Extended Kalman Filter for SLAM

Autonomous Systems Project Report

Diogo Costa 99919, Gonçalo Frazão 99945, João Loureiro 99987, Ricardo Silva 100071

***Abstract*—This paper investigates the application of an *Extended Kalman filter* for online simultaneous localization and mapping (*SLAM*). The study presents the results and subsequent analysis of applying the filter to both simulated environments in *Gazebo* and real-world data. The research reveals the challenges encountered when utilizing the extended Kalman filter in the context of laser-based measurements and provides a comprehensive explanation of these issues. Despite the encountered problems, successful simulations and real-world applications showcase the filter's effectiveness in estimating the robot's pose and mapping the environment. The paper further explores the impact of the robot's behavior on the filter's mapping capabilities.**

I. Introduction and Motivation

SLAM, or Simultaneous Localization and Mapping, refers to the computational problem of constructing a map of an unknown environment while simultaneously determining the location of the agent or sensor within that environment, using sensor measurements and motion data. EKF SLAM combines the estimation of the robot's pose (position and orientation) with the creation of a map of the environment. The algorithm maintains a belief about the robot's pose and the map, updating them iteratively as sensor measurements and control inputs are received. It uses a probabilistic framework to model uncertainty and employs the extended Kalman filter to estimate the optimal robot pose and map given the noisy sensor data.

Taking advantage of the exceptional opportunity presented by this course to work with tools that are rarely accessible to most people, our group eagerly embraced the chance to work with the laser scanner. We made a firm commitment to implement the EKF most effectively and efficiently as possible, carefully considering the specific working environment and its distinctive features. The challenging factor in this project proved its importance in fuelling us through all obstacles found.

II. Methods and Algorithms

A. Extended Kalman Filter

The Extended Kalman Filter presents itself as an improvement to the Kalman filter family as it allows nonlinearities and uncertainties in real-world environments. Despite its computational complexity, which grows quadratically with the number of landmarks, the EKF has shown promising results for medium-scale scenes. Therefore due to its simplicity and effectiveness, the EKF is still a practical solution for many Online SLAM applications.

Algorithm 1 shows the pseudo code that guided us through out this project to implement this filter.

Algorithm 1 EKF for Online SLAM

Require: *Previous state* μ_{t-1}, *previous covariance matrix* Σ_{t-1}, *control* u_t, *observarions* z_t, *noise covariance matrix associated with odometry motion model* R_t *and noise covariance matrix associated with observation model* Q_t.

A motion prediction function $g(u_t, \mu_{t-1})$ *and a measurment prediction function* $h(\bar{\mu}_t)$

EKF Prediction Step

Compute the predicted pose estimate $\bar{\mu}_t = g(u_t, \mu_{t-1})$

Compute the low dimension Jacobian of the motion model $G_t^x = \frac{\partial g}{\partial (x,y,\theta)^T}$

Compute the predicted covariance matrix $\bar{\Sigma}_t = \begin{pmatrix} G_t^x \Sigma_{xx} G_t^{xT} + R_t & G_t^x \Sigma_{xl} \\ (G_t^x \Sigma_{xl})^T & \Sigma_{ll} \end{pmatrix}$ ▷ Landmark covariance matrix is left unchanged!

EKF Correction Step

for *all extracted features* z_t^i **do**
  if z_t^i *is associated with landmark* j **then**
    Compute $\hat{z}_t^i = h(\bar{\mu}_{t_{x,j}})$
    Compute the measurement Jacobian $H_t^i = h'(\bar{\mu}_{t_{x,j}})$
    Compute the measurement covariance $S_t^i = H_t^i \Sigma_{t-1} H_t^{iT} + Q_t$
    Compute the Kalman Gain $K_t^i = \bar{\Sigma}_t H_t^{iT} S_t^{i-1}$
    Update the predicted pose estimate $\bar{\mu}_t = \bar{\mu}_t + K_t^i (z_t^i - \hat{z}_t^i)$
    Update the predicted covariance matrix $\bar{\Sigma}_t = (I - K_t^i H_t^i)\bar{\Sigma}_t$
  else
    Parse feature through a validation gate
    if z^i *has been seen multiple times* **then**
      Upgrade z^i *to a landmark*
      Go through steps $6 - 11$ ▷ Notice that the initialization makes the innovation equal to zero
    end if
  end if
end for

B. Motion Model

From the algorithm 2 it becomes clear that

$$g(u_t, \mu_{t-1}) = \begin{pmatrix} x \\ y \\ \theta \end{pmatrix} + \Delta t \cdot \begin{pmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & & 1 \end{pmatrix} \begin{pmatrix} v_x \\ v_y \\ v_{\text{yaw}} \end{pmatrix}$$

Therefore we are now able to compute the motion model Jacobian G_t^x as follows:

$$G_t^x = \frac{\partial g}{\partial (x,y,\theta)^T} = \begin{pmatrix} 1 & 0 & -\Delta t \cdot (sin(\theta)v_x + cos(\theta)v_y \\ 0 & 1 & \Delta t \cdot (cos(\theta)v_x - sin(\theta)v_y) \\ 0 & 0 & 1 \end{pmatrix}$$

Algorithm 2 Odometry Motion Model

Require: u_t, μ_{t-1_x}

$Unpack\ from\ u_t\ v_x,\ v_y,\ v_{yaw}\ and\ \Delta t$
$x' = x + \Delta t \cdot (cos(\theta)v_x - sin(\theta)v_y)$
$y' = y + \Delta t \cdot (sin(\theta)v_x + cos(\theta)v_y)$
$\theta' = \theta + \Delta t \cdot v_{yaw}$

return $\bar{\mu}_{t_x} = (x', y', \theta')^T$

C. Landmarks

Lines have been selected as landmarks due to the predominant presence of walls or similarly structured obstacles in the environment. Recognizing that the number of landmarks poses a significant vulnerability in the Extended Kalman Filter (EKF), it is crucial to mitigate computational complexity by leveraging this particular representation type.

We employed the orthogonal projection of the map origin in polar coordinates to depict these landmarks as shown in figure 1. It is important to note that this representation necessitates normalizing the angle $\alpha_{f(i)}$ within the range of 0 to 2π.

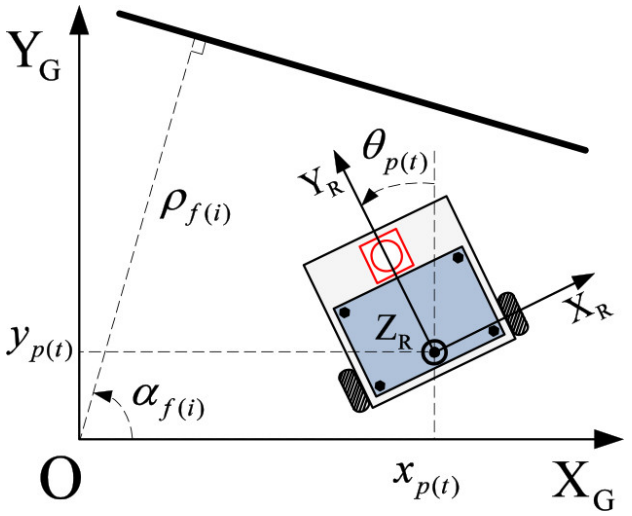


Fig. 1: Landmark representation

D. Feature Detection

In order to utilize the desired landmark representation, it is necessary to apply one of several algorithms available for line feature detection. In our study, we successfully implemented and enhanced an algorithm with linear complexity, as proposed in the paper [1].

This algorithm is composed of three main steps explained in the following subsections.

1) Seed-segment detection:

Each complete line segment originates from a seed segment detected during the current time step. Consequently, the seed points must adhere to a stringent set of conditions outlined by specific guidelines. These guidelines impose restrictions on various factors, including the distance between adjacent 2D laser points, the individual point distance from the fitted seed line equation, and the distance between each point and the intersection point formed by the seed line segment and the line connecting the robot and the corresponding 2D laser point.

The following algorithm depicts what was explained above.

Algorithm 3 Seed-segment detection

Require: $N_p, \epsilon, \delta, \tau, S_{num}, P_{min}, bp_{index}$
for $i = bp_{index} \rightarrow (N_p - P_{min})$ **do**
  $flag = True$
  $j \leftarrow i + S_{num}$
  fit **Seed**(i, j)
  for $k = i \rightarrow j$ **do**
    obtain the predicted point P'_k
    $d_1 \leftarrow distance\ from\ P'_k\ to\ P_k$
    if $d_1 > \delta$ **then**
      $flag = false$
      break
    end if
    $d_2 \leftarrow distance\ from\ P_k\ to$ **Seed**(i, j)
    if $d_2 > \epsilon$ **then**
      $flag = false$
      break
    end if
    $d_3 \leftarrow distance\ from\ P_k\ to\ P_{k-1}$
    if $d_3 > \tau$ **then**
      $flag = false$
      break
    end if
  end for
  if flag == true **then**
    return **Seed**(i, j)
  end if
end for

2) Region growing:

Region segment growing involves the utilization of the outer and innermost points of the seed segment, with the aim of expanding this seed to generate a preliminary line segment before further refinement. The criteria imposed on new points during this growth process closely resemble those employed in the seed segment detection algorithm.

Algorithm 4 Region growing

Require: **Seed**$(i, j), N_p, P_{min}, L_{min}, \epsilon, \delta, \tau$
Initialization: **Line**$(P_b, P_f) \leftarrow$**Seed**$(i, j), L_i = 0, P_i = 0$
$P_f = j + 1, P_b = i - 1$
while $distance(P_f, \textbf{Line}) < \epsilon$ and $distance(P'_f, P_f) < \delta$ and $distance(P_f, P_{f-1}) < \tau$ **do**
  if $P_f > N_p$ **then**
    break
  else
    refit **Line**(P_b, P_f)
  end if
  $P_f \leftarrow P_f + 1$
end while
$P_f \leftarrow P_f - 1$
while $distance(P_b, \textbf{Line}) < \epsilon$ and $distance(P'_b, P_b) < \delta$ and $distance(P_b, P_{b-1}) < \tau$ **do**
  if $P_b < 1$ **then**
    break
  else
    refit **Line**$(P_b, P_f$
  end if
  $P_b \leftarrow P_b - 1$
end while
$P_b \leftarrow P_b - 1$
obtain L_l, P_l from **Line**(P_b, P_f)
if $(L_l \geq L_{min})$ & $(P_l \geq P_{min})$ **then**
  return Line(P_b, P_f) with **Parameters**(m, b)
end if

3) Overlap region processing:

Lastly, overlap region processing is employed as a means to refine the obtained lines. During this stage, it is crucial to identify and address collinear lines or those with slight overlaps, as such occurrences can adversely affect the integrity of the landmark representation in subsequent stages.

Algorithm 5 Overlap region processing

Require: N_l (The number of line segments)
for $i = 1 \rightarrow N_l - 1$ **do**
  $j \leftarrow i + 1$
  Endpoint index of $Line_i$:(m_1, n_1)
  Endpoint index of $Line_j$:(m_2, n_2)
  if $Line_i$ and $Line_j$ have similar slopes **then**
    refit **Line**(m_1, n_2)
    continue
  end if
  if $m_2 \leq n_1$ **then**
    for k = $m_2 \rightarrow n_1$ **do**
      $d_k^i =$ distance$(P_k, line_i)$
      $d_k^j =$ distance$(P_k, line_j)$
      if $d_k^i < d_k^j$ **then**
        continue
      else
        break
      end if
    end for
    $n_1 \leftarrow k - 1$
    $m_2 \leftarrow k$
  else
    break
  end if
  refit **Line**(m_1, n_1)
  refit **Line**(m_2, n_2)
end for
return line segments without overlap region

E. Measurement Model and Data Association

Using figure 1 to aid our thought process we can define the measurement model as follow:

$$\hat{z}_t^i = h(\bar{\mu}_t)^i = \begin{pmatrix} \rho_{f(i)} - \bar{x}_{p(t)} \cdot cos(\alpha_{f(i)}) - \bar{y}_{p(t)} \cdot sin(\alpha_{f(i)}) \\ \alpha_{f(i)} - \bar{\theta}_{p(t)} \end{pmatrix}$$

Where $(\bar{x}_{p(t)}, \bar{y}_{p(t)}, \bar{\theta}_{p(t)})^T$ represents the laser scanner position. Therefore we are now able to compute the low dimension Jacobian of the measurement model[1].

$$^{low}H_t^i = \frac{\partial h}{\partial(\bar{x}, \bar{y}, \bar{\theta}, \rho_{f(i)}, \alpha_{f(i)})^T} =$$
$$= \begin{pmatrix} -cos(\alpha_{f(i)}) & -sin(\alpha_{f(i)}) & x_s^R(sin(\bar{\theta}) \cdot cos(\alpha_{f(i)}) - cos(\bar{\theta}) \cdot sin(\alpha_{f(i)})) & 1 & sin(\alpha_{f(i)}) \cdot (\bar{x} + cos(\bar{\theta})x_s^R) - cos(\alpha_{f(i)}) \cdot (\bar{y} + sin(\bar{\theta})x_s^R) \\ 0 & 0 & -1 & 0 & 1 \end{pmatrix}$$

To establish data association, a nearest-neighbor approach is employed, which involves calculating the distances between the predicted measurements for each landmark and the observed feature. If the smallest distance is below a specified threshold for the association, we can confidently determine that the observed feature corresponds to the previously stored landmark.

III. Implementation

A. Uncertainty Estimate

Accurate estimation of error covariance matrices is essential for achieving optimal performance with the Extended Kalman Filter (EKF). While automating tests for odometry can be challenging, we have opted to manually fine-tune the matrix R_t due to its complexity. On the other hand, estimating σ_ρ^2 is a relatively straightforward process. It involves collecting data from the laser scanner to measure the actual distance to a specific object. Lastly, we can also manually adjust σ_θ^2 to a value that offers satisfactory performance.

In practice, as observed in table I, the experimentally

estimated value for σ_ρ^2 was not utilized with actual data. The rationale behind this decision will be elaborated upon in the subsequent section. It is also worth mentioning that we made the assumption of independence among the random variables, leading to a zero covariance outside the main diagonal of the matrices.

$$R_t = \begin{pmatrix} \sigma_x^2 & 0 & 0 \\ 0 & \sigma_y^2 & 0 \\ 0 & 0 & \sigma_\theta^2 \end{pmatrix} \quad Q_t = \begin{pmatrix} \sigma_\rho^2 & 0 \\ 0 & \sigma_\varphi^2 \end{pmatrix}$$

Parameter	Value	Description
Feature Detection		
L_{min}	$0.6\ m$	*Minimum line length*
P_{min}	10	*Minimum number of points*
S_{num}	6	*Number of points in seed-segment*
ϵ	$0.05\ m$	*Distance threshold from a point to a line*
δ	$0.05\ m$	*Distance threshold from a point to the predicted point*
τ	$0.08\ m$	*Distance threshold for adjacent points*
Motion Model		
R_t	$\begin{pmatrix} 10mm^2 & 0 & 0 \\ 0 & 10mm^2 & 0 \\ 0 & 0 & 0.5mrad^2 \end{pmatrix}$	*Motion model error covariance matrix*
Measurement Model		
Q_t	$\begin{pmatrix} 0.01m^2 & 0 \\ 0 & 0.1rad^2 \end{pmatrix}$	*Measurement model error covariance matrix*
Data association		
d_l	$0.25\ m$	*Maximum distance to associate a feature and a landmark*
Feature Upgradability		
d_f	$0.03\ m$	*Maximum distance to associate two features*
$counter$	3	*Minimum number of times a feature needs to be seen to upgrade into a landmark*
Δt	$10\ s$	*Maximum time a feature is stored*

TABLE I: Parameters for the implementation of EKF

B. Simulation in Gazebo

Considering the highly controlled noise environment provided by *Gazebo*, it was deemed the optimal choice for testing our program. Since *Gazebo* served as our micro-simulator, the parameters listed in table I were slightly adjusted to better align with the dynamics within this environment.

In this subsection, we have included maps obtained with and without the correction step. Astute readers may raise questions about the stark contrast between the map produced without the update step of the EKF and the presence of peculiar representations in the map where correction was applied. Regarding the latter point, it becomes evident that the accumulating linearization error in the kinematic model has a significant impact as the robot's rotational speed increases (it is important to note that this even happens within a relatively **low-noise** environment)[2]. As for the former concern, we strongly believe that it is a result of a combination of the factor mentioned earlier, namely the accumulation of error, and the influence of the confidence placed in the correction step. Both of these factors have implications for the interpolation of sensor data and its representation in the world frame. These issues will be better explained further down the line.

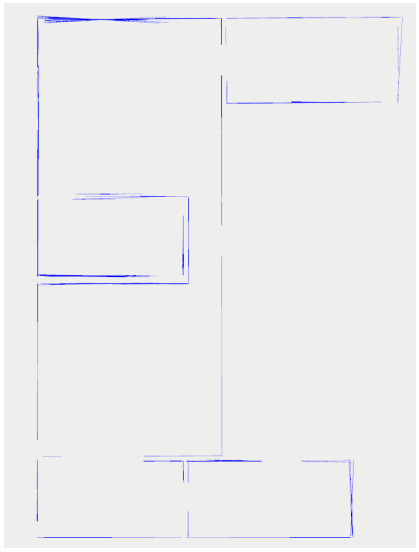


Fig. 2: *Gazebo* house map with EKF correction step

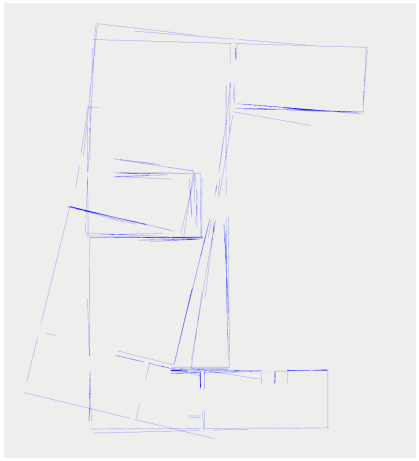


Fig. 3: *Gazebo* house map without EKF correction step

[1]Notice that x_s^R represents a relative position from the laser scanner with respect to the robot's pose. Since y_s^R and θ_s^R are both zero, this is the only relative position present in the Jacobian.

[2]We were surprised by the unexpected observation that, even with highly accurate positioning, the kinematic motion model exhibited poor performance in accurately determining the true angle of the robot, particularly when significant changes were made to the rotation speed.

IV. Experimental Results

A. Qualitative analysis

Our real-world test data primarily consisted of *rosbags* collected from the laboratory floor. While we observed that the odometry in the turtlebot was reasonably well calibrated, figure 4 reveals similarities between some behavioral patterns observed in the micro-simulator. Although not immediately evident, it is apparent that certain features were significantly affected by the linearization error of the kinematic model. Consequently, to ensure accurate data association between these features and their corresponding landmarks, we were compelled to raise the value of d_l. However, this adjustment does not come without consequences. Increasing this threshold reduces the subsequent map resolution representation, resulting in challenges such as the slight misalignment of the robot's positioning when associating, for instance, walls and doors. Despite the challenges, our efforts led to a successful outcome, resulting in a significantly improved representation of the 5th floor as seen in figure 5.

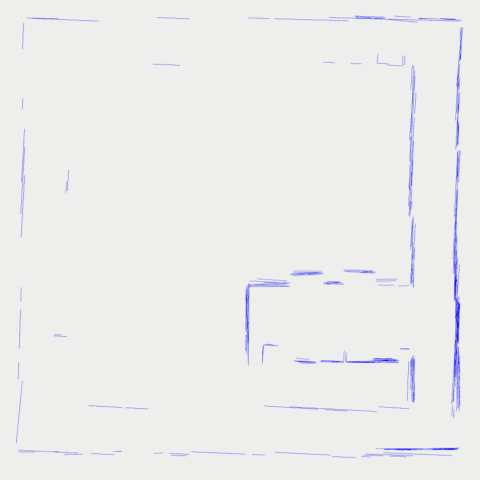


Fig. 4: Floor map without EKF correction step

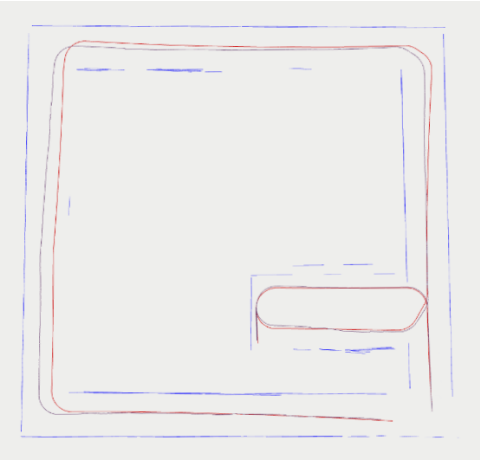


Fig. 5: Floor map with EKF correction step. In purple is represented the filter path and in red the odometry data path

B. Quantitative analysis

To understand how accurate our map representation was, we measured the corridors' length and width through the tools available under *RViz*. The results in Table **??** show the differences between real-world measurements and the results produced through the filter. Our main explanations for this difference rely on the fact that we had to have a higher *threshold* for associations, which meant that there was no distinction between walls and doors, resulting in small error accumulations. This represents nothing less than a consequence of our limitations.

	Map [m]	Measuring Tape [m]	Error [%]
Right corridor length	15.64	15.75	0.70
Top corridor length	15.82	15.76	0.38
Left corridor length	15.66	15.75	0.57
Bottom corridor length	15.84	15.76	0.51
Corridor's width	1.667	1.67	0.18

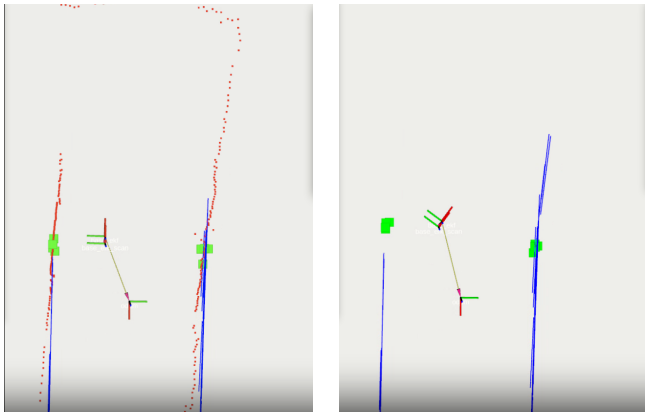
TABLE II: Comparison between obtained and real map measurements

C. Issues, limitations and future work

As highlighted in previous sections, placing unwavering confidence in pure dead reckoning-based prediction has proven detrimental in some scenarios. This is primarily attributed to the unpredictable slippage and the cumulative effect of linearization errors within the kinematic model, particularly when the robot's rotational speed undergoes rapid variations. One might question why other EKF based *SLAM* solutions using, for instance *ArUco Markers*, do not diverge under the same circumstances. The answer lies exactly in the difference between the measurement models! While our data associations are based on orthogonal projections, the associations of *ArUco Markers* are determined using simple point-to-point distance. Consequently, although both solutions are subject to similar errors during the prediction step, a minor misalignment of two lines can lead to significantly divergent orthogonal projections, whereas the distance between the same data point (the *ArUco tag*) from both lines only experiences slight alterations. In summary, this means that from the perspective of the measurement model, the inaccuracy carried from the prediction step is amplified, thus leading to filter divergence in some case scenarios[3]. Figure 6 showcases this reality.

Fortunately, there are viable solutions available to tackle this challenge. As mentioned in the paper [2], which recognizes the issue described earlier, leveraging the ICP (iterative closest point) algorithm to evaluate the dead reckoning outcome and provide necessary adjustments, it is proved possible to achieve more robust results, albeit with increased computational complexity. We leave this as a recommendation for future implementations!

[3]This phenomenon also elucidates why Q_t exhibits higher values than anticipated in a solution utilizing *LIDAR* technology. In fact, it also explains why we found high sensitivity to parameter tuning.



(a) Feature and landmark comparison

(b) Feature upgrades to landmark

Fig. 6: Incapacity to perform correct data association due to the accumulated error

V. Conclusions

At the outset of this project, we encountered numerous academic explanations of *SLAM* that all posed the same fundamental question: Why is *SLAM*, not a straightforward problem? While several technical reasons contribute to this fact, and some were even encountered during this project, fundamentally, we can provide a more overarching explanation. Unique and creative solutions to solve *SLAM* have different perspectives of problems found in real-world scenarios! Nevertheless, despite its limitations, we believe we produced a reasonable solution for this problem.

The challenges faced across development required critical thinking and deep research into its roots. With little to no information about these problems, we had to construct an explanation for their existence. By itself, this proved to be very enriching.

References

[1] Haiming Gao Xuebo Zhang Yongchun Fang Jing Yuan. A line segment extraction algorithm using laser data based on seeded region growing. *International Journal of Advanced Robotics Systems*, 4(2):1–10, 1 2018.

[2] Jixin Lv ukinori Kobayashi Takanori Emaru Ankit A. Ravankar. Indoor slope and edge detection by using two-dimensional ekf-slam with orthogonal assumption. *International Journal of Advanced Robotics Systems*, 12(2):1–10, 4 2015.

[3] Sebastian Thrun Wolfram Burgard Dieter Fox. *Probabilistic Robotics*. The MIT Press, 2005. ISBN 0262201623.