Lab 1: SLAM for Karel in 1D

Simultaneous Localization and Mapping

César Borja and Nerea Gallego

February 11, 2024

1 Introduction

This report contains the explanations of the different tasks of Lab 1. We show the completition of the SLAM algorithm in 1D, a second implementation that uses Sequential Map Joining and a computational analysis.

2 SLAM in 1D algorithm

We were asked to complete **update_map** and **add_new_features** functions.

add_new_features This function is responsible for adding into the map those features seen by the sensor that are not already in the map. This is done before each step of the robot.

Being R_k the robot in step k and $\{F_1, F_m\}$ the features (*beepers*) already included in the map at step k, the map at step k is build as:

$$\mathbf{x}_{k|k-1} = \begin{bmatrix} x_{R_0R_k} \\ x_{R_0F_1} \\ x_{R_0F_2} \\ \vdots \\ x_{R_0F_m} \end{bmatrix}, \mathbf{P}_{k|k-1} = \begin{bmatrix} \sigma^2_{R_k} & \rho_{R1}\sigma_{R_k}\sigma_{F_1} & \cdots & \rho_{Rm}\sigma_{R_k}\sigma_{F_m} \\ \rho_{R1}\sigma_{R_k}\sigma_{F_1} & \sigma^2_{F_1} & \cdots & \rho_{1m}\sigma_{F_1}\sigma_{F_m} \\ \vdots & \vdots & \ddots & \vdots \\ \rho_{Rm}\sigma_{R_k}\sigma_{F_m} & \rho_{1m}\sigma_{F_1}\sigma_{F_m} & \cdots & \sigma^2_{F_m} \end{bmatrix} \tag{1}$$

Where $\rho_{Ri}|i \leq m$ is the correlation between the robot and the feature i and $\rho_{ij}|i,j \leq m$ is the correlation between the features i and j.

Then, when the sensor see n new features that are not in the map, we want to add them in the map as follows:

$$\mathbf{x}_k = \begin{bmatrix} x_{R_0R_k} \\ \mathbf{x}_{R_0F} \\ x_{R_0R_k} + \mathbf{z}_n \end{bmatrix} = \begin{bmatrix} \mathbf{I}_{m\times m} & \\ \mathbf{1}_{n\times 1} & \mathbf{0}_{n\times m-1} \end{bmatrix} \mathbf{x}_{k|k-1} + \begin{bmatrix} \mathbf{0}_{m\times n} \\ \mathbf{I}_{n\times n} \end{bmatrix} \mathbf{z}_n = \mathbf{A}\mathbf{x}_{k|k-1} + \mathbf{B}\mathbf{z}_n \tag{2}$$

$$\mathbf{P}_k = \mathbf{A}\mathbf{P}_{k-1|k}\mathbf{A}^T + \mathbf{B}\mathbf{R}_n\mathbf{B}^T \tag{3}$$

Where

$$\mathbf{z}_n = \begin{bmatrix} z_1 \\ \vdots \\ z_n \end{bmatrix}, \mathbf{R}_n = \begin{bmatrix} \sigma^2_{z_1} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \sigma^2_{z_n} \end{bmatrix} \tag{4}$$

We implement this linear combination to get the new state of the map.

update_map This function updates the map when the sensor has seen features that are already in the map, using Kalman Filter.

To do so, first we need to create $\mathbf{H}_k$

$$\mathbf{H}_k = \begin{bmatrix} -1 & 0 & \cdots & \cdots & \cdots & \cdots & 0 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ -1 & 0 & \cdots & 0 & 1 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ -1 & 0 & \cdots & \cdots & 0 & 1 & 0 & 0 \end{bmatrix}_{f \times m+1} \tag{5}$$

Where f is the number of features seen by the sensor in the step k that are already in the map. Each row has a 1 in the column corresponding with the id. of the feature.

Then we use Kalman Filter to update the map $(\mathbf{x}_{k|k}, \mathbf{P}_{k|k})$ as follows

$$\begin{aligned} \tilde{y}_k &= z_k - H_k \hat{x}_{k|k-1} \\ S_k &= H_k P_{k|k-1} H_k^T + R_k \\ K_k &= P_{k|k-1} H_k^T S_k^{-1} \\ x_{k|k} &= x_{k|k-1} + K_k \tilde{y}_k \\ P_{k|k} &= (I - K_k H_k) P_{k|k-1} \end{aligned}$$

2.1 Results

Once we have implemented these functions, we can run the MATLAB code to SLAM in 1D. Fig. 1 shows the correlation matrix, the estimated position of the robot and the estimated map (position of every beeper) at the end of the execution.

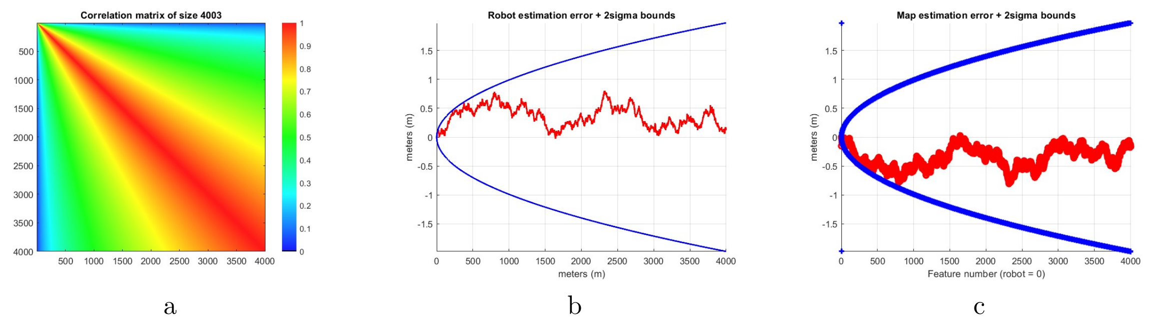


Figure 1: Correlation matrix (a), robot estimation error (b) and map estimation error (c) after the execution of the 1D SLAM. In b, the y axis represents the robot location error in meters. In c, every red dot represents a *beeper*. The y axis represents the location error of each *beeper*. Blue points in b and c represent the gaussian error.

3 Sequential Map Joining

Sequential Map Joining consists in building n sequential maps and join them together to obtain the full map. The objective of this approach is to reduce the total computational cost.

The sequential map joining it is computed as linear combination of gaussian variables (the maps to join, M). To construct the global map, it is necessary to concatenate two maps to another iterating over all local maps. The strategy followed to get the global map is to iteratively concatenate the maps in position two in advance to the map in first position of the list.

The algorithm used to construct the map is implemented as follows:

Algorithm 1: Sequential Map Joining

```
Data: List of map,s $M$
Result: Full map, $m_f$
1  $x_f = M[1].x$
2  $P_f = M[1].P$
3  $m_f.R0 = M[1].R0$                    // We keep the reference location of the first map
4  for  m in M[2:end] do
5      $x_{aux} = m.x$
6      $P_{aux} = m.P$
7      $A =$ eq.6
8      $B =$ eq.6
9      $x_f = Ax_f + Bx_{aux}$
10     $P_f = AP_fA^T + BP_{aux}B^T$
11 $m_f.x = x_f$
12 $m_f.P = x_P$
```

$$\mathbf{A} = \begin{bmatrix} \mathbf{I}_{k_f \times k_f} & \\ \mathbf{1}_{k_{aux}-1\times 1} & \mathbf{0}_{k_{aux}-1\times k_f-1} \end{bmatrix}, \mathbf{B} = \begin{bmatrix} 1 & \mathbf{0}_{1\times k_{aux}-1} \\ \mathbf{0}_{k_f-1\times k_{aux}} & \\ \mathbf{0}_{k_{aux}-1\times 1} & \mathbf{I}_{k_{aux}-1\times k_{aux}-1} \end{bmatrix} \tag{6}$$

Where k_f and k_{aux} correspond to the amount of features (including the robot) of the final map and the auxiliary map to be joined.

4 Computational analysis

In this section we will compare the computational cost of the single map approach and the sequential map joining approach. For this experiment we have set the number of steps to 4000. For the map joining, we have used 4 maps of 1000 steps each.

Fig.2 shows the cost per step of both methods. We can see that the cost per step increases exponentially with the number of steps saved in the map. For that reason, the cost per step in the joined map tends to be very low. Those peaks on the joined map graph appears because of the join operation of two maps. The computational time of the join operation is also exponential based on the size of the maps to be joined. Also, the Fig.3 shows the cumulative cost comparison. We can see that the total spent time for the map join approach is $\sim 4\%$ of the single map approach. This is because we are "mitigating" the exponential cost by building sequential smaller maps.

We can conclude that the sequential map joining method is a better way to build the map in terms of computational cost.

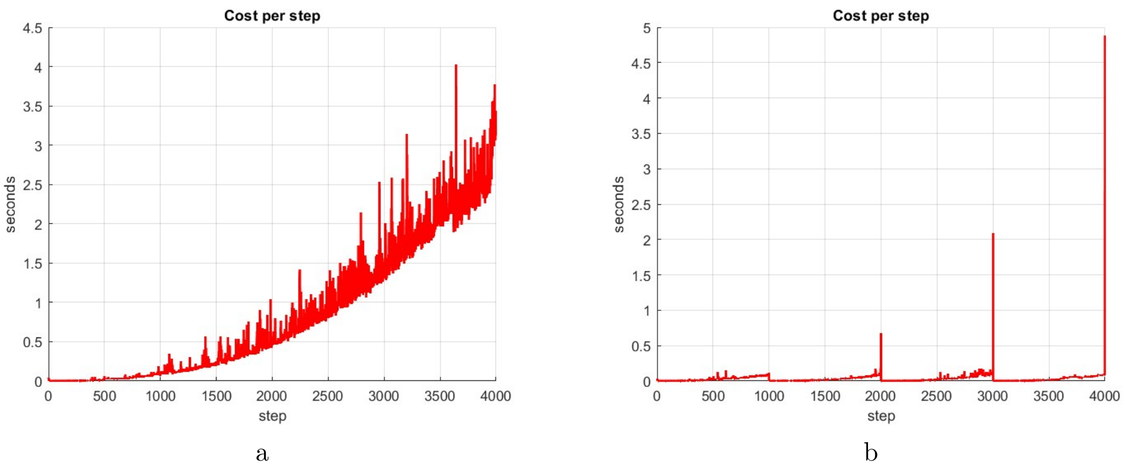
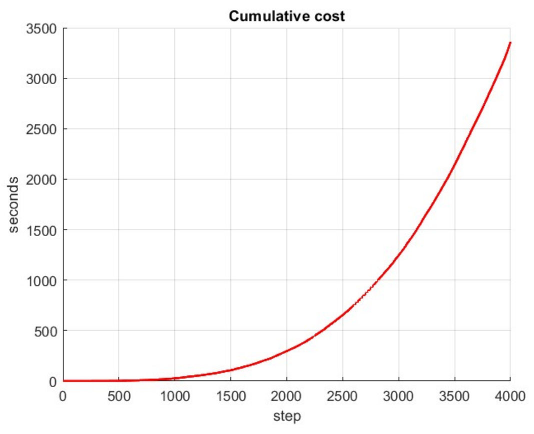
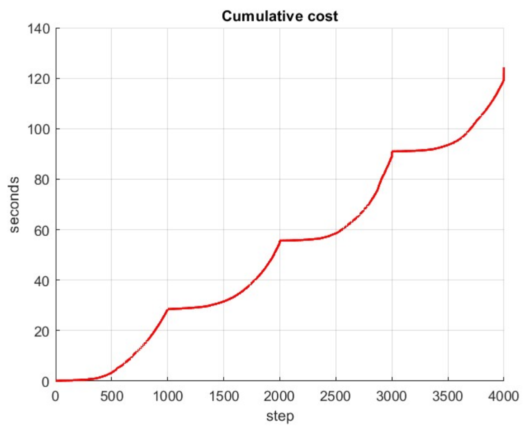


Figure 2: Cost per step comparison



a



b

Figure 3: Cumulative cost comparison