t \mathbf{H}_t^T + \mathbf{Q}_t)^{-1},$$
$$\boldsymbol{\mu}_t = \bar{\boldsymbol{\mu}}_t + \mathbf{K}_t(\mathbf{z}_t - \mathbf{h}(\bar{\boldsymbol{\mu}}_t)),$$
$$\boldsymbol{\Sigma}_t = (\mathbf{I} - \mathbf{K}_t \mathbf{H}_t)\bar{\boldsymbol{\Sigma}}_t,$$

Implementation

Motion Model:

Three coordinates are needed to describe the state: the position (x,y), and the orientation (θ) and the respective velocities in each component.

$$\mu_t = \begin{bmatrix} x_t \\ \frac{dx_t}{dt} \\ y_t \\ \frac{dy_t}{dt} \\ \theta_t \\ \frac{d\theta_t}{dt} \end{bmatrix}$$

A first order system is considered without a control signal (u) to describe the motion of the robot. The velocity in each component is approximately constant, with the introduction of Gaussian noise in the velocity components.

$$\begin{cases} x_t = x_{t-1} + \frac{dx_{t-1}}{dt}\Delta t \\ \frac{dx_t}{dt} = \frac{dx_{t-1}}{dt} + \varepsilon_x \\ y_t = y_{t-1} + \frac{dy_{t-1}}{dt}\Delta t \\ \frac{dy_t}{dt} = \frac{dy_{t-1}}{dt} + \varepsilon_y \\ \theta_t = \theta_{t-1} + \frac{d\theta_{t-1}}{dt}\Delta t \\ \frac{d\theta_t}{dt} = \frac{d\theta_{t-1}}{dt} + \varepsilon_\theta \end{cases}$$

Measurement Model:

The sensor used is a RGB-D camera that measures projective distances from an imaginary plane orthogonal to the camera's orientation and the environment it is in.

When all the angles, except for the yaw, are 0º, the values to obtain are trivial. They are just the values of the pixels corresponding to the horizontal line.

When the drone moves, it tilts. This creates a problem since the approach used is a 2D one. With the tilt, the distances in the horizontal plane should be measured anyway. Considering the projective geometry of the camera, and the pin-hole model, the intersection between the horizontal plane and the focal plane corresponds to the line that should be obtained in the image.

Having the pixel coordinates of the horizontal line, the real distance needs to be computed. Using the intrinsic and the extrinsic parameters of the camera, the coordinates of the horizontal line can be computed. The distances comes directly from X.

$$w \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \mathbf{K}\begin{bmatrix} \mathbf{R} & \mathbf{0} \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

To serve as ground truth or a measure, the yaw, which is equivalent to the orientation, can be obtained from the IMU. Having the projection of the heading vector (X axis) of the body in the horizontal plane and in the X axis of the world, the angle between both can be easily calculated through the cosine.

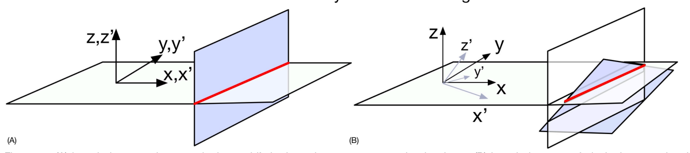

Figure 2 - (A) in red, the array that contais the world's horizontal component seen by the drone, (B) in red, the array of pixels that contains the world horizontal component of the plane when the drone gets a YX rotation.

Observation Model:

The goal is to determine the distances between an imaginary plane orthogonal to the camera's principal point and the obstacles seen in the camera's field of view when the robot's pose is the one predicted by the prediction step of EKF that is given by:

$$h(\overline{\mu}) = \left\| p_p^* - p_i^* \right\| =$$
$$= \sqrt{(x_r^* + dr\cos(\beta) - x_i^*)^2 + (y_r^* + dr\sin(\beta) - y_i^*)^2}$$

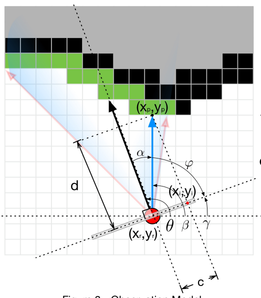

Figure 3 - Observation Model

Experimental Results

Camera Rotation:

A camera rotation was simulated and the pixels corresponding to the horizontal plane were successfully obtained.

Complexity:

The time complexity is highly dependent on the complexity of the observation model, due to the high number of iterations.

It depends on the number of points obtained from the image in the camera and the distance from the camera to the obstacle.

The time complexity increases approximately linear with the increase in the number of pixels computed.

The figure 4 representes the time elapsed to compute the observation model depending on the number of points obtained from the camera's image.

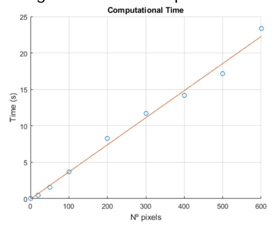

Figure 4 - Computational Time

Simulation:

Using the first alternative where the localization algorithm does not use data from the IMU, it was not possible to find a combination of initial conditions and covariance values that would make the algorithm converge to the real pose of the robot, as it can be seen in the figure 5.

Using the second alternative, where the rotation yaw was used in the observation, some tests in different maps were made.

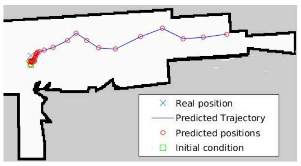

Figure 5 - Simulation performed without IMU

Square Map:

Analysing the figure 6, it is possible to see that the prediction of the EKF are good and the robot follows the real path. However, this map represents a lot of ambiguities.

Real Map:

Some tests were performed in order to find a good configuration for the covariance matrixes that would converge the algorithm for different paths tested.

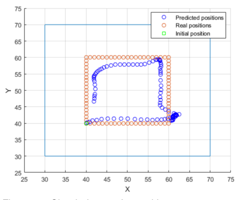

Figure 6 - Simulation performed in a square map

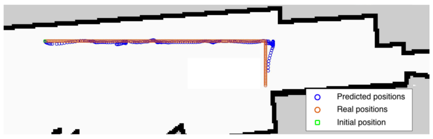

Figure 7 - Simulation performed in the real map

Real Time:

The algorithm was tested using real time results and it did not converge to the real position, because of:

- the map, because there is a big difference when compared to the reality;
- the angle yaw that was constantly drifting due to erros of the magnetometer;
- the camera that could only measure distances bigger than 40/ 50 cm.

Conclusions

There must be a trade-off between the number of pixels obtained and the time that it takes to compute the observation model, in order not to compromise the localization process and without taking too much time;

There are errors associated that provoke the divergence of the algorithm.

Without the IMU, the model couldn't converge in simulations;

The effect of choosing the movement model of first order can be detected because sometimes the robot has a movement similar to the uniform movement;

The covariance matrices can be adjusted to increase the uncertainty of the state and observation in order to be able to converge the algorithm.