Capstone Group 6

Baraa Al Jorf, Eddie Han, Manuel Padilla, Priyanshu Mishra

Table of Contents

1. Abstract

This capstone project aims to implement a fully autonomous library navigation robot that is able to assist librarians in daily library organization duties. To achieve this goal, the robot will have unsupervised operational capabilities including self-localization and pathfinding, as well as being able to interact with the environment through its 6 degrees-of-freedom arm manipulator. The robot will also utilize AI for its various subtasks, with its capabilities being trained in a simulated virtual environment to accelerate its learning process.

The four major subcategories the project relies upon are; self-localization and mapping (SLAM) to be able to understand itself and the surroundings, navigation and pathfinding to be able to move within its given environment, robot arm manipulation to perform object movement tasks necessary to achieve its goal and simulation to train and attest the operation of the robot.

2. Problem Definition

2.1. Problem Analysis

1. Who has the problem?

Vehicular automation offers numerous valuable advantages across various industries. Automation presents a promising technology, particularly in the realm of menial manual labor, and the global workforce increasingly relies on its benefits. Once tasks are automated, industries can experience scalable development. For instance, in product assembly lines, automation has significantly enhanced deployability and efficiency. Similarly, in healthcare applications, the automation of surgical procedures shows great promise in ensuring higher accuracy and reducing human error.

With this in mind, our project aims to develop a robot capable of automating the manual labor aspects of a librarian's job, specifically sorting and finding books. We believe this automation will bring considerable benefits to librarians worldwide, and our initial focus is on the NYUAD library, where we plan to deploy our project.

However, it is important to acknowledge that autonomous vehicle technology has the potential to disrupt society, particularly in terms of job displacement. The ability of autonomous vehicles to replace nearly 10% of the transport industry workforce is a notable concern (5). Although this figure primarily applies to the United States and includes jobs that will be minimally impacted by autonomous vehicle technology, it still suggests a significant portion of the workforce may become unemployable through no fault of their own.

Thus, this capstone project serves as both an exploration of the potential jobs that may be replaced by autonomous vehicle technology and a demonstration of the achievable complexity by an undergraduate student team. Our goal is to prove that autonomous vehicles can be valuable tools in facilitating and improving existing jobs, rather than solely replacing them.

2. What does the problem seem to be?

One of the challenges faced by librarians is the significant amount of time spent on menial tasks that restrict productivity but can be readily automated. Particularly, tasks such as locating a specific book, sorting it, or delivering it to the individual who requested it may appear straightforward and suitable for robot assistance. However, these tasks consume a substantial portion of librarians' time. It is important to note that while this aspect of the job may be susceptible to automation, manual labor represents only a fraction of the work performed by librarians.

3. When does the problem occur?

The issue arises when there is a reliance on low-skilled manual labor for tasks involving navigation and transportation of objects in a warehouse-like setting. With the rise of e-commerce and the increasing demand for "x-day delivery" marketing strategies, such jobs have become more prevalent in recent years.

4. Where does the problem occur?

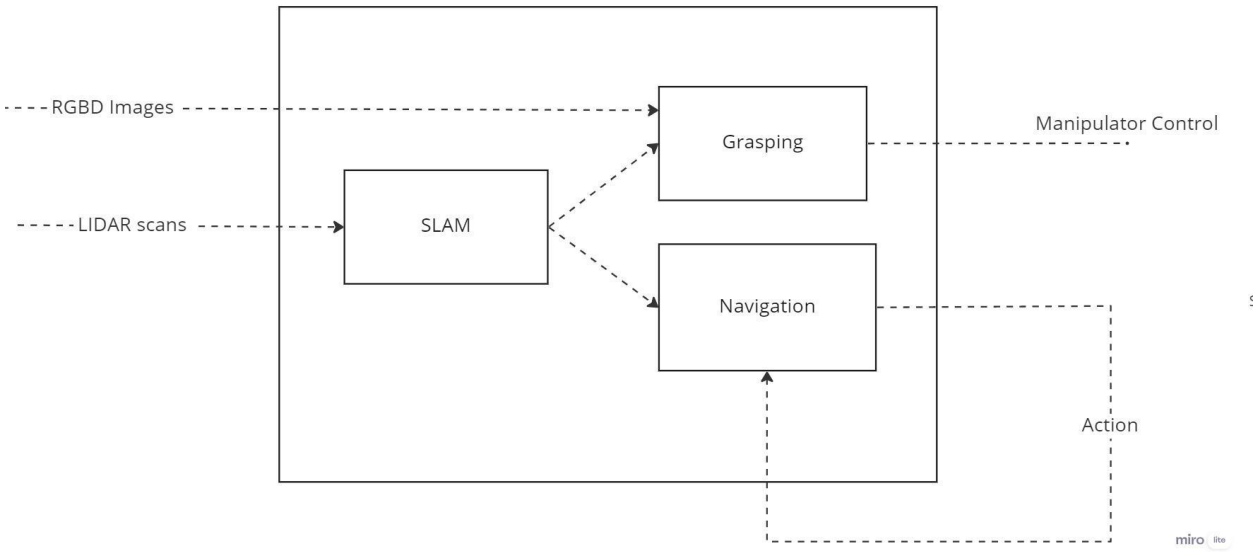
The problem occurs in any environment where manual labor is prevalent.

5. Why are we contributing to the problem by designing a system that threatens these jobs?

It is crucial for researchers, including the capstone team, to focus on implementing the autonomous robot system. This approach allows for a deeper understanding of the jobs that may be at risk and the complexity of the technologies involved. By gaining this knowledge, researchers can make more accurate predictions and provide valuable advice to society. It is essential to acknowledge that the progress of technology is unstoppable and irreversible. Therefore, it is wiser to approach the issue with caution and prudence rather than avoidance.

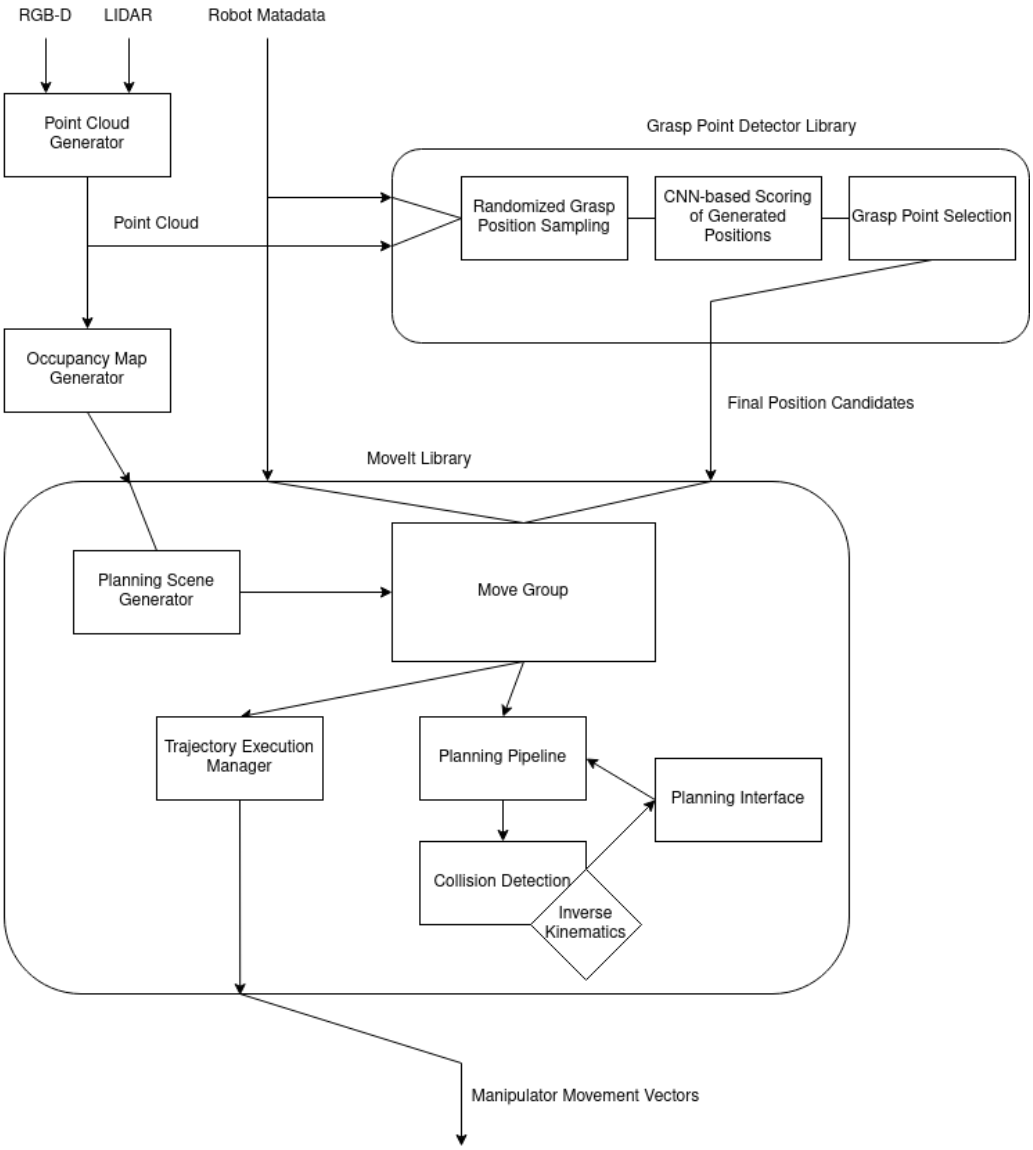
2.2. Problem Clarification - Black-Box Modeling

Full Black-Box Modeling



Detailed Boxes-

Grasping -



INPUTS:

Point cloud data of object to grasp

Point cloud data of the immediate environment

Metadata about manipulator

OUTPUTS:

Movement vectors for robot arm manipulator

Navigation - Describe Policy and Reward function + Flowchart

INPUTS:

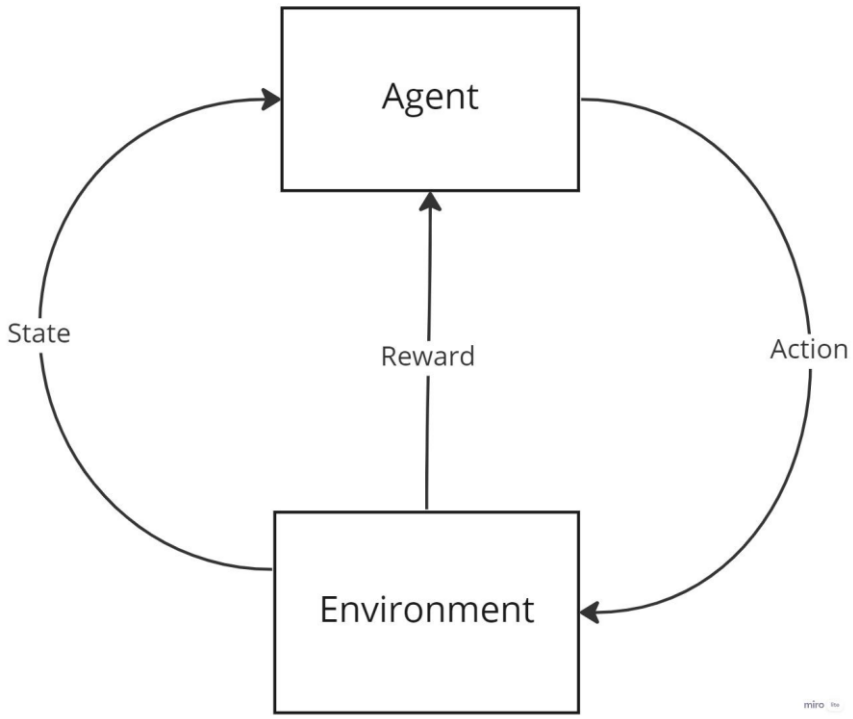
Point cloud data of environment

Metadata about the current vehicle's location and its functionality

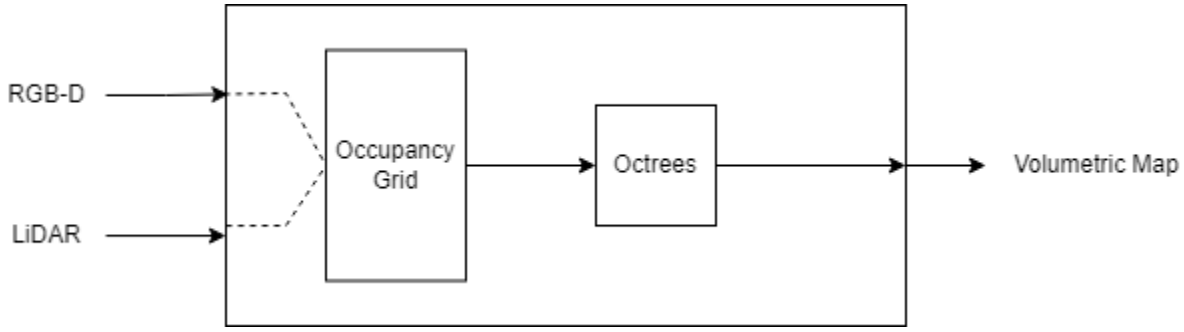
Task to be performed

OUTPUTS:

Movement vectors for robot wheels



SLAM - Describe SLAM algorithm and how input is turned into point cloud



INPUTS:

LiDAR data

RGB data

Metadata about the current vehicle's status and operational functionality

OUTPUTS:

Volumetric Map of environment

Simulation

INPUTS:

3D Models (with aptly assigned properties)

Integration of all other components into the simulation

OUTPUTS:

General Simulation Environment for testing with other project components

2.3. Problem Statement

"Build a robot that is able to self-navigate the NYUAD Library. Build a robot that is able to autonomously place a book on the shelf."

Description:

Idealized Description;

A student should be able to borrow a book from anywhere in the library. Upon request, the robot should be able to locate the book in the library, retrieve it from the rows of books, and then deliver it to the student. The robot should also be able to take a student's book and return it to a correct

spot on the shelf.

Realistic Description;

A user in the library should be able to request a book from the shelf. Upon this request, the robot should be able to navigate from its ‘home’ to a known location where the book is, retrieve the book from a sparse shelf, and navigate towards the designated ‘goal’ location, and drop off the book. The robot should also be able to pick up the book from its carriage and place it on a sparse shelf.

Key Technologies

Computer Vision, Automation, Simulation, Machine Learning

- Note; there exists other important aspects of required technologies such as hardware control, computer vision for object detection, and more - these were considered not part of the designated capstone students’ responsibilities and were delegated to project collaborators.

3. Design Constraints

3.1. Performance Constraints

Movement of the vehicle at walking speed (approx. 1m/sec) or less; the objective is to reach between 0.5 m/s to 1 m/s movement speed.

Grasping and placing the book on the shelf should take less than 3 minutes to execute to completion

Full completion of the task should take no more than 1 hour to complete; the objective is to reach approx. 30 minutes of execution time, depending on the complexity of the task and the distance required to travel

Able to navigate to its charging station and charge itself during non-operation

Continuously operational for approx. 5 hours before requiring charging

- During use-case scenario should enable last the entire Library operational day (9 am to 12 pm) without disruptions due to charging

Operational Uptime of 99%

- Non-operational Uptime of 80% (service and maintenance during Library closing hours)

Target navigational error of <50 cm, pathfinding navigational error of <5cm

3.2. Safety Constraints

Since this capstone relies heavily on the use of robotics, the guidelines detailed by the American National Standards Institute in the current revision of the industrial robot safety standard are followed. Among the most applicable guidelines, this capstone project wants to ensure that

1. The robot should not collide with stationary objects (walls, shelves, tables)

2. The robot should not collide with moving objects (people navigating the library)
    - The robot should slow down / stop movement upon potential collision
    - Desired extension; robot should perform predictive evasive action

3. The robot should not cause damage to people, property, nor itself upon accidental collision

3.3. Software Constraints

All deployed software should be a free-to-use license
Runnable on provided hardware GPU: NVIDIA RTX 3070 Ti or 3080 Ti

3.4. Non-technical Constraints

In order to allow the robot to know where the book is, the robot must also be integrated with pre-existing library systems that catalog and identify the location of the book.

Note; these ‘constraints’ are more akin to ‘desirable functionalities’ and are not required for the minimum viable functionality of the robot.

4. Criteria for Design Evaluation and Testing

To evaluate the functionality and efficiency of the proposed design, we have established the following testing criteria:

- **Integration**: The individual sections of the project must be integrated onto a single platform, with no version compatibility issues, to ensure that the robot can perform all its designated tasks.
- **Navigation Accuracy**: The robot must be able to navigate autonomously to a set location accurately, with an accuracy of +/- 10cm.
- **Obstacle Avoidance**: The robot must be able to detect and avoid stationary and moving objects in its path

- **Safety**: The robot must not cause any damage to people, property, or itself upon accidental collision, with a success rate of 100%.
- **Grasping Efficienc**y: The robot must be able to grasp a book accurately and efficiently within 3 minutes, with a success rate of 95% or higher.
- **Testing Environment:** The robot must be tested in an environment that mimics the NYUAD library warehouse, with a variety of different objects and obstacles to simulate a real-world environment.

5. Deliverables Statement

5.1. Hardware

The purchased vehicle that has been used for testing and deployment should be included

- Vehicular part
- Robot arm manipulator part
- Computer & GPU running the software
- RGB-D camera
- LIDAR sensor

5.2. Software

- Github repository & step-by-step instructions on installing and running the software on similar hardware
- Trained ML model saved on the deployed hardware

5.3. Other

- Capstone report and poster
- Records of robot function in the simulated environment
- Video demonstration of the robot’s operation

6. Conceptualization

6.1. Background Research

Vehicular Automation, also known as 'self-driving' technology, revolutionizes the way vehicles operate by minimizing human input. It is a rapidly evolving field with the potential to bring about

significant changes in the world. The impact of this technology is already being felt in society, demonstrating its immense potential.

However, when deployed in real-world scenarios, autonomous vehicles still encounter several challenges that undermine public trust. Issues such as inaccurate object detection and pathing calculation glitches raise concerns about the reliability of autonomous vehicles.

Fortunately, these problems become less critical when working with smaller-than-human vehicles that operate at lower speeds, minimizing the risk of bodily harm in the event of a crash. This makes it an achievable goal for a student capstone project. Therefore, the objective of this project is to develop a fully-functioning autonomous vehicle capable of operating within the NYUAD library. The vehicle will incorporate deep-learning-based grasping, object detection, and navigation techniques within the library's three-dimensional space.

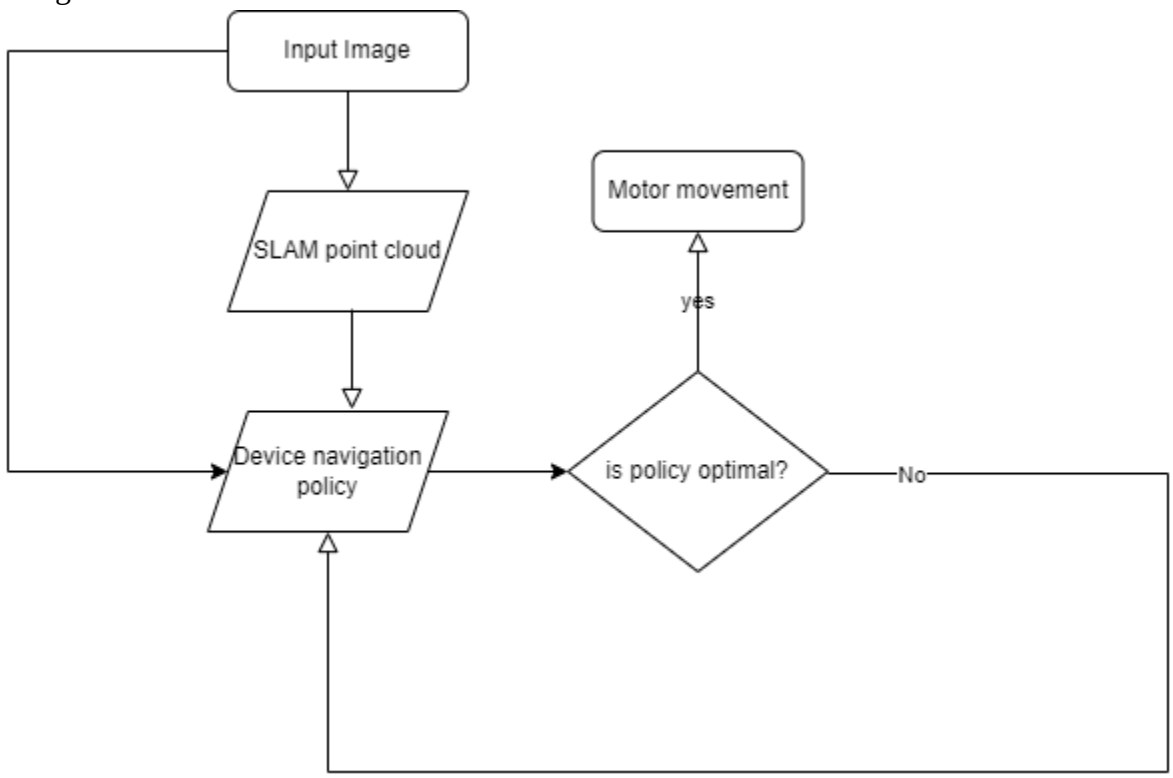
While there are no existing examples of a library navigational robot, the individual components of this task have been successfully implemented in various industries. Robot arm manipulation using inverse kinematics has found application in the manufacturing sector, while simultaneous localization and mapping (SLAM) and pathfinding algorithms are actively being developed in the automotive industry. The aim of this project is to integrate these disparate solutions into a cohesive platform tailored for library navigation.

6.2. Concept Generation with Morphological Chart

Morphological Chart for Comparing Different Design Parts of the Project		
Navigation	Reinforcement Learning using https://docs.omniverse.nvidia.com/app_isaacsim/app_isaacsim/tutorial_ros_navigation.html	Reinforcement Learning using navigation on Gazebo Simulator: https://www.neobotix-docs.de/ros/ros1/simulation.html
SLAM	Hector-Slam	G-mapping
Environment	Omniverse/Isaac Sim based real-time tracking and simulation	Gazebo based pre-planned simulation
Grip Planning	Based off https://ros-planning.github.io/moveit_tutorials/doc/moveit_grasps/moveit_grasps_tutorial.html	MoveIT ROS framework, TrajOpt algorithm: https://rll.berkeley.edu/trajopt/doc/sphinx_build/html/

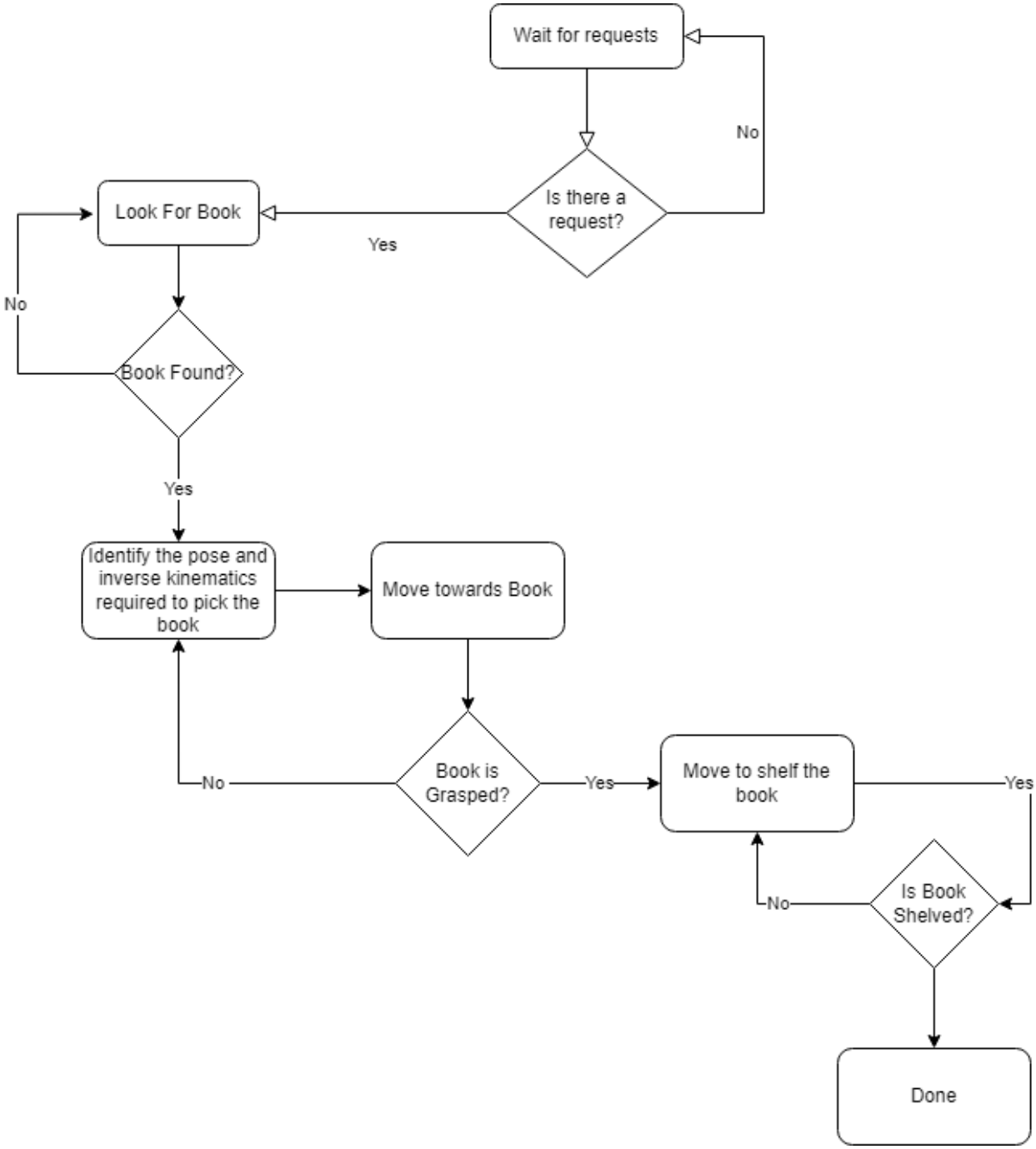
7. Modeling, Simulation, and Optimization / Experimentation Plan

Navigation + SLAM Flowchart



The flowchart shows the flow of information as the input images are processed. The input images are fed into SLAM and along with the Lidar signals produce a point cloud map. This point cloud is then used to create the navigation policy that the robot is set to follow. Using reinforcement learning, the policy is evaluated based on the overall loss (reward.) Only when the policy is evaluated to be optimal does the robot start moving.

Grasping Flowchart



For the grasping, the robot waits until there is a request for a book to be shelved. Once a request comes, the robot navigates to the book, finds it, uses inverse kinematics to grasp it and then shelves it.

8. Final Design Expected

In this section, we will provide an overview of the proposed project's final expected design, which we aim to have fully implemented by the end of May 2023. The design revolves around the project's intended environment, the library, where the robotic system will operate.

In the final design, the robotic system will showcase the ability to navigate autonomously through the library environment, effectively avoiding any obstacles it encounters along the way. This navigation is facilitated by utilizing the point cloud information obtained from the LiDAR and RGB-D sensors, coupled with the SLAM and navigation components specifically developed for this project, as described earlier. The objective is to enable the robot to travel between two predetermined points within the library.

Once the robot reaches its assigned location, the grasping aspect of the project comes into play. The robotic system is designed to detect objects, specifically books in this case, and generate a feasible plan of action for picking up and placing the object. This plan is then executed using the dedicated grasping modules developed specifically for this project, as detailed in the preceding section. Once the object, such as a book, is securely grasped, the final design aims to enable the robot to autonomously navigate to another assigned position and carefully release or place the book at its designated location. This process can be repeated for other objects, each with their respective assigned locations within the library.

In summary, the final design encompasses a robotic system capable of autonomous navigation in the library environment, obstacle avoidance, object detection and grasping, and precise object placement. The objective is to enable the robot to perform the assigned tasks of picking up books, navigating to designated positions, and placing the books accordingly, all while maintaining the highest level of precision and reliability.

8.1. SLAM Modeling

Simultaneous Localization And Mapping is a technique used to enable a vehicle to build a map of the surrounding unknown environment using sensor data (usually LiDAR) whilst simultaneously allowing the vehicle to localize itself within the given space. In the scope of this project, SLAM uses the aid of both RGB-D sensors and a 2D LiDAR sensor. The main challenges faced by the SLAM techniques are noisy or dynamic enviroments and large storage requirements. For navigation to be successful SLAM has to work alongside robot controls in order to assert that the intended motion signaled by the controls is being done correctly. Redundant sensors and measurements are used to improve the localization and information of the robot’s motion.

Interoceptive Sensors are used to measure scalar or vector quantities such as wheel encoders, gyroscopes, and accelerometers.[7]

Exteroceptive Sensors require more computations on the measurements in order to apply them to

the environment building. This include range sensors such as LiDAR and RADAR, and cameras such as stereo cameras and kinect cameras [7].

In this project we will be using an online SLAM technique. In other words, all computations are done based only on the current pose of the robot and previous measurements are updated recursively. There are three main requirements that must be met for robot autonomous navigation in a library [6]:

1. Probabilistic representation: SLAM creates a map based on the probabilistic occupancy estimation information generated by the exteroceptive sensors' measurements. The measurements will always be afflicted with uncertainty; however, the robot might face random measurements caused by dynamic obstacles or certain surface reflections. Therefore, a probabilistic representation is needed to have an estimate of the current state of the environment of the robot.
2. Modeling of unmapped areas: When performing path planning, a collision free path has to be done towards the target which would normally avoid unmapped areas. However, the robot is supposed to be able to autonomously create the map and also update the previous map according to new measurements of the unmapped areas.
3. Efficiency: The generated map is required to be efficient according to both access time and storage for the robot to successfully navigate and perform calculations in real time.

3D representations can be done in many ways including, point clouds, multi-leveled surface map, elevation maps, or volumetric representations. The Library Robot will use the volumetric representation for the following reasons. The point clouds are unable to model unknown areas and are unable to deal with high levels of noise or dynamic obstacles. They are mostly suitable for high precision static environments. Additionally, the amount of memory occupancy keeps increasing without an upper bound. The elevation map results in a grid in which each cell retains one value representing the elevation at that point. This representation assumes that a 2.5D model is normally enough. The drawback is that the generated map does not as a volumetric representation and is therefore useful for path planning but it is not efficient for localization as it does not represent the actual environment. Lastly, the multi-leveled surface representation takes the multi-volume occupancy approach in which a volumetric representation is created; however, it is assumed that the extent of the map is known beforehand and map updates are more computationally expensive.

The octrees data structure (representation) is able to update properly and does not require the size of the map beforehand. Although there have been other approaches that use octrees, the Library robot is using a compression technique that maintains the probability information required for future updates as well as tackles the problem of overconfident maps and multi-resolution queries.

The sensor data is integrated using an occupancy grid. "The probability of a leaf node n to be occupied given the sensor measurements z_1:t is estimated according to:

$$P(n \mid z_{1:t}) = \left[1 + \frac{1 - P(n \mid z_t)}{P(n \mid z_t)} \frac{1 - P(n \mid z_{1:t-1})}{P(n \mid z_{1:t-1})} \frac{P(n)}{1 - P(n)}\right]^{-1}$$

This update formula depends the on the current measurement z_t , a prior probability P(n), and the previous estimate P(n|z_1:t-1). [6]" The octree is developed using an occupancy tree generated by the sensors which basically is a series of basic boolean inputs that say whether a certain space is occupied at moment t.

- Navigation Modeling

  Navigation technology refers to being able to generate an efficient path that a robot is able to take, adapting to changing environmental conditions as required. Navigation is a critical part of automation, though it relies on SLAM and robot control processes to operate correctly. In this capstone, Navigation is modeled using reinforcement learning. Reinforcement learning models the environment to be navigated using a Markov Decision Process where a 4-tuple (S, A, P, R) determines the best course of action

  - S refers to a set of states called the state space
  - A is a set of actions called the action space
  - $P_a(s, s') = \Pr(s_{t+1} = s' \mid s_t = s, a_t = a)$ refers the probability that action a in state s at time t will lead to state s' at t+1
  - R(s,s') is the immediate reward received after transitioning from state s to s' after action a.

The goal is to find the best "policy" that optimizes the reward function. The agent reacts with the environment in discrete time steps with the goal of optimizing $\pi : A \times S \to [0, 1], \pi(a, s) = \Pr(a_t = a \mid s_t = s)$.

8.2. Grasping

Grasping technology plays a crucial role in the robot's capability to detect objects, formulate a plan for picking up and placing them, and execute the plan accordingly. Given the complexity of this task, the students' responsibility primarily involves generating the manipulator movement vectors based on the provided inputs.

The grasping black box assumes that RGB-D data, LIDAR data, and the robot manipulator's metadata are the inputs. However, the SLAM algorithm also generates point cloud data and occupancy map data, which can be utilized by the grasping section. In the actual implementation, the point cloud and occupancy map are sourced from the preceding section.

The first step involves utilizing the robot's metadata and point cloud data with the Grasp Point Detector (GPD) library. This library generates a set of potential positions from which the gripper can attempt to grasp the object. It accomplishes this by randomly sampling positions relative to the given point cloud data and using a CNN (Convolutional Neural Network) to assign scores to these positions based on their likelihood of resulting in a successful grasp. A selector then identifies a limited number of the most promising grasps based on their scores.

Next, the robot's metadata, occupancy map, and the selected grasp candidates are fed into the MoveIt library. MoveIt transforms the occupancy map into its own environment called the "planning scene" and utilizes its inverse kinematics suite to generate the desired movement vectors for the manipulator. These vectors are derived through the planning pipeline, and the generated plans are tested within the move group's internal scene before the final movement vectors are produced.

In summary, the grasping technology involves leveraging the Grasp Point Detector library to generate potential grasp positions based on point cloud data, and using a selector to identify the most promising grasps. Additionally, the MoveIt library is employed to convert the occupancy map into a planning scene, generate manipulator movement vectors using inverse kinematics, and validate the plans before finalizing the movement vectors.

8.3. Simulation

Simulation technology refers to creation of a virtual environment upon which the robot is able to conduct experiments in. As all the components rely heavily on machine learning, the simulation is a critical step in enabling the robot to learn its operations. The project involves scanning four primary locations around the campus at NYU Abu Dhabi, namely the Performance Gym, Library (both of which are high priority, and the Highline (ground and second floors, both of which are low priority as of now), using the Unitree B1, the Dashgo D1 (and a stationary LiDAR scanner FARO Focus). The Unitree B1 is equipped with a 3D LIDAR sensor that generates a 3D map of the environment and detects obstacles, enabling safe navigation. The point cloud data generated by the LiDAR sensors requires processing before it can be used for generating or using in a virtual environment. This virtual environment, when created by the process of triangulation and mesh computation and conversion, can be used for testing and incorporating the project’s implemented autonomous navigation, obstacle avoidance and grasping mechanisms as discussed above.

9. Implementation Details

9.1. Grasping

9.1.1. Perception

Perception of the environment was assumed to have been provided by a RGB-D camera. In practice, this assumption was implemented by using the default RGB-D camera provided by the MoveIt! program and attaching it to the robot arm at a plausible location (edge-most arm where the manipulator is also located) through MoveIt!’s manipulator group generation program. This camera location can be

changed depending on the live run of the robot without significantly impacting the performance of the robot.

The RGB-D information was converted into point cloud information and then processed using the simple_grasping package provided by ROS. This enables the robot to find and execute manipulator positions that can grasp and release an object. Furthermore, this package is able to differentiate between a level surface and a graspable object in question. The point clouds of the target object and the surface are then filtered for noise and then pushed to two separate ROS publishers. Finally, all these tasks were packaged as a single ROS service that can be called upon.

9.1.2. Motion Planning

Motion Planning was done by coding ROS nodes from scratch using the moveit_python package. It calls upon the perception service created above and uses the live feed to constantly recalculate the potential trajectory to take to grasp the object. Moveit_python mostly takes care of inverse kinematics and collision detection issues on its own, only requiring that the moveit group be configured correctly. The actual task of moving the manipulator itself is also executed through the moveit_python package. As of now, a custom location was created as the dropoff location, but modifications to both the perception service and the motion planning service can be made to be able to adaptively find a desirable dropoff location.

The implementation was done as a simple state machine. It normally sits idle until called upon. Once it is called upon, the camera scans and finds an object to be picked, refreshes the simulation scene, and then calls the moveit_python pick_with_retry() function with the point clouds of the surface and the object as inputs. The robot then performs the pickup motion, throwing an error when either no objects are detected or an unresolvable collision is detected.

9.1.3. Issues Faced with Grasping

The most notable issue faced with grasping was the assumption that the perception service works as intended - it was noted that if the pipeline fails to detect a graspable object, either due to the failure of the simple_grasping package and/or the camera moving away from the view of the target object, then the whole pipeline throws an error and stops executing. There has yet to be sufficient testing in regard to how an external point cloud information service would affect the pipeline created.

9.2. Navigation

9.2.1. Twin Delayed Deep Deterministic Policy Gradient:

Twin Delayed Deep Deterministic Policy Gradient (TD3) is a deep reinforcement learning algorithm that is designed to learn continuous control policies. It is an extension of the original DDPG algorithm that addresses some of its limitations and improves its performance. The TD3 algorithm utilizes two deep neural networks, called actor and critic, to learn the policy and the value function, respectively. The actor-network is responsible for mapping the states to actions, while the critic network estimates the expected rewards for a given state-action pair. The TD3 algorithm uses three key techniques to improve the performance of the original DDPG algorithm:

1. Twin networks: Instead of using a single critic network, TD3 employs two critics that independently estimate the value function. This helps to reduce the overestimation of the Q-values and improves the stability of the learning process.
2. Delayed updates: The TD3 algorithm updates the actor and critic networks less frequently than in DDPG. This helps to reduce the impact of noisy and correlated updates and improves the stability of the learning process.
3. Target networks: TD3 uses target networks to improve the stability of the learning process. The target networks are copies of the actor and critic networks that are periodically updated with the weights of the online networks. This helps to reduce the variance of the learning process and stabilizes the value estimates.

9.2.2. DRL Repository:

Deep Reinforcement Learning (DRL) is a powerful method to teach robots to learn through experience. DRL-robot-navigation is a GitHub repository that contains code for training a robot to navigate through a simulated environment using DRL. The repository utilizes the Twin Delayed Deep Deterministic Policy Gradient (TD3) neural network to teach a robot how to navigate to a random goal point in a simulated environment while avoiding obstacles. To use this repository, one should have ROS Melodic, PyTorch, and Tensorboard installed on their system. A Velodyne sensor simulator is also used in the implementation. To install the repository, first clone the repository using the command line. The next step is to compile the workspace by navigating to the catkin_ws directory and running the catkin_make_isolated command. After the workspace is compiled, set up the sources using the provided commands in the README file. The training is done by running the train_velodyne_td3.py file using the command line. The script can be found in the TD3 directory. To visualize the training process, one can use Tensorboard by navigating to the TD3 directory and running the tensorboard

command. After the training is completed, the model can be tested by running the test_velodyne_td3.py file using the command line. The script can be found in the TD3 directory. This repository eliminates the need for SLAM as TD3 does not require a mapping of the environment for it to operate successfully.

9.3. Simulation

9.3.1. Environment Scanning:

For the Environment Scanning stage of the implementation, the project involves scanning four primary locations around the campus at NYU Abu Dhabi, namely the Performance Gym, Library, and the Highline (ground and second floors), using the Unitree B1, the Dashgo D1 (and a stationary LiDAR scanner FARO Focus). The Unitree B1 is equipped with a 3D LIDAR sensor that generates a 3D map of the environment and detects obstacles, enabling safe navigation, which is all controlled by a human operator handling the robot dog through a physical controller that sends control signals to its controller system. The Dashgo D1, on the other hand, has a 2D LIDAR sensor that creates a detailed 2D map for navigation and object recognition, which is controlled precisely through a virtual controller running on a laptop, which regulates the unit's velocity and position. The LiDAR sensors from both of the aforementioned robotic units are used in conjunction to generate a dense point cloud information and positional data while the onboard RGBD sensors are used to capture color information for the environment (along with additional depth information).

9.3.2. Simulation Generation:

For the aspect phase of the Simulation, the project requires the use of the aforementioned captured point cloud and positional information from the robot units in the environmental scanning phase of the project. The point cloud data generated by the LiDAR sensors requires processing before it can be used for generating or using in a virtual environment. This processing is done using several open-source tools, most notably using software like CloudCompare (for filtering and pre-processing the point cloud data) and MeshLab (which is used for the triangulation of meshes from the point cloud information). To elaborate a bit on the software being used, CloudCompare and MeshLab are both open-source 3D point cloud and mesh processing software that provides a wide range of tools for point cloud processing, such as filtering, registration, and segmentation.

In the preprocessing stage, we first filter the point cloud data in CloudCompare to remove any noise or outliers. This is achieved by applying a statistical outlier removal filter to the point cloud data. Once the point cloud data has been filtered, we then perform registration of the pre-processed point cloud data, which is the process of aligning multiple point clouds into a common coordinate system. This is done using an iterative closest point (ICP) algorithm to align the point clouds. Once the point cloud data has been filtered and registered in CloudCompare, we move on to MeshLab to perform triangulation of meshes on the same. This is achieved by converting the point cloud data into a mesh representation using the Poisson surface reconstruction algorithm. The output of this algorithm is a converted triangulated mesh representing the scanned environment.

Now that the scanned environment is processed, we proceed to use it in a virtual setting by using the triangulated mesh generated in the previous step to generate the environment in Gazebo, which is an open-source physics-based simulator that allows for the simulation of complex robotic systems in a virtual environment. In Gazebo, after importing the processed mesh as listed before, we add the necessary physics properties to the model, such as friction and collision properties, to simulate the real-world environment accurately. Finally, we also add obstacles to the environment to test the autonomous navigation and obstacle avoidance systems. The obstacles are added to the environment as Gazebo models, and their physics properties are also defined accordingly. This provides us with the apt virtual environment for testing and incorporating the project's implemented autonomous navigation, obstacle avoidance and grasping mechanisms as explained in this report.

10. Issues

10.1. Navigation

The biggest issue with navigation was related to library dependencies in most navigation frameworks online. Since ROS is a constantly evolving backbone, it was thus a struggle to ensure all library dependencies worked with the capstone project's ROS version. After a lengthy trial and error, it was settled that the best ROS version to use that worked for all installed libraries and used frameworks was ROS melodic which is a sub-distribution of ROS 1.

10.2. Grasping

By far the biggest issue of implementation of the grasping task was the technical errors caused by ROS. These included, but were not limited to, version errors with Linux and ROS Noetic/Melodic, conflicting simultaneous python library version requirements, conflicting setup of catkin workspace, conflicting packages, and more. Due to the strict version requirements of most ROS packages, and the fact that many of the packages used legacy systems meant that even a minor technical error would cause major roadblocks. Furthermore, there appears to be major issues when attempting to identify an object to pick up from point cloud data. This latter issue could have been resolved by the simple_grasping package, but technical issues have prevented the package from being implemented on any available robot files.

10.3. Simulation

In the current stage of the simulation aspect of the project, there are two primary issues: generalized scanning through obstacle/noise ridden environments, and point cloud to environment conversion in Gazebo. As far as the first of these issues is concerned, the main difficulty is associated with selecting and using the right scanners and scanning practices to obtain the best possible dense point cloud data, in this case obtained from a Unitree B1 and primarily from a stationary Faro Focus LiDAR scanner. While most of the scans obtained from the scanners, especially the stationary one when used in a place like the performance gym were very consistent, when trying to scan the library (which was full of obstructions and reflections), the used scanners proved to be difficult to use to get a single dense scan (and so to use them, would require the merging and noise-cleaning for multiple scans). This brings us to the latter of the two issues, which involves using the generated point cloud scans as environments within Gazebo. Here, the process followed works very well for converting the point cloud data, by computing the normals and converting it into a mesh, But the technical issue here is the importing and usage of the aforementioned mesh files into workable Gazebo world files, rending us currently unable to utilize the scanned environments (Performance Gym, Library) as environments for testing the other two aspects of the project.

11. Outcomes

11.1. Navigation

The initial expectations here were fully realized as the robot is able to train on any environment, familiar or not, and develop a policy to follow to reach a given goal. The project was successful in using a training function that saves the policy to be used later by a testing function developed by the capstone team.

11.2. Grasping

The initial expectations for the grasping section of the project was the ability of the MMO 500 robot platform to identify an object of interest and pick it up, and then also place it down at a pre-designated location with minimal input from the user (no guidance). The delivered project shows the MMO 500 robot manipulating its movements from point A to point B based on prior movements manually set by the user on the MoveIt! GUI.

11.3. Simulation

The initial expectations for the simulation system of the project was to utilize the two robots Unitree B1 & Faro Focus scanners to scan the mentioned environment and use the obtained point cloud information to convert to meshes and use as virtual environments for simulating and testing the other aspects of the project(navigation and grasping). The delivered projects, takes the aforementioned issues into account and presents the scanned environments and a sample Gazebo simulation for testing the navigation and grasping.

12. Future Scope

12.1. Navigation

Navigation is an area where we showed a lot of encouraging progress. The future steps in this project would be to do more testing of the robot in different environments and test whether adding moving obstacles to the environment improves the overall performance of the robot.

12.2. Grasping

A docker package of functioning MoveIt! A simulation where all the version issues, library conflicts, and workspace setup has been resolved (up to and including the simple_grasping package) would be highly desirable for any future work. This would enable the training of ML algorithms in the Gazebo simulation environment.

12.3. Simulation

A direct implementation of the converted simulation Gazebo world, generated from the point cloud information from the environment scans, all packaged into a Gazebo world and urdf file, allowing direct, and more realistic testing of the navigation and grasping aspects of the project, allowing for a more polished implementation of the simulation, based on future work.

13. **Ethics**

The preceding sections of this paper have extensively covered the technical design aspects of the proposed project. However, it is equally essential to consider the ethical implications and concerns associated with this project in order to develop a comprehensive plan for its execution. As outlined above, the primary elements of this project involve designing a robotic system within a pre-scanned environment, capable of detecting and navigating around obstacles, with a grasping system mounted on top to handle the retrieval of objects, specifically books from library shelves. In this specific context, there are certain ethical considerations that can be categorized into two types: case-specific and general.

The case-specific concerns arise mainly from the physical environment in which the project is implemented, namely the library. While these concerns are not overly inhibiting, they warrant careful attention:

1. Positioning and movement of the robot within the library may be disconcerting or disruptive to individuals in close proximity.
2. The robot's navigation system could inadvertently collide with people walking in the environment, causing minor inconveniences.
3. The grasping system of the robot, responsible for handling objects, may occasionally lead to unintended issues in terms of book organization.
4. The physical components of the robot may be susceptible to unknown factors within the library environment, necessitating regular checks.

The general concerns regarding this project pertain to the broader concepts of navigation, mapping, and object grasping, and can offer valuable insights for project improvement or the exploration of alternative engineering designs:

1. The robotic system designed for this project, like any robotic system, possesses certain vulnerabilities, both in terms of hardware and software, that could potentially be exploited for malicious or unintended purposes.
2. The current design of the robotic system relies on point cloud information for object grasping. However, accurately identifying the specific object to grasp from a group of closely aligned objects without implementing a separate object detection mechanism linked to a comprehensive database remains a ongoing concern.
3. The ultimate aim of this robotic system is to replace a specific workflow typically assigned to a librarian or similar roles, which raises its own ethical dilemma.

By acknowledging these ethical considerations, we can proactively address potential challenges and seek ways to mitigate any adverse effects.

14. Bill of Materials

No	Part	
1	Realsense d435 RGB-D Camera	https://store.intelrealsense.com/buy-intel-realsense-depth-camera-d435.html
2	Lambda Tensorbooks (High-Performant Laptops)	https://drive.google.com/file/d/1jfTbzAxFIPDl6Vhvh8nSUxEIOP8mhwmt/view?usp=sharing
3	LIDAR Scanner (Faro Scanner)	https://www.faro.com/en/Products/Hardware/Focus-Laser-Scanners
4	Rosbot Mini (TX)	https://www.amazon.com/Generic-Rosbot-Mini-Pi/dp/B0B1P5QNGD/ref=sr_1_2?crid=2V70PX4D5KX3H&keywords=RO

No	Part	Webpage	Price in $	Quantity	Total	Purchase Justification	Note (optional)	Purchase State

1	Realsense d435 RGB-D Camera	https://store.intelrealsense.com/buy-intel-realsense-depth-camera-d435.html	314	3	942	This project aims to utilize a combination of these sensors mounted on a 3D printed bracket on a remotely controlled trolley/robot to accurately scan environments (NYUAD library), used in conjunction with the LIDAR sensor.	Required soon	Needs to be bought
2	Lambda Tensorbooks (High-Performant Laptops)	https://drive.google.com/file/d/1jfTbzAxFIPDl6Vhvh8nSUxEIOP8mhwmt/view?usp=sharing	3499	2	6998	Based on our plans and calculated assumptions, we'll be needing at least 2 high-performance laptops (with a dedicated Nvidia RTX GPU or equivalent), required for each of the main task divisions, running remotely for the duration of our capstone, as the simulation software/environment (Gazebo) that we're trying to run requires a Linux based environment and requires a RTX series/equivalent GPU. Also given that most of the current work structure for the project is divided into multiple	Required ASAP (we're assuming that these have already been ordered)	Accessible now (since the 1st week of November)

						indviualized tasks, its paramount that we have access to separate machines/laptops capale of executing/testing the code and simulation aspects of our project.		
3	LIDAR Scanner (Faro Scanner)	https://www.faro.com/en/Products/Hardware/Focus-Laser-Scanners	-	1	-	The main purpose of the Faro LiDAR scanner to scan the environment (the library in this case), and generate the point cloud information, required for creating the mesh to be imported into our virtual simulation (inside gazebo), and usue that in conjuction with the RGB-D sensors for mapping and navigation.	Required soon	Could be used/borrowed from the Kinesis lab
4	Rosbot Mini (TX)	https://www.amazon.com/Generic-Rosbot-Mini-Pi/dp/B0B1P5QNGD/ref=sr_1_2?crid=2V70PX4D5KX3H&keywords=ROSBOT&qid=1667107863&sprefix=rosbot%2Caps%2C343&sr=8-2&ufe=app_do%3Aamzn1.fos.4dd97f68-284f-40f5-a6f1-1e5b3de13370&th=1	2499	1	2499	This is the tentative robot to be purchased and used for navigating within our real environment (by utilizng our SLAM design/builtin libraries), and using the planned Gripper (used for grasping - deatails to be available by Spring 2023)	Required by Spring 2023	Needs to be purchased

15. Project Management

15.1. WBS

Library Robot				
Stage	Task	Duration	Planned dates	
			Start	End
1. Research On Existing Projects	1.1 Review exisiting algorithms for navigation using reinforcement learning	10	9/2/2022	9/22/2022
	1.2 Review existing methods for simulating a 3D enviroment on omniverse	10	9/2/2022	9/22/2022
	1.3 Review existing methods for grasping	7	9/16/2022	9/23/2022
	1.4 Identify the problem statement and delegate each tasks to group members to focus in	9	9/23/2022	9/30/2022
	1.5 Decompose the tasks needed to solve the problem and delegate to group members	9	9/24/2022	10/1/2022
2. Plan Implementation	2.1 Look into possible simulation environments	7	10/1/2022	10/7/2022
	2.2 Look into possible way to assemble all of the parts into one working robot	7	9/30/2022	10/7/2022
	2.3 Look into hardware sensors needed for the best performance	6	10/1/2022	10/7/2022
	2.4 Request devices needed	3	10/7/2022	10/10/2022
	2.5 Regroup member's individual research and select best concept design	7	10/8/2022	10/14/2022
3. System Development	3.1 Get simple version of your task running for a place to start with	14	10/14/2022	10/28/2022
	3.2 Identify Improvements to be done	2	10/28/2022	10/30/2022
	3.3 Perform improvements to the algorithms and system	19	10/30/2022	11/18/2022
4. Build Software Prototype	4.1 Build a simulation platform	7	11/18/2022	11/25/2022
	4.2 Integrate the separate portions to create one simulation	7	11/25/2022	12/2/2022
	4.3 Test the simulation	7	12/2/2022	12/9/2022
5. Test Software Prototype	5.1 Develop testing criteria for possible hardware implementation	5	1/22/2023	1/27/2023
	5.2 Perform tests	7	1/27/2023	2/3/2023
6. Build Hardware Model	6.1 Build hardware model	7	2/3/2023	2/10/2023

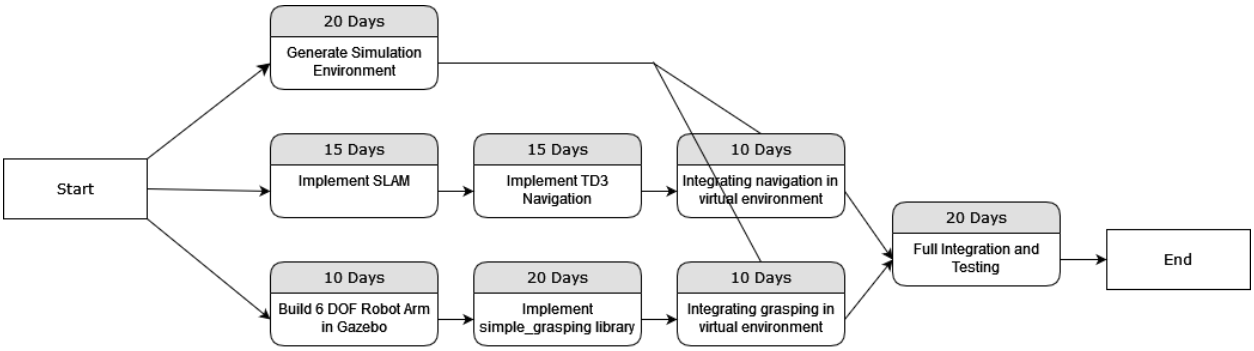
	6.2 Test sensors	5	2/3/2023	2/8/2023
	6.3 Train model	14	2/10/2023	2/24/2023
	6.4 Test and refine model	21	2/24/2023	3/17/2023
7. Document and report	7.1 Document design process	8	3/16/2023	3/24/2023
	7.2 Refine analysis and proposed solution	6	3/24/2023	3/30/2023
	7.3 Prepare final report	12	4/1/2023	4/12/2023
	7.4 Prepare poster	7	4/12/2023	4/19/2023

15.2. DSM

Library Robot																					
TASK		A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
1.0 Abstract Of the Project	A	A	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
2.1 Problem analysis	B		B																		
2.2 Black-box	C		X	C	X																
2.3 Problem statement	D		X	X	D																
4.1 Background Research	E		X	X	X	E															
4.2 Concept Generation	F			X	X	X	F	X													
4.3 Concept Selection	G					X	X	G	X												
5.1 Modeling	H		X	X			X	X	H	X											
5.2 Optimization	I						X	X		I		X	X								
6.1 Initial design proposed	J								X	X	J	X									
6.2 Changes made to initial design	K										X	K	X	X	X						
7.0 Budget	L							X			X	X	L			X	X				

8.1 Description of success	M							X	X		X		M		X	X				
8.2 Issues faced	N							X	X	X	X		X	N	X		X			
8.3 Changes made	O						X		X	X				X	O	X			X	X
8.4 initial result	P															P				
9.0 Ethics	Q		X		X	X											Q			
10.1 Criteria for testing	R		X			X												R		
10.2 Test Data	S																		S	
10.3 Discussion on testing data	T																		X	T

15.3. Critical Path



15.4. Gantt Chart

Color code:	Blue: in progress	Green: complete	Red: problem	Grey: uncertain (future)									
Capstone team (Baraa, Eddie, Manuel and Priyanshu)	Sep 20 - Sep 24	Sep 27 - Oct 1	Oct 4 - Oct 8	Oct 11 - Oct 15	Oct 18 - Oct 22	Oct 25 - Oct 29	Nov 1 - Nov 5	Nov 8 - Nov 12	Nov 15 - Nov 19	Nov 22 - Nov 26	Nov 29 - Dec 3	Dec 6 - Dec 10	Dec 13 - Dec 17
IMPORTANT DEADLINES													Project Design Submission
Establish Management System													
Preliminary Python Code													
Testing Code													
Migrate Code													
Testing in Gazebo													
Project Design Document Finalization													

16. References

1. "Transportation Economic Trends: Transportation Employment - Industry," *Bureau of Transportation Statistics*. [Online]. Available: https://data.bts.gov/stories/s/Transportation-Economic-Trends-Transportation-Empl/caxh-t8jd/. [Accessed: 30-Nov-2022].
2. "What is an autonomous car? – how self-driving cars work," *Synopsys*. [Online]. Available: https://www.synopsys.com/automotive/what-is-autonomous-car.html. [Accessed: 30-Nov-2022].
3. CatClifford, "Elon Musk: Tesla's work 'supersedes political parties, race, Creed, religion'," *CNBC*, 06-Nov-2018. [Online]. Available: https://www.cnbc.com/2018/11/05/elon-musk-teslas-work-is-important-to-the-future-of-the-world.html. [Accessed: 30-Nov-2022].
4. McCarroll, C., Cugurullo, F. Social implications of autonomous vehicles: a focus on time. *AI & Soc* 37, 791–800 (2022). https://doi.org/10.1007/s00146-021-01334-6
5. Author: Devin C. Gladden | Apr. 26, Author: Devin C. Gladden, Author: D. C. Gladden, The Author Devin C. Gladden 2020-21 Fellow, and Devin C. Gladden 2020-21 Fellow, "When will we trust self-driving cars?," *Belfer Center for Science and International Affairs*. [Online]. Available: https://www.belfercenter.org/publication/when-will-we-trust-self-driving-cars. [Accessed: 30-Nov-2022].
6. A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "OctoMap: An Efficeint Probabilistic 3D Mapping Framework Based on Octrees," 31-Dec-2012. [Online]. Available: http://www.arminhornung.de/Research/pub/hornung13auro.pdf. [Accessed: 30-Nov-2022].
7. "ANSI/RIA R15.06-2012- The Industrial Robot Safety Standard," *ISHN RSS*, 23-Jan-2018. [Online]. Available: https://www.ishn.com/articles/107815-ansiria-r1506-2012--the-industrial-robot-safety-standard. [Accessed: 30-Nov-2022].
8. "Reinforcement learning," *Wikipedia*, 29-Nov-2022. [Online]. Available: https://en.wikipedia.org/wiki/Reinforcement_learning. [Accessed: 30-Nov-2022].
9. An Introduction to Simultaneous Localization and Mapping (SLAM) for Robots - Technical Articles (control.com)
10. https://arxiv.org/pdf/2103.07119.pdf