SLAM performance predicition

Mauro Tellaroli
Statistical Methods for Maching Learning
Università degli Studi di Milano

I declare that this material, which I now submit for assessment, is entirely my own work and has not been taken from the work of others, save and to the extent that such work has been cited and acknowledged within the text of my work. I understand that plagiarism, collusion, and copying are grave and serious offences in the university and accept the penalties that would be imposed should I engage in plagiarism, collusion or copying. This assignment, or any part of it, has not been previously submitted by me or any other person for assessment on this or any other course of study.

1 Introduction

In the field of autonomous mobile robot, the Simultaneous Localization And Mapping (SLAM) represents one of the most significant problems. As the name suggests, the goal of the SLAM is incrementally obtain from a mobile robot a map of the environment and simultaneously locate the robot's position inside it. The difficulty of the SLAM problem is that to precisely localize itself, the robot needs an accurate map, while to obtain an accurate map, it needs an accurate localization inside the map. This type of problems is called chicken-and-egg problem.

1.1 SLAM methods

In the last 30 years several approaches have been developed to solve the SLAM problem leading to methods extensively used in both common and industrial scenarios. The main paradigms [4] used are:

- Extended Kalman Filter: a vector containing the estimates of the robot position and the environment landmark is used. In addition, a matrix composed of the correlations between the positions and landmark estimates is also used. Both the vector and the matrix are updated using the extended Kalman Filter.
- Particle Filter: the Monte Carlo method is used to sample the possible states of the robot. A random sample is called a particle and it's composed of the robot position and a map estimate. The main steps are: predict the robot's pose for each particle, update particle weights based on sensor data, resample particles according to their weights, and update each particle's map.
- Graph-based: a graph is used where nodes represent robot poses or landmarks, and edges represent spatial constraints derived from sensor observations or odometry. The SLAM problem is formulated as a graph optimization task, where the goal is to find the configuration of nodes that best satisfies all constraints, typically using nonlinear optimization techniques.

1.2 Accuracy of SLAM methods

The heterogeneity of SLAM methods makes it difficult to find a measure that can evaluate their godness. Many approaches rely on manual evaluation or utilize information derived from the specific algorithm. Nowadays the most used metric is the Absolute Pose Error (APE)[1]. The APE measures the difference between the estimated trajectory produced by the SLAM algorithm and the ground truth trajectory, focusing on the global consistency of the estimated poses. The big advantage of the APE is that it allows for a quantitative assessment of how well the SLAM algorithm reconstructs the robot's trajectory over time, independent of the internal workings of the specific method used.

The APE is composed of two sub-metrics:

1. The Absolute Translational Error (ATE): it measures the difference in position, typically using the Euclidean distance between corresponding poses.

2. The Absolute Rotational Error (ARE): it measures the difference in orientation, often computed as the angular distance between corresponding rotations.

Figure 1 and 2 show respectively the ATE variation over time and the difference between the ground truth and the robot estimated trajectory during an environment exploration.

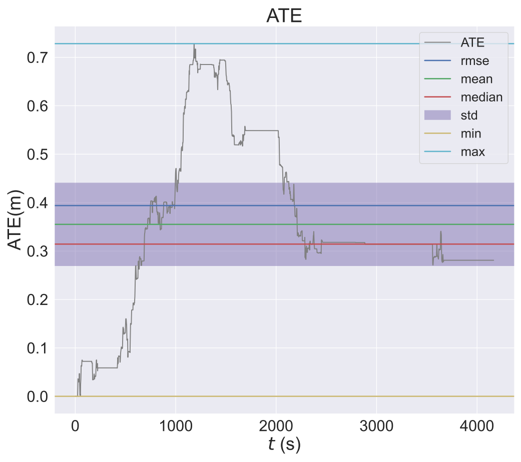

Figure 1: ATE variation over time

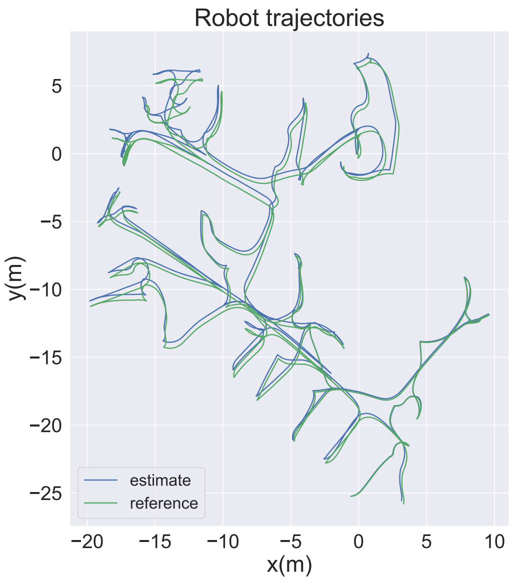

Figure 2: Difference between real (in green) and estimated (in blue) trajectory

1.3 APE calculation

As anticipated in the previous section, the APE metric is based on the difference between the estimated robot poses and the corresponding ground truth poses.

Formally, let $\boldsymbol{x}_{1:T}$ be the poses of the robot estimated by a SLAM algorithm from time step 1 to T, and let $\boldsymbol{x}^*_{1:T}$ be the corresponding ground truth poses.

Let $\delta_{i,j}$ be the relative transformation that moves the robot from pose x_i to x_j:

$$\delta_{i,j} = x_j \ominus x_i$$

$$\delta^*_{i,j} = x^*_j \ominus x^*_i$$

Let $trans(\cdot) \in \mathbb{R}^2$ be the function that extracts the translational (position) component from a transformation, and let $rot(\cdot) \in \mathbb{R}^2$ extract the rotational (orientation) component as Euler angle.

Finally, to compute the APE:

$$\text{APE} = \underbrace{\frac{1}{N}\sum_{i,j} trans(\delta_{i,j} \ominus \delta^*_{i,j})^2}_{\text{ATE}} + \underbrace{\frac{1}{N}\sum_{i,j} rot(\delta_{i,j} \ominus \delta^*_{i,j})^2}_{\text{ARE}}$$

1.4 Project goal

Typically, the APE is calculated a posteriori, after the robot has explored the entire environment. The main goal of this project is to propose a predictor for the SLAM performance, specifically the APE, before running the algorithm or during its execution, using features derived from the input data or intermediate results. This type of prediction can improve the robustness and efficiency of autonomous mobile systems since the position error (APE) can be used to correct the robot trajectory.

The work in [3] proposes two main APE predictors, one based on linear regression, and another one based on a Gaussian Process via an RBF Kernel. Although Gaussian Process regression showed slightly better performance compared to linear regression model, the authors preferred linear regression because they considered it a more interpretable model. In both cases, a feature extractor is used [2] obtaining a set of structural features to represent the environment as a vector, which is then passed to the model.

2 Model

In this project, a modern Convolutional Neural Network (CNN) is used, in order to delegate the feature extraction to the model and directly using the environment floorplan image. Formaly, the model h we want to learn can be expressed as follows:

$$h : [0, 255]^{n,n} \times \mathbb{R}^+ \to \mathbb{R}^2$$

Where $[0, 255]^{n,n}$ is a floorplan image, encoded as a $n \times n$ grayscale image, $\mathbb{R}^+$ is the area of the floorplan expressed in m^2 and $\mathbb{R}^2$ contains ATE (m) and ARE (rad).

2.1 Data preprocessing

References

[1] Rainer Kümmerle et al. “On Measuring the Accuracy of SLAM Algorithms”. In: *Autonomous Robots* 27 (Nov. 2009), pp. 387–407. DOI: 10.1007/s10514-009-9155-6.

[2] Matteo Luperto and Francesco Amigoni. “Extracting Structure of Buildings Using Layout Reconstruction”. In: *Intelligent Autonomous Systems 15*. Ed. by Marcus Strand et al. Cham: Springer International Publishing, 2019, pp. 652–667. ISBN: 978-3-030-01370-7.

[3] Matteo Luperto, Valerio Castelli, and Francesco Amigoni. “Predicting Performance of SLAM Algorithms”. In: *CoRR* abs/2109.02329 (2021). arXiv: 2109.02329. URL: https://arxiv.org/abs/2109.02329.

[4] Cyrill Stachniss, John J. Leonard, and Sebastian Thrun. “Simultaneous Localization and Mapping”. In: *Springer Handbook of Robotics*. Ed. by Bruno Siciliano and Oussama Khatib. Cham: Springer International Publishing, 2016, pp. 1153–1176. ISBN: 978-3-319-32552-1. DOI: 10.1007/978-3-319-32552-1_46. URL: https://doi.org/10.1007/978-3-319-32552-1_46.