Discussion of the application of several landmark extraction and matching algorithms

Francisco Rodrigues* and João Marafuz Gaspar†
Department of Electrical and Computer Engineering, Instituto Superior Técnico
*francisco.rodrigues.09@tecnico.ulisboa.pt, †joao.marafuz.gaspar@tecnico.ulisboa.pt

Abstract—In this paper, we present an analysis of the application of various landmark extraction and matching algorithms in robotic navigation. We chose to explore RANSAC and Hough Transform for landmark extraction, and Nearest Neighbor and Maximum Likelihood for landmark matching, in order to form four models. Additionally, we also evaluated the use of Iterative Closest Point as an additional model. Our experiments were conducted using pre-recorded ROSbags in different scenarios on a TurtleBot3 robot, and a previously implemented FastSLAM algorithm. The results of our study demonstrate the potential of these algorithms for use in real-world robotic navigation applications, and provide insights for future research in this field.

1 Introduction

In this paper, we delve into the exciting field of landmark extraction and matching algorithms for robotic navigation. The ability of a robot to accurately perceive and understand its environment is crucial for it to navigate and perform tasks autonomously.

By exploring the use of techniques such as RANSAC and Hough Transform for landmark extraction, and Nearest Neighbor and Maximum Likelihood for landmark matching, we aim to explore what is possible in this field. RANSAC and Hough Transform are both widely used for feature extraction in computer vision, and have proven to be robust and efficient. Nearest Neighbor and Maximum Likelihood, on the other hand, are commonly used for feature matching and have shown to be effective in different scenarios.

In this research, we will be combining these algorithms to form four models, as well as exploring the use of Iterative Closest Point (ICP) as a replacement for odometry as an additional model.

Our research will be tested in real-world scenarios using pre-recorded ROSbags on a TurtleBot3 robot and a previously implemented FastSLAM algorithm.

2 Motivation

Landmark extraction and matching algorithms play a vital role in the process of a robot accurately perceiving and understanding its environment by enabling it to identify and track distinctive features in its environment.

Current landmark extraction and matching algorithms have limitations in terms of efficiency, robustness, and resilience. For example, some algorithms may struggle in low-light or cluttered environments, while others may not be able to handle large amounts of data. This motivated us to explore the use of state-of-the-art techniques prevously described.

In summary, the motivation for this work is to explore the efficiency, robustness and resilience of landmark extraction and matching algorithms for robotic navigation, which can help to push the boundaries of what is currently possible in the field of robotics and increase the capabilities of autonomous systems.

3 Landmarks

In this paper, lines are utilized as landmarks in indoor environments, such as walls, due to their prevalence. It's important to note that using a high number of landmarks is not practical in our implementation (see Section 5), which uses a particle filter with individual maps for each particle. Thus, lines serve as an efficient option as one line can represent multiple points. To understand the representation of lines, imagine a line perpendicular to the landmark passing through the origin. Any single landmark can then be identified by $[\rho\ \varphi]^T$, where ρ is the distance from the origin to the intersection point and φ is the angle of the x-axis with the perpendicular line – represented in Fig. 1.

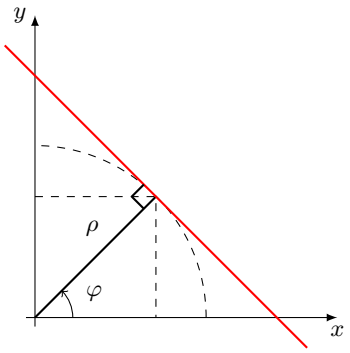


Fig. 1: Landmark geometric definition.

4 Algorithms

4.1 Landmark Extraction

Landmark extraction is the process of identifying and extracting relevant features from sensor data for use in localization and mapping.

4.1.1 Random Sample Consensus

Algorithm 1 represents the adapted RANdom SAmple Consensus (RANSAC) pseudocode, which is used to extract landmarks from the scan data. It operates by randomly selecting a sample of S laser readings, all within D degrees from each other and determining the best-fit line through these readings using a least-squares approximation. Then, the number of laser readings that are near, within X meters, the best-fit line are counted. If this number exceeds a pre-determined threshold C, a line has been found. This process is then repeated until no more points are available or the maximum iteration number N is reached.

Algorithm 1 RANSAC for Line Extraction

Require: Set of points $\mathcal{P}$, threshold distance X, minimum number of points C, initial number of samples S, maximum degree separation D and number of iterations N.
The **RndSelect**$(S, \mathcal{P}, D)$ function, which randomly selects S points from set $\mathcal{P}$ that span no more than D degrees on the original scan. The **LSTSQ**$(\bigstar)$ function, which returns the best-fit line for the set $\bigstar$. The **distance**$(.)$ function, which returns the euclidean distance.
Ensure: Set of lines $\mathcal{L} = \left\{\ell \in \mathcal{L} : \ell = [\rho\ \varphi]^T\right\}$.
Algorithm: **RANSAC**$(\mathcal{P}, X, C, S, D, N)$ **return** $\mathcal{L}$

$\mathcal{L} \leftarrow \{\}$ ▷ intialization
for $i = 1$ **to** N **do**
  if $\#\mathcal{P} < S$ **then** ▷ check if there are still enough points to keep going
    break
  end if
  $\mathcal{A} =$ **RndSelect**$(S, \mathcal{P}, D)$
  $\ell_i =$ **LSTSQ**$(\mathcal{A})$ ▷ best-fit line for the set $\mathcal{A}$
  $\mathcal{B} = \{p \in \mathcal{P} :$ **distance**$(p, \ell_i) < X\}$
  $\mathcal{P} = \{p \in \mathcal{P} :$ **distance**$(p, \ell_i) < 3X\}$
  if $\#\mathcal{B} > C$ **then**
    $\ell_f =$ **LSTSQ**$(\mathcal{B})$ ▷ best-fit line ℓ_f for the set $\mathcal{B}$
    Calculate r^2 for this fit ▷ r^2 is the coefficient of determination
    if $r^2 \geq 0.9$ **then**
      $\mathcal{L} = \mathcal{L} \cup \ell_f$ ▷ add line ℓ_f to $\mathcal{L}$
    end if
  end if
end for
return $\mathcal{L}$

As depicted in Fig. 2, the points that fall within the orange boundaries are considered inliers and are used to define the dashed red line, while points outside of the threshold are considered outliers and discarded.

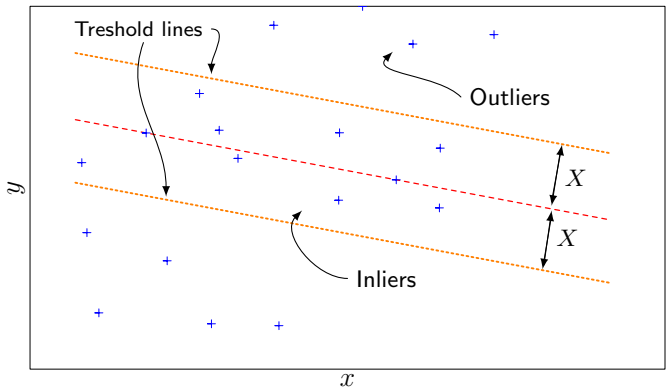


Fig. 2: RANSAC visualization. [1]

One of the main advantages of RANSAC is that it can robustly fit a model to a set of data that may contain outliers. However, the choice of the number of iterations and the threshold for determining inliers can affect the results and it can be sensitive to the presence of a large number of outliers.

4.1.2 Hough Transform

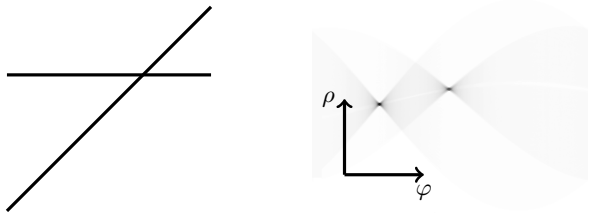
The Hough Transform is a widely used image processing technique used to detect lines (or any other shape that can be described mathematically) in an image. It is based on the idea that a line in an image can be represented as a point in the Hough Space, which is a parameter space that describes all possible lines in the image. The Hough Lines algorithm (see Algorithm 2) is a specification of the Hough Transform that converts each edge point in the image to a set of polar coordinates $[\rho\ \varphi]^T$ and then accumulates all the polar coordinates in a two-dimensional accumulator. The accumulator is then searched for local maxima, which correspond to lines in the original image.

Algorithm 2 Hough Lines

Require: Image $\mathcal{I}$, threshold t, resolution $\Delta\rho, \Delta\varphi$ of the accumulator $\mathcal{A}(\varphi, \rho)$.
Ensure: Set of lines $\mathcal{L} = \left\{\ell \in \mathcal{L} : \ell = [\rho\ \varphi]^T\right\}$.
Algorithm: **HoughLines**()

$\mathcal{A} \leftarrow \mathbf{0},\ \mathcal{L} \leftarrow \{\}$ ▷ intializations
for each edge point $[x\ y]^T$ in $\mathcal{I}$ **do**
  for $\varphi = 0$ **to** 2π **do**
    $\rho = x\cos(\varphi) + y\sin(\varphi)$
    $\mathcal{A}(\varphi, \rho)$ ++ ▷ increment the accumulator at position (ρ, φ)
  end for
end for
$\mathcal{A} = \{a \in \mathcal{A} : a/\max(\mathcal{A})\}$ ▷ normalization
$a_{\max} = \{a \in \mathcal{A} : a$ is local maxima $\wedge\ a > t\}$ ▷ find local maxima in the accumulator above threshold t
$\mathcal{L} = \left\{[\rho\ \varphi]^T : \mathcal{A}(\varphi, \rho) = a_{\max}\right\}$ ▷ list of lines corresponding to the local maxima
return $\mathcal{L}$

Fig. 3 shows an example of an original image on the left, with a white square and two black lines crossing it, and the corresponding Hough Space on the right.



(a) Input Image. **(b)** Hough Space.

Fig. 3: Example showing the results of a Hough transform on a raster image containing two thick lines.

One of the main advantages of the Hough Transform is that it can detect lines of any orientation, even if they are partially obscured or broken. However, it can be computationally expensive for large images.

4.2 Landmark Matching

Landmark matching is the process of comparing and matching the extracted features from sensor data to those in the map to estimate the current pose of the robot.

4.2.1 Nearest-neighbour

To match identified landmarks with previously seen ones, the Nearest-neighbor (NN) approach may be used. For each previously seen landmark, the particle's position $[x\ y]^T$ is projected onto it and onto the newly identified one, using

$$\textbf{project}\left([x\ y]^T\right) = \frac{1}{a^2+b^2}\begin{bmatrix} b^2x - aby - ac \\ a^2x - aby - bc \end{bmatrix}, \qquad (1)$$

where and the landmark is given by $ax + by + c = 0$.
If the squared Euclidean distance between the two projections is under a defined threshold, we assume that it is the same landmark.
One of the main advantages of NN is that it is simple and easy to implement. It is also computationally cheap, making it suitable for real-time applications.

4.2.2 Maximum Likelihood

Maximum Likelihood (ML) is a statistical method that is commonly used in landmark matching to determine the best correspondence between two sets of landmarks. It is based on the assumption that the probability of a given correspondence

between two landmarks is determined by the similarity between the landmarks.

To find the best correspondence, the ML approach compares each possible correspondence between the two sets of landmarks and calculates the likelihood of each correspondence. The correspondence with the highest likelihood is considered the best match.

Algorithm 3 Maximum Likelihood [2]

Require: Measurement z_t, control u_t, pose x_t and the N_{t-1} features known (map) with their mean $\mu_{j,t-1}$ and covariance $\Sigma_{j,t-1}$, for $j = 1, \ldots, N_{t-1}$. A measurement prediction function $h(\mu_{t-1}, x_t)$, its Jacobian $h'(\mu_{t-1}, x_t)$ and the measurement model noise Q_t. The importance factor p_0 used for new features.
Ensure: Weight w and index $\hat{c}$ of maximum likelihood correspondence or feature.
Algorithm: **MaximumLikelihood**(z_t, u_t, x_t, N_{t-1}) **return** $\langle w, \hat{c} \rangle$

for $j = 1$ **to** N_{t-1} **do** ▷ measurement likelihoods
  $\hat{z}_j = h(\mu_{j,t-1}, x_t)$ ▷ measurement prediction
  $H_j = h'(\mu_{j,t-1}, x_t)$ ▷ calculate Jacobian
  $Q_j = H_j \Sigma_{j,t-1} H_j^T + Q_t$ ▷ measurement covariance
  $w_j = |2\pi Q_j|^{-1/2} \exp\left[-\frac{1}{2}(z_t - \hat{z}_j)^T Q_j^{-1} (z_t - \hat{z}_j)\right]$ ▷ likelihood correspondence
end for

$w_{N_{t-1}+1} = p_0$ ▷ importance factor, new feature
$w = \max_{j=1,\ldots,N_{t-1}+1} w_j$ ▷ maximum likelihood correspondence
$\hat{c} = \arg\max_{j=1,\ldots,N_{t-1}+1} w_j$ ▷ index of ML feature
return $\langle w, \hat{c} \rangle$

One of the main advantages of ML estimation is that it provides a way to estimate the parameters of a probability distribution that best explain a given set of observations. However, since the ML method is a probabilistic method, it does not guarantee a unique solution.

4.3 Scan Matching

Scan matching is a technique used in robotic mapping and localization to align two or more scans of a scene, taken at different times or from different viewpoints, to estimate the relative pose of the robot with respect to the environment.

4.3.1 Iterative Closest Point

Iterative Closest Point (ICP) algorithm is a widely used method for scan matching, which has been applied in various fields such as robotics and computer vision. The ICP algorithm aims to align the current scan with a map of the environment, by iteratively finding the closest point correspondences between the two (see Fig. 4). The algorithm starts with an initial alignment and then iteratively improves it by minimizing a distance metric between the correspondences. The algorithm converges to the optimal alignment when the point correspondences are in a local minimum of the distance metric.

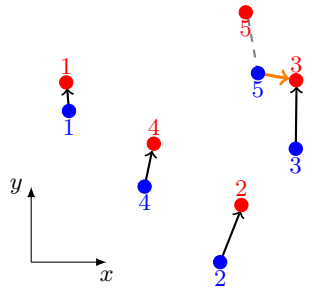


Fig. 4: ICP application on two sets of points. Note that the orange arrow denotes a wrong matching from NN.

The basic idea behind the ICP algorithm is that it estimates the transformation that aligns the two sets of points in the least squares sense. The algorithm repeats the following steps:

- Finding the closest point correspondences between the current scan and the map;
- Estimating the transformation that aligns the correspondences, using a method such as singular value decomposition;
- Updating the alignment by applying the estimated transformation to the current scan

The above can be translated to the pseudocode presented in the **Algorithm 4**.

Algorithm 4 Iterative Closest Point [3]

Require: Two consecutive sets of scans $\mathcal{S}_k$ and $\mathcal{S}_{k+1}$ with N_k and N_{k+1} scans each respectively, the minimum error $\varepsilon_{\min}$ and the maximum number of iterations, $i_{\max}$.
The **NearestNeighbor**$(\beta \in \mathcal{S}_{k+1}, \mathcal{S}_k)$ function, which finds the nearest point in the set $\mathcal{S}_k$ to the point β.
Ensure: The vector $\vec{u}_{CM}$ of the translational projection between the centers of mass and the rotation matrix T of the transformation $\mathcal{S}_{k+1} \to \mathcal{S}_k$.
Algorithm: **IterativeClosestPoint**$(\alpha \in \mathcal{S}_k, \beta \in \mathcal{S}_{k+1}, \varepsilon_{\max}, i_{\max})$ **return** $\vec{u}_{CM}, T$

$\mu_{\mathcal{S}_k} = \frac{1}{N_k}\sum_{i=1}^{N_k} \alpha_i$ and $\mu_{\mathcal{S}_{k+1}} = \frac{1}{N_{k+1}}\sum_{i=1}^{N_{k+1}} \beta_i$ ▷ Compute the centers of mass
$\vec{u}_{CM} = \mu_{\mathcal{S}_{k+1}} - \mu_{\mathcal{S}_k}$
$\beta_i = \beta_i - \vec{u}_{CM},\ i = 1, \ldots, N_{k+1}$ ▷ Translate center of mass $\mathcal{S}_{k+1} \to \mathcal{S}_k$
$(\varepsilon, \varepsilon_{\text{old}}, i, T) \leftarrow (\infty, \infty, 1, I_3)$ ▷ Initializations
while $\varepsilon_{\text{old}} = \infty$ **or** $\varepsilon/\varepsilon_{\text{old}} < \varepsilon_{\min}$ **or** $i < i_{\max}$ **do**
  for each $\beta_i \in \mathcal{S}_{k+1}$ **do** ▷ nearest neighbor
    $\gamma_i =$ **NearestNeighbor** $(\beta_i, \mathcal{S}_k)$
  end for
  $T_{\text{aux}} = \arg\min_{T_{\text{aux}}} \varepsilon(T_{\text{aux}}) = \arg\min_{T_{\text{aux}}} ||\gamma - T_{\text{aux}}\beta||_2^2$ ▷ Solve the optimization problem with least squares solution
  $(\varepsilon, \varepsilon_{\text{old}}, i, T) = \left(\min_{T_{\text{aux}}} ||\gamma - T_{\text{aux}}\beta||_2^2, \varepsilon, i+1, TT_{\text{aux}}\right)$ ▷ Updates
end while
return $\vec{u}_{CM}, T$

The ICP algorithm has the advantage of being simple to implement and computationally efficient, it is widely used for scan matching in SLAM, however, it is known to be sensitive to initialization and local minima. In this work, we use the ICP algorithm in our hybrid method, as a substitude for odometry, and still perform all other steps (RANSAC, matching, etc...) as in all other methods.

5 Implementation

5.1 Assumptions

For this work we considered the same motion and measurement models as in [1], which are

$$g(x_{t-1}, u_t, R_t) = \begin{bmatrix} x + \hat{\delta}_{\text{trans}} \cos\theta \\ y + \hat{\delta}_{\text{trans}} \sin\theta \\ \theta + \hat{\delta}_{\text{rot}} \end{bmatrix} \tag{2}$$

and

$$h_j(x_t, l_j) = \begin{bmatrix} \rho - x\cos\varphi - y\sin\varphi \\ \varphi - \theta \end{bmatrix}, \tag{3}$$

where index j refer to the j-th landmark, as well as the values for the odometry and laser sensor covariances, R_t and Q_t, represented below, seeing as the used robot (TurtleBot3) data was the same.

$$R_t = \begin{bmatrix} \sigma_x^2 & 0 & 0 \\ 0 & \sigma_y^2 & 0 \\ 0 & 0 & \sigma_\theta^2 \end{bmatrix} = \begin{bmatrix} 10\,\text{mm}^2 & 0 & 0 \\ 0 & 10\,\text{mm}^2 & 0 \\ 0 & 0 & 0.05\,\text{mrad}^2 \end{bmatrix} \tag{4}$$

$$Q_t = \begin{bmatrix} \sigma_\rho^2 & 0 \\ 0 & \sigma_\varphi^2 \end{bmatrix} = \begin{bmatrix} 10\,\text{cm}^2 & 0 \\ 0 & 300\,\text{mrad}^2 \end{bmatrix} \tag{5}$$

5.2 FastSlam implementation

The task of creating a map of an environment while simultaneously determining the location of a moving robot is known as SLAM (Simultaneous Localization And Mapping). One method for solving this problem is FastSLAM, which uses a technique called particle filtering to generate the map and localize the robot. This approach has several advantages over traditional methods, such as those based on Extended Kalman Filters (EKF).

In order to test the previously proposed algorithms, the FastSLAM implementation from [1] was used as a baseline (the code is available here).

5.3 Models Overview

The combination of the presented algorithms is represented in the Table 1, where it is possible to verify the name assigned to each combination, in order to make the writing that follows much easier.

Table 1: Algorithm combination identification.

	Algorithm	
Name	Landmark Extraction	Landmark Matching
Model I	RANSAC	Nearest-neighbour
Model II	RANSAC	Maximum Likelihood
Model III	Hough Transform	Nearest-neighbour
Model IV	Hough Transform	Maximum Likelihood
Model V	Iterative Closest Point + RANSAC + NN	

5.4 Parametrization

The parameters used for the final implementation were the following:

Table 2: Used Parametrization.

Parameter	Value	Description
	RANSAC	
N	150	Max attempts to find lines
S	6	Number of initial samples
D	10	Max degree separation [°]
X	0.01	Max distance a reading may be from line [m]
C	23	Number of points that must lie on a line for it to be taken as a line
	Hough Lines	
lines_max	5	Maximum number of lines to detect
t	50	Minimum number of votes a line must have
ρ	$[-5, 5]$	ρ interval [m]
φ	$[0, \pi]$	φ interval [rad]
$\Delta\rho$	0.12	Distance resolution of the accumulator [m]
$\Delta\varphi$	$\pi/4$	Angle resolution of the accumulator [rad]
	Nearest Neighbor	
$d_{\max}$	0.5	Distance threshold [m]
	Maximum Likelihood	
p_0	1×10^{-12}	Importance factor
	Iterative Closest Point	
$\varepsilon_i/\varepsilon_{i+1}$	0.95	Maximum error ratio
$i_{\max}$	5	Maximum number of iterations

6 Experimental Results

6.1 Qualitative analysis

All the experiments were made on the headquarters of Instituto Superior Técnico – University of Lisbon.

6.1.1 Results on ROSbag `RoundTrip_30_maio.bag`

The obtained results for this ROSbag, which was taken in the 5th floor of the North Tower (**Setting I**), are represented in the Figures 7-12 in the **Annex**.

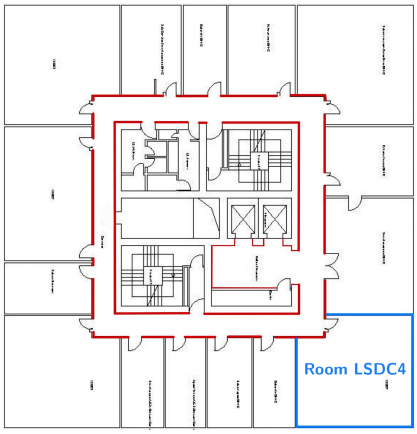


Fig. 5: Floor plan of **Setting I**.

We can see that both **Model I** and **Model III** yield pretty accurate maps of the 5th floor of the north tower. However, **Models II** and **IV** were not able to provide a full map. **Model V** was able to construct a full map, although with quite big inaccuracies in the distance measurements. In terms of the map's shape (that is, ignoring the actual distances measured), the best performing method seems to be **Model I**. Nevertheless, we can see that **Model III**, despite initially not matching the starting corridor with the ending one, eventually lines them up, thus showing some recovery capacity.

6.1.2 Results on ROSbag `Civil_22_junho.bag`

We also tested these same models using a ROSbag taken on the ground floor of the civil pavilion (**Setting II**) – `Civil_22_junho.bag`, which results are represented in the Figures 13-18 in the **Annex**.

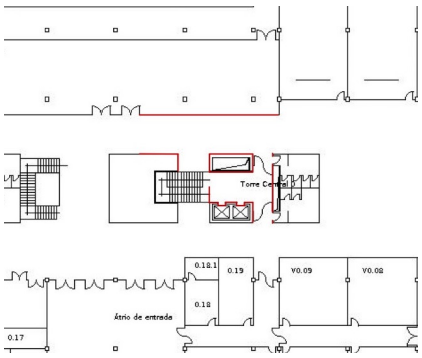


Fig. 6: Floor plan of **Setting II**.

As we can see, the odometry data for this bag is much more accurate, and when running models **I-IV** the results we get are similarly good. **Model V** had some issues mapping the starting area due to how similar it looked while traveling down the corridor. However, we can observe in Fig. 18 that the elevator section is mapped quite well.

6.2 Quantitative analysis

Utilizing the *"measure"* tool from `RViz` it was possible to measure some distances on the obtained map and then compare them with the real ones, as can be seen in the Table 3.

Table 3: Comparison of obtained measurements.

	1st corridor length [m]	2nd corridor length [m]	1st corridor width [m]
Measuring Tape	15.57	15.76	1.69
Model I	15.69 ± 0.43	15.61 ± 0.43	1.64 ± 0.43
Model II	15.97 ± 0.30	15.34 ± 0.22	1.70 ± 0.22
Model III	15.06 ± 0.55	15.12 ± 0.43	1.66 ± 0.43
Model IV	16.1 ± 0.31	15.88 ± 0.25	1.69 ± 0.20
Model V	11.7 ± 0.43	12.06 ± 0.43	1.66 ± 0.43

Table 4: Relative errors for each model.

	1st corridor length	2nd corridor length	1st corridor width
Model I	0.77%	0.95%	2.96%
Model II	2.57%	2.66%	0.59%
Model III	3.28%	4.06%	1.78%
Model IV	3.40%	0.76%	0%
Model V	24.86%	23.48%	1.78%

Note that the uncertainty value was computed by measuring the width of each of the corresponding walls.

The values closest to the ones measured directly with a measuring tape are the ones obtained using **Model I**. We can see that for this model all the expected values sit within the margin of error. Models **III** and **IV** also perform quite well, with only one value that lying outside the error margin.

As expected, **Model 5** was not able to get an accurate measurement of the length of the corridors because without odometry data, laser scans far away from corners look very similar, and so it is difficult to extrapolate any translation data.

Table 5: Model runtime per landmark.

Model	Run Time [μs/landmark]
HL + NN	367.09
HL + ML	229.52
RANSAC + NN	273.73
RANSAC + ML	224.14
ICP	8884.4

In Table 5 we can see the time it took to process the scans divided by the number of existing landmarks for the various models. We can see that models **I-IV** all perform similarly, with **Model V** being an outlier and being more than ten times slower. This shows us that for the first 4 models, execution is not a bottleneck and so we are free to choose the method that yields better results.

7 Discussion

As shown in the previous section, the various models seem to perform differently in the two studied environments. While models **I-IV** were all able to generate a reasonable map in both situations, we can see that for **Setting I**, **Model III** and **Model IV** were able to recover from some of the accumulated errors and correct their course once they returned to a previously visited place. Even so, all models were able to take the inaccurate odometry data (especially in the measurement of the robot's rotation) showing resilience. As described before, in terms of efficiency, the only outlier is **Model V**, with all the other ones being able to run in real-time. In **Setting II**, **Model IV** seems to be the most accurate, correcting the small angle inaccuracies in odometry data.

In regards to **Model V**, we can see it yields a lot worse results than all the other methods. This makes sense as we're ignoring the very important odometry data from the robot, but it's still interesting that it is able to create a map of both environments, even it very flawed. The idea behind this method was to be used in case odometry data from the motor sensors was not available, either because of a system failure or because it was not available to begin with. Due to our naïve Python implementation, this method also ended up being very slow, to the point where it would not be feasable to run in real-time. We also tried a C++ implementation, however we did not manage to get the whole FastSLAM node working in C++ to integrate with ICP within the deadline of the project.

8 Conclusions & Real World Applications

In this paper, we have presented an analysis of the application of various landmark extraction and matching algorithms in robotic navigation. We have evaluated the performance of RANSAC and Hough Transform for landmark extraction, combined with Nearest Neighbor and Maximum Likelihood for landmark matching, as well as the use of Iterative Closest Point (ICP) as an additional model. Our experiments were conducted using pre-recorded ROSbags in different scenarios on a TurtleBot3 robot, and a previously implemented FastSLAM algorithm.

The results of our study demonstrate the potential of these algorithms for use in real-world robotic navigation applications, and provide some insights for future research in this field.

In terms of real-world applications, these algorithms have the potential to be used in a wide range of robotic systems, such as autonomous vehicles, drones, and service robots. Furthermore, the ability to accurately extract and match landmarks can improve the performance of robotic navigation systems, leading to more efficient and reliable navigation in complex and dynamic environments.

For future work, we could explore the use of these algorithms in real-world environments, in addition to further evaluating their performance in different scenarios.

References

[1] Francisco Rodrigues, João Marafuz Gaspar, Manuel Graça, and Tiago Lourinho. Application of FastSLAM in TurtleBot3. Technical report, May 2022.

[2] Enrique Fernández Perdomo. Test and evaluation of the fastslam algorithm in a mobile robot. 12 2013.

[3] Ismail Elkhrachy. *Towards an Automatic Registration for Terrestrial Laser Scaner Data*. PhD thesis, 02 2008.

Annex

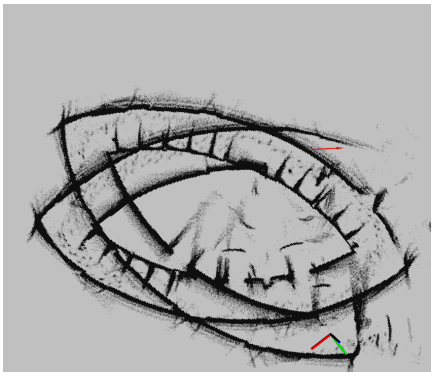


Fig. 7: Setting I map using only odometry data.

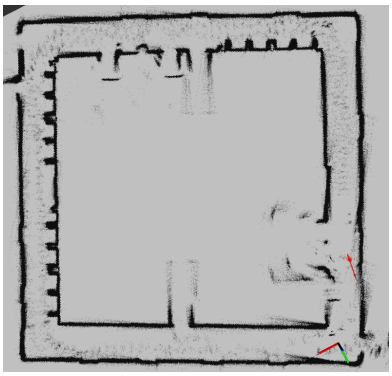


Fig. 8: Setting I map using **Model I**.

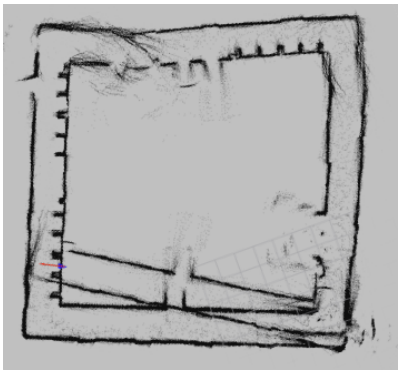


Fig. 9: Setting I map using **Model II**.

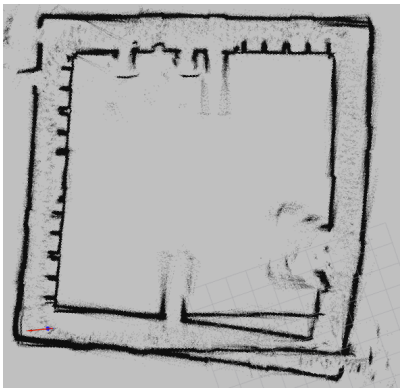


Fig. 10: Setting I map using **Model III**.

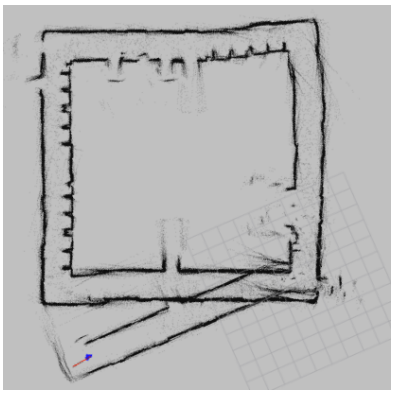


Fig. 11: Setting I map using **Model IV**.

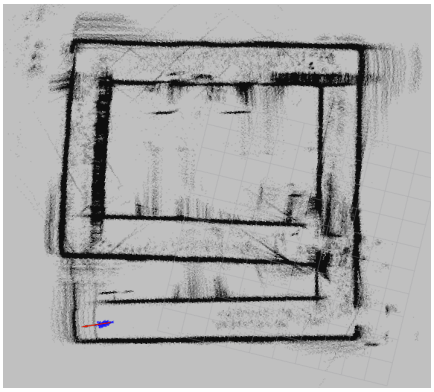


Fig. 12: Setting I map using **Model V**.

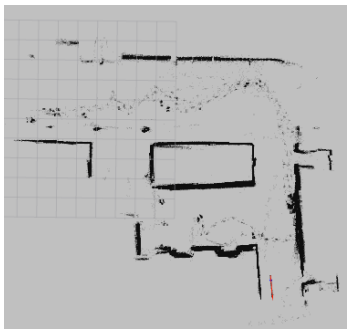


Fig. 13: Setting II map using only odometry data.

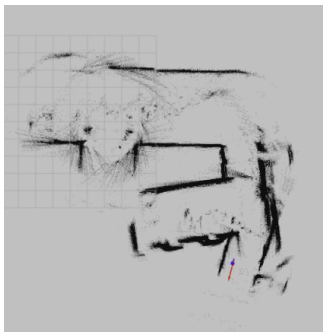


Fig. 14: Setting II map using **Model I**.

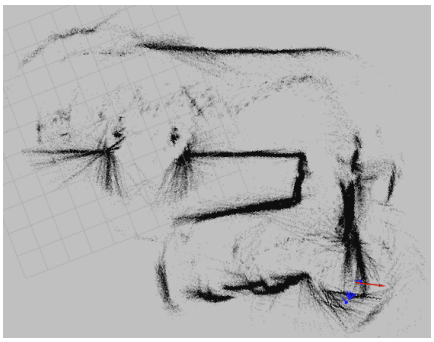


Fig. 15: Setting II map using **Model II**.

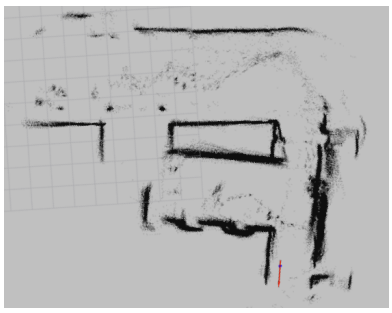


Fig. 16: Setting II map using **Model III**.

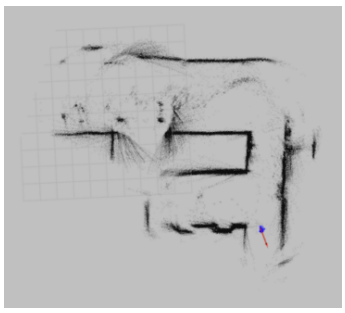


Fig. 17: Setting II map using **Model IV**.

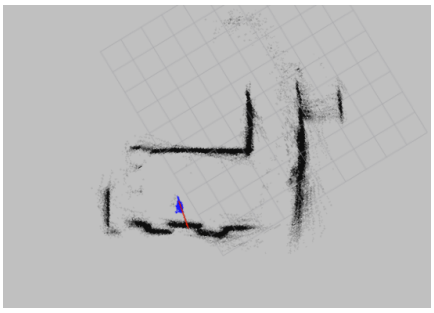


Fig. 18: Setting II map using **Model V**.