Simultaneous Localization & Mapping

Varun Gupta

***Abstract*— Simultaneous localization and mapping is the problem of constructing a map of an unknown environment while simultaneously keeping track of the robot's location in it. In this paper, we will discuss the Monte Carlo localization technique. We will use a LiDAR range sensor to generate a map of the environment for a humanoid robot.**

I. INTRODUCTION

Identifying the location of a robot in different environments as well as constructing a map of the environment simultaneously is the most important feature every mobile robot must possess in order to better understand or interact with the world and/or perform tasks optimally according to its environment. Various statistical techniques have been implemented to solve the problem of SLAM but the most popular ones are particle filters and extended Kalman filter. These techniques have been implemented in self-driving cars, Unmanned Aerial Vehicles, Autonomous Underwater Vehicles as well as personal robots.

In this project, we implement a Monte Carlo localization technique that basically uses a particle filter to represent the distribution of likely states, with each particle representing a hypothesis of where the robot is. The algorithm typically starts with a uniform random distribution of particles over the configuration space, meaning the robot has no information about where it is and assumes it is equally likely to be at any point in space. Whenever the robot moves, it shifts the particles to predict its new state after the movement. Whenever the robot senses something, the particles are resampled based on recursive Bayesian estimation, i.e., how well the actual sensed data correlate with the predicted state. Ultimately, the particles should converge towards the actual position of the robot

II. SYSTEM CONFIGURATION

The humanoid robot uses a Hokuyo UTM-30LX/LN Scanning Laser Range Finder, also called the LiDAR sensor, placed at the head of the robot as shown in Fig1, that provides the distances of the obstacles from the sensor for a 270° scan angle at an angular resolution of 0.25° and a measurement resolution of $1mm$ at the rate of 40Hz. The specifications of the sensor can be found in [1]. The LiDAR also provides the absolute odometry, i.e. the 2D position and yaw orientation of the LiDAR in the global frame which is also used for the incremental odometry estimation for localization. Additionally, the robot uses an Inertial Measurement Unit placed at the center of mass for providing valuable orientation information about the robot in the global frame which is also used for the purpose of localization. The joint configurations of the robot obtained from the joint encoders is also required in order to transform all the data to the global frame in order to construct the map of the environment in its exact form rather than how the robot perceives it. For the humanoid robot being experimented on, only the head's pitch angle and the neck's yaw angle are sufficient joint angles to transform the scans to global frame. A kinect also placed on the head captures RGB-D data but we do not use this data in the localization or mapping in this project.

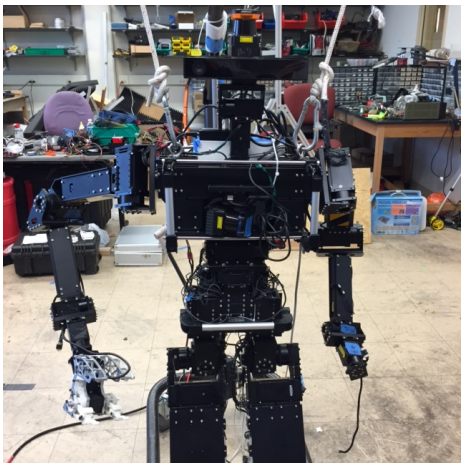


Fig. 1. Humanoid Robot

III. ALGORITHM

Here we present the algorithm followed to implement the particle filter based localization using measurements from LiDAR and IMU. Fig2 shows a flowchart describing the algorithm implemented to solve the SLAM problem. The algorithm can be broadly classified into preprocessing of sensor outputs, representation of LiDAR sensor data in global reference frame which we call mapping, and the particle filter algorithm which consists of particle initialization, prediction and update steps. We will discuss all the steps of the algorithm in the following sections but not in the sequence of the implementation.

A. Data Preprocessing

The head and neck angles obtained from the joint data are the true angles. The distance measurements obtained by the LiDAR however need to be filtered because the laser range finder has a guaranteed detection range of $0.1 - 30m$, and so the scans at distances lower or higher than these are eliminated before construction of the map. The next most important step for the 2D map construction is to

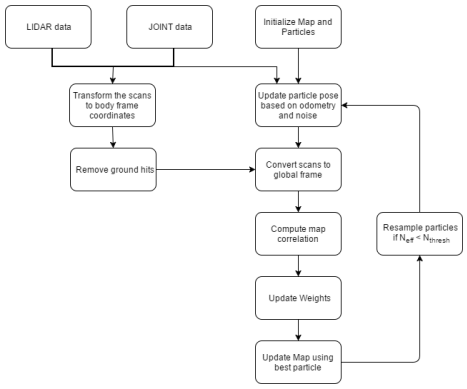


Fig. 2. Algorithm flowchart

ensure that laser rays that hit the ground and get reflected need to be removed. To identify the scans that detect the ground as obstacle, the scans have to be transformed to body frame coordinates and then the scans that have a negative z-component taking the base of the robot as reference are treated as ground scans and are therefore removed. The transformations are discussed in III-B. The robot's pose data is used without any filter but only the difference between subsequent data are used for pose estimation rather than the true values.

B. Mapping

In this section, we will discuss the transformation of the LiDAR scans to global frame assuming we know the robot's true location. This involves a series of transformations from LiDAR frame to head frame, from head frame to body frame, and from body frame to global frame. We will use the following notations throughout the paper. α is the head angle, β is the neck angle, θ is the scan angle. d is the raw scan input. x, y and z represent the coordinates in frames specified by their subscripts. We will use the subscript l for LiDAR frame, h for head frame, b for body frame and w for world frame. We will denote the yaw angle of the robot (or the particles in the particle filter) using ψ.

The coordinates in LiDAR frame are obtained for each scan using the below equations.

$$\begin{bmatrix} x_l \\ y_l \\ z_l \end{bmatrix} = \begin{bmatrix} d \times \cos\theta \\ d \times \sin\theta \\ 0 \end{bmatrix}$$

where θ is the angle of the angle of the scan ranging from -135° to 135°.

The transformation from LiDAR frame to head frame is obtained by two rotational transformation of α and β about the pitch and yaw axes respectively, which we will call R_y and R_z respectively, and a translation from the LiDAR center to the head which is just a translation of $0.15m$ along the z axis in the head frame, which we will call T.

$$R_z = \begin{bmatrix} \cos\beta & -\sin\beta & 0 & 0 \\ \sin\beta & \cos\beta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$R_y = \begin{bmatrix} \cos\alpha & 0 & \sin\alpha & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\alpha & 0 & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0.15 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} x_h \\ y_h \\ z_h \\ 1 \end{bmatrix} = R_z \times R_y \times T \times \begin{bmatrix} x_l \\ y_l \\ z_l \\ 1 \end{bmatrix}$$

To convert the scans from head frame to body frame using the robot's base as the reference, we perform a translation of $1.26m$, which is the height of the head of the robot, in the negative z-direction. That is,

$$\begin{bmatrix} x_b \\ y_b \\ z_b \end{bmatrix} = \begin{bmatrix} x_h \\ y_h \\ z_h + z_{head} \end{bmatrix}$$

Now, let us suppose that the robot's coordinates in the global frame are given by $(x_r, y_r, 0)$ and a yaw given by ψ. The scans in the global frame coordinates are obtained using the equation:

$$\begin{bmatrix} x_w \\ y_w \\ z_w \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\psi & -\sin\psi & 0 & 0 \\ \sin\psi & \cos\psi & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_b \\ y_b \\ z_b \\ 1 \end{bmatrix}$$

Here, we discussed the transformations to obtain the global scan coordinates for the humanoid robot. Now we will discuss the particle filter algorithm that forms the crux of the SLAM algorithm.

C. Particle Filter SLAM

The location of the robot cannot be precisely predicted using odometry which warrants a particle filter based model to represent the distribution of states the robot is likely to be in. The algorithm typically starts with the assumption that the robot is at the origin of the map and after each observation, a motion update is made for each particle and the best particle decides the robot's estimated pose. The mapping is done using this pose estimate and the particles are resampled for the next iteration. The whole process is detailed in the subsections below.

1) Initialization: The number of particles is fixed and chosen to be 100 in this project. At the beginning of the algorithm, all the particles are initialized to be at the origin with a yaw angle of 0° and all the particles are assigned a uniform weight of 0.01. A global map of a fixed dimension ($80m$ by $80m$) and resolution ($0.05m$) in both directions is initialized with a value of 0 indicating unexplored grids. The first scan is used to update the initial map by adding a log odds value of $log(9)$ to points in the map where the laser hit, excluding the ground points; and subtracting the same log odds value of $log(9)$ to points that lie along the path joining the LiDAR on the robot to the obstacle points where laser beams hit. Certain other algorithm related parameters such as upper limits on the log odds and the noise covariance.

2) Localization Prediction: In this step, the pose of the particles is predicted using odometry. The odometry is the change in the pose of the robot which is determined using the difference in the subsequent measurements of the pose from the LiDAR. The odometry obtained is converted to local frame and then converted to the corrected global frame because the odometry provided by the kinematic model of the humanoid is not in the same global frame of reference as the map. The algorithm followed for the prediction step is explained below.

The delta increment in the pose is obtained by the difference in the measured pose.

$$\begin{bmatrix} dx & dy & d\theta \end{bmatrix} = pose_t - pose_{t-1}$$

The pose is converted to local frame using the previous yaw.

$$\begin{bmatrix} dx \\ dy \end{bmatrix}_{local} = \begin{bmatrix} \cos\psi_{t-1} & -\sin\psi_{t-1} \\ \sin\psi_{t-1} & \cos\psi_{t-1} \end{bmatrix}^T \times \begin{bmatrix} dx \\ dy \end{bmatrix}$$

$$d\theta_{local} = d\theta$$

This entire operation can be achieved using a Smart Plus operator in the SE(2) space. The Plus operation is done between the current pose and the negative of the previous pose. The next step is to transform it back to the global frame using the current yaw.

$$\begin{bmatrix} dx \\ dy \end{bmatrix}_{global} = \begin{bmatrix} \cos\psi_t & -\sin\psi_t \\ \sin\psi_t & \cos\psi_t \end{bmatrix} \times \begin{bmatrix} dx \\ dy \end{bmatrix}_{local}$$

$$d\theta_{global} = d\theta_{local}$$

Now the updated pose can be calculated as follows:

$$x_t = x_{t-1} + dx_{global}$$

$$y_t = y_{t-1} + dy_{global}$$

$$\theta_t = \theta_{t-1} + d\theta_{global}$$

In the localization prediction step, the pose for each particle is updated using the odometry along with an additional noise sampled from a 0 mean distribution having a predefined covariance. This covariance matrix is tuned differently for different map generations in the current implementation. The pose is therefore predicted according to the expression:

$$p_t = p_{t-1} \oplus (o_t - o_{t-1}) \oplus \mathcal{N}(0, \mathcal{W})$$

3) Localization Update: This step is the most significant step since it decides the basis on which the map should be updated and also defines the distribution of the particles for the next iteration. The first step is to identify the best particle which could represent the pose of the robot without significantly violating the previously generated map. The idea is to correlate the scans for each particle with the previously updated map and identify the one that not just provides the best correlation but also has significant weight from the previous iteration. The particle weights are updated using the following expressions:

$$a_t^{(j)} = \frac{a_{t-1}^{(j)} \times corr(z_t^{(j)}, m_{t-1})}{\sum_i a_{t-1}^{(i)} \times corr(z_t^{(i)}, m_{t-1})}$$

The correlation is computed by simply finding the number of occupied grids in the new scan that already exist as occupied grids up to the previous scan. Another idea is to compare the grids around the scanned grids with the previous map to find any nearby robot pose that could produce better correlation. However, this does not work well when using a linear weight update method. Instead the weight updates could be made using the likelihood of the particles which is measured as follows:

$$p_h(z_t^{(i)} \mid m_{t-1}) = \frac{\exp(corr(z_t^{(i)}, m_{t-1}))}{\sum_j \exp(corr(z_t^{(j)}, m_{t-1}))}$$

The weight update equations in this case appear to be as follows.

$$a_{t|t-1}^{(j)} = \log a_{t-1}^{(j)} + corr(z_t^{(j)}, m_{t-1})$$

The updated weights are normalized differently here as shown in the expressions below. This is because most of the particles have extremely high correlation and an exponent of that yields extremely large weights. Therefore even a small difference in the correlation gets penalized significantly.

$$a_{t|t}^{(j)} = a_{t|t-1}^{(j)} - \max_k a_{t|t-1}^{(k)} - \log \sum \exp a_{t|t-1}^{(j)} - \max_k a_{t|t-1}^{(k)}$$

This normalization technique handles the issue of overflow in the particle weight update. Once the weights are updated, the best particle is chosen to be the one with the highest weight and the maps are updated using the scans corresponding to that particle.

The next step is to resample the particles so that the stochasticity prevents the pose prediction to drift away from the true state. It is not required to resample at every instant. Therefore, a certain metric called effective particles is computed using the expression:

$$N_{eff} = \frac{1}{\sum_j (a_t^{(j)})^2}$$

If this effective particles is less than a certain threshold, the particles are resampled. This implies that whenever a majority of weights get accumulated within a few particles, the particles are resampled based on the current weight and are equally weighted for the next iteration. A systematic sampling approach was implemented which ensures that more particles were chosen from the particle with more weight and less particles from those that had less weight.

So the localization update step gives the estimated pose of the robot along with the updated particle poses and weights. A pseudocode in Algorithm 1 describes the entire particle filter algorithm that helps solve the problem of SLAM.

Algorithm 1 Monte Carlo Localization Algorithm for SLAM

Input: z_t, o_t

Output: m_{t+1}

Initialization : $p_0^n \leftarrow (0,0,0)$, $a_0^n = \frac{1}{N}$ for $n = 1, \ldots, N$

for *each* t *in* $range(0, T)$ **do**
  p_t^* = max(p_t^n, a_t^n) for $n = 1, \ldots, N$
  m_{t+1} = Mapping(p_t^*, z_t, m_t)
  p_{t+1}^n = LocalizationPrediction(p_t^n, o_{t+1}, o_t)
  p_{t+1}^n, a_{t+1}^n = LocalizationUpdate(p_t^n, a_t^n, z_t, m_{t+1})
end for
return m_T, p_t for $t = 1, \ldots, T$

IV. PERFORMANCE

In this section, we will discuss the techniques and approaches that were implemented and compare the results by analyzing the output map. Let us first describe the setup of the system before we analyze the results.

The map is initialized for an $80m \times 80m$ with a resolution of $0.1m$ in both the dimensions. The Particle filter is initialized with 100 particles with equal weights, all at the same starting pose defined to be the origin or the center of the map. The following robot configurations are defined: height of center of mass = $0.93m$, height of head = 0.33m above the center of mass, height of LiDAR at perfectly standing condition = 0.15m above the head. The head and neck angle are the joint configurations of the robot required in order to map the scans from the LiDAR exactly to the global coordinates.

The SLAM algorithm requires data from LiDAR which are filtered to remove scans that are unreliable below and above the minimum ($0.1m$) and maximum threshold ($30m$) respectively. Also certain scans that are observed to be at a height of $0.10m$ from the base are also eliminated since they may not be accurate as the resolution for the system is defined to be $0.1m$. It also requires the following parameters to be defined and tuned : the number of particles, noise covariance for the particles, threshold on effective number of particles for resampling, the log-odds for occupancy of grid, and the upper and lower caps on the log-odds, in order to get perfect mapping of the environment and this tuning is also flexible. In all our experiments, we fix the number of particles to be 100 which works well. The noise covariance is chosen

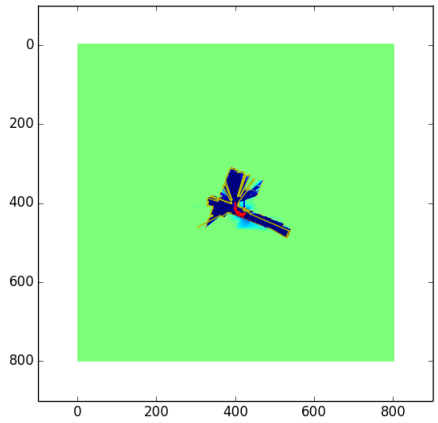


Fig. 3. Map for scan dataset 0

to be $[0.001, 0.001, 0.001]$ where the first two parameters are the covariances for the x and y coordinates in meters and the third parameter is the covariance for the yaw of the robot given in radians. This ensures that particles are sampled from a standard deviation of 0.25 degrees on either side of the predicted yaw angle from odometry. The threshold on the effective number of particles depends on the map correlation technique and the particle weight update used. In our experiments, we use the boolean map to obtain the normalized map correlation which gives a correlation in the range 0 to 1. The particle weights are updated using the linear weighting of the previous weights with the corresponding map correlation. For this scheme of weight updates, the threshold for the effective number of particles is chosen to be 90% of the initialized number of particles. An importance resampling is used when the estimated number of effective particles is less than this threshold. The whole procedure as explained in the algorithm section results in a log-odds map based on the choice of the log-odds and the upper and lower limits. We choose $\log(9)$ for the obstacle and $-0.5 \times \log(9)$ for free space. The upper and lower caps on the log-odds for the map is set to $100 \times \log(9)$ and $-100 \times \log(9)$ respectively. The results obtained following the above mentioned parameters and settings are shown in the Figures 3, 4, 5, 6. In all these figures, the dark blue region indicates free zone, light blue indicate scanned free zone but with less confidence, the green zone remains the zone that couldn't be scanned by the LiDAR. The red track in each of the figure traces the path followed by the robot corresponding to that map. The yellow lines indicate the contour lines made by the last scan points made by the robot.

In the Figures 7, 8, the results for the test data set are shown. The Fig 7 shows the positions and orientations of the robot with time for 2 cases: one being the output of prediction step alone, and the other being the output using particle filtering with 100 particles. It can be observed that the trajectories observed for the two cases are different. A better insight as to what is happening can be observed in Fig

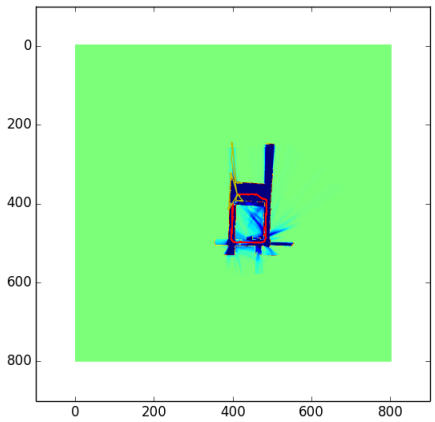


Fig. 4. Map for scan dataset 1

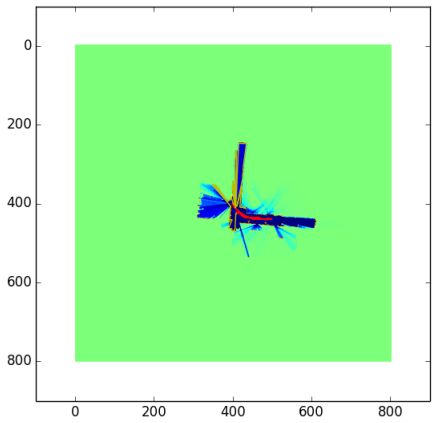


Fig. 5. Map for scan dataset 2

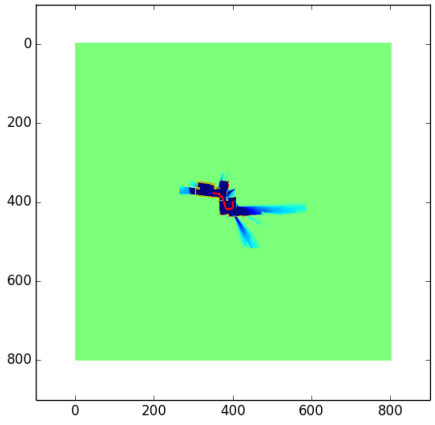
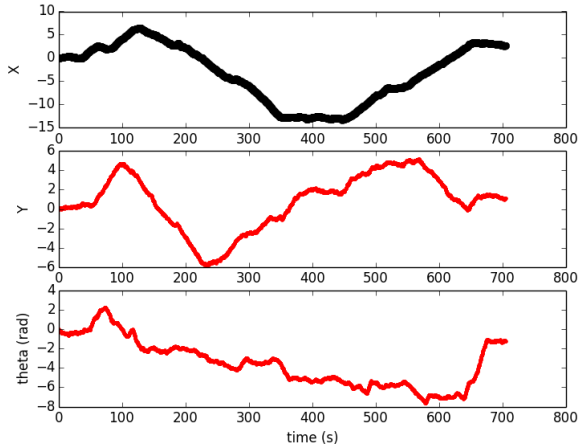
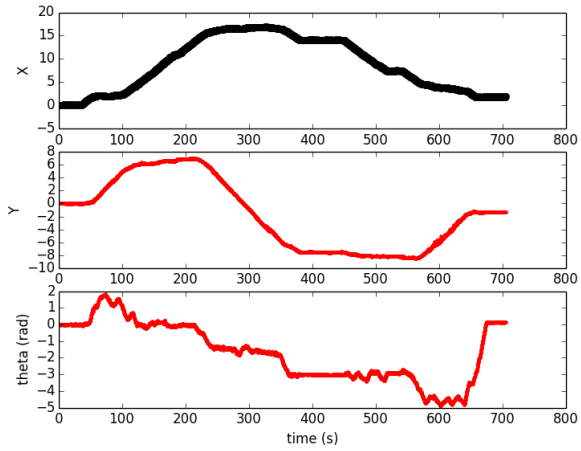


Fig. 6. Map for scan dataset 3



(a) Using a single particle and odometry prediction only



(b) Using 100 particles and map correlation based update

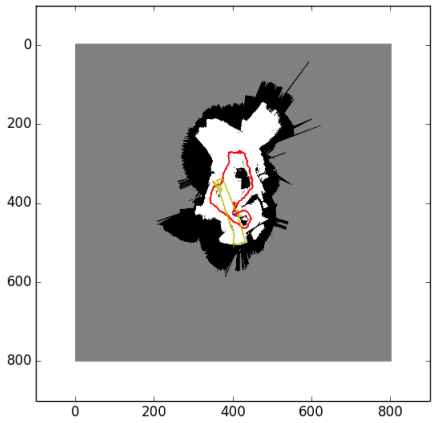
Fig. 7. Positions and orientations for test dataset

8 where the constructed map and the localization is shown for the two cases. The prediction alone does not produce good results since certain measurements go awry due to sensor inaccuracies. The result obtained using a particle filter with 100 particles is much more sensible.

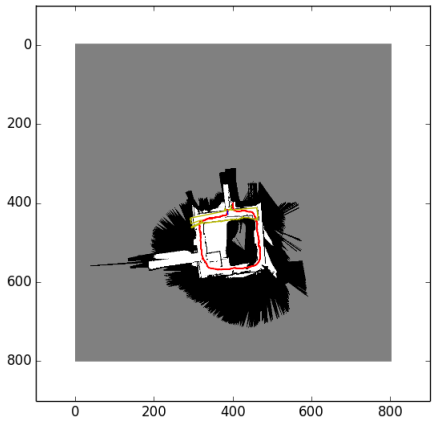
V. CONCLUSIONS

In this project, a particle filter based localization technique has been implemented to estimate the pose of a humanoid robot in the map that is simultaneously constructed using the LiDAR range sensor scans from the best particle estimate. The best particle is decided based on its scan's correlation with the previously constructed map.

Loop closure is one of the problems that is not forcefully handled by the current implementation but is inherently incorporated when minor shifts in the scans are incorporated in the map correlation. A better localization update could be achieved using a correlation with point clouds of the past scans but would require extensive memory and computing.



(a) Using a single particle and odometry prediction only



(b) Using 100 particles and map correlation based update

Fig. 8. Map outputs for test dataset

Further improvement can be made by refining the odometry by applying Kalman filter on gyro and accelerometer measurements from the IMU. This project used only LiDAR scans for generating a map of the environment. More recently, features from visual data are used to construct a more comprehensive map of the environment.

ACKNOWLEDGMENT

Thanks to University of Pennsylvania for providing me an opportunity to work on interesting projects by conducting a course on Learning in Robotics. Special thanks to Professor Nikolay A. Atanasov and the Teaching Assistants for providing the datasets and guiding me during the project.

REFERENCES

[1] hokuyo UTM-30LX specifications
[2] Nikolay A. Atanasov. Lecture Notes for ESE 650: Learning in Robotics, 2017.