Mobile Agent Localization in Unknown Environment Using EKF SLAM

Agata Szeremeta
Earth & Space Science (ESS)
York University
agatasz@my.yorku.ca

Abstract - This project focuses on localizing a moving agent in and mapping an unknown environment through the use of artificial, visual landmarks. An Extended Kalman Filter in a Simultaneous Localization and Mapping (EKF SLAM) algorithm was applied on a dataset provided by the Autonomous Space Robotics Lab at the University of Toronto Institute for Aerospace Studies (UTIAS). Then, it was attempted to apply EKF SLAM on self-collected data acquired by a moving Microsoft Kinect RGB-D sensor. Unfortunately, it was not possible to obtain correct results for the sensor localization and environment map. Nevertheless, the procedure taken to perform EKF SLAM in the latter manner is still given, as the algorithm is expected to function given few future modifications.

1. Introduction

As an agent continues to move in any environment, its odometry data inevitably becomes affected by errors that propagate throughout its motion. Such noise results in poor pose (position and orientation) estimation of the agent in the environment. This can further affect subsequent findings, such as the relative position of features. To reduce effect of odometry associated errors and optimize the estimated agent pose, additional data can be collected using auxiliary sensors and data fusion techniques can be applied. On such data fusion technique is Simultaneous Location and Mapping Using an Extended Kalman Filter.

Simultaneous Localization and Mapping (or SLAM, for short) is a method of mapping the surrounding environment, while simultaneously localizing the moving mapping sensor within it. Often thought of as a chicken and egg problem, SLAM requires information regarding the mapping agent's pose to map the environment while also requiring such a map to determine the relative pose [1]. Several types of SLAM approaches exist. These include FastSLAM, ORB SLAM, Grid-Based SLAM, and RGB-D SLAM. Perhaps the most significant SLAM approach is using Extended Kalman Filtering (otherwise known as EKF SLAM) [2].

Kalman Filtering (KF) is an optimal method of estimating and filtering states in linear Gaussian systems [2]. The Extended Kalman Filter (EKF) is an extension to the (standard) Kalman Filter which allows it to be applicable to cases where the state transitions and measurements are nonlinear. EKF SLAM is thus a method of localizing an agent with nonlinear motion in an environment by predicting its future position and adjusting its previous position based on the observations of features in the environment. In addition to localizing the agent, EKF SLAM also estimates the position of all landmarks observed by the agent as it moves [2]. While performing both tasks, it minimizes the uncertainty in the pose of the agent and position of landmarks.

For this project, a robust EKF SLAM software was written using educational data collected from the University of Toronto Institute of Aerospace Studies (UTIAS) by the Autonomous Space Robotics Lab.

2. Data/System Used

2.1. Autonomous Space Robotics Lab System

The dataset collection created by Leung et al. that was used to generate the initial EKF SLAM algorithm consisted of nine, 2D indoor datasets containing odometry data and measurement data to landmarks from 5 different robots, in

various environment situations, such as with and without barrier obstructions [3]. Because this dataset was designed to be used in performing Multi-Robot Localization and Mapping, it was beneficial for the purposes of this project as it provided all required information to implement the algorithm. The system of robots and landmarks can be seen in Fig 1.

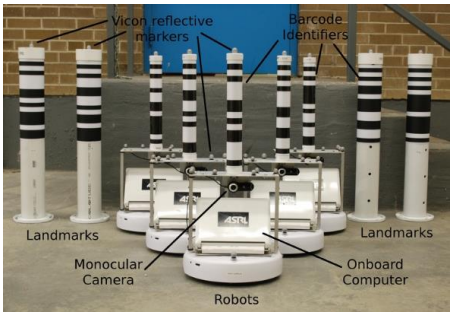

Figure 1. The Robot and Landmark Design [3]

In this dataset, the environment consisted of a 15mx8m space in which 15 cylindrical landmarks were placed at arbitrary locations. Each robot was identical in construction and built on the iRobot Create, which uses two-wheel differential drive [3]. A monocular camera was used for sensing a unique barcode on each of the landmarks and robots. The captured images, at resolutions of 960x720, were rectified and processed to detect these barcodes. Because each barcode was unique, the agent was able to distinguish to which landmark/robot it was observing upon extracting the range and bearing measurements. The dataset collection also included ground truth data for all robot poses and landmark positions. This was beneficial for validating obtained results. To capture such ground truth data, a 10-camera Vicon motion capture system was used.

In addition to the odometry data of each of the robots, their measurements in each test scenario, and ground truth data, the authors also provided various tools for users of their datasets to use for convenience. These included the following 3 Matlab scripts [3]:

I. *loadMRCLAMdataSet*
II. *sampleMRCLAMdataSet*
III. *animateMRCLAMdataSet*

The first script parsed the data (originally provided through various text files) into Matlab variables. The second sampled the data at a specified fixed time interval. The default sampling frequency used was 50Hz or 0.02s. The final provided tool animated the collected ground truth data against the obtained EKF SLAM results. For ease of use in the code, these scripts where transformed into functions.

Although the dataset collection was primarily created for the study of the cooperative localization (or simpler, for localizing using multiple/pairs of observations and sensors), the data from a single agent in a single scenario could be extracted and used to perform EKF SLAM [3]. For this project, Robot #1 from Dataset #9 and Robot #4 from Dataset #1 were used. Furthermore, although an identification number to the observed feature was provided with each measurement, it was not used, as EKF SLAM with unknown data correspondence (discussed below) was implemented.

3. Background

3.1. The Kalman Filter (KF)

The Kalman filter is a technique for state filtering and prediction, when using linear systems. It implements the Bayes filter and makes use of the following assumptions [2, 4]:

a. The probability of the next state and measurement probability are linear Gaussian functions, with added Gaussian noise
b. The initial state belief is a Gaussian random vector of known mean and covariance

The KF is a recursive algorithm that makes use of both mathematical and statistical models to determine an optimal (unbiased and minimal

variance) state solution, represented through its mean and covariance [2, 4]. The filter follows two main steps: state prediction and state correction. State prediction is based on the previous estimated state and state covariance. It involves modifying a belief based on an action (the control input). The state correction step is based on the taken/true measurement. Based on the disparity between the true measurement and expected measurement (dependent on the estimated pose), a corrective term (known as the Kalman Gain) can be computed and used to apply an update value to the estimated state. This update step decreases the uncertainty in belief.

The KF algorithm, obtained from Table 3.1 of Probabilistic Robotics by Thrun, Burgard, and Fox [2], is summarized below:

$$\bar{\mu}_t = A_t \mu_{t-1} + B_t u_t + \epsilon_t \quad (1)$$
$$\bar{z}_t = C_t \bar{\mu}_t + \delta_t \quad (2)$$
$$\bar{\Sigma}_t = A_t \Sigma_{t-1} A_t^T + R_t \quad (3)$$

$$K_t = \bar{\Sigma}_t C_t^T (C_t \bar{\Sigma} C_t^T + Q_t)^{-1} \quad (4)$$
$$\mu_t = \bar{\mu}_t + K_t (z_t - C_t \bar{\mu}_t) \quad (5)$$
$$\Sigma_t = (I - K_t C_t) \bar{\Sigma}_t \quad (6)$$

Equations 1 and 3 correspond to the prediction step. In them, $\bar{\mu}_t$ and $\bar{\Sigma}_t$ are the estimated state vector and state covariance at the current time t, respectively, while μ_{t-1} and Σ_{t-1} is the previous state vector and state covariance (which have already been corrected for). Furthermore, u_t is the control vector at time t. Both A_t and B_t are matrices which, when multiplied by the state and control vectors or covariance matrix, ensure that the state transition functions become linear in their arguments – a requirement for Kalman Filtering. The variable ϵ_t is a Gaussian random vector that models the randomness in the state transition and the matrix R_t models the covariance of the mean posteriori state (that is, represents the noise in the control vector). The variable $\bar{z}_t$ is the expected measurement given the estimated state vector and, as such, C_t is a matrix describing the relationship between the state and measurement. Finally, like ϵ_t, δ_t is a Gaussian random error.

Equations 3 to 5 correspond to the correction step. K_t denotes the Kalman gain, while Q_t is a matrix that provides information on the measurement noise. Lastly, I is the identity matrix.

3.2. The Extended Kalman Filter (EKF)

When state transitions and measurements are represented by nonlinear functions, the Extended Kalman Filter (EKF) can be applied in lieu of the standard KF. As previously mentioned, EKF is a generalization of the KF that overcomes the required linearity assumption. EKF adjusts the previous KF equations of 1 and 2 to [2]:

$$\bar{\mu}_t = g(u_t, \mu_{t-1}) + \epsilon_t \quad (7)$$
$$\bar{z}_t = h(\bar{\mu}_t) + \delta_t \quad (8)$$

where g and h are nonlinear functions. An EKF linearizes these functions using first order Taylor Series Expansion, and obtains a linear approximation to the function's value from its slope. This allows calculations resembling those in Kalman Filtering to be subsequently applied.

The EKF algorithm, obtained from Table 3.3 of Probabilistic Robotics by Thrun, Burgard, and Fox [2], is summarized below:

$$\bar{\mu}_t = g(u_t, \mu_{t-1}) \quad (9)$$
$$\bar{z}_t = h_t(\bar{\mu}_t) \quad (10)$$
$$\bar{\Sigma}_t = G_t \Sigma_{t-1} G_t^T + R_t \quad (11)$$

$$K_t = \bar{\Sigma}_t H_t^T (H_t \bar{\Sigma} H_t^T + Q_t)^{-1} \quad (12)$$
$$\mu_t = \bar{\mu}_t + K_t \big(z_t - h_t(\bar{\mu}_t)\big) \quad (13)$$
$$\Sigma_t = (I - K_t H_t) \bar{\Sigma}_t \quad (14)$$

Variable notation in the above equations hold the same as those in equations 1-6, apart from G_t and H_t which represent the Jacobians of $g\big(u_t, \mu_{t-1}\big)$ and $h\big(\bar{\mu}_t\big)$, respectively.

Although EKF is not entirely just the linearized version of the Kalman filter, as there are some

differences in underlying belief assumptions and thus some limitations, it remains a valid approximation of the state [2]. It should be noted that the correctness of the filter is dependant on the degree of nonlinearity of the functions and the degree of uncertainty – approximately linear functions, with lower covariance are better approximated than those that are more nonlinear with wider Gaussian beliefs [2].

3.3. EKF SLAM

Given a nonlinear control input (or motion) of an agent and measurements to landmarks taken by the auxiliary sensor on the agent, EKF SLAM techniques can be implemented to determine the agent's pose. While, knowing the pose and observations to landmarks relative to that pose, a map of the location of features can simultaneously be created.

As described by Sola in [5], SLAM consists of the following three recursive operations:

I. Agent moves
II. Agent discovers interesting features (landmarks) in the environment
III. Agent observes landmarks that have been previously mapped

In the first, the agent moves in the environment. This motion increases the uncertainty of its localization, as it is likely to be affected by some error. Then, the agent detects landmarks in the environment which need to be incorporated in the simultaneously built map. Like the motion model, the observation of these landmarks will also likely contain some uncertainty. It should be noted that error from the motion will also propagate into the observation error. Finally, the agent will improve its self-localization and the localization of landmarks in the map by re-observing landmarks and correcting its pose and map based off the new, redundant observations. This step, also known as loop closing, will cause uncertainty of the agent's pose and location of landmarks to decrease [1].

Often, EKF SLAM is used in robotics to determine the pose of a robot which moves in a 2D plane, using a velocity-based motion model [1]. The algorithm can be used to localize and map in 3D space, however, due to the environment spaces used for project, only the 2D case is considered. In considering 3D space, the proceeding vectors/matrices can be augmented with the appropriate variables.

The pose (or state) of the robot, in 2D space can be defined through:

$$\mu_t = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix}_t$$

where *x* and *y* are its position and θ is its heading in the plane. This state has the covariance:

$$\Sigma_t = \begin{bmatrix} \sigma_{xx} & \sigma_{xy} & \sigma_{x\theta} \\ \sigma_{yx} & \sigma_{yy} & \sigma_{y\theta} \\ \sigma_{\theta x} & \sigma_{\theta y} & \sigma_{\theta\theta} \end{bmatrix}_t$$

With a velocity-based motion model, the motion of the robot can be mathematically represented through:

$$u_t = \begin{bmatrix} -\frac{v_t}{\omega_t}\sin\mu_{t-1,\theta} + \frac{v_t}{\omega_t}\sin\left(\mu_{t-1,\theta} + \omega_t\Delta t\right) \\ \frac{v_t}{\omega_t}\cos\mu_{t-1,\theta} - \frac{v_t}{\omega_t}\cos\left(\mu_{t-1,\theta} + \omega_t\Delta t\right) \\ \omega_t\Delta t \end{bmatrix} \quad (15)$$

Often, the robot observes point landmarks using a range-bearing sensor [1]. Thus, its measurement vector becomes:

$$z_t = [r_t^i \quad \phi_t^i]^T$$

When the robot observes a *new* landmark, its state vector is augmented to include the estimated position of the landmarks in the plane (denoted m_i). N denotes the total number of landmarks present in the environment.

$$\mu_t = \begin{bmatrix} x \\ y \\ \theta \\ m_{1_x} \\ m_{1_y} \\ \vdots \\ m_{N_x} \\ m_{N_y} \end{bmatrix}_t = \begin{bmatrix} x_t \\ m_1 \\ \vdots \\ m_N \end{bmatrix}$$

A newly observed landmark's position is determined through an inverse observation function and this function's Jacobians. The new state of the robot is determined by applying the EKF algorithm. With this new landmark augmentation, the state covariance becomes the variance-covariance matrix of the pose of the agent ($\Sigma_{x_t x_t}$) and positions of landmarks ($\Sigma_{m_N m_N}$)

$$\Sigma_t = \begin{bmatrix} \Sigma_{x_t x_t} & \Sigma_{x_t m_1} & \cdots & \Sigma_{x_t m_N} \\ \Sigma_{m_1 x_t} & \Sigma_{m_1 m_1} & \cdots & \Sigma_{m_1 m_N} \\ \vdots & \vdots & \ddots & \vdots \\ \Sigma_{m_N x_t} & \Sigma_{m_N m_1} & \cdots & \Sigma_{m_N m_N} \end{bmatrix}$$

However, if the agent observes an already existing landmark, it uses that measurement to correct its state (not augment it). A new landmark is identified through a data association step. That is, the landmark is compared to those previously identified. If the association is high, the robot is likely observing an already identified landmark. Otherwise, then the robot is likely observing a new landmark. Often, the number of landmarks is unknown prior to performing SLAM and, instead, the state vector is updated incrementally.

As suggested by Stachniss in [1], EKF SLAM developed on the EKF algorithm by following the simple filter cycle:

I. Robot State Prediction
II. Robot Measurement Prediction
III. Landmark Measurement
IV. Landmark Data Association
V. State Update

As shown in Table 1 of the EKF SLAM guide by Sola in [5], the similarities between the EKF and SLAM are:

Table 1: Comparison of EKF and SLAM [5]

Event	EKF	SLAM
Agent Moves	EKF Prediction	Agent Motion
Sensor Detects New Landmark	State Augmentation	Landmark Initialization
Sensor Observes Known Landmark	EKF Correction	Data Association & Map Correction

The most significant development difference is the data association step. In this step, the agent must either have a priori knowledge of which landmark it is observing or use some form of data association technique to compare the observed landmark to the previously observed ones. The simplest, commonly used data correspondence technique is Maximum Likelihood [1].

3.4. Maximum Likelihood Correspondence

If the agent is unsure of which landmark is it observing, an incremental maximum likelihood (ML) estimator can be used to determine correspondence [1].

ML estimation (or MLE) is a statistical process of determining population parameters that maximize the likelihood function of given sample observations [6]. That is, it is an estimate from sample data that estimates of parameter values that maximize the probability of obtaining the observed data [6]. Given a likelihood function, $\zeta(\theta; x)$, where θ is the model parameter, MLE states the estimator $\hat{\theta}$ is:

$$\hat{\theta} \in argmax(\zeta(\theta; x))$$

To apply MLE to determine data correspondence between landmarks, it is assumed that a new

landmark is being observed and an initial estimation of the landmark position is first created. Again, this is performed using an inverse observation function and its Jacobians. If the Mahalanobis distance, defined through the equation below, between this suggested new landmark position (denoted x) and all previously observed landmarks (denoted μ), with covariance S, is greater than a threshold γ, then the observed feature is potentially a new landmark [7].

$$Dist._M = \sqrt{(x-\mu)^T S^{-1}(x-\mu)} \qquad (16)$$

Unfortunately, due to limitations of the ML technique, the landmark cannot be classified as 'new' based solely on how far it is from other landmarks [1]. The landmark under consideration must, in addition, be sufficiently different compared to the other landmarks. For example, even if the computed distance between two landmarks is sufficiently large, it may not be a true representation as the positions of the landmarks contain error. As such, the difference between them should be above a threshold value which ensures that the agent is not 'confusing' the landmarks due to uncertainties [2]. Otherwise, the landmark is believed to be a previous landmark and the algorithm proceeds by correcting/updating the state vector.

4. Algorithm Implementation

The EKF SLAM algorithm was implemented using Matlab software. Since it was not required to 'remotely' control a robot and instead only perform computations using matrices, the software was considered best to use due to its mathematical environment advantages. In addition, the software was more optimal for use compared to others due to its available image processing toolboxes (particularly useful for analyzing the Kinect sensor data).

4.1. EKF SLAM Algorithm Pseudocode

The pseudocode used for performing EKS SLAM with Maximum Likelihood data association, obtained from Table 10.2 of Probabilistic Robotics [2], with few modifications from equations presenting in [1], was as follows:

$$N_t = N_{t-1} \qquad (17)$$

$$F_x = \begin{bmatrix} 1 & 0 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & \underbrace{0 \quad \cdots \quad 0}_{2k-2} & & \end{bmatrix} \qquad (18)$$

$$\bar{\mu}_t = g(u_t, \mu_{t-1}) = \mu_{t-1} + u_t$$

$$\bar{\mu}_t = \mu_{t-1} + F_x^T \begin{pmatrix} \alpha \\ \beta \\ \omega_t \Delta t \end{pmatrix} \qquad (19)$$

$$\Gamma = \mu_{t-1,\theta}$$

$$\alpha = -\frac{v_t}{\omega_t}\sin\Gamma + \frac{v_t}{\omega_t}\sin(\Gamma + \omega_t \Delta t)$$

$$\beta = \frac{v_t}{\omega_t}\cos\Gamma - \frac{v_t}{\omega_{t_t}}\cos(\Gamma + \omega_t \Delta t)$$

$$G_t = I + F_x^T \begin{bmatrix} 0 & 0 & \beta \\ 0 & 0 & -\alpha \\ 0 & 0 & 0 \end{bmatrix} F_x \qquad (20)$$

$$\bar{\Sigma}_t = G_t \Sigma_{t-1} G_T^T + F_x^T R_t F_x \qquad (21)$$

$$Q_t = \begin{bmatrix} \sigma_r^2 & 0 \\ 0 & \sigma_\phi^2 \end{bmatrix} \qquad (22)$$

For all observed features, $z_t^i = \left(r_t^i, \phi_t^i\right)^T$ do:

$$\bar{\Gamma} = \bar{\mu}_{t,\theta}$$

$$\begin{pmatrix} \bar{\mu}_{N_t+1,x} \\ \bar{\mu}_{N_t+1,y} \end{pmatrix} = \begin{pmatrix} \bar{\mu}_{t,x} \\ \bar{\mu}_{t,y} \end{pmatrix} + r_t^i \begin{pmatrix} \cos\left(\phi_t^i + \Gamma\right) \\ \sin\left(\phi_t^i + \Gamma\right) \end{pmatrix} \qquad (23)$$

For $k = 1$ to $N_t + 1$ do:

$$\delta_k = \begin{pmatrix} \delta_{k,x} \\ \delta_{k,y} \end{pmatrix} = \begin{pmatrix} \bar{\mu}_{k,x} - \bar{\mu}_{t,x} \\ \bar{\mu}_{k,y} - \bar{\mu}_{t,y} \end{pmatrix} \qquad (24)$$

$$q_k = \delta_k^T \delta_k \qquad (25)$$

$$\hat{z}_t^k = \begin{pmatrix} \sqrt{q_k} \\ \tan^{-1}\left(\delta_{k,y}/\delta_{k,x}\right) - \bar{\mu}_{t,\theta} \end{pmatrix} \qquad (26)$$

$$F_{x,k} = \begin{bmatrix} 1 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 1 & 0 & \cdots & 0 & 0 & 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & \underbrace{0 \quad \cdots \quad 0}_{2k-2} & & & 1 & 0 & 0 & \underbrace{0 \quad \cdots \quad 0}_{2N-2k} & & \\ 0 & 0 & 0 & & & & 0 & 1 & 0 & & & \end{bmatrix} \quad (27)$$

$$H_t^k = \frac{1}{q_k}\begin{bmatrix} -\sqrt{q_k}\delta_{k,x} & -\sqrt{q_k}\delta_{k,y} & 0 & \sqrt{q_k}\delta_{k,x} & \sqrt{q_k}\delta_{k,y} \\ \delta_{k,y} & -\delta_{k,x} & -1 & -\delta_{k,y} & \delta_{k,x} \end{bmatrix} F_{x,k} \quad (28)$$

$$\Psi_k = H_t^k \bar{\Sigma}_t (H_t^k)^T + Q_k \quad (29)$$

$$\pi_k = \left(z_t^i - \hat{z}_t^k\right)^T \Psi_k^{-1} \left(z_t^i - \hat{z}_t^k\right) \quad (30)$$

End for

$$\pi_{N_t+1} = \gamma \quad (31)$$

$$j(i) = argmin(\pi_k) \quad (32)$$

$$N_t = \max\{N_t, j(i)\} \quad (33)$$

$$K_t = \bar{\Sigma}_t \left(H_t^{j(i)}\right)^T \Psi_{j(i)}^{-1}$$

End for

$$\mu_t = \bar{\mu}_t + \sum_i K_t^i \left(z_t^i - \hat{z}_t^{j(i)}\right) \quad (34)$$

$$\Sigma_t = \left(I - \sum_i K_t^i H_t^{j(i)}\right) \bar{\Sigma}_t \quad (35)$$

The variable s_t^i denotes the landmark ID (if known) while N_t denotes the number of known landmarks in the environment at time t while γ denotes the threshold value required for a new landmark to be added to the map. The Mahalabonis distance to each landmark is observed, and if this value is exceeded, a new landmark is created. Although the landmark ID's were provided in the external dataset, the EKF SLAM Maximum Likelihood algorithm was written to exclude this.

As observed in the above equations, to predict the state, it is necessary to use the previously found corrected state. Thus, to initialize the algorithm, the initial state vector and state covariances must be set. Often, these are:

$$\mu_0 = \begin{bmatrix} x_0 \\ y_0 \\ \theta_0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

$$\Sigma_0 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

Upon initializing a new landmark, their positions were estimated to be (0,0) while their covariance a large number.

4.2. Implementation on External Data

To implement the EKF SLAM algorithm on the data provided by Leung et al., the given tools that loaded and sampled the dataset were first used at their default sampling rates. Then, the algorithm provided through the pseudocode in the previous section was applied. However, since it was known that the robot's used to collect data followed a two-wheel differential drive motion pattern, it was necessary to adjust the previously defined motion function, u_t, to [8]:

$$u_t = \begin{bmatrix} v_t \Delta t \cos\left(\Gamma + \frac{\omega_t \Delta t}{2}\right) \\ v_t \, \Delta t \sin\left(\Gamma + \frac{\omega_t \Delta t}{2}\right) \\ \omega \Delta t \end{bmatrix} \quad (36)$$

Furthermore, for plotting purposes, the initial state vector, μ_0, was defined to be the initial observed ground position of the robot. The initial covariance was set to small values of approximately 0. Using initial values of 0's was not expected to interfere with the algorithm. However, was expected to cause visual confusion when using the given animation function since coordinates would be relative to a 0 position start. As such, the ground coordinates were used. Lastly, as the errors associated with the control input (or motion) and measurement, were not provided, they were arbitrarily chosen to be:

$$Q_t = \begin{bmatrix} 10cm^2 & 0 \\ 0 & 10^{\circ 2} \end{bmatrix}$$

$$R_t = \begin{bmatrix} 10cm^2 & 0 \\ 0 & 10^{\circ 2} \end{bmatrix}$$

Such values were selected through a "common-sense" approach. Given such a small scale of the environment and, likely, good sensors on the robots, the uncertainties were not expected to be large. The effect of using different values was

analyzed and is discussed further in the proceeding section.

The Mahalanobis distance and significance in difference level used to justify that an observed landmark was 'new' to the environment were chosen as 0.8446 and 1.5, respectively. The Mahalanobis distance was chosen according to a χ^2 distribution probability [9]. A value of 70% confidence was used as a threshold for the Mahalanobis Distance. This value was chosen after trial and error with other confidence value. Since it was known that 15 landmarks existed in the environment, a criterion for the success of these threshold values was whether they provided the expected number of landmarks to be identified.

To animate the results, the given tool was used. In addition, scripts to directly plot the estimated trajectory against the ground truth and residuals between them were written.

5. Results

5.1. Using External Data

To complete the EKF algorithm on data by Robot #1 of Dataset # 9, the written software required approximately 1.07 minutes. This time length is dependent on the number of measurements taken and the sampling time/frequency. Nevertheless, its rather large value suggested that the algorithm/software may not be optimal for implementation in a real-time processing situation.

It was known that Dataset #9 included barrier walls and occlusions in the environment, placed in positions, as below.

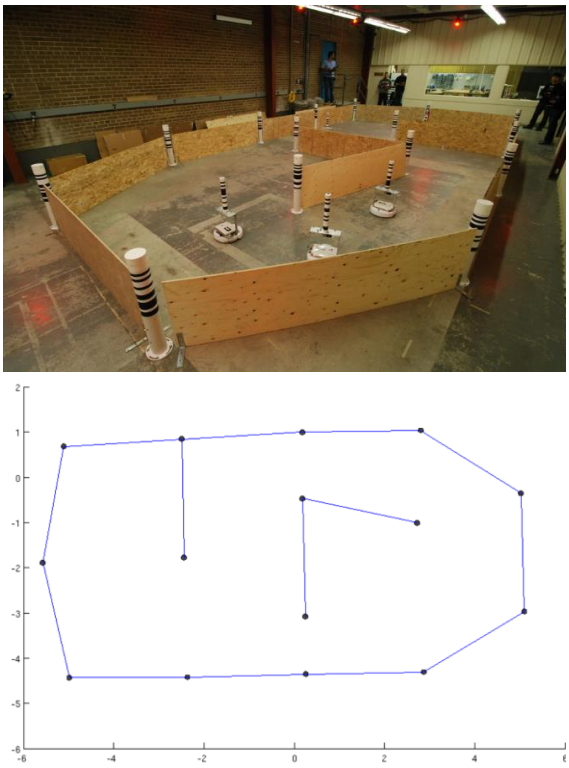

Figure 2: Barriers and Obstructions in Environment [3]

By animating the state of the robot, its motion (or position and orientation) in the environment was observed at each timestep. Upon doing so, it was noted that the robot did not remain within the barrier wall, at all times, as was expected. Instead, its trajectory and observations appeared rather sporadic and incorrect. Visually comparing to the animation of the provided ground truth trajectory further proved of the unlikely path. As, the ground truth seemed to follow a more reasonable path. These animations served as a visual confirmation of the likely existence of an error in the algorithm, since it was physically impossible for the robot to be localized in space beyond the wall. A sample image of the robot in its animated path with indicators to which landmark it is observing is provided below.

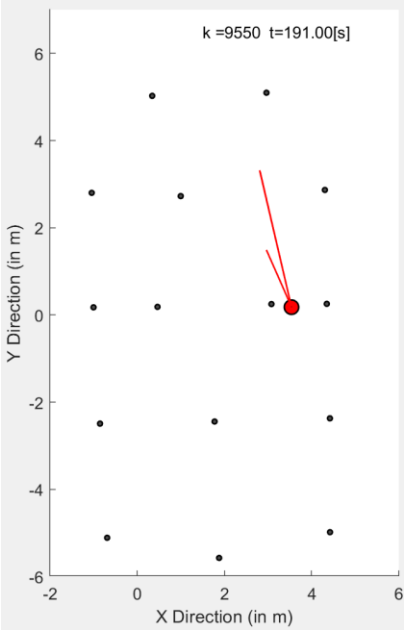

Figure 3: Sample Animation of Robot #1 of Dataset #9 in Environment

Statistics of the disparities between the ground truth pose and estimated pose (found by subtracting ground truth from estimated) are summarized in the proceeding table.

Table 2: Statistics of Disparity Between Ground Truth and Estimated Robot Pose of Robot #1 of Dataset #9

Disparity Statistic	X Value (m)	Y Value (m)	θ Value (rad.)
Minimum	-16.2101	-14.1675	-6.2701
Maximum	2.9653	7.7197	6.2249
Standard Deviation	-6.2890	-2.6183	-0.0790
Mean	4.8670	5.9465	2.4976

Given a successful algorithm, it was expected that the magnitude of differences between the ground truth and estimated poses to be small, particularly because of the rather small variances of the motion model. Unfortunately, errors for this dataset are greater than both the variances defined for both the motion and measurements and even the size of the environment itself. It was however noted the errors are also dependent on the motion and measurement noises, which were arbitrarily selected. Given better approximations for those values, it was expected that the magnitude of error between the ground truth and estimated decrease.

After repeating on data acquired by Robot #4 of Dataset #1, significant improvements in results were observed. Again, all landmarks were observed and, with greater number of measurements the run time of the software was still less than that previous – at approximately 22 seconds. The following figure provides an animation of the robot in the environment and the position of landmarks.

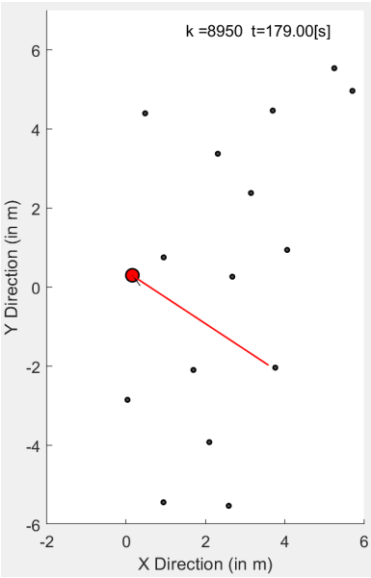

Figure 4: Sample Animation of Robot #4 of Dataset #1 in Environment

In addition, the estimated trajectory was much less random than that estimated in the previous case, even resembling the ground truth more (as seen in figure below).

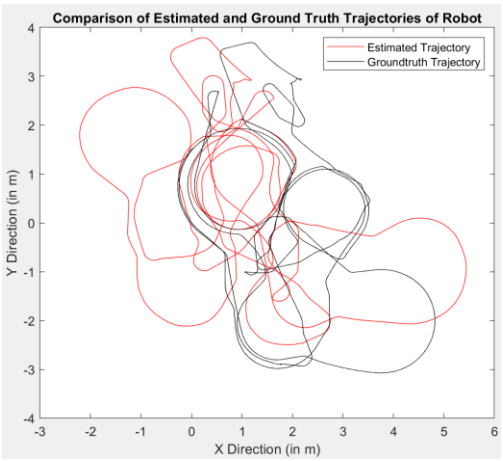

Figure 5: Comparison of Estimated and Ground Truth Trajectories of Robot #4 in Dataset #1

The figure above suggests the robot's ground truth trajectory follows some patterns. This is mainly clear in the repeated circular paths in the middle of the environment. Unfortunately, the estimated trajectory still deviates from this ground truth path and continues to be sporadic in certain regions of the environment (i.e. to the mid-left). However, it was noted that there was poor placement of landmarks in this region (can be observed in figure 4). By not having any landmarks to observe in that region, the landmark measurements cannot be used to minimize the error propagation that occurs due to motion of the robot in that region. As such, the pose of the robot is poorly estimated.

Statistics of these differences in trajectories (found by subtracting ground truth from estimated) are summarized below:

Table 3: Statistics of Disparity Between Ground Truth and Estimated Robot Pose of Robot #4 of Dataset #1

Disparity Statistic	X Value (m)	Y Value (m)	θ Value (rad.)
Minimum	-3.8137	-0.7301	-6.2665
Maximum	1.0992	1.2084	6.1825
Standard Deviation	-0.6431	0.3678	0.0083
Mean	1.1851	0.4531	0.8723

Plotting the magnitude of differences at each time step yielded:

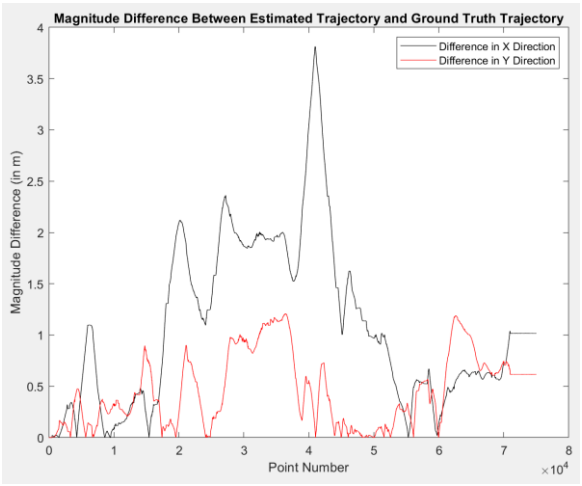

Figure 6: Magnitude of Trajectory Differences Found by Subtracting Ground Truth from Estimated

Or, plotting the differences found by subtracting ground truth from estimated:

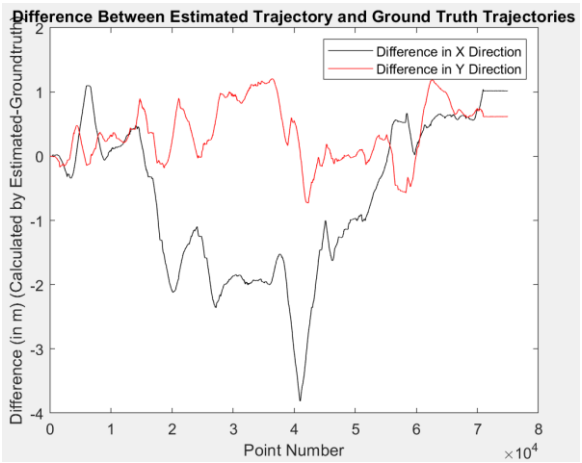

Figure 7: Trajectory Differences Found by Subtracting Ground Truth from Estimated

In general, large disparity values are not observed between the Y values. Instead, values seem to fluctuate around 0m, with greater positive differences. Greater disparity is observed between the X values, particularly around the 15000 to 50000 point number marks. Evaluating the given observation file for observations taken at such points (between approximately 400 and 1000 seconds) showed

that fewer landmarks were observed compared to others. As mentioned previously, with fewer observations, the pose of the robot may be estimated incorrectly.

Despite still existing, the disparity between ground truth and estimated poses greatly decreased from the solution found using Dataset #9. Estimated values were closer to what was expected. With the continuous existence of random or outlier regions, there is still a possibility for an error in the written software. However, code results seem to also suggest that Dataset #9 possibly contained some unknown error.

Repeating with larger variances in the measurement and motion noise, and using the data of Robot #4 of Dataset #1, resulted in approximately equivalent error statistics between the estimated path and ground truth. Repeating with smaller variances also resulted in similar statistics. This was contrary to initial beliefs, as it was expected that increased uncertainty would result in the algorithm having more difficulty in localizing the agent and identifying landmarks. However, such results can be due to the possible existence of errors in the software.

6. Future Improvements

There exist several various methods to improving the EKF SLAM with Unknown Data Correspondence algorithm.

As suggested by Thrun et. Al in [2], one such method involves limiting the number of landmarks considered in the data association step. By only considering landmarks close to the robot and/or within its sight (that is, in its observation range and field of view), the number of iterations for which data association is performed is reduced, as it is known that the robot observation does not have a high correspondence with landmarks outside of the view. Such restrictions can improve the algorithm run time and, thus, make the current software more efficient to apply in real-time situations.

Another suggested improvement method for EKF SLAM in [2] is the use of a provisional landmark list to compensate for extreme outliers that are measured. Thrun et. al suggests that poor measurements can result in extreme outliers which, in turn, can result in the creation of untrue landmarks. They recommend to first add such landmarks to a provisional list, and not using their associated measurements to update the pose of the robot. Instead, the landmark can be augmented into the state vector after it has been consistently observed and its uncertainty ellipse has decreased [2]. However, this method may only be beneficial when the scale of the environment is large and the auxiliary sensors poor, as measurements are likely to contain more error.

To improve the software written to process the external data, additional improvements to those listed above, could include adjusting the data association algorithm. As suggested previously, the Maximum Likelihood algorithm is limited by its brittleness of landmark confusion. To reduce the likelihood of poorly associating landmarks and/or classifying ‘new’ and ‘old’ features, the landmarks can be arranged further apart or include signatures [2]. The former, reduces the probability of confusing landmarks while the latter allows landmarks to be more distinguishable in the environment and from each other. However, although beneficial to an extent, adjusting the spatial arrangement of features may not be an ideal improvement. This is because, with a sparser environment and less features, EKF SLAM struggles to perform agent localization which in turn may re-introduce confusion of landmarks [2].

7. Additional EKF SLAM Implementations

In addition to applying EKF SLAM on externally collected data, it was attempted to implement EKF SLAM on self-collected Microsoft Kinect

RGB-Depth (or RGB-D) sensor data. Unfortunately, upon submission of the project, the software contained errors, which did not provide expected results. However, given the modifications suggested in this report, it is predicted that the system may function as expected.

7.1. Microsoft Kinect RGB-D Sensor System

The additional system consisted of a (calibrated) Kinect V.1 RGB-D system that was manually moved on an approximately level floor (in a horizontal plane). It was hoped that the system would be implementable on a 4-wheel cart upon algorithm success. This would allow a larger environment to be considered. The system lacked auxiliary sensors (such as wheel encoders) that provided odometry data and ground truth pose/position. It was calibrated using Matlab's Camera Calibration toolbox.

7.2. Implementing EKF SLAM Algorithm

The EKF SLAM algorithm was implemented in two distinct manners - in a 1D case (where the sensor strictly moved towards a single landmark) and in a 2D case (where the sensor was manually moved in a small-scale environment in a horizontal plane). Unfortunately, EKF SLAM could not be conducted in a real-time continuous manner. Instead it was required that the user manually trigger the sensor to capture the RGB-D images after motion of the camera.

7.2.1. Algorithm in the One Directional Motion Case

When performing EKF SLAM in the 1D case, the sensor followed a motion model based on my steps. That is, each iteration, the camera was moved one step closer to the target. To determine the distance corresponding to a step, several trials were performed and measured. The average and standard deviation of a step were then found. In addition, the sensor followed a measurement noise model that was found by finding the disparity between the observed depth (range) and true (real-world) depth to an object.

From the images taken at each point, the landmark was assumed to be the principal point of the depth image, with pixel coordinates found through the calibration process. The camera was strategically positioned so that the principal point would fall on a smooth, vertical wall that did not have significant depth. This would ensure that the observed depth would not be greatly dependent on the point on the wall that the principal point fell on (since this position would be a different point in the real-world). The measurement model consisted of the range and bearing, where range was the depth to the wall, and the bearing was 0, as the sensor observed 'straight ahead.'

Since motion was only in one direction, it could be represented by a linear function. As such, a standard KF could be applied. This required modifications to the EKF SLAM pseudocode provide above. The previously given KF equations 1-6, were used in place of the EKF defined state vector, covariance, Kalman gain, etc. Although observing only one landmark, it was still necessary to perform data association through ML and the Mahalanobis Distance to ensure that multiple features were not mistakenly mapped. This component did not differ from the previous EKF SLAM algorithm.

7.2.2.Algorithm in the Two Directional Motion Case

Because the sensor system did not provide odometry data and did not follow a strict linear path, as in the previous case, it was necessary to implement visual odometry methods to determine the motion of the camera from sequential images. Visual odometry was implemented by modifying the Monocular Visual Odometry Matlab guide available at [10]. In short, matching features were identified within sequential images and their corresponding indices would be used to compute and geometric transformation matrix. The matrix provided the

current camera pose and orientation, relative to the previous camera pose. With this information, the velocity in motion (or, change in position between frames) and angle (along the rotating axis) could be found.

Crosshair targets, such as that below, were used as landmarks in the environment.

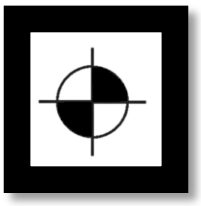

Figure 8: Sample Crosshair Target Used as Landmark

First, the centroids of these targets image were found in the RGB. The target identification function did so by assessing the centroid of contrasting polygons in a black and white image. To ensure that the targets were identified, it was necessary to have them against a strongly contrasting background, thus they were bordered with a black frame. A sample result of this target recognition is shown below.

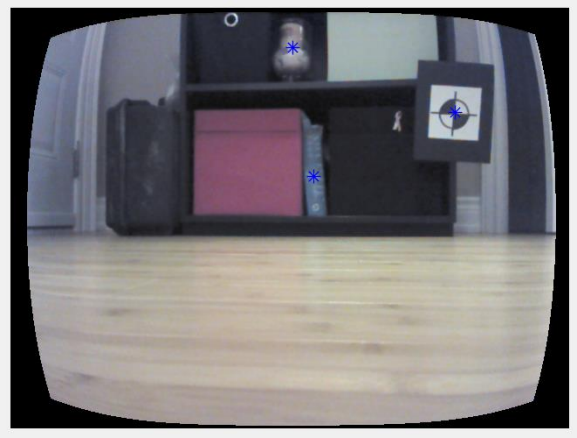

Figure 9: Sample Target Identified in Image

Unfortunately, the target identifier did not guarantee the correction of just the target centroid in the image (as shown above). Such detection was dependent on the location of the target, obliquity of the image frame, and objects surrounding the target. The depth to the target centroids were then extracted from the depth image and the angle to the target centroid from the RGB image principal point was calculated using trigonometry methods. These became the range and bearing angles, respectively.

Having collected motion and observation information regarding the current camera position, the EKF SLAM pseudocode, provided above, was implemented.

7.3. Results

Unfortunately, the algorithm was not able to correctly localize the sensor and map the observed landmarks, in either case. Sample results of the linear case are summarized below.

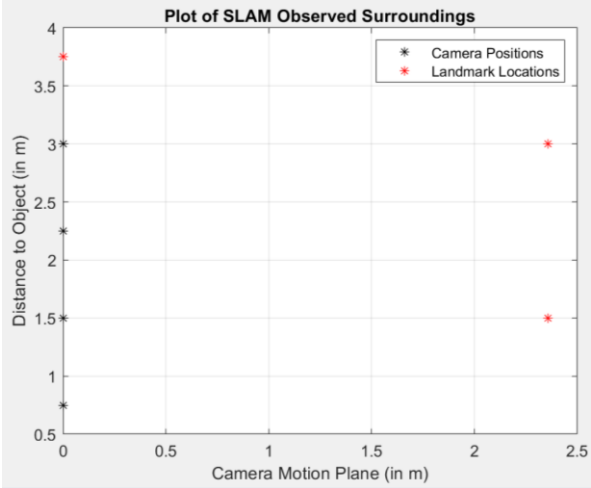

Figure 10: Sample EKF SLAM Results in 1D Kinect Sensor Case

Table 4: Estimated Location of Kinect Sensor in 1D Case

Y Position (m)	X Position (m)
0.75	0
1.499999	0
2.249999	0
2.999996	0
3.749996	0

As seen in the figure above, the camera sensor was estimated to follow a straight-line pattern, as expected. Its estimated distance motion also appears to follow the defined motion model. However, the latter results seem odd as they do not deviate much from the 75cm forward motion despite having great error. Unfortunately, it is also observed that the single landmark is not mapped correctly. The sensor and algorithm confuse the single landmark for multiple landmarks, in positions impossible to view from the sensor (outside of it's field of view). This is likely an error arising from the data association algorithm. It was likely that the sensor could not observe correct depths from its viewing angle. The sensor was positioned and manually moved on the floor, so majority of its images consisted of portions of the floor, as shown below.

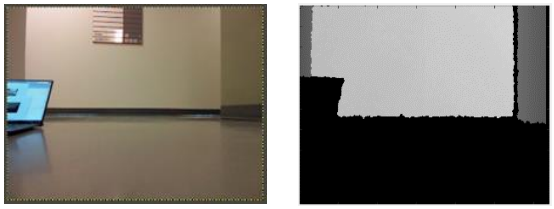

Figure 10: Sample Images Taken in 1D Test Case (RGB – Left Image, Depth – Right Image)

Since the sensor has a depth observing range of approximately 0.8m to 4m [8], it was not able to provide depth readings for the floor. It was believed that this incorrect depth value could have been included in determining the depth of the principal point and, with such an extreme depth difference between images, the algorithm believed a new landmark is observed and added it to the map. Further improvements would need to be made to the 1D case, to account for actual target observations like in the 2D case – there is no guarantee for the principal point to be the same world point in all images.

7.4. Possible Reasons for Software Failure

The algorithm is considered unsuccessful due to its inability to correctly localize the sensor and map the targets, compared to their known truth. Although this could be caused from several different errors, the following are believed to be the main contributors to its failure:

I. Poor Odometry Estimation
II. Inconsistent Coordinates Frames
III. Poor Measurement to Target Centroids

The implemented EKF SLAM algorithm for the latter scenario struggled to provide good approximations to the ground true odometry. The odometry was determined from matching features in sequential frames and, if not enough key features, the visual odometry algorithm would fail to provide a result. The algorithm is greatly dependant on the strength and number of matches in the images. Thus, if the general EKF algorithm was successfully implemented in this case, it would still be greatly limited in performance and mapping capabilities.

The results of the visual odometry process are provided in terms of the previous camera's coordinate frame, not the real-world coordinates. As such, it cannot be mapped with landmarks, observed in the real-world frame (their position is determined from real-world depth). Because the scaling term to convert such camera coordinate frame values to real-world was unknown, the resulting map from the EKF process was not truly correct.

To account for this difference in coordinate frames, a structure from motion algorithm approach could be used. That is, the 3D coordinates of features points could be determined by observing them in three sequential images and triangulating [11]. However, this would require the feature to be present in at least 3 consecutive frames – in the first, to identify the feature, in the second to re-observe and triangulate into 3D space, and in the 3rd to calculate the transformation. Requiring additional images also limits the efficiency of the EKF SLAM algorithm.

Finally, a likely contributor to the failure of the algorithm is the poor detection of and measurement to targets/landmarks. As seen in

the sample image above which showed targets located in a single RGB image frame, the algorithm confused other features as landmarks. Although, the algorithm can still function if including them, they are 'fake landmarks'. This hold great significance if not observed in any additional image frames. As mentioned above, to account for these fake identifications or extreme outliers, a landmark provisional list can be used.

8. Conclusions

A successful EKF algorithm was implemented for a particular dataset of the collection offered by Leung et. all. It is expected that their offered Dataset #9 contains an error, thus not enabling the written EKF SLAM algorithm to function accordingly. It is expected that the software work on all remining datasets and robots. A difference between estimated and ground truth trajectories was observed, as was expected. Kalman Filtering and Extended Kalman Filtering simply provide optimal solutions to predicting the state. Although EKF SLAM was not implemented correctly on self-collected data, the foundation and understanding for doing so was developed. Knowledge of the likely errors causing algorithm failures exists. Given their removal through few modifications to the software, a fully functioning EKF SLAM algorithm is expected.

References

[1] C. Stachniss, "Robot Mapping: EKF SLAM," Albert-Ludwigs-Universitat Freiburg, Frieburg im Breisgau.

[2] S. Thrun, W. Burgard and D. Fox, Probabilistic Robotics, Draft ed., 2000.

[3] K. Y. K. Leung, Y. Halpern, T. D. Barfoot and H. H. T. Liu, "The UTIAS Multi-Robot Cooperative Localization and Mapping Dataset," *International Journal of Robotics Research,* vol. 30, no. 8, pp. 969-974, July 2011.

[4] J. Wang, "Kalman Filter and Its Applications," York University, Toronto, 2018.

[5] J. Sola, "Simultaneous Localization and Mapping with the Extended Kalman Filter," 2014.

[6] Merriam-Webster Dictionary, "Maximum Likelihood," 2019. [Online]. Available: https://www.merriam-webster.com/dictionary/maximum%20likelihood. [Accessed 12 01 2019].

[7] "Mahalanobis Distance: Simple Definition, Examples," 21 11 2017. [Online]. Available: https://www.statisticshowto.datasciencecentral.com/mahalanobis-distance/. [Accessed 11 01 2019].

[8] N. Ganganath and H. Leung, "Mobile Robot Localization Using Odometry and Kinect Sensor," Calgary.

[9] R. Warren, R. E. Smith and A. K. Cybenko, "Use of Mahalanobis Distance for Detecting Outliers and Outlier Clusters in Markedly Non-Normal Data: A Vehicular Traffic Example," Air Force Research Laboratory, Dayton, 2011.

[10] MathWorks, "Monocular Visual Odometry," [Online]. Available: https://www.mathworks.com/help/vision/examples/monocular-visual-odometry.html. [Accessed 05 01 2019].

[11] K. Yousif, A. Bab-Hadiashar and R. Hoseinnezhad, "An Overview to Visual Odometry and Visual SLAM: Applications to Mobile Robotics," Melbourne, 2015.