EECS 568 / NAVARCH 568 / ROB 530 : Mobile Robotics
UMICH NCLT SLAP

Sravan Balaji*, Sabrina Benge†, Dwight Brisbin‡, Hao Zhang§, and Yushu Zhang¶
University of Michigan, Ann Arbor
{ *balajsra, †snbenge, ‡dbrisbin, §hzha, ¶yushuz }@umich.edu

Abstract

This project discusses the methods and approaches one should consider when implementing sequential localization and planning of a mobile robot in an uncertain environment. We implement a chatbot interface using semantic language processing for users to provide start and end locations for navigation. Then we look into the implementation of a particle filter for localization given sensor data from the North Campus Long-Term Vision and LiDAR dataset. Once location is determined, we consider A algorithm for path planning and model predictive control to design a velocity controller for the robot.*

***Keywords**: Particle Filter, Semantic Language Processing, Model Predictive Control, A* Path Planning.*

***Project Resources**: Project Website, Repository, Video Summary, Presentation, and Generated Models from NCLT Data*

1. Introduction

A lot of work has been done in the field of autonomous systems to be able to navigate and perform tasks in an unknown environment. These systems rely on sensor data and pre-collected information, but sensor data can be noisy, wheels can slip, and unexpected disturbances like wind can have a large impact on performance. Thus, engineers have come to rely on probabilistic methods for dealing with uncertainty.

When navigating around a map, autonomous systems need methods of determining their location within it using the data collected from sensors. One approach to this is a particle filter. This involves starting with random guesses as to the robot's location, then evaluating the likelihood of observing the collected sensor data if the robot were at each of these guessed positions. By comparing the actual measurements to the expected measurements if the robot were at a particular location, the particle filter allows the robot to create a probability distribution of the likelihood of its location. Over time, as more landmarks (objects of known location within the map) are observed, the particles become more concentrated around the likeliest location of the robot, thus providing a good estimation of position within the known map.

Once a position is determined, a robot can plan a path to reach its target destination using algorithms such as A*. This approach assigns a cost to locations in a map. This cost is based on the number of steps taken to reach the position and a heuristic based evaluation of distance from the target. Thus, a robot can explore a map and find a path to the destination that reduces this cost.

Once a path is determined, a robot needs a control strategy to follow the path it found. This is where model-predictive control (MPC) comes in. MPC allows for the design of a controller that takes into account system constraints on states and inputs while solving an optimization problem that will determine the necessary system inputs to perform a task like trajectory tracking.

In this project, we explore methods for sequential localization and planning (SLAP) of a segway robot around a map. We made use of the University

of Michigan North Campus Long-Term Vision and LiDAR (NCLT) Dataset [1] for point clouds collected by a 3D LiDAR and ground truth pose information. To control robot velocity, we use model predictive control. Additionally, we implement a semantic language processing interface for users to interact with the simulation. Inputted sentences are processed and keywords, like North Campus building names, are extracted to determine start and end locations for navigation.

2. Methodology

Figure 1 shows the overall architecture of the interaction between the different components of sequential localization and planning. To begin, the user provides building names for the robot's start and end position via a chatbot-like interface. Semantic language processing is used to extract building names from the provided sentence and determine GPS coordinates to be used by our path planning algorithm. Once a starting location is determined, our path planning algorithm performs a search of the ground truth map to find a route that will get the robot closer to the desired destination. This path information is used by our velocity controller to send motor commands to the simulated robot. Then the particle filter uses point cloud data provided in the NCLT dataset to localize the robot in the map of North Campus. All data from these components is saved to a data file that is used to visualize our results. We have implemented semantic language processing, path planning, localization, and velocity control individually, but have not integrated all components together. We discuss each component separately in this section.

2.1. Semantic Language Processing

To enable users to have a more natural experience with our system, we have created a chatbot to interface with the remaining system. [1] The user can request the buildings they would like to go to from there current location in the form of a sentence such as: "Please take me to BBB from the duderstadt" and in turn the navigation will confirm selection then begin navigating.

1. Example chatbot code available.

2.1.1 Chat Bot

The chat bot utilizing keras backend to create a bag of words model, spacy.io to parse sentences into their grammatical types, and python natural language toolkit (NLTK) to manage the input data itself [2]. The chatbot construction begins with setting up intents. Intents are used to train our chatbot on the types of commands it is likely to receive and the appropriate responses. The intents include a tag: used for referencing the intent, patterns: the commands the chatbot is likely to receive, and responses: appropriate responses by the chatbot. The patterns are parsed using a bag of words model to extract features from the example patterns. This allows the chatbot to match new input commands similar to the patterns in the intents file. Once the features are extracted, the bags are sent through a LSTM network using keras backend to create a model that we can compare features to with new commands. Therefore when the chatbot receives a command, it utilizes bag of words to extract and compare features to know bags of words in our model. The model gives us a list of the most likely tag that matches our input command. The chatbot then randomly selects a response from the intents file based on the tag. This sets up a basic back and forth conversation. To extend the chatbot to start navigation, we utilized spacy, a library that parses sentence structure and a second bag of words model. If navigation is the returned tag, the chatbot uses the python library spacy to parse out the sentence's proper nouns near the key words to and from. Once the nouns are found, the chatbot checks if they are a building available on the map using a second bag of words model to match. The second bag of words model was trained on buildings instead of sentences as input. Therefore, the chatbot is able to recognize BBB as the Bob and Betty Beyster Building. A view of the chat bot is seen in 2.

2.1.2 Change to Local Frame

After determining the start and end building names, the chatbot changes the names to the local frame (x,y,z) coordinates from the GPS coordinates. The GPS coordinates are selected from the two nearest doors of the start and end building locations. The local frame coordinates follow the following directions x:north, y:east, and z:down. To convert from GPS coordinates to the local frame, we linearized around

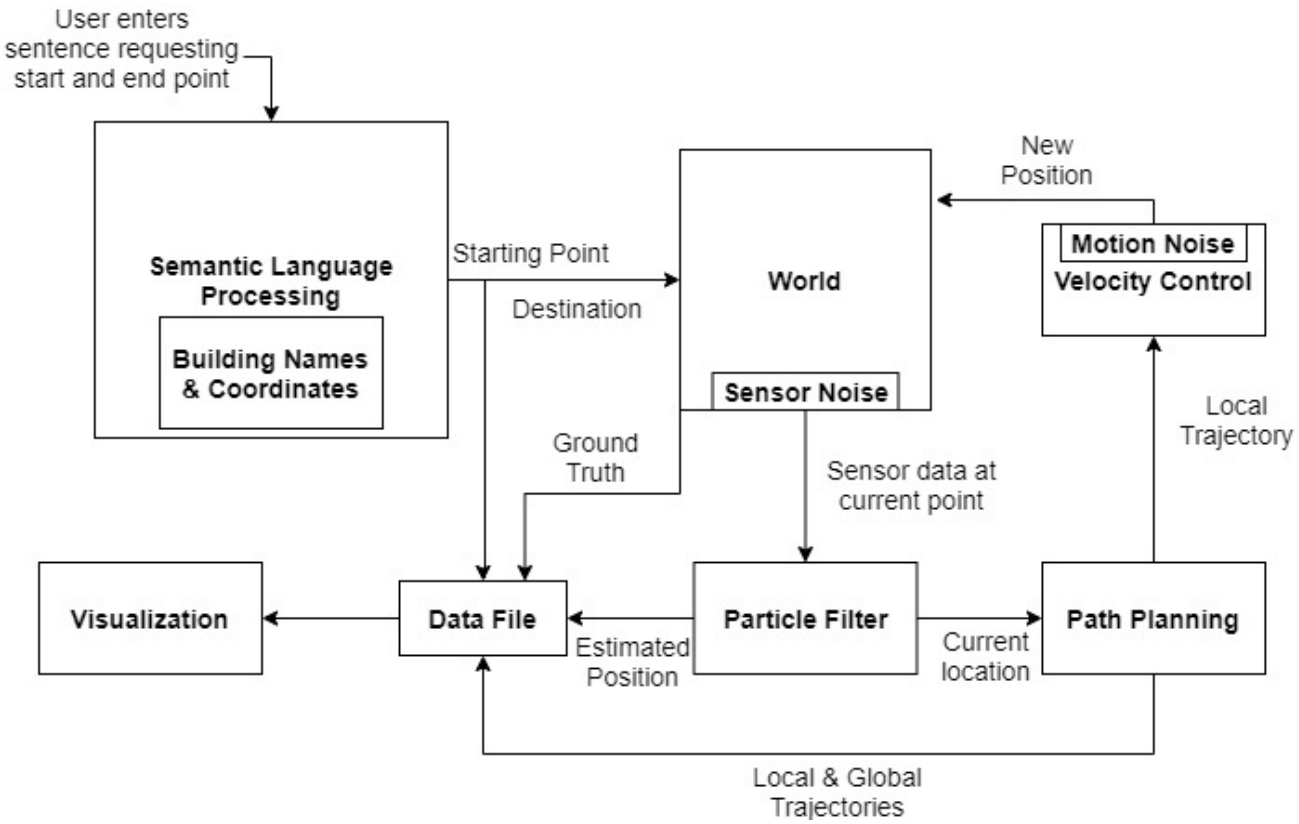

Fig. 1: Architecture diagram for segway robot simulation

the origin (42.29 latitude, -83.71 longitude, 270 m altitude) [2] based on the approximate radii of the earth along with the latitude and longitude. This is shown in eq. 1:

$$\mathbf{r_{ns}} = \frac{(r_e r_p)^2}{((r_e \cos lat_o)^2 + (r_p \sin lat_o)^2)^{\frac{3}{2}}}$$
$$\mathbf{r_{ew}} = \frac{r_e^2}{\sqrt{(lat_o)^2 + (r_p \sin lat_o)^2}} \quad (1)$$

where r_{ns} is the earth's radius north to south, r_{ew} is the earth's radius east to west, r_p is the earth's polar radius (6,356,750 m), and r_e is the earth's equatorial radius (6,378,135 m). With these linearization constants, we calculate the local frame coordinate using eq. 2:

$$\mathbf{x} = \sin(lat - lat_o) r_{ns}$$
$$\mathbf{y} = \sin(lon - lon_o) r_{ew} \cos lat_o \quad (2)$$
$$\mathbf{z} = alt_o - alt$$

Note that the positive direction of the z axis is oriented downwards. Once the start and stop locations are finalized in the local frame, the navigation can begin.

2.2. Feature Extraction and Localization via Particle Filter

Due to time and hardware limitations discussed in section 4, we ultimately decided to make use of the existing open-source pole landmark extraction implemented by Schaefer et. al. [3]. Our project video shows the particle filter process, which is described in [3]. This is comprised of pole detection, mapping, and localization. In modern architecture, poles are found everywhere and generally do not move over time, thus making them ideal to serve as landmarks for localiza-

2. Exact coordinates available on first line of gps data files from nclt database in radians

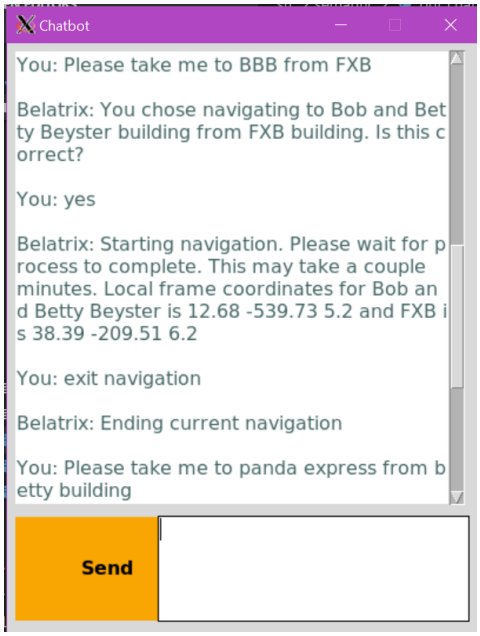

Fig. 2: Chatbot interface

tion even if weather or lighting conditions change. As the segway robot drives around, point clouds are collected from the LiDAR sensor, which can be used to estimate the average relative location and width of the poles from the robot's current location. Using ground truth poses provided from the NCLT dataset, these relative pole locations can be transformed into the global frame, to build up the 2D map of pole locations.

The process of building up this global reference map is not as simple as transforming the local frame measurements to global frame coordinates however. Due to the arbitrarily large size of this global map and finite memory storage restrictions, Schaefer et. al. [3] suggest constructing intermediate local grid maps that can later be combined into the arbitrarily large global map. This process involves defining a fixed size local map, collecting LiDAR within this map, and registering overlapping features so they can be combined into one global map. The registration process and global map creation can be made easier by keeping the axes of the local maps aligned with the global map. Then overlapping poles detected in multiple local grid maps can be averaged together in the global map, resulting in an estimate of pole locations and widths.

This global map is treated as ground truth when new trajectories are generated for the purposes of localization. One could generate the global map using the NCLT dataset, then drive a robot around north campus with a LiDAR sensor and use a particle filter to determine where in the map the robot is at a given time. The results of a simulated version of this process are shown in section 3.

2.3. Path Planning

A* algorithm is a graph-search based method for path planning. It is optimized Dijkstra's algorithm; A* algorithm finds the optimal path to one location, while Dijkstra's algorithm finds the paths to multiple locations [4]. Furthermore, A* makes use of a heuristic which usually reduces the number of explored nodes. In our implementation, we used the euclidean distance heuristic which is admissible for our graph. For the implementation, we made use of a priority heap to track the nodes to be explored next (i.e., those with the least estimated cost to the goal given by the heuristic). Since our dataset does not have any restrictions on distance between neighboring nodes, we utilized a KD-tree to efficiently find the nearest neighbors when a node is selected for exploration. The result of our A* implementation is a set of k pose indices which index into all poses in the dataset. We call the corresponding positions β_i for $0 \leq i \leq k$.

Once we have the path, we can parameterize configurations $X(s)$ in task space by a variable s in the range $[0, 1]$ such that $X(0) = \beta_0, X(1) = \beta_k$. To find the position at any s, we first derive the twist corresponding to the screw motion. This is obtained by

$$\log X_{start}^{-1} X_{end}$$

for X_{start}, X_{end} in the world frame. Using the twist we simply multiply the motion by s to obtain:

$$X(s) = X_{start} \exp\left(\log\left(X_{start}^{-1} X_{end} s\right)\right).$$

The twist between each pose in the path can be obtained and this will fully describe the motion from the start to the goal by iteratively using the corresponding twist on the current pose described in the appropriate Lie Group. We implemented this on the group SE(2), but as defined, the approach works for any SE group.

A final note on path planning is that we applied cubic interpolation to the poses obtained from A*.

This approach yields a smooth position over time; however, higher order moments are not smooth at the via points used. We could apply a higher order polynomial interpolation technique to smooth the higher moments including velocity, acceleration, jerk, etc. [5].

2.4. Velocity Control

Model Predictive Control (MPC) is a multi-variable control method using linear-time-invariant dynamic model [6]. The MPC controller predicts behavior across the prediction horizon at each time interval, and then computes optimal control output based on that prediction.

MPC is widely used for trajectory tracking. In our project, we proposed to do a short-term trajectory tracking since the current location estimate from Particle Filter is always changing over time and the controller would actually take only the control output of the first interval even though the MPC calculates control outputs of all intervals. Short-term trajectory tracking would also benefit the whole computing time.

A non-linear model is used for velocity control. The model has 4 states and 2 outputs:

$$\boldsymbol{x} = \begin{bmatrix} x & y & v & \theta \end{bmatrix}, \quad \boldsymbol{u} = \begin{bmatrix} a & \beta \end{bmatrix} \tag{3}$$

where (x, y) is the global 2d coordinates of the robot location, v is the linear speed of the robot and θ is the heading angle with respect to x-axis. a is the linear acceleration and β is the angular velocity.

The non-linear kinematic is described as below:

$$\begin{cases} \dot{x} = v\cos(\theta + \beta) \\ \dot{y} = v\sin(\theta + \beta) \\ \dot{v} = a \\ \dot{\theta} = \frac{v}{l_r}\sin(\beta) \end{cases} \tag{4}$$

where l_r is the length of the robot body along the heading direction. The model is linearized and discretized since the MPC controller can only use LTI dynamic model.

The cost function is

$$f = \sum_{i=1}^{k-1}[(|x_i - x_i^r| + |y_i - y_i^r|)(10 - 2i) + 10(i+1)\beta] + 100[(x_k - x_k^r)^2 + (y_k - y_k^r)^2] \tag{5}$$

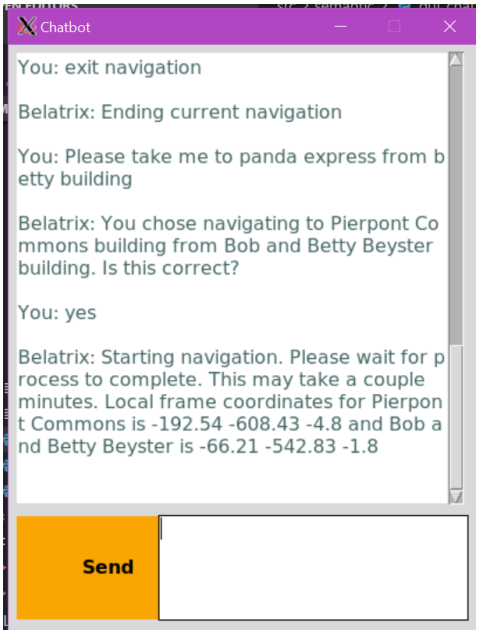

Fig. 3: Snippet of chatbot conversation showing command to start navigation and conversion to local coordinates.

where k is the number of poses ahead of the current pose, and (x^r, y^r) are the pose coordinates of reference trajectory.

3. Results

3.1. Semantic Language Processing

Upon completion of the training of the chatbot using the bag of words models, the chat is able to hold a conversation about navigation, control the flow of the conversation, and convert building names into local frame coordinates. This can be seen in Figure 3.

3.2. Localization using a Particle Filter

As mentioned in section 2, the poles detected in the robot's local coordinate frame can be transformed into the global map frame using the ground truth poses provided by the NCLT dataset. From this, we can perform localization within the global map to generate a plot of the driven trajectory of the segway robot along with locations of pole landmarks. Figure 4 shows the locations of poles extracted from the point cloud data collected by a Velodyne HDL-32E LiDAR sensor as blue dots. The red line represents the localized trajectory of the segway robot as it

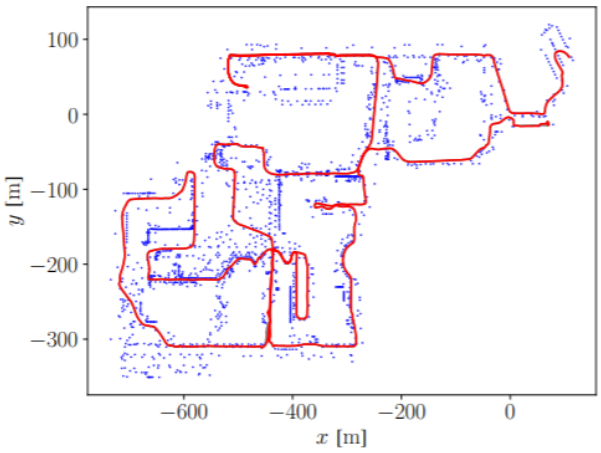

Fig. 4: Pole landmark map created from the NCLT dataset [1], figure reproduced from [3].

drove around the University of Michigan's North Campus, making use of the known locations of these landmarks.

Using the global reference map built from one or more data collection sessions, the robot's pose can be localized for a new trajectory. As shown in the project video, the particle filter starts at a known position and uses odometry data to propogate the distribution of particles forward. The distribution of particles spreads outwards, accounting for wheel slippage and other unaccounted for motion uncertainties until landmarks are observed. Particles that deviated far from the true pose become less likely given the observed position of known landmarks, thus concentrating the re-sampled particles around the true pose. We overlayed the resulting pole landmark map with trajectory from the 2013-01-10 session of the NCLT dataset on top of google maps [7] road and satellite images to qualitatively assess the accuracy. After some scaling and stretching, the trajectory lines up very well with the roads and sidewalks visible from the satellite image, confirming the results presented in [3].

3.3. Path Planning

We validated the implementations of each of the motion planning techniques already discussed. Our A* implementation performed quickly and always achieved an optimal path. The twist sequence varies depending on the path, but usually has around $1,000$

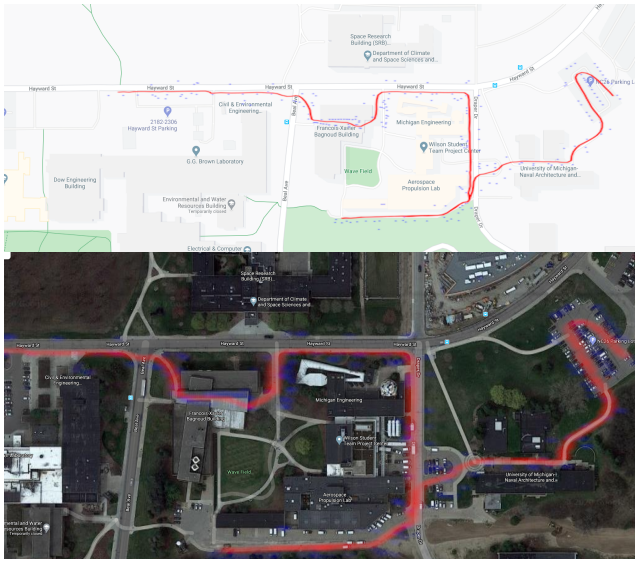

Fig. 5: Localized trajectory for 2013-01-10 session overlayed on google maps [7] road and satellite view of area around University of Michigan North Campus FXB building.

twists which when executed arrive at the correct end pose. The interpolation is graphically shown to yield smooth position as intended.

3.4. Velocity Control

We choose $k = 5$ path poses generated by A* ahead of the current pose as the reference trajectory. The computing of optimal control outputs utilizes *cvxpy* to minimize the cost function (Figure 6).

4. Discussion

4.1. Feature Extraction Methods

To localize within the dataset, we explored multiple methods of extracting features from sensor data. These included registration of point clouds collected by a LiDAR sensor, image identification using SIFT matching on camera data, and feature extraction and matching using Visual Bag of Words.

Registration of LiDAR data utilizes the iterative closest point (ICP) algorithm to match point clouds and calculate any transformations and rotations needed to compare the clouds [8]. This methodology is great for finding the optimal transformation between point clouds that are similar, but would not work well for our application. We need to quantify

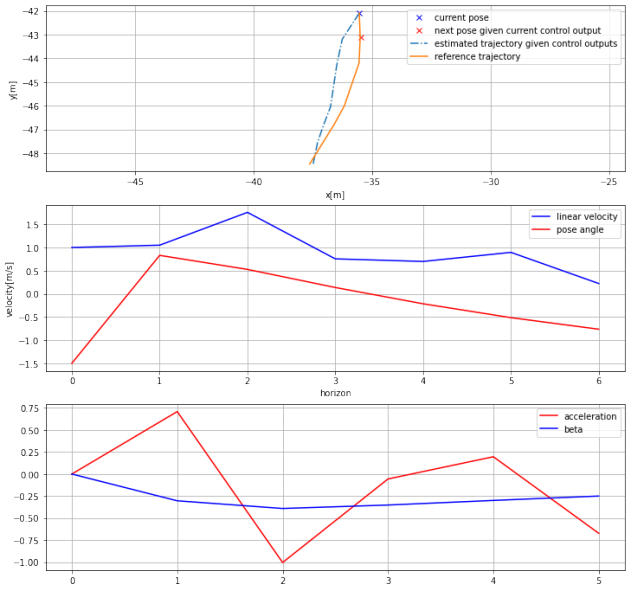

Fig. 6: MPC outputs across the prediction horizon.

the difference between measured sensor data and ground truth information, but a direct application of ICP does not provide this capability since the point clouds and ground truth data significantly vary. Thus, any transformation we find would not be appropriate because the point clouds are not meant to be transformed to match. Additionally, we were not able to find a robust error calculation that would provide a likelihood of observing a particular point cloud given a ground truth map. Therefore, we moved onto the next approach.

Next, we analyzed methods to compare images (rather than point clouds) to take a more computer vision based approach to solve localization. This methodology would compare an image from a camera on the robot to the images collected from other data collection sessions. To elaborate, we would build a map of images collected from one session of the NCLT dataset as ground truth, then use images collected from a different session as the sensor data. Thus, we would not have to artificially add noise to the sensor data before comparing it to ground truth. SIFT feature detection is one approach that reliably extracts scale-invariant features from images, thus making them appropriate for determining if two images of the same object (perhaps taken at different angles) are similar. By extracting SIFT features from sensor data and ground truth data, we could determine the so called distance between the images which will inform the particle filter's likelihood calculation [9]. This methodology was quickly abandoned after we realized the processing power and storage that would be required. Each session video from the NCLT dataset was 37 - 130 GB. Feature extraction from this many images would not have been feasible in our time constraints.

The alternative methodology we attempted was visual bag of words. This allows use to extract features using a SIFT algorithm on point clouds, group the features together into a vocabulary, and create a histogram training image of the number of times each of the features in the vocabulary appears. Then we can train a classifier using the histogram vector to generate bags of visual words to enable localization. This method requires many components and models. The current models are not appropriate for LiDAR data or required more processing power than we had available [10], [11]. While we did attempt these methods, our results were computer crashes and the terrifying blue screen of death. One computer ended up needing to have windows completely reinstalled. Due to the high workload required to complete this and size of the LiDAR data, approximately $\sim$10 - 93 GB per session, this was not feasible with our local processing power either.

Therefore, with other areas explored, we moved on to the feature extraction using pole landmarks method described in section 2.

4.2. Future Work

As mentioned in this report, due to limited time and computing resources, we were unable to explore the implementation of image-based localization in depth. This seems like a very promising path given the tremendous advances in computer vision that have been made through the use of feature detection and convolutional neural networks. While pole detection is adequate, the replacement using visual bag of words would be a significant improvement and allow for applications where poles are sparse or non-existent. Implementing bag of words within the chatbot gave us insight into the algorithm and how it can be used. Extending this to also do visual bag of words would be a research path the would greatly improve this project. Using processing power

on campus would have allow for greatly improved results by speeding up processing and analysis of the different methodologies to parse point clouds.

We also would have liked to fully integrate all components of the project together. Due to technical limitations, such as Python version and library dependencies, this was not pursued further.

5. Conclusion

From the current work completed, we were able to explore and evaluate the various components necessary to simulate a robot moving around campus based on path planning and velocity control for navigation and chatbot for user control using point cloud data to localize.

The chatbot was able to give us insight into bag of words implementation and give a simulation into how a user could easily interface with a system. This implementation is easily editable and deployable on other systems. Instructions available on our project website.

The localization methods we researched and attempted included registration of point clouds collected by a LiDAR sensor, image identification using SIFT matching on camera data, feature extraction and matching using Visual Bag of Words, and pole feature extraction and localizing using a particle filter. Each of these methods have their own pros and cons as discussed in 2 and 4. This shows that multiple methodologies are required in robotics as we move forward due to limitations in robotics around processing power. While most research focuses largely on accuracy, our implementation was largely focused on computing power needed. This highlights a need for more lightweight systems and algorithms in robotics.

All further documentation and video can be found on the project website.

6. Acknowledgments

We would like to recognize the course staff of Mobile Robotics for their help this semester. Lecturer Maani Ghaffari, Graduate Student Instructor Tzu-Yuan (Justin) Lin, and Graduate Student Instructor Peter Westra were instrumental in helping us learn many robotics concepts through lectures and homework assignments. They were very flexible in transitioning the class to an online format due to the COVID-19 pandemic while still providing a valuable educational experience.

References

[1] N. Carlevaris-Bianco, A. K. Ushani, and R. M. Eustice, "University of Michigan North Campus long-term vision and lidar dataset," *International Journal of Robotics Research*, vol. 35, no. 9, pp. 1023–1035, 2015.

[2] D. Team, "Python chatbot project - learn to build your first chatbot using nltk keras," Jan 2020. [Online]. Available: https://data-flair.training/blogs/python-chatbot-project/

[3] A. Schaefer, D. Büscher, J. Vertens, L. Luft, and W. Burgard, "Long-term urban vehicle localization using pole landmarks extracted from 3-d lidar scans," in *2019 European Conference on Mobile Robots (ECMR)*. IEEE, 2019, pp. 1–7.

[4] J. Lerner, D. Wagner, and K. Zweig, *Algorithmics of large and complex networks: design, analysis, and simulation*. Springer, 2009, vol. 5515.

[5] K. M. Lynch and F. C. Park, "Modern robotics: Mechanics, planning, and control," 2017.

[6] A. Franco and V. Santos, "Short-term path planning with multiple moving obstacle avoidance based on adaptive mpc," in *2019 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC)*. IEEE, 2019, pp. 1–7.

[7] Google, "University of michigan - north campus," April 2020. [Online]. Available: https://goo.gl/maps/95KVt3n43tPsrZwL8

[8] W. F. Costa, J. P. Matsuura, F. S. Santana, and A. M. Saraiva, "Evaluation of an icp based algorithm for simultaneous localization and mapping using a 3d simulated p3dx robot," in *2010 Latin American Robotics Symposium and Intelligent Robotics Meeting*, 2010, pp. 103–108.

[9] E. Karami, M. S. Shehata, and A. J. Smith, "Image identification using SIFT algorithm: Performance analysis against different image deformations," *CoRR*, vol. abs/1710.02728, 2017. [Online]. Available: http://arxiv.org/abs/1710.02728

[10] J. Martínez-Gómez, V. Morell, M. Cazorla, and I. García-Varea, "Semantic localization in the PCL library," *CoRR*, vol. abs/1601.08158, 2016. [Online]. Available: http://arxiv.org/abs/1601.08158

[11] M. A. Uy and G. H. Lee, "Pointnetvlad: Deep point cloud based retrieval for large-scale place recognition," *CoRR*, vol. abs/1804.03492, 2018. [Online]. Available: http://arxiv.org/abs/1804.03492