Lab 1 report: SLAM for Karel in 1D

Juan Lorente Guarnieri, Hugo Mateo Trejo

1 Add New Features

1.1 Problem Statement

The `add_new_features` function is responsible for incorporating newly detected features into the map. This requires expanding the state vector and covariance matrix while ensuring consistency with previous data.

1.2 Solution Approach

To properly add new features:

1. Construct transformation matrices J_1 and J_2, which extend the state vector while preserving existing features. These matrices are defined as:

$$J_1 = \begin{bmatrix} 1_{1\times1} & 0_{1\times m} \\ 0_{m\times1} & I_{m\times m} \\ 1_{n\times1} & 0_{n\times m} \end{bmatrix}, \quad J_2 = \begin{bmatrix} 0_{(1+m)\times n} \\ I_{n\times n} \end{bmatrix} \tag{1}$$

   where:

   - J_1 ensures that the existing state remains unchanged.
   - J_2 integrates the new feature positions z_n into the state vector.

2. Update the state estimate $\hat{x}$ by appending new feature positions to the existing state vector:

$$\mathbf{x}_{k|k} = J_1\mathbf{x}_{k|k} + J_2 z_n \tag{2}$$

3. Adjust the covariance matrix $\hat{P}$ to account for new features, ensuring correct uncertainty propagation:

$$\mathbf{P}_{k|k} = J_1\mathbf{P}_{k|k}J_1^T + J_2 R_n J_2^T \tag{3}$$

4. Maintain consistency by updating the list of true feature IDs and their corresponding positions in the map.

5. Update the feature count n to reflect the inclusion of new landmarks.

2 Update Map

2.1 Problem Statement

The `update_map` function refines the map estimate using the Kalman Filter update step. This process integrates new measurements to improve the estimated feature positions and reduce uncertainty in the map.

2.2 Solution Approach

The function follows the standard Kalman update equations to incorporate new observations efficiently. The steps involved are:

1. **Compute the measurement matrix** H_k: This matrix establishes the relationship between the observed measurements and the state vector. It consists of:

$$\mathbf{H}_k = \begin{bmatrix} -1 & 0 & \cdots & \cdots & \cdots & \cdots & 0 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ -1 & 0 & \cdots & 0 & 1 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ -1 & 0 & \cdots & \cdots & 0 & 1 & 0 & 0 \end{bmatrix}$$

   - A column of -1 values corresponding to the reference frame.
   - A sparse identity-like structure assigning the correct feature positions.

2. **Calculate the innovation** y_k: This represents the difference between the actual measurements and the predicted values given the current state estimate.

3. **Determine the innovation covariance** S_k: It quantifies measurement uncertainty and ensures the correct weighting of new information.

4. **Compute the Kalman gain** K_k: This matrix optimally balances prior estimates and new observations.

5. **Update the state estimate** $\hat{x}$: The estimated feature positions are refined by incorporating the measurement residuals.

6. **Update the covariance matrix** $\hat{P}$: This step reduces uncertainty by accounting for the influence of new observations.

2.3 Mathematical Formulation

The Kalman update equations applied in this function are:

$$
\begin{aligned}
\tilde{y}_k &= z_k - H_k \hat{x}_{k|k-1} && (4)\\
S_k &= H_k P_{k|k-1} H_k^T + R_k && (5)\\
K_k &= P_{k|k-1} H_k^T S_k^{-1} && (6)\\
x_{k|k} &= x_{k|k-1} + K_k \tilde{y}_k && (7)\\
P_{k|k} &= (I - K_k H_k) P_{k|k-1} && (8)
\end{aligned}
$$

where:

- H_k is constructed based on the feature associations.
- y_k captures measurement deviations.
- S_k models uncertainty propagation.
- K_k determines the correction magnitude.
- $\hat{x}$ and $\hat{P}$ are updated accordingly.

3 Sequential Map Joining

3.1 Problem Statement

The `sequential_map_joining` function integrates multiple local maps, generated during distinct trajectory segments, into a single unified global map. This process is essential for constructing a comprehensive environmental model from discrete local observations.

3.2 Solution Approach

The map joining procedure follows a series of steps to ensure the correct alignment of state vectors, covariance matrices, and associated features:

1. Transformation matrices J_1 and J_2 are defined to appropriately merge the state vectors and covariance matrices from the local map and the global map. These matrices ensure that the state vector and covariance matrix from each map are incorporated into the global map with consistency. These matrices are defined as:

$$
J_1 = \begin{bmatrix} 1_{1\times 1} & 0_{1\times m} \\ 0_{m\times 1} & I_{m\times m} \\ 1_{n\times 1} & 0_{n\times m} \end{bmatrix}, \quad J_2 = \begin{bmatrix} 1_{1\times 1} & 0_{1\times m} \\ 0_{m\times 1} & 0_{m\times m} \\ 0_{1\times 1} & 0_{1\times m} \\ 0_{(n-1)\times 1} & I_{(n-1)\times m} \end{bmatrix} \tag{9}
$$

where:

- J_1 ensures that the global frame will be added to the local map.
- J_2 integrates the local map into the global map.

2. The global state vector $\hat{x}$ is updated by combining the previous global estimate with the newly integrated features from the local map, using the transformation matrices.

3. The covariance matrix $\hat{P}$ is updated to reflect the uncertainty associated with the combined map, considering both the prior map and the new local observations.

4. Feature consistency is maintained by updating the global list of feature IDs, positions, and statistical properties. This includes recalculating error metrics and adjusting the associated feature covariance values.

5. The total number of features is updated to reflect the inclusion of new features from the local map.

4 Results

In this section, we present the results obtained when applying the Kalman Filter algorithm using two strategies for map update: *full map* and *map joining*. The following graphs are included:

1. Correlation matrix.
2. Map estimation error with its corresponding 2σ bounds.
3. Robot estimation error with its corresponding 2σ bounds.

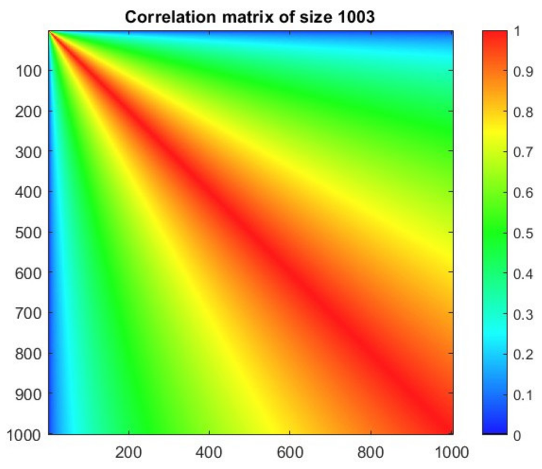

Figure 1: Correlation matrix (full map)

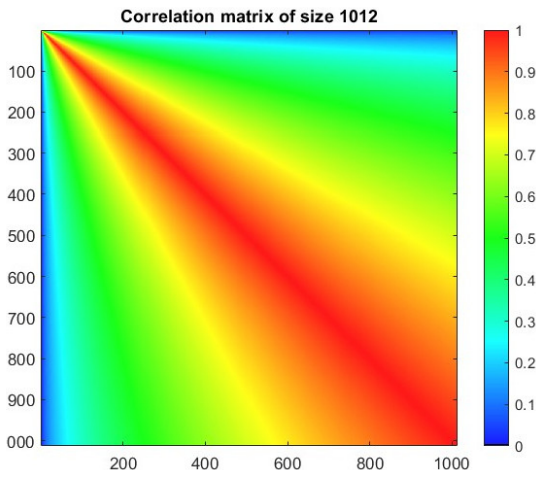

Figure 2: Correlation matrix (map joining)

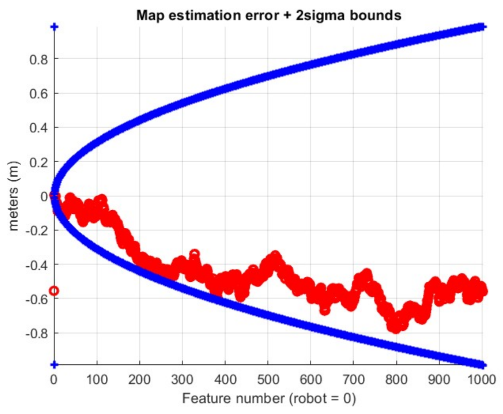

Figure 3: Map estimation error (full map)

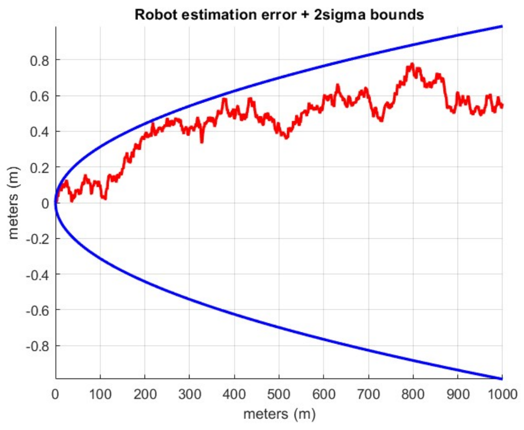

Figure 4: Robot estimation error (full map)

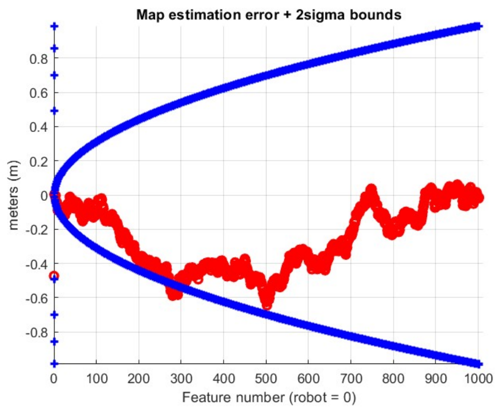

Figure 5: Map estimation error (map joining)

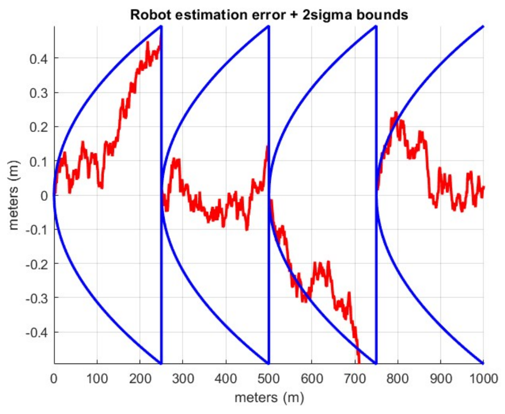

Figure 6: Robot estimation error (map joining)

After analyzing the results, the following observations can be made:

- **Correlation Matrix:** No significant differences are observed between the *full map* and *map joining* methods, indicating that both maintain a consistent structure in the correlation of features.
- **Map Estimation Error:** Both the *full map* and *map joining* approaches show a sub-linear increase in the map estimation error and the 2σ bounds, as theoretically expected.
- **Robot Estimation Error:** A notable difference is observed in this case. In the *full map* approach, the 2σ bounds increase progressively and the robot estimation follows a continuous evolution. In contrast, with the *map joining* method, every 250 steps the 2σ bounds and the robot estimation reset to values

close to zero, reflecting the division of the map into segments and a local update of the state.

5 Computational Cost Analysis

5.1 Analysis of Cost per Step

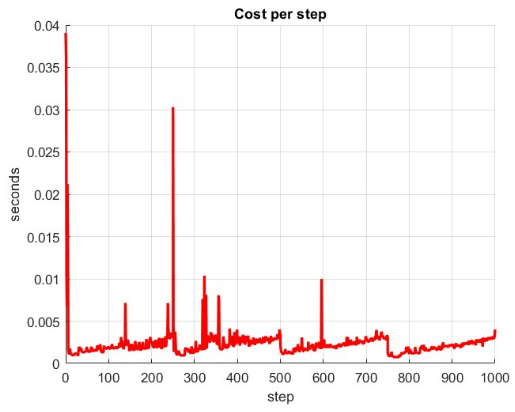

Figure 7: Cost per step (with map joining)

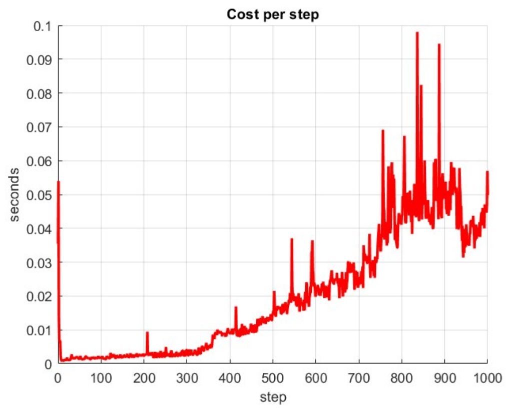

Figure 8: Cost per step (without map joining)

Full Map: In the full map approach, each iteration processes the state vector and covariance matrix, which grow with the number of features. This means that update operations (especially those related to covariance) have a cost that increases as the number of features n grows.

- **Complexity:** Approximately $O(n^2)$ per step.
- **Observation:** The graph shows that the cost per step increases, reaching up to 0.06 seconds at iteration 1000.

Sequential Joining Map: In this method, the map is divided into sub-maps (in this case, 4 partitions), and each is updated independently.

- **Complexity:** Each sub-map remains small, so update operations have an almost constant cost, meaning $O(1)$ per step.
- **Observation:** Although cost peaks (close to 0.005 s) appear at iterations where a sub-map reaches its limit (e.g., at steps 250, 500, 750, and 1000), the cost resets almost to zero when a new sub-map starts. This prevents the cost accumulation observed in the full map approach.

5.2 Analysis of Cumulative Cost

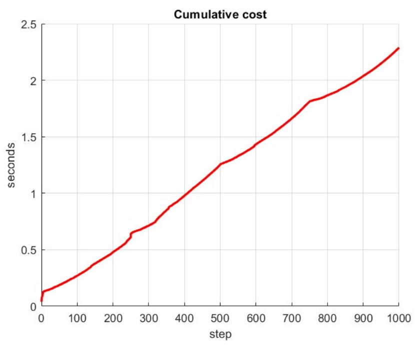

Figure 9: Cumulative cost (with map joining)

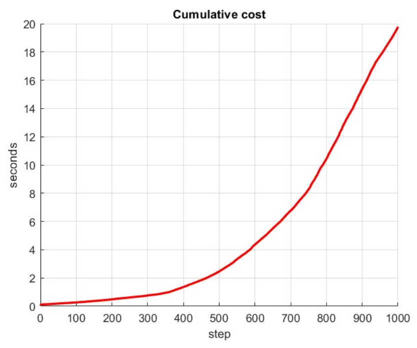

Figure 10: Cumulative cost (without map joining)

Full Map: Since the cost per step follows $O(n^2)$ complexity, summing the cost over all steps up to iteration n results in a total complexity of $O(n^3)$.

- **Observation:** The graph shows that the cumulative cost reaches 20 seconds at iteration 1000, confirming the cubic complexity.

Sequential Joining Map: Because the cost in each sub-map resets once it reaches a limit, the total accumulated cost does not grow exponentially but remains nearly linear.

- **Observation:** The cumulative cost graph shows a linear trend. Although there are periods with higher costs (due to the $O(n^3)$ complexity within each sub-map until it reaches its threshold), restarting in the next sub-map keeps the overall growth linear.

5.3 Conclusions

Scalability: The full map approach does not scale efficiently as the number of features increases, since its cost per step grows as $O(n^2)$ and the cumulative cost as $O(n^3)$. This results in processing times that can become prohibitive in real-time applications (0.06 s per step and 20 s accumulated at 1000 iterations).

Efficiency of Sequential Joining Map: By splitting the map into sub-maps, the computational cost is significantly reduced. Each sub-map maintains a limited number of features, allowing the cost per step to remain nearly constant $O(1)$ and the cumulative cost to grow linearly. This strategy is especially useful in environments requiring real-time processing or systems with limited resources.

Overall Conclusion: The *sequential joining map* implementation optimizes the process by avoiding the quadratic and cubic growth of computational cost associated with handling a full map. This leads to a significant performance improvement, making the Kalman Filter SLAM system more viable for applications with large data volumes or real-time constraints.

6 Problems

6.1 Problem 3

Assume that Karel has perfect (zero error) odometry. Analyzing the KF SLAM equations, KF SLAM is possible. The variance of the position of the robot $(1, 1)$ will be zero. Since feature positions are correlated by the robot odometry and not the sensor, they will have zero covariances, and the whole matrix will be diagonal.

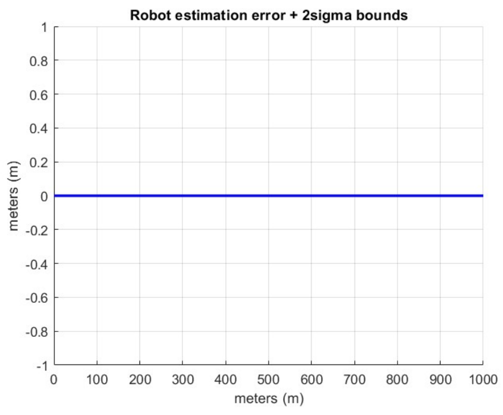

Figure 11: Robot estimation error

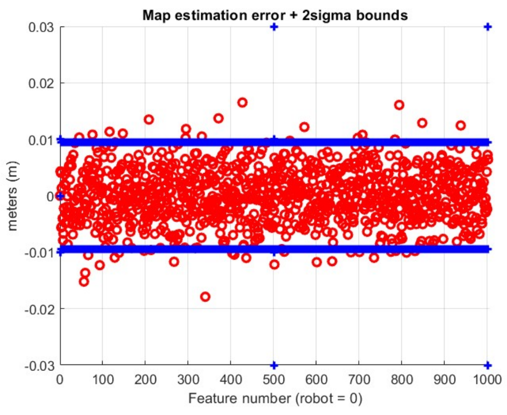

Figure 12: Map estimation error

6.2 Problem 4

Assume that Karel has a perfect (zero error) sensor. KF SLAM is also possible, and the input variance of the measurements will be zero. The covariance matrix will be similar, with a smaller variance for the features, while covariances remain the same.

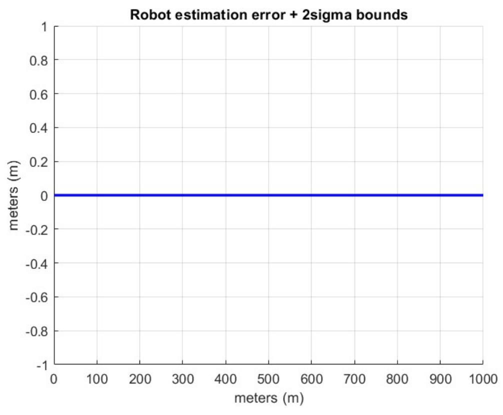

Figure 13: Robot estimation error

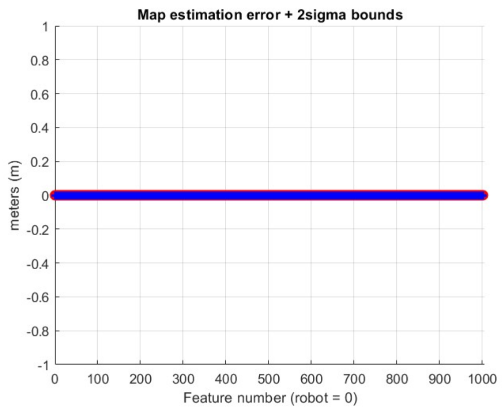

Figure 14: Map estimation error

6.3 Problem 6

If GPS is available with a certain accuracy s^2_{GPS}, KF SLAM will work similarly, with GPS measurements refining the robot position and variance alongside odometry.

6.4 Problem 7

If Karel's initial location uncertainty is not zero, the covariance matrix will have larger values in both diagonal and covariances, but correlation remains unchanged.

6.5 Problem 8

If Karel has no odometry but a reasonable motion estimate, KF SLAM is not possible since there is no bound on the measurement error variance.

7 Work Distribution

All work was done jointly by both participants during the practice sessions.